بسم الله الرحمن الرحیم

مباحث منتخب (مدیریت و زمانبندی پروژه)

رشته : کارشناسی ارشد مهندسی صنایع مدیریت سیستم و بهره‌وری

نام استاد : دکتر بهروز افشار نجفی

دانشگاه : دانشگاه آزاد واحد قزوین

جلسه اول

یک قطعه‌ای را از کوره بیرون آوردیم باید صبر کنیم تا خنک شود و بعدش برای کارهای بعدی انجام شود.

یک جلسه Classification (طبقه‌بندی) دارد درجلسه که ۴ یا ۵ است.

مسائل زمانبندی پروژه را تقسیم می‌کنیم. بعد سراغ مسائل زمانبندی unconstrain (مسائل بدون محدودیت منابع)، یعنی ما برای اینکه زمانبندی پروژه را شروع کنیم اولش مسئله را راحت شروع می‌کنیم یعنی فرض می‌کنیم محدودیت منبع نداریم بعد از اینکه همه آیتم‌ها را صحبت کردیم به سراغ resource constrain می‌رویم. (زمانبندی پروژه با محدودیت منابع) خود این مسئله نسخه‌های مختلفی دارد که تا آخر ترم در رابطه با آنها صحبت خواهد شد. روش‌های دقیق را می‌گوییم، روش‌های ابتکاری، روش‌های نسخه‌های دیگه از مسئله مطرح می‌شود.

RCPSP ⟵ Resource Constrained Project Scheduling Problem

(زمانبندی پروژه با محدودیت منابع)

یک اصطلاح خلاصه و بسیار به جا می‌باشد. مثلاً CPM در ذهن شما جا افتاده است.

MRCPSP ⟵ بعد از مسئله RCPSP در مورد مسئله زمانبندی پروژه با Multi mode بحث می‌شود. M به خاطر multi mode بودن است .

می‌خواهیم کانالی برای لوله‌گذاری، تأسیسات حفر کنیم. این کانال به طول ۳۰ متر، عمق ۰.۵ متر و ۵۰ سانت عرض ، چقدر زمان برای حفر کانال به طول ۳۰ متر می برد ؟ با استفاده از ۲ کارگر مثلاً ۳ روز طول می‌کشد. با استفاده از ۵ کارگر یک روز طول می‌کشد.

می‌توانید با کارگر بیشتر زودتر انجام دهید. به هر کدام از اینها یک execution mode یا یک روش اجراء می‌گویند. یعنی برای انجام کار دو روش دارید با دو کارگر و یا با ۵ کارگر می‌توانید انجام دهید.

RCPSP یعنی هر فعالیت فقط یک روش اجراء دارد.

MRCPSP یعنی هر فعالیت چندین روش اجراء برای انجام آن وجود دارد.

حالا فقط بحث کارگر روز نیست . این پنجره‌هایی که اینجا نصب شده است می‌توانست چه طوری باشد ؟

Mode ها یا حالت‌های آن را بگویید؟

این پنجره‌ها می‌توانست گالوانیزه، آلومینیوم، PVC و ... باشد مثلاً اگر آلومینیومی بود ۸۰۰ هزار تومان و اگر گالوانیزه بود ۲۰۰ هزار تومان در اینجا یعنی کیفیت خوب ولی گران تر.

حال در مسئله RCPSP ما باید این mode ها را چی کار کنیم ؟ انتخاب یک mode براساس سود و نفع

الان در مثال دو کارگر کدام یک را ترجیح می‌دهید؟

نمی‌توانید دقیقاً بگویید چون هر کدام یک مزیت و یک بدی دارد. خوبی اولی این است که تعداد کارگر کمی می‌خواد ولی زمان بیشتری نیاز است تا فعالیت انجام شود. در مورد دومی درست است که تعداد کارگر بیشتر است و زمان انجام فعالیت کم است ولی هزینه‌اش زیاد می‌باشد. به این مسائل ، مسائل Time cost trail off می‌گویند. Trail off یعنی چی ؟ یعنی موازنه ، موازنه زمان و هزینه . در واقع عین ترازو می‌باشد یک کف که پایین می‌رود کفه دیگر بالا می‌آید. یعنی هر چقدر بتوانید این دو تا را متوازن پیش ببرید موفق تر هستید.

یعنی در MRCPSP ما دنبال این هستیم که این mode ها را طوری انتخاب کنیم که این موازنه بین زمان و هزینه برقرار شود. یعنی قطعاً همه‌اش دنبال این باشیم که mode هایی با کارگر زیاد انتخاب کنیم درست است که پروژه زود تمام می‌شود اما آن وقت هزینه خیلی بالا می‌رود و بالعکس. ما دنبال این هستیم که بین آنها موازنه انجام دهیم.
آخرین بحثی که در این کتاب مطرح است دو مورد می‌باشد که وقت کلاس برای بیان آنها کم می باشد.

- Stocastic scheduling project (زمان پروژه احتمالی)
- React scheduling(زمانبندی پایه)

معرفی دو Refrence

- کتاب اول آقای دیوید و اردونت دو نویسنده کتاب Project scheduling research هستند.
- کتاب نیومن

این سه نفر افراد صاحب نام در بحث زمانبندی پروژه هستند.

مقدمه بر مدیریت پروژه :

تعریف پروژه :

یک پروژه یک فرآیند Unique (منحصر به فرد) است که شامل یک مجموعه از activities (فعالیت‌ها) است که دسترسی به یک هدف را با رعایت الزامات شامل زمان، هزینه و ... به عهده گرفتند.
تعریف بالا یک تعریف علمی است.

ویژگی‌های پروژه :

- هدف منحصر به فرد (unique Purpose) یعنی دارای یک هدف مشخصی است.
- موقتی است (temprory) یعنی پروژه ماهیتش بر اساس موقتی است.
- نیاز به منبع دارد که معمولاً از حوزه‌های مختلف است (require resource from various areas)
- دارای Sponser اولی است. یعنی پروژه برای شرکت، سازمان، فردی است که انجام می‌شود و آن قرار است هزینه انجام آن را تأمین کند.
- مهم ترین ویژگی : پروژه شامل uncertainty (عدم قطعیت) است.

چرا عدم قطعیت در پروژه وجود دارد ؟ چون unique است یعنی قبلاً این پروژه نبوده است ، سابقه‌ای ازش نداریم. پروژه یک بار وجود دارد.
مثال : کسی که در نانوایی نان پخت می‌کند کار آن یک پروژه نیست یک عملیات (Operation) است . Operation یک کار تکراری است. آن کار را هر روز دارد انجام می‌دهد پس عدم قطعیت در آن نیست.
پل سازی که ۱۰ سال طول می‌کشد یک پروژه است ، با اینکه پل سازی در جاهای دیگر هم انجام شده است ولی چون جای پل سازی، نوع خاک ، وزش باد ، تیمی که فعالیت دارند متفاوت است باعث می‌شود پروژه unique باشد. پلی که چند ماه قبل ساخته شده پیمانکارش خوش قول بوده است ولی در این مسئله پیمانکار بد قولی می‌کند. در واقع کیفیت کارش آن است بنابراین حتی اگر شما تمام document های پل‌های قبلی که ساخته شده را داشته باشید می‌توانید در این پروژه استفاده کنید ولی باز هم این پروژه متفاوت از پروژه‌های قبلی خواهد بود.

در مواردی که صبحت شد : هدف ، پیچیدگی، موقتی، عدم قطعیت ، چرخه عمر پروژه ⟵ اینها عناصر عمومی پروژه هستند.

عمر پروژه : CP عمر پروژه شامل پنج فاز است.

۱- فاز طراحی
۲- فاز تعریف
۳- فاز Planning
۴- فاز Scheduling
۵- فاز ختم پروژه

فاز طراحی :

نحوه شکل‌گیری فاز طراحی به چه صورت می‌باشد؟

لحظه شروع Live site یک پروژه از اولی ملاقات Sponser با شما است. یک مشتری به شما مراجعه می‌کند و پیشنهادی به شما مطرح می‌کند. همچنین چیزی را می‌خواهم بسازیم این می‌شود شروع فاز طراحی.

فاز طراحی فقط شامل جلساتی می‌باشد (جلسات بین مشتری و پیمانکار) که درخصوص چهارچوپ کلی پروژه صبحت می‌گردد و ریز جزئیات عنوان نمی‌گردد. مثلاً سوالاتی از قبیل : این پروژه در رابطه با چی هست ؟ کجا قرار است اجراء گردد ؟ بودجه‌اش چه مقدار می‌باشد ؟ با چه هدفی ساخته می‌شود ؟ چه زمانی قرار است به اتمام برسد ؟

فاکتور تعیین کننده در این فاز **بودجه** می‌باشد.

شما در واقع بودجه‌ای که برای این پروژه در نظر گرفته‌اید که به پیمانکار پیشنهاد می‌دهید (بعد از چندین جلسه) ، اگر پیمانکار با این چهارچوب موافقت کرد قرار داد اولیه شکل می‌گیرد و اگر نه تمام .

وقتی توافق‌های اولیه شکل گرفت یک گام به جلوتر هدایت می‌شوید.

فاز تعریف پروژه :

فاز تعریف یعنی قراردادها امضاء شده است یعنی توافق‌های اولیه صورت گرفته است با مبلغ کلی، با زمان کلی، در واقع ما پذیرفتیم این پروژه را در طی دو سال انجام بدهیم. مبلغ آن هم a تومان است. منابع هم قرار شده تعیین کنیم.

در فاز تعریف اهداف، Scope و استراتژی‌ها استخراج می‌شود اهداف پروژه مشخص می‌شود Scope یعنی چی ؟

مثلاً این ساختمان را به عنوان پروژه تعریف کنید این Scope این ساختمان چی می‌شود؟ دانشکده صنایع پروژه است ؟ چه چیزهایی را Scope تعریف می‌کنید؟

فقط اسکلت ساختمان را بسازید و بقیه را شما کاری ندارید. یعنی تا چه مرحله‌ای از کار به عنوان پروژه تعریف شده است. آیا مثلاً محوطه سازی هم جزء کار است ؟ شما نمی‌دانید هست یا نیست آن باید نوشته شود. یعنی باید اقلامی از کار باید به عنوان پروژه پوشش داده شود اینجا به اصطلاح نوشته شود.

در فاز اول راجع به جزئیات صحبت نمی‌شود بحث خیلی کلی است ولی در فاز دوم جزئیات را مد نظر قرار می دهید.

مثلاً Maintance این پروژه با شما است یا نه ؟ یعنی بعد از اتمام پروژه خدمات نگهداری‌اش چه طوری می باشد ؟

استراتژی‌ها : تأمین مصالح با کی می‌باشد ؟ کدام قسمت کار قرار است برون‌سپاری شود یا حمل و نقلش به عهده کی می‌باشد ؟ بحث اسکان پرسنل چه طوری می‌باشد ؟ این استراتژی‌ها در واقع در اینجا مشخص می‌شود.

خلاصه : این فاز همان فاز اولی است ولی با زوم بیشتر با جزئیات بیشتر. اگر در فاز یک نکات مبهمی باقیمانده باشد در این فاز برطرف می‌گردد. معمولاً با احتمال کمی ممکن است باز پروژه در این مرحله Cut گردد. یعنی توافق‌ها صورت نگیرد ولی احتمالش کمتر از فاز یک است.

استراتژی‌های یکسری رویکردهایی است که تصمیم گرفتن راجع به آنها در طول پروژه ، حیاتی هستند. به طور مثال تأمین منابع ، مصالح است شما مبلغی را برای پروژه تعیین کرده‌اید ، Scope اش را هم مشخص کرده‌اید. ولی در

ذهن من پیمانکار این است که این مبلغ با فرض این است که شمای کارفرما مصالح را تأمین می کنید ولی شما در ذهنتان دقیقاً عکس این باشد در استراتژی‌ها در واقع این بحث‌ها را در واقع تقسیم می‌کنیم که مثلاً برای کدم بخش‌ها تأمین مصالح با پیمانکار است و کدام بخش‌ها به عهده کارفرما است یا می تواند همه‌اش با پیمانکار یا همه اش با کارفرما باشد.

بحث دیگر در رابطه استخدام نیرو می‌باشد که آیا به عهده پیمانکار است یا به عهده کارفرما. یا بحث‌های برون سپاری است ، کدام قسمت کار را پیمانکار به عهده نگیرد. مثلاً من پیمانکار بخش تأسیسات را به عهده نگیرم و به عنوان استراتژیک آن را برون سپاری کنم که این باید در قرارداد ثبت شود.

فاز برنامه‌ریزی (Planning)

WBS یا شکست کار

- براساس شکست کار پروژه را از کل به جزء تا به پایین ترین سطوح تقسیم می کنیم.
- پایین ترین سطوح همان Acitivies (فعالیت‌ها می‌باشد)
- من بعد هر جا که فعالیت هست یعنی پایین ترین سطوح هستند.

معمولاً۶ یا ۷ لایه در پروژه‌های بزرگ لازم است بشکنیم. به طور مثال، نمی گوییم نصب کلید شمارهٔ ۴ کلاس ۳۱۰ ، یعنی اینقدر وارد جزئیات نمی‌شویم. به اندازه‌ای ریز می‌کنیم که منقطی باشد. به اصطلاح از نظر کنترلی به عنوان یک فعالیت تعریف بشود.

هر چقدر پروژه بزرگتر باشد این بسته‌های کاری بزرگتر هستند. ممکن است این آخر سر نصب کلید پریزهای کل ساختمان به عنوان یک کار به حساب بیاید.

OBS (organization B... Structure) : از WBS یک کپی بگیرد ولی خالی به جای فعالیت‌ها (به جای گفتن نصب کلید پریزرها) مسئول آن کار را بنویسید. مثلاً آقای فلانی، بخش فلانی ... یعنی در OBSوظیفه‌ها مشخص می‌شود یعنی مشخص می‌شود که از نظر کنترل پروژه مسئول را پیگیری کنید. مثلاً این پنجره نصب شده است ولی کنارش خوب نصب نشده یا به اصطلاح درزگیری نشده ، الان چه کسی باید مقصر بدانیم؟ چی کار بکنیم؟ WBS را روی OBS بگذارید و ببیند این فعالیت در OBS مسئولش چه شخصی بوده است که الان باید پاسخگو باشد.

در OBS مسئولیت‌ها مشخص است که شما فرد که در این پروژه درگیر هستید کدام فعالیت را باید پاسخگو باشید.

PN (project Network) شبکه پروژه :

در شبکه پروژه فقط پیش‌نیازی مشخص می‌شود . این شکل الان یک PN است.

جهت فلش‌ها یعنی جهت پیش‌نیازی ها، روی PN چه چیزهایی نوشته می‌شود اسم فعالیت (مثلا فعالیت ۳) زمان آن و منبع آن .

این مثال را الان چه طور می‌خوانید ؟

فعالیت ۳ با تعداد ۱۶ کارگر و با هفت پاکت سیمان به مدت ۱۰ روز به طول می‌انجامد.

یعنی PN ، پیش‌نیازها، duration ها ، فعالیت ها و منابع ثبت می‌شود

PN را هم می‌توان با شکل و هم می‌توان با جدول نشان داد.

در هفته‌های بعد مثال‌ها که زده خواهد شد درخصوص زمانبندی پروژه است از فاز طراحی شروع نمی‌کنیم.

زمانبندی پروژه : تعیین زمان شروع و پایان پروژه را زمانبندی پروژه می‌گویند.

و آخرین فاز هم فاز termination (تحویل کار یا تسویه حساب) می‌باشد.

Cost ◄——— هزینه Staff ———► نیروی کار

این نمودار نشان می‌دهد در طول پروژه به چه میزان هزینه و چه میزان نیروی کار لازم است. البته ۲ تا نمودار است این نمودار پایین نرخ آن است و نمودار بالایی نمودار تجمعی است.

در فاز اول (فاز design) که اوائل پروژه است توجه کنید این نمودار با شیب کم به چه معنی است ؟

اوایل پروژه هزینه زیادی پرداخت نمی‌کنیم و نیروی کار زیادی نیز لازم نیست چون معمولاً ۳، ۴ ، ۵ نفر قرار مذاکره کنند و هزینه‌ها هم در حد هزینه‌های اداری است ولی از این قسمت به بعد روند صعودی است که نشان دهنده اوج هزینه‌ها و نیروی کار است که در فاز انجام پروژه می‌باشد. یعنی بیشترین هزینه و بیشترین نیروی کار در انجام پروژه داریم. دوباره که به سمت آخر پروژه می‌رویم هزینه و نیروی کار رو به کاهش می‌رود.

این نمودار همین را نشان می‌دهد فقط تجمعی‌اش است یعنی تا اینجا چقدر هزینه شده است اگر دقت کنید در نمودار تجمعی شیب مهم است کجا بیشترین شیب را داریم جایی که وارد Implementation می‌شود متوجه می‌شوید که هزینه و نیروی کار زیادی اضافه می‌شود و در آخرهای پروژه نمودار مسطح می‌شود.

درست است که مسأله را زمانبندی کردیم ولی این فازها سری نیستند یعنی فاز J انجام می‌شود و تمام خوب حال فاز دو انجام و تمام. این فازها با همدیگر Overlap (همپوشانی) دارند.

مثلاً در این شکل :

مذاکرات اولیه که صورت می‌گیرد همزمان که قضیه کاملاً جدی می‌شود فاز برنامه‌ریزی هم به موازات آن شروع می‌شود یا حتی فاز اجرایی هم داره شکل می‌گیره (مثلاً نقشه‌ها را دارید آماده می‌کنید)

مذاکرات یکجا بالاخره تمام می‌شود اما فاز برنامه‌ریزی را نگاه کنید یک فازی نیستند که شما فکر کنید این در واقع یک کار تقریبا تا نزدیکی‌های آخر پروژه هنوز این برنامه‌ریزی انجام می‌دهید به خاطر اینکه کلی Replan باید انجام شود. Replan یا برنامه‌ریزی مجدد . برنامه می‌ریزد در واقع مغایرت دارد و اشکال داره ، دوباره اشکال داره ، دوباره اصلاح . یعنی بارها و بارها می‌توانید برنامه را شما اصلاح کنید. اگر دقت کنید دوباره فاز همان executing همان implementation فوقش من اسلایدهاش هست. اصطلاحاتش عوض شده ولی مفاهیمش همان است اینم فاز اجرای پروژه ببینید فاز اجرای پروژه در پروژه بیشترین حجم کار را در واقع داره به خودش اختصاص می‌دهد از اول تا آخر. کدام کار تقریباً از اول تا آخر وجود دارد با تقریباً وزین یکسان فاز کنترل پروژه ، کنترل پروژه یک فرآیند تقریباً دائمی است از اول تا آخر. هیچ کدام دیگر یک همچنین ویژگی را ندارد.

و نهایتاً فاز closing پروژه یا فاز بستن پروژه (تحویل پروژه) :

تحویل پروژه یک کار لحظه‌ای نیست ببنید از کجا شروع، کار تمام نشده ولی دارید مثلاً یک محوطه را نرده می‌کشید یعنی لازم نیست که حتماً کل کار صفر و کامل بشه و در یک روز یا در یک هفته تحویل بدهید. گام به گام کار که تکمیل می‌شود الان در واقع اینجا چه اتفاقی افتاده است یک قسمتی از کار به طور کامل تکمیل شده آماده تحویل می‌شود و دیگر آخرهاش هم تحویل نهایی کار است. این در واقع بیشتر می‌خواهد بگوید که کار به اصطلاح تیم‌های مختلف درگیر پروژه هستند کارشون همپوشانی دارد یعنی کار مستمری را دارند. اینجا اشاره کرده تیتر را بخوانید لطفاً .

امتیازات استفاده از مدیریت پروژه ، آره علمی ، رسمی. یک مدیریت پروژه آنهایی که مثلاً تاریخی صحبت می‌کنند پروژه‌ها را به کی نسبت می‌دهند؟ تاریخ پروژه‌ها را. اهرام مصر، نمی‌دونم معابد ماهایا، پروژه دیوار چین اینها هم یک پروژه‌اند به هر حال با یک علمی با یک دارایتی با یک فوت و فنی اینها مدیریت شدند تا پروژه ساخته شده است ولی آن چیزی که به اسم مدل پروژه نوین می‌شناسیم تاریخچه‌اش خیلی جدید است و اینجا در واقع سعی می‌کنیم این امثال‌ها را معرفی کنیم. میگه شما اگر از مدت پروژه علمی استفاده کنید این ویژگی‌ها را دارد :

- کنترل بهتر منابع مالی، فیزیکی و منابع انسانی
- روابط بهبود یافته با مشتری
- زمان‌های توسعه کوتاه
- هزینه‌های پایین
- کیفیت بالا و قابلیت اطمینان بالا
- حاشیه سود بالا
- بهره‌وری بالا
- هماهنگی داخلی بهتر
- وجدان کاری بالا

دانش مدیریت پروژه به دنبال این است که این امتیازات را برای شما فراهم کند خوب همه آنها امتیاز بودند، همه اینها یعنی هزینه ها کم بشوند، بهره‌وری بالا برود، سود بالا برود از منابع بهتری استفاده بشود کیفیت کار بهتر بشود وجدان کاری بهتر بشود اینها همشان حسن است.

ما توی یک پروژه با سه تا محدودیت با ۳ تا آرمان مواجه هستیم :

- Scope goals (اهداف نهایی): پروژه داره سعی میکند که به چی برسد ؟
- Time goals (اهداف زمان) : چقدر این پروژه باید طول بکشد تا تکمیل بشود؟
- Cost goals (اهداف هزینه) : چقدر باید هزینه داشته باشد؟

جواب‌ها :

سوال اولی : پروژه دنبال این است که به چه کار برسد ؟ به یک کار خیلی خوب، کیفیت خوب

سوال دومی: پروژه دنبال این است که کی تمام شود ؟ خیلی زود

سوال سومی : پروژه چقدر باید هزینه داشته باشد ؟ خیلی کم

مشخص است من دوست دارم پروژه‌ام در زمان کم با هزینه کم و با بهترین کیفیت، با بهترین Scope پوشش داده شود.

اگر خوب دقت کنید این سه تا هدف با همدیگر چی دارند؟ با هم تضاد دارند.

اگر شما بخواهید یک کار خوب بخواهید انجام بدهید پس باید عجله نکنید و باید هم پول خرج کنید نمی‌شود که با هزینه کم و خیلی سریع یک کار خوب بدست بیارید.

سوال دوم : اگر بخواهید پروژه خیلی زود تمام شود. پس به کیفیت اهمیتی قائل نشوید هزینه‌اش هم نمی‌توانید بگویید کم باشد باید پول خرج کنید که زود انجام بشود.

سوال سوم : اگر بخواهید هزینه نکنید پس دیگر کیفیت هم نخواهید داشت و عجله هم ندارید.

یعنی هر کدام را که می‌خواهید به اصطلاح بیارید بالاتر آن دو تای دیگر پایین می‌روند. مدیر پروژه وظیفه اش اینطوری تعریف می‌شود آن وظیفه مدیر پروژه است که بتواند بین این سه تا مورد بالا یک تعادل و توازن خوب را اجرا کند که به عنوان وظیفه محوری یک مدیر پروژه است. یک آرمان اگر آن بالا تعریف کنید باید آن ایده‌آل بشود برای اینکه به آن نقطه ایده‌آل برسید این ۳ تا به اصطلاح وجود دارد. به اصطلاح تضاد دارند حالا ما باید بتوانیم با استفاده از مفاهیم جدید و علوم نوین تا حد امکان به این نقطه آرمانی نزدیک بشویم.

یک مثال توجه بفرمایید یک نمونه از پروژه‌های IT . Standish group یک گروه مطالعاتی یک گروه به اصطلاح مشاور در آمریکاه است سال ۹۵ پروژه‌های IT را بررسی کردند. یادتان نرود زمان کی ؟ ۹۵ . چی رو در این پروژه‌ها

دیده است. بررسی کرده که پروژه‌های IT این نتایج را دارد فقط ٪۱۶/۲ پروژه‌های IT موفق بودند و ٪۳۱ آنها قبل از اینکه به تکمیل برسند کنسل شده اند.

تعریف موفق پروژه یعنی چی ؟ به نتیجه رسیده است یعنی به آن اهدافی که تعریف شده است رسیده است و با آن بودجه ، با آن منابع و با آن زمان

۹۵ یادتان باشد. بعد همین موسسه سال ۲۰۰۱ دوباره همین پروژه‌های IT را بررسی می‌کند یعنی ۶ سال بعد . خروجی را ببینید به چی رسیده است؟ time overruns ها به طور قابل توجه‌ای کاهش پیدا کرد و به ٪۶۳ در مقایسه با، از ٪۲۲۲ ، time overruns رسیده به ٪۶۳ است . time overruns یعنی چی ؟ آره در زمانی که باید تخطی کند (تجاوز کنی) می‌شود time overruns. مثلاً به شما یک پروژه می‌دهم که سه هفته دیگه تحویل بدهید ولی شما به جای سه هفته ، ۴ هفته بعد تحویل می‌دهید و نمی‌توانید در مدت سه هفته به موقع تحویل بدهید. یک هفته time overruns داشتید یعنی چند درصد ؟ یک هفته تقسیم بر سه هفته می‌شود ٪۳۳ . یعنی ٪۳۳ time overruns داشته‌اید.

سال ۹۵، time overruns های ٪۲۲۲ بود ولی ۶ سال بعد اوضاع بهتر شده و شده ٪۶۳ . cost overruns ها ٪۱۸۹ بوده ولی الان ٪۴۵ شده است. بازم ظاهراً پروژه‌های IT بهتر شده است. توی سال ۹۵ موفقیت پروژه‌های IT ، ٪۱۶ بوده است ولی الان ٪۲۸ شده است. به جای درصد تعدادی‌اش را هم گفته است.

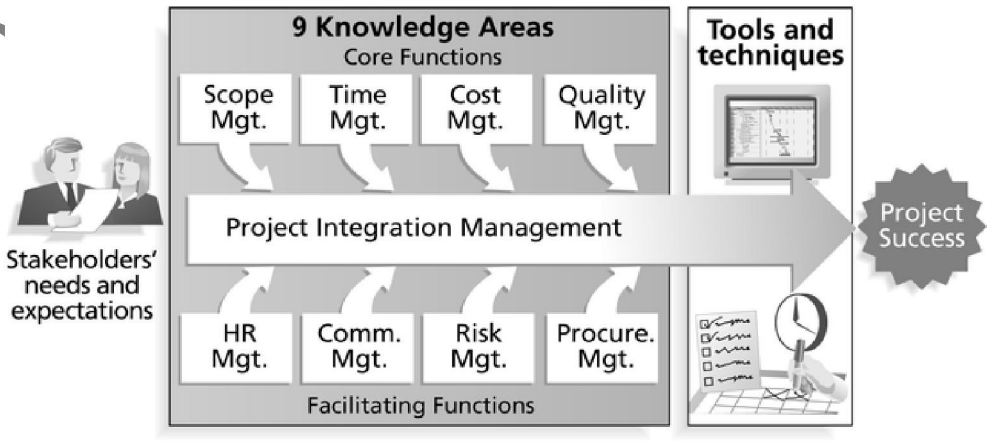
خلاصه اینکه اوضاع در ۶ سال بهتر شده است بعد آمده ریشه‌یابی کرده است. چرا اوضاع بهتر شده است ؟ چه موضوعی باعث بهتر شدن آن شده است ؟ دلیل برای افزایش در پروژه‌های موفق، ابزارهای بهتر ایجاد شدند برای monitor و کنترل بیشتر و مدیران پروژه ماهرتر با فرآیندهای مدیریت بهتر. می‌گوید در واقع استفاده از دانش‌های نوین مدیریت پروژه مثلاً پروژه IT مثلاً به این صورت در واقع نمود داشته است.

حالا چرا در آن بازه زمانی ؟ به خاطر اینکه در آن برهه سال ۹۰ به بعد اوج پیدایش دانش نوین مدیریت پروژه است یک برهه‌ای است که در واقع ان اوج گیری ، بیشترین بحث را داشته است.

ما یک موسسه‌ای داریم به اسم PMI (Project Management Institute) (موسسه مدیریت پروژه) . این تعاریف بر طبق PMI است.

Project Management را چی تعریف کرده است ؟ کاربرد دانش، مهارت‌ها ، ابزارها، و تکنیک برای فعالیت‌های پروژه به منظور تحقق ملزومات پروژه است. پس به کارگیری دانش، مهارت، ابزار و تکنیک برای رسیدن به اهداف پروژه را مدیریت پروژه می‌گویند.

۹ تا حوزه کلی داریم



که ۴ تا وظیفه اول یعنی (Scope, Time , Cost , Quality management) را Core function می‌گویند. یعنی اینها حوزه‌های اصلی مدیریت پروژه هستند.
اما ۴ تا وظیفه بعدی یعنی Human resource, Communication , Risk , Procure را facilitating Functions می‌گویند در واقع تسهیل کننده هستند.
Integration هم که خوب از اسم اش مشخص است یکپارچه کننده است. پس ما دنبال این هستیم که پروژه چی بشود؟ در Scope تعریف شده در زمان کم با هزینه کم با بهترین کیفیت انجام بشود اما برای اینکه به آن اهداف برسیم، لازم است که شما بتوانید Human resource, Communication , Risk , Procure management خوب داشته باشید و با یکپارچگی آنها بتوانید به آن اهداف برسید. به چه هدفی ؟ به این که نیازها و انتظارات Stakeholders ها یا همان ذینفع‌ها را تبدیل کنید به یک پروژه موفق این در واقع به واسطه استفاده از چیه ؟ ابزارات ؟؟؟ Stakeholders ها یا ذینفع‌های پروژه چه کسانی هستند؟ اصلی ترین آنها مشخص است که Sposor ها یا همان مشتری هستند.

- Sponsor ها (مشتریان)
- کارکنان پشتیبانی
- مشتری‌ها
- کاربرها
- تأمین‌کننده‌ها
- مخالفان پروژه

مخالفان پروژه هم حتی ذینفع به حساب می‌آیند. چه طوری ذینفع هستند ؟ شکست آنها ذینفع است. کسی که در واقع رقیب سازمان است از شکستن ذینفع است. آیتم آخر با بقیه کمی متفاوت تر است.
خوب اینجا امتیازات دیگری از مدیریت پروژه لیست شده است:

- روسا، مشتریان و سایر ذینفع ها دوست ندارد که surprise بشوند. در یک پروژه suprise شدن به چه معنا می‌باشد؟ در یک پروژه surprise شدن هیچ وقت به معنی پولش نیست به معنی بعدش است. مثلاً یک کاری باید در ۳ روز انجام می شد ولی سه ماهه که انجام نشده است. این پروژه باید یک میلیون هزینه بر می‌داشت ولی تا الان ۱۵ میلیون هزینه داشته است . در واقع می‌گوید که هیچ کدام از اینها دوست ندارند که این شکلی در واقع این اعداد به هم بریزد.
- مدیریت پروژه اطمینان را خوب فراهم می کند و ریسک را کاهش می‌دهد.
- مدیریت پروژه ابزاری برای برنامه‌ریزی، monitor (نظارت) ، ردیابی و مدیریت زمانبندی‌ها، منابع، هزینه‌ها و کیفیت فراهم می‌کند.
- در دانش مدیریت پروژه همه چی باید document شود ثبت شود این به چه دردی می‌خورد برای پروژه‌های ؟؟ مدیریت پروژه (PM) فراهم می‌کند یک تاریخچه‌ای را که در واقع metric base هست برای برنامه‌ریزی های آتی و همچنین documentation
- اعضای پروژه یاد می‌گیرند و رشد می‌کنند با کارکردن در یک محیط تیمی ساختار وظیفه‌ای یا وظیفه محور

دانش مدیریت پروژه دو قسمت دارد: یک دانش خاص مدیریت پروژه است و یک دانش از علوم دیگر است. مثلاً یک مدیر پروژه قطعاً باید خود مفاهیم مدیریت پروژه را بداند. و قطعاً باید یه مقداری چه بداند؟ به اصطلاح Applicationa area knowledge and practice یعنی مثلاً شما وقتی مدیر پروژه یک IT هستید باید

یک چیزهایی از IT بدانید جدا از اینکه مدیر پروژه هستید. اما چون مدیر پروژه ERPهست باید ERP بدانید نه به طور مفصل. پس صرفاً با گذراندن یک درس مدیریت پروژه فرد نمی‌تواند مدیر پروژه باشد. قطعاً باید از خود پروژه هم اطلاعاتی داشته باشد.

مدیریت پروژه به این مفهوم که داریم صبحت می‌کنیم تاریخچه‌اش زیاد نیست modern projenct management شروع شده از پروژه منهتم . پروژه منهتم هم فکر کنم که همه می شناسید. همان پروژه‌ای است که منجر به توسعه ساخت بمپ اتم شد. اینم با هم صحبت کردیم امروز نمودار، گانت،

اینجا را بخوانید . ۱۹۷۰ نیروی نظامی آمریکا برای اولین بار از نرم‌افزارهای مدل پروژه در واقع در پروژه‌های ساختش استفاده کرد و خط آخر از ۱۹۹۰ تقریباً تمام صنایع به نوعی از مدیریت پروژه استفاده می‌کنند و آن هم اوج‌گیری است که اشاره کردم پس خیلی تلاش زیادی ندارد . شاید معمولاً ۳۰ سال بعدها به این فرمتی که ما الان داریم می‌بینیم در حالی که خود پروژه‌ها تاریخچه‌شون به چند هزار سال برمی‌گرده ولی مدیریت پروژه به اصطلاح علمی‌اش خیلی بحث قدیمی نیست سال ۹۶ مدیریت پروژه شغل Number one انتخاب کرده بود. عرض کنم موسسات حرفه‌ای مثل PMI در واقع به طور قابل توجه‌ای رشد کرده بود حالا اشاره کرده به متوسط حقوق مدیران پروژه

PMI اولین کنفرانس را در سال ۲۰۰۰ کرد.

PMBook که یک در واقع راهنما و یک به اصطلاح محور علمی مدیریت پروژه است استاندارد ANSI دارد که یک استاندارد معروفی است.

صدها کتاب جدید، مقاله و کنفرانس‌ها که ارائه می شوند مربوط به مدیریت پروژه است. PMI یک Certificate هم دارد به اسم PMP کهمعمولاً افرادی که مدعی باشند که مدیر پروژه هستند و عملی تخصص دارند در این قضیه و PMP در واقع Certificate محکمی بر این است که فرد واقعاً یک مدیر پروژه هست.

مدیریت پروژه هم نرم‌افزار زیاد دارد که بحث از موضوع کلاس خارج است به سه قسمت تقسیم می‌شوند قیمت‌هاش هم بستگی به میزان کاربری که قراره ازش استفاده می‌کند دارد. این Scan کلی از موضوع من فقط خواستم جدا کنم که آن جایگاه زمانبندی را جدا کنم یک مسیر بسیار طولانی وجود داشته این وسطش یک تیکه از کار به اسم زمانبندی را فقط داریم راجع به صحبت می‌کنیم بقیه بحث‌ها بیشتر کار شده‌اند. زمانبندی کمتر. همان اول که شروع کردیم زمانبندی پروژه مباحثش کمتر است شاید می‌شه گفت مقاله‌هایی که در زمانبندی پروژه است بیشترشون مربوط به همین ۷ و ۸ سال اخیر است یعنی بسیاری از مدل‌های زمانبندی پروژه هستند که اصلاً هیچ کس به آنها وارد نیست مباحثش خیلی بکر است و اگر در واقع جای بسیار کار کردن دارد. هفته بعد با پارت ۲ متمرکز با زمانبندی پروژه است شروع می شود ترم‌های قبل یک پروژه داشتیم و یک تمرین که الزامی بود که آنها در این ترم اختیاری است حالا من در واقع جلوتر بازم می‌گم آن جلسه‌ای که طبقه‌بندی پروژه را باید انجام بدهید آن جلسه به شما می گویم موضوع پروژه را، چه شکلی شد تا آن موقع شما عملاً مدیریت پروژه کاری انجام نمی دهید اما تمرین ها برنامه‌نویسی است آنهایی که برنامه‌نویسی بلد هستند و یا می‌خواهند یاد بگیرند شاید تمرین‌ها را بپذیرند در ترم‌های قبل محور اصلی در ارزیابی پروژه بوده بعد تمرین بوده و بعد پایان نامه الان با اختیاری شدن پروژه و تمرین یعنی عملاً تمام ارزیابی کلاس براساس پایان ترم است پایان ترم در واقع محور بحث است اما آنهایی که به دلخواه پروژه تمرین انتخاب کنند که این موضوع ۴ تا ۵ هفته طول می کشد تا مشخص شود آن وقت پروژه در اولویت است بعد تمرین بعد میشه عین ترم قبل ولی حالت روتین این است که کلاس بیاید گوش کنید و امتحان بدهید که این ساده‌ترین حالت است. ولی انجام پروژه حسن دارد درد سر هم دارد. باید یک مقاله را انتخاب کنید و از سیر تا پیاز آن را مسلط توضیح بدهید. حسن آن این است که شما می توانید از روی این مقاله پایان نامه‌تان را تعیین کنید.

جلسه دوم

در بحث محاسبات سخت معنی دیگری دارد. سخت یعنی زمانبر. زمانی که یک معادل دیفرانسیل داده می‌شود فکر می کنید که حل کنید اما نمی تونید این سخت، سخت فکری می‌باشد اما در سوال دیگر از عدد یک تا میلیون را در نظر بگیرید. تمام اعدادی که رقم تکراری ندارند را پیدا کنید با هم جمع کنید و عدد آخر را به من بگویید. سوال سختی نیست اما احتمالا ۴ تا ۵ روز طول می‌کشد که این سوال را حل کنید. دونه دونه ارقام را لیست کنید و اعداد که ارقام تکراری ندارند را پیدا کنید و بعد با هم جمع کنید. سخت فکری نیست همین الان می‌توانید شروع کنید که حل کنید ولی خوب زمانبر می‌باشد.

یک پرانتز باز کنید (به مدت ۲۰ دقیقه از مطلب دیگر توضیح می‌دهد)

یک مسأله چه طوری به جواب می‌رسد خوب مشخص است با کمک یک الگوریتم حل می‌شود. مثلا برنامه‌ریزی خطی با Simplex حل می‌شود. برنامه مساله تخصیص (روش مجارستانی) n تا کار داریم به n تا ماشین تخصیص بدهیم، مسأله برنامه ریزی عدد صحیح با روش شاخه و شاخه حل می‌کردیم. در برنامه ریزی تولید یا در ارزیابی کار و زمان مثلاً مساله بالانس خط تولید را حل می کردیم. مسأله توالی عملیات را با روش جانسون حل می کردیم. در واقع ما برای هر مسأله ای یه روش حلی داریم. هر مسأله یک بعدی دارد یعنی چی ؟ از امتحان که بیرون می‌آید می‌گوییم مسأله کوچک داده بوده چه طور کوچک و بزرگ بودن

مثلاً یک برنامه‌ریزی خطی سه متغیره یا ده متغیره بدهم بعد ده متغیره سخت می‌باشد. ۵ تا کار رو به ۵ تا ماشین تخصیص بدهیم یا ۱۰۰ تا کار رو به ۱۰۰ تا ماشین تخصیص بدهیم وقتی تعداد کار و ماشین زیاد می‌شود یعنی ابعاد مسأله زیاد می‌شود. خوب از نظر منطقی هر چقدر بعد بیشتر شود زمان حلش بیشتر می‌شود. مثلاً یک پروژه‌ای داشتیم که هشتاد تا فعالیت داشت به جای هشت تا فعالیت.خوب معلومه که هشتاد تا باشه حلش بیشتر طول می‌کشد. پس می‌پذیریم که هر الگوریتمی که مسأله را حل می‌کند T (زمان) زمان حلش تابعی از بعد آن می‌باشد. زمان حل هر مسأله‌ای تابعی از بعدش است. یعنی بعدش بیشتر باشد زمان حل هم بیشتر می‌شود. به این T(n) می‌گویند تابع پیچیدگی محاسباتی .

سوال : تابع پیچیدگی محاسباتی برای مسأله است یا الگوریتم ؟

هر مسأله یک بعدی دارد. هر الگوریتمی که آن مسأله را حل می‌کند یک زمانی دارد که این زمان تابعی از آن بعد می‌باشد. این تابع هم حتماً صعودی است.

این T(n)می‌تواند خطی باشد.

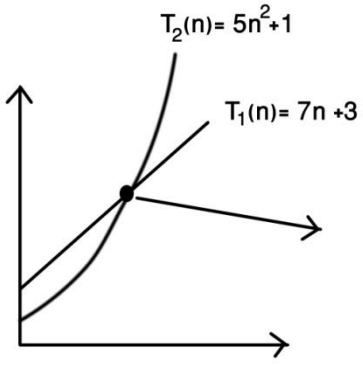
می‌تواند سهمی باشد، می‌تواند نمایی باشد و درجه ۲ باشد.

اگر خوب دقت کنید به این شکل کدام تابع پایین تر می‌باشد پایین تر بهتر است چون زمانش کمتر است. تابع سهمی پایین تر است تا این نقطه و لی از این نقطه خط پایین تر است سهمی بالاتر است.

مسئله (بعد)n

الگوریتم

T(n) زمان حل مسئله تابعی از بعد است تابع پیچیدگی محاسباتی



جواب

$$T_1(n) \in O\ T_2(n)$$

از یک نقطه‌ای به بعد همه T_1 کوچکتر از T_2 است و بهتر است.

مسائل به دو دسته سخت و آسان تقسیم می‌شوند هر مسأله‌ای که الگوریتمی برای حل آن داشته باشیم که

$T(n) = O(n^k)$ چند جمله‌ای

به آن مسئله P می‌گوییم و آسان گفته می‌شود و هر مسأله‌ای که الگوریتم برای حل آن نداشته باشیم که $T(n) = O(n^k)$ باشد. مسئله سخت است (NP) چند جمله‌ای نیستند نمایی هستند.

مسائل NP را نمی شود حل کرد؟ خیر دارند ولی T(N) ، نمایی است و زمان آن طولانی است.

هر مسئله‌ای دو تا نسخه دارد :

۱- Decision Problem (DP) نسخه DP با آیا شروع می‌شود؟ آیا این پروژه را می‌شود ۱۰ ماهه تمام کرد ؟ در انگلیسی به این سوال Yes, No Question می‌گویند. آیا می‌توانید در یک هفته انجام دهید؟
۲- Optimization Problem (OP) مسئله بهینه‌سازی : پیدا کنید این جوابی را که پروژه را در زودترین زمان ممکن تمام شود. جواب بهینه می‌خواهد.

DP – آیا کسی هست که نمره کنترل پروژه است بالاتر از ۱۶ باشد؟ راحت تر است.

OP– دنبال دانشجویی هستم که بالاترین نمره را گرفته؟ یعنی کل فضا را باید سرچ کنم تا بتوانید جواب را بیابید.

سوال مهم : مسائل بهP , NP تقسیم می‌شوند منظور DP هست یا OP ؟

منظور DP است که تقسیم می‌شود به P و NP

مسئله NP-complete ؟ سخت‌ترین مسائل در گروه NP را Np-Hard می‌گویند. یک دانه نیست گروه هستند.

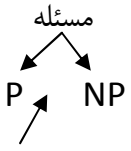
NP-Hard چیست ؟

مسئله‌ای که نسخه DP اش Np-Complete (یعنی سخت باشد) نسخه OP اش NP-hard می‌شود

سوال آخر کدام سخت تر است ؟ NP-hard چون کلاً OP ها از DP ها سخت‌تر هستند پس NP-hard ها از NP-complete هم سخت تر هستند. الگوریتمی برای آن نیست که چند جمله‌ای باشد.

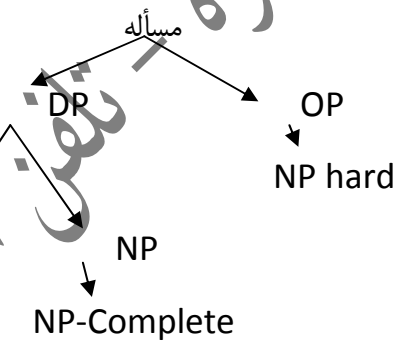
تمرین : در گوگل تایپ NP hard یا NP Complete کنید. یک لیست دویست ، سیصد تایی برای شما می‌آورد.

هر مسئله‌ای که الگوریتمی برای حل داشته باشد که T(n) متعلق به پیک N,O به توان K است می‌شود مسائل آسان به P



DP آیا می‌شود پروژه یک ماهه تمام کرد

جواب بهینه OP پیدا کنید جوابی که پروژه را در زودترین زمان ممکن تمام شود.



سخت ترین مسائل NP

در گوگل NP-hard List و NP-complete را Search کنید.

Reduction (کاهش)

اگر یک مسئله‌ای به مسئله‌ای دیگر Reduce شود معنی اش این است که B از A سخت‌تر است. Reduce بشه یعنی چی؟ خودش یک فلسفه است. Reduce شدن به معنی کوچک شدن نیست. اگر A به B Reduceشود می‌گویند B از A سخت تر است. این ماژیک مشکی یک مسئله است این مسئله جدید است نمی دونم سخت است یا آسان . اگه شما با استفاده از منطق که در این درس است این مسائلی که قبلا ثابت شده است مسئله A را کاهش بدهید به B نشان می دهد B از A سخت تر است.

کاهش Reduce

$A \infty B$

B از A سخت تر است.

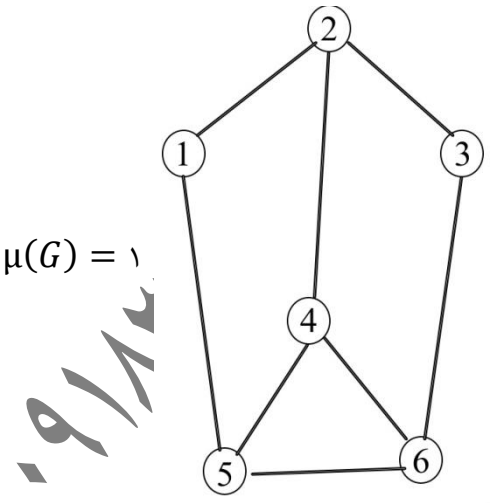
کریشنا دمورتی و دوو مسئله حداقل فعالیت مجازی را با NP-hard نشان دادند.

نویسندگان NP-hardness مسئله حداقل فعالیت مجازی را با فراهم کردن Vertex(node) Color problem که کاهش‌پذیر مسئله حداقل فعالیت مجازی است پایه گذاری کردند.

یک شبکه در نظر بگیرید این شبکه است شش تا گره وجود دارد کدام گره همه بردارها مستقیماً همه وصل هستند. نمیشه اشکال نداره. کدام دو تا گره همه بردارها را پوشش می‌دهند. بازهم نمی‌شه خوب اشکال نداره سه تا گره بگید که همه بردار به این سه تا وصل اند ۵ و ۲ و ۶ می‌شود یا نه ؟ با یکی نشد با دو تا نشد ولی Minimum Node cover این شبکه سه است. به چه دردی می‌خورد این مسأله

فرض کنید بردارها جاده‌ها باشند گره‌ها هم قرار ایستگاه امداد تأسیس شود ما می‌خواهیم با کمترین سرمایه‌گذاری به همه جا پوشش بدهیم یا در شهر نیروی انتظامی از این ایستگاه‌ها سیار بزنن که همه خیابان‌ها را پوشش بتونه بدهد.

چه اصراری داریم که مجازی‌ها را مینیم کنیم. چون ظاهراً بعدش کوچکتر می‌شود. تعداد فعالیت‌ها به معنی بعد مسئله است ؟ یعنی مطمئن هستند که تعداد فعالیت‌ها را کم کنیم مسئله راحت‌تر حل می شود.



$\mu(G) = ۱$

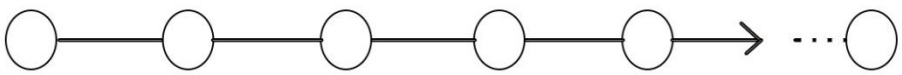
با ۳ تا

min 3

۸ فعالیت

دو تا پروژه می‌دهیم یکی ده تا فعالیت داره یکی ۵۰ تا فعالیت. اونی که ۵۰ تا فعالیت دارد چون تعداد بیشتر سخت تر حل می‌شه ولی لزوماً این طوری نیست آقای بین سال ۹۲ موضوع را به چالش کشید که اونی که تعداد فعالیت‌ها زیاد باشد فکر کنید پروژه سخت شده مثلاً یک پروژه همان باشد چند تا فعالیت داره، ۸ تا فعالیت دارد. یک پروژه دیگه این جا رسم می‌کنیم ۱۰۰۰ تا فعالیت دارد. حالا فکر کنید کدام راحت‌تر حل می‌شوند ولی چون خطی اند کدام راحت تر حل می‌شوند ۱۰۰۰ تا فعالیت دارند خوب داشته باشه ولی چون خطی اند پیشنیازهاش راحت‌تر اند و راحت‌تر حل می‌شود. فعالیت‌ها به تنهایی نمی‌توانند شاخص باشند. می تواند فعالیت‌ها کم باشه ولی سختر باشه. یک اصلاح دیگه معرفی می‌کند به جای تعداد فعالیت‌های یک عددی معرفی میکند به اسم $\mu(G)$ پیچیدگی کاهشی (موجی) یک عدد صحیح نامنفی است . یعنی چنده ؟ صفر، یک، دو، یعنی اعشاری منفی نیست با استفاده از این شاخص این شبکه $\mu(G) = ۰$ است $\mu(G) = ۱$ است . چه طوری حساب شد ؟ هنوز صحبت نکردیم.

۱۰۰ تا فعالیت



خطی است راحت تر است

این راحت‌تر است تعداد فعالیت‌ها به تنهایی شاخص نیستند.

$\mu(G) = ۰$ ⟵ نود استفاده نکردیم

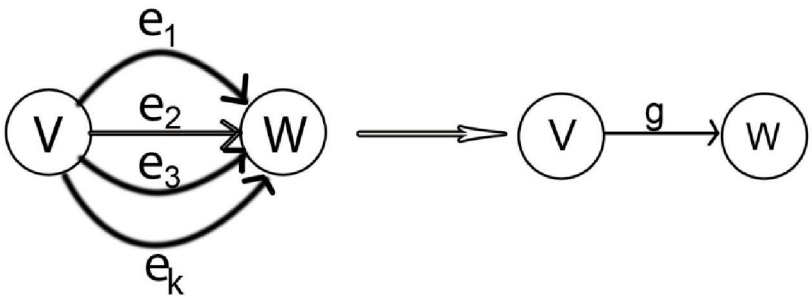
سه تا تعریف را با هم چک می‌کنیم:

کاهش موازی

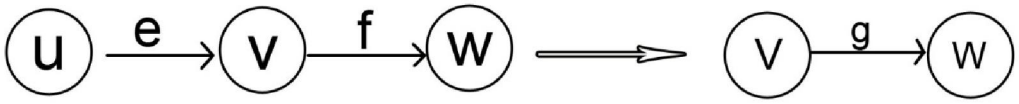
کاهش سری

کاهش گره

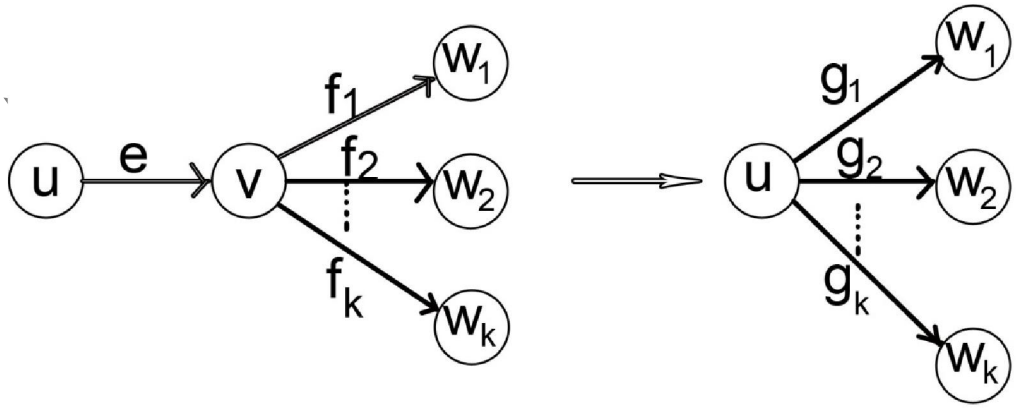
کاهش موازی : در گره‌های V,W جایگزین دو یا چند بردار $e_1, e_2, \dots, e_k$ که V را به Wوصل کردند با یک بردار تک به اسم g یعنی پس بین دو تا گره چند تا بردار موازی دارید که آنها را پاک کنید یک بردار به اسم g بگذارید کاهش موازی گویند.



کاهش سری : در گره‌های V ممکن است وقتی که e از u و v آمده است، e هست تنها بردار وارد شده به f از v و w آمده خارج شده از v وقتی که گره V یک بردار وارد و یک بردار خارج شده باشد دو بردار با هم سری هستند، بردارهای e و f را جایگزین کنند با g (در واقع روی کپی داریم کار می‌کنیم)



کاهش گره : در گره v ممکن است یک بردار وارد و چند تا خارج شده باشد یا چند تا وارد و یک خارج شده باشد و e از u به v تنها بردار وارد شده به V باشد و فرض کنید $f_1, f_2, \dots, f_k$ بردارهای خارج شونده باشند جایگزین کنید $g_1, g_2, \dots, g_k$ را به جای e و $f_1, f_2, \dots, f_k$



با استفاده از این سه مفهوم، شبکه را کوچک و کوچکتر می‌کنیم که فقط یک بردار بماند تا حد امکان موازی و سری استفاده کنیم و Node استفاده نکنیم اگر مجبور شدیم Node استفاده کنیم، در پایان تعداد دفعاتی که Node استفاده کردیم می‌شماریم به آن $\mu(G)$ می‌گوییم.

مینیم تعداد استفاده از Node را $\mu(G)$ می‌گوییم

$\mu(G)$ ⟵ کمترین تعداد دفعاتی که Nodeتا زمانی که به یک تک بعدی برسیم

۱۴ تا فعالیت دارد و ۹ تا مایلستون
اول سری یا موازی استفاده می‌کنیم که ندارد،
نتیجه (۱) مثلاً ۶ را انتخاب می‌کنیم، سپس Node استفاده می‌کنیم.
بعد موازی و سری را نمی‌شماریم بردارها حذف می‌شوند تا دیگر نباشد
دوباره Node(۲) استفاده می‌کنیم.
بعد موازی و سری را نمی‌شماریم بردارها حذف می‌شود تا دیگه نباشد
دوباره Node (۳)
بعد موازی و سری را نمی‌شماریم بردار ها حذف می‌شود تا دیگر نباشد تا به تک بردار برسیم

Min ⟶ $\mu(G) = ۳$

جلسه سوم

۷تا تعریف داریم که تعریف ها پیش نیاز یک الگوریتم که همان الگوریتم محاسبه موجی است.

تمرین : لیست مسائل NP-complete چی هست و در آن لیست مسئله minimize کردن تعداد فعالیت مجازی و مسئله Minimum – node را پیدا کنید.

تعریف اول : (۱,n,dag)

تعریف ساده‌ای است از یک گراف است که دارای ۳ ویژگی می‌باشد.

۱- اول گراف باید گراف تو ترمینال باشد. گراف تو ترمینال گرافی است که یک شروع و یک پایان دارد. شروع و پایان را ترمینال می‌گویند. یک شروع و یک پایان می‌شود دو ترمینال

۲- این گراف باید جهت دار باشد. یعنی باید بردار داشته باشد.

۳- شماره‌گذاری گره‌ها باید به اصطلاح توپولاژکی باشد. یعنی در جهت بردارها شماره‌ها افزایشی باشد.

در این اسلاید ترمینال شروع را به نام گره یک و ترمینال پایان را به نام گره n می‌نامند.

۱,n,dag ⟵ dag حروف اول Directed Acyclic Graph می‌باشد.

به عنوان مثال : این شکل گراف ۱,n,dag است چرا ۱,n,dagاست ؟ چون سه تا ویژگی دارد :

- یک شروع دارد و یک پایان
- بردار دارد یعنی جهت دار است
- جهت بردار هم از ۳به ۶ است یعنی از کم به زیاد است.
-

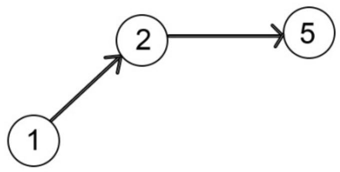
تعریف دوم :

یک بردار V Dominate می‌کند (یعنی غلبه می‌کند) بردار دیگری را مثل w را اگر هر مسیر از یک به w شامل v باشد به اصطلاح v ، w را dominate می کند.

مثال : از یک می‌خواهم به ۵ برویم چه اتفاقی می‌افتد از کجاها باید رد شود باید از ۲ رد بشیم راه دیگه‌ای ندارد. نتیجه : ۵ را dominate می‌کند. چون از یک به ۵ باید از ۲ رد بشود، ۲ پنج را dominate می‌کند. سوال ؟ ۴ هشت را dominate می‌کند یا خیر؟ جواب خیر است. چون از یک به هشت می‌توانید از ۱ ← ۳ ← ۷ ← ۸ و ۱ ← ۴ ← ۷ ← ۸

از یک به ۵ می‌رویم.

Dominate وقتی است که مجبور شوید.



تعریف سوم :

عکس تعریف دوم را به اصطلاح Reversal می‌گویند. اگر هر مسیر از v به n (گره آخر) شامل w باشد می‌گوییم w ، reverse-dominator ، (v) می‌باشد. عکس آن است آنجا از یک به w بود اینجا از v به n یعنی با یک کاری ندارد با n کار دارد (یعنی ۹). مثلاً از ۴ به ۹ چندتا مسیر است ؟ از ۴ می‌خواهم به n (یعنی ۹) دو تا مسیر وجود دارد

۴–۷–۸–۹ یا ۴–۸–۹ اگر دقت کنید هر دوی این مسیر از ۸ عبور می‌کند دو تا مسیر از ۴ به n داریم نتیجه ۸ چهار را reverse dominate می‌کند. چون تعریف اش به ۹ است اما اگر از یک صحبت می‌کردیم می‌شد مفهوم dominate کردن .

Reverse-dominate با یک کار ندارد با n کار دارد. ولی dominate با شروع یعنی با یک کار دارد.

اگر از هر مسیر از V به n شامل w باشد ، W ، reverse از ۴ به ۹ می‌خواهیم برویم.

۴ ← ۷ ← ۸ ← ۹

۴ ← ۸ ← ۹

۸، ۴ Reverse می‌کنیم

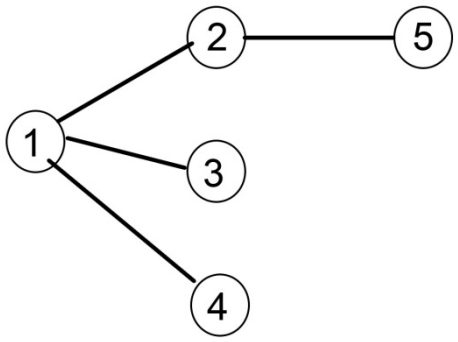
تعریف چهار : immediate dominator

W ، dominator نزدیکترین به w (غیر از خودش) بهش می‌گویند immediate dominator. به طور خلاصه dominatorای که نزدیکتر است را immediate dominator می‌گویند. نزدیک منظور شماره‌اش می‌باشد. مثلاً ابتدا یک سوال بپرسم یک همه را Dominateمیکند ؟! این جمله درست است ؟!

بله درست می‌باشد هر مسیری از ۱ به ۶ ، از ۱ به ۷، از ۱ به ۹ باید از یک عبور کند. یک همه راه Dominate می‌کند. جمله دوم : ۹ همه را Reverse dominate میکند. بله

سوال بعدی : کدام‌ها ۵ را dominate می‌کنند ؟ ۱ و ۲

حالا : کدام شمارش به ۵ نزدیکتر است ؟ ۲ ، پس ۲ ، immediate dominator یک هم هست ولی فقط dominator است اینکه شمارش نزدیکتر است می شود Immediate dominator



تعریف پنج : immediate reverse dominator

Immediate dominator ایی که شماره‌اش نزدیک تر است بازم مفهوم نزدیکتر

سوال : کدام‌ها ۴ را reverse-dominate می‌کنند ؟ ۸ و ۹

۹ همه را reverse dominate می‌کند.

سوال :

immediate dominator دو کدام است ؟ یک

immediate dominator سه کدام است ؟ یک

immediate dominator چهار کدام است ؟ یک

immediate dominator پنج کدام است ؟ دو

immediate dominator شش کدام است ؟ یک

اول تعریف را یکبار بررسی کنیم :

immediate dominator اونی است که اول dominator باشد بعد نزدیک

تعریف dominator این بود که مجبور باشد رد شوید.

برای اینکه به ۶ بروید مجبور نیستید از ۳، از ۲ یا غیره رد شود.

immediate dominator هفت کدام است ؟ یک

immediate dominator هشت کدام است ؟ یک

immediate dominator نه کدام است ؟ یک

تعریف ۶: dominator-tree

یک گرافی است که immediate dominator ها را نشان می‌دهد. با هم توافق می‌کنیم که dominate tree را با (T_1G) نشان دهیم.

تعریف ۷: reverse dominator tree

یگ گرافی است که immediate reverse dominator ها را نشان می‌دهد (T_2G)

الگوریتم

یک گرافی داریم که به عنوان ورودی مسأله داده شده است. می‌خواهیم $\mu(G)$ یا درجه سختی‌اش را محاسبه کنیم؟

اول باید T_1G و T_2G را محاسبه کنیم ، سپس از T_1G و T_2G به عنوان ورودی استفاده کنید. یک گراف دیگه به اسم Complexity Graph به اختصار CG را محاسبه می‌کنید

CG یک گراف است ولی $\mu(G)$ یک عدد است

بعد از محاسبه CG ، Minimum node cover این CG می‌شود $\mu(G)$

حساب کردن T_1G و T_2G

ورودی مسأله گراف است

الگوریتم اول :

اول دو تا پیش نیاز دارد.

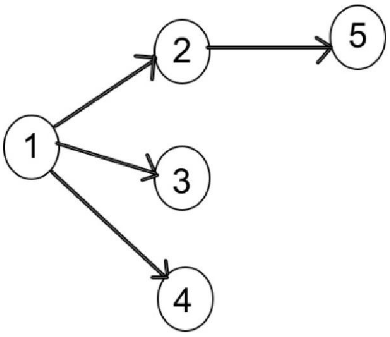
Observation ۱:

فرض کنید G یک dag باشد منظور همان ۱,n,dag است

فرض کنید گره w عضو گره V هست فقط یک بردار وارد شونده (Incoming Arc)داشته باشد.

به W یک بردار از V بهش وارد شده پس V ، immediate dominator ، W است.

به طور خلاصه : وقتی به یک گره فقط یک بردار وارد شود همان یک بردار immediate dominator اش را تعیین می‌کند.



مثلاً در شکل :

به ۵ چند تا بردار آمده است ؟ یک بردار از ۲

نتیجه : ۲ قطعاً Immediate dominator ۵ است

به ۲ چند تا بردار آمده است ؟ یک بردار از ۱

نتیجه : یک قطعاً immediate dominator ۲ است.

۱ ← immediate dominator دو، سه و چهار است.

۲ ← immediate dominator پنج است.

فقط یک گره یک بردار وارد شده باشد.

در مورد بقیه چیزی نمی‌توان گفت چون بیش از یک بردار وارد شده است.

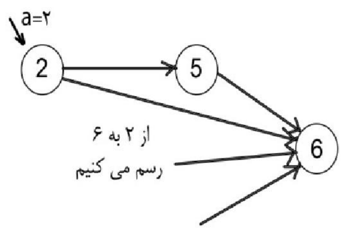
Observation ۲ :

فرض کنید یک گراف داشته باشیم که dag است یعنی همان ۱,n,dag

فرض کنید گره W بیش از یک بردار incoming Arc دارد .

W یک گره‌ای است که چند تا بردار بهش وارد گردیده و V یک پیش‌نیاز فوری که بالاترین شماره را دارد.

مثلاً راجع به شمارهٔ ۶ صبحت می‌کنیم. یعنی W=۶ به ۶ چند تا بردار وارد شده است ؟ ۳ بردار



V=۵ می‌شود پیش‌نیازها ۳,۲,۵ هستند ۵ شماره بزرگتر است.

حال اگر ۸ =w باشد V=۷ است

اگر W=۹ باشد V=۸ است .

پس جایگزین کنید برداری که از V به W آمده است به جاش A به W را اضافه کنید.

A در واقع Immediate dominator ، V می‌باشد.

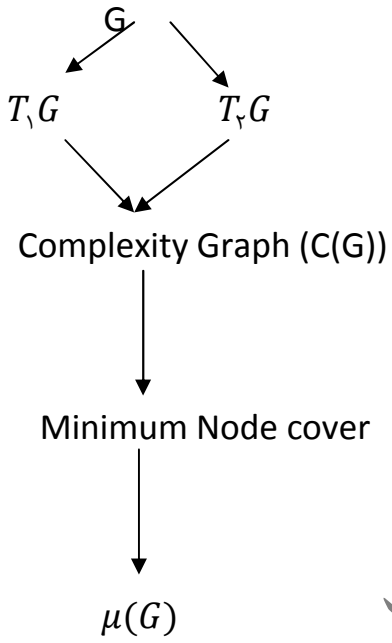
W=۶ و V=۵ نتیجه بردار ۵ به ۶ را پاک کنید به جاش از A به W استفاده کنید که A همان ۲ است immediate dominator ، V که همان ۵ است این بردار پاک کنید از ۲ به ۶ یک بردار بزنید.

مثال بعدی:

W=۸ و V=۷ نتیجه از ۱ به ۸ بردار می‌زنیم چرا چون یک immediate dominator ، ۷ است .

مثال بعدی

W=۹ و V=۸ نتیجه از ۱ به ۹ بردار می‌زنیم چرا چون یک immediate dominator ، ۸ است .



الگوریتم T_1G

۴ تا گام ساده دارد:

گام یک : ساخت ماتریس همسایگی یا ماتریس پیش‌نیازی (incidence Matrix)

اول ماتریس پیش‌نیازی بسازید.

ماتریس پیش‌نیازی : یک ماتریسی که n×n است به تعداد گره‌ها. n تعداد گره‌ها می‌باشد.

در این مثال ۹×۹ است . کجاها یک است آنجاهایی که پیش‌نیاز است. مثلاً ۳ پیش نیاز ۷ است خانه‌هایی که یک است مستقیماً پیش‌نیاز است.

سوال : این ماتریس با این شکل چه فرقی می‌کند؟

پاسخ : هیچ فرقی نمی‌کند. این شکل با این ماتریس فرقی نمی‌کند مثل این می‌ماند که من از روی لیست اسم شما را بخوانم یا کد دانشجویی. هیچ فرقی نمی‌کند کد دانشجویی معرف شما است. اون یک‌ها هم همین بردار هستند.

اگر این ۹، ۹۰۰ تا یا ۱۰۰۰ تا فعالیت بود که نمی‌توانستیم روی کاغذ نشان بدهیم که کلاً باید موضوع را به صورت عددی و کد نشان بدهیم.

گام دوم : Observation ۱ ,۲ را ستون به ستون انجام دهید.

گام سوم : Observation ۱ : اگر یک گره فقط یک بردار ورودی از v دارد پس v immediate dominator اش است پس این به اصطلاح بردار یا عدد را همان طوری بذارید بماند.

گام چهارم : Observation ۲: اگر یک گره بیش از یک بردار وارد شونده (incoming Arc) دارد به طور مکرر، جایگزین کنید بردار v به W را با A که A immediate dominator ، V است.

ستون اول چی می‌شه ؟ هیچی
ستون دوم چند تا یک دارد؟ به تعداد یک عدد یعنی وقتی یک عدد یک در ستون است مفهوم گرافی‌اش به این معنا است که یک بردار وارد شده بذارید بماند
ستون سوم و چهارم و پنجم همه مشمول ۱ Obeservation هستند هیچ‌کاری با آنها نداریم.
ستون ششم : سه تا یک دارد یعنی به اصطلاح سه بردار بهش وارد شده خوب V کدام می‌شود ؟ V=۵ یعنی یکی از همه پایین‌تر است پاک کنید به اصطلاح بردار V به W را پاک کنید به جاش از A به W اضافه کنید A همان immediate dominator پنج می‌باشد یعنی دو. پس از ۲ به ۶ اضافه می‌کنید
به طور مکرر انجام می‌دهید.
این بار از ۳ به ۶ را پاک کنید Dominator سه کدام است ؟ یک . از ۱ به ۶ بردار اضافه کنید.
بازم دو تا . از ۲ به ۶ را پاک کنید به جاش از ۱ به ۶ که هست.
نتیجه : یک بردار دارد. یک immediate dominator شش هست.
ستون ۷ :
از ۴ به ۷ حذف، Dominator چهار ، یک است. از ۱ به ۷ اضافه می‌کنیم.
۱از ۳ به ۷ حذف، Dominator سه ، یک است. از ۱ به ۷ اضافه می کنیم
نتیجه : یک immediate dominator هفت هم ۱ هست.
ستون ۸:
از ۷ به ۸ حذف، Dominator هفت ، ۱ است. از ۱ به ۸ اضافه می‌کنیم.
از ۴ به ۸ حذف، Dominator چهار ، ۱ است از ۱ به ۸ اضافه می‌کنیم
نتیجه : یک immediate dominator هشت هست.
ستون آخر
از ۸ به ۹ حذف، Dominator هشت ، ۱ است.
از ۶ به ۹ حذف، Dominator شش ، ۱ است.
از ۵ به ۹ حذف، Dominator پنج ، ۲ است. از ۲ به ۹ اضافه می‌کنیم.
بازم ۲ به ۹ صفر، Dominator دو ، ۱ است.
تمام همه ستون‌ها یک است کدام خانه‌ها یک است آنها را به هم وصل کنید
از ۱ به ۲ ، از ۱ به ۳، از ۱ به ۴ ، از ۲ به ۵، تمام
این گرافی در که می‌بینید همان Dominator tree است

اول incidence Matrix را بسازید
بعد ستون به ستون چک کنید
ستون‌های که یک عدد یک دارند کار نداشته باشید.
ستون‌هایی که بیش از یک عدد یک دارند را طبق ۲ Observation کاهش بدهید خلاصه یک عدد ۱ بماند.
یک‌های که آخر ماند به هم وصل کنید
T_1G یا dominator tree بدست می‌آید.

:T_2G

در شبکه پروژه ، جهت فلش‌ها برگردانید یعنی به جای ۹ پایان باشد ۹ شروع جهت فلش‌ها هم برعکس است یک پایان همین کار را دوباره انجام دهید. تمام بعد از انجام دادن این خروجی اش است.

Reverse dominator tree را خودتان انجام دهید.

یک کاغذ بردارید شبکه اول پروژه را رسم کنید فقط جهت‌ها را برعکس رسم کنید. همین چهار تا گام را انجام دهید. پایین ترین یک را صفر کنید. انگار پروژه از ۹ به ۱ است هم ردیف ۹ تا یک است. ۱ پایان است.

W=۷

b=۳

(۳,۷) ⟶۰

(۲,۷) ⟶۰

(۱,۷) ⟶۰

v=۷

c=۶

(۷,۶) ⟶۰

(۷,۷) ⟶۰

(۷,۸) ⟶۰

(۷,۹) ⟶۰

ساخت CG یا Complexity Graph

نمی‌توانید CG را محاسبه کنید قبلش T_1G, T_2G را بدست آورده باشید.

Data structure (ساختار داده) این الگوریتم یک ماتریس بالا مثلثی است که کدام خانه‌ها یک است I,G یک است اگر Node I مستقیماً یا غیر مستقیماً پیش نیاز node G باشد و در غیر اینصورت صفر.

سوال:

این همان Incidence Matrix است یا خیر؟

در Incidence Matrix کدام خانه‌ها یک بودند ؟ آنهایی که پیش‌نیاز بودند در Incidence Matrix آنهایی که مسقیماً پیش‌نیاز بودند ولی در این جا مسقیم یا غیر مستقیم پیش‌نیاز است یعنی یک‌های این ماتریس بیشتر است..

مثلاً ۳ پیش‌نیاز ۸ است یا نه ؟

۳ مستقیماً پیش‌نیاز ۸ نیست به طور غیرمستقیم و به واسطه ۷ پیش‌نیاز ۸ است.

در این ماتریس خانه (۳،۸) را یک بنویسم

پس از آماده سازی این ماتریس حال دو گام زیر را انجام می‌دهیم

گام یک :

برای هر گره‌ای مثل W ، immediate dominator به اسم b را پیدا کنید بله از T_1G آن را پیدا کنید و در این ماتریس کدام خانه‌ها صفر بگذارید ؟ عنصر (v,w) را . w گره‌ای که من الان در آن هستم ،Vچی هست ؟ یک شمارنده است کمتر مساوی b

فرض کنید W هفت باشد فرض کنید از T_1G Dominator = ۳ می‌باشد کدام خانه ها صفر می شوند؟

اندیس دوم ثابت است.

خانه‌های (۳،۷) (۲،۷) (۱،۷) صفر می‌شوند.

گام دو :

برای هر گره‌ای مثل V این بار immediate reverse dominator را پیدا کنید به اسم C ، عنصر (V,W) را صفر بگذارید. فرق این است اندیس اول V ثابت است اندیس دوم بزرگتر مساوی تغییر پیدا می‌کند.

فرض کنید

V=۷

immediate reverse dominator=۶

کدام خانه‌ها صفر می‌شود؟

(۶،۷) (۷،۷) (۷،۸) (۷،۹)

مثال :

گام یک از روی T مثلاً ۵ را بگید

شروع

برای ردیف i=۱,...,n

برای ستون j=۱,...,n

اگر سلول (i,j) مجاز باشد پس قرار بده a را یک

حذف کن ردیف i و ستون j

Label with dashes :

Labeled –set=Φ;

برای ردیف i=۱,...,n

در غیر اینصورت ۱ را در ردیف i قرار بده سپس

Labeled-set=labeled-set+{row i};

Label row i with '-';

{۱ ۴ ۵ ۶ ۷ ۸ ۹}

Labeling :

درحالی که labeled-set≠Φ انجام بده

انتخاب کن labeled row i ;

Labeled-set=labeled-set-{row i};

برای ستون j=۱,...,n

اگر سلول [i,j] مجاز باشد و ستون j نباشد labeled سپس

Label column j with row number i;

اگر ستون j هیچ یکی نداشته باشد.

پس

Breakthrough

در غیر اینصورت

Label row with j;

Labeled-set=labeled-set+{row i}

Breakthrough:

قرار a را یک در سلول [i,j] ;

در حالی که ردیف ‘-’i≠ label انجام بده

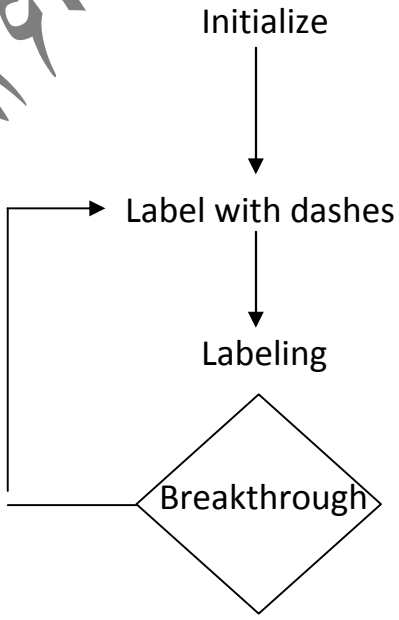
J=row label i;

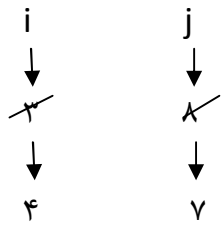
پاک کن ۱ از سلول [column label, row label] ;

I=column label [i,j] ;

قرار بده a را ۱در سلول [i,j] ;

پاک کن labels و برو به label with dashes ;





(۴,۷)

↓

Start

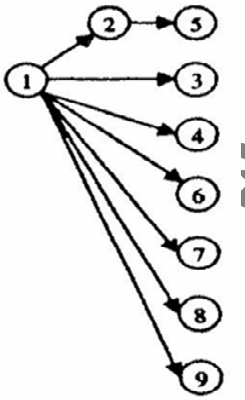
(۷,۶) (۷,۷)(۷,۸) (۷,۹)

این اندیس دوم را از ۶ بروید تا آخر (منظور n) است.

این دو تا که صفر شد بحث تمام گردیده است.

مثال :

	1	2	3	4	5	6	7	8	9
1	-	1	1	1	1	1	1	1	1
2		-	0	0	1	1	0	0	1
3			-	0	0	1	1	1	1
4				-	0	0	1	1	1
5					-	1	0	0	1
6						-	0	0	1
7							-	1	1
8								-	1
9									-



	1	2	3	4	5	6	7	8	9
1	-	0	0	0	0	0	0	0	0
2		-	0	0	0	1	0	0	1
3			-	0	0	1	1	1	1
4				-	0	0	1	1	1
5					-	1	0	0	1
6						-	0	0	1
7							-	1	1
8								-	1
9									-

Data structure ها را آماده کردم ماتریس ، کدام خانه ها یک هستند آنهایی که مستقیم یا غیرمستقیم پیش نیاز است.

گام یک : از روی T_1 را کپی کردم و قبلاً حساب کردم مثلاً ۵ را بگویید.

۵ Dominator کدام است ؟ ⟵ ۲ ⟵ پس کدام خانه‌ها صفر می‌شوند؟ اندیس دوم را ثابت نگه دارید . (۱,۵) (۲,۵) را رنگی کردیم برای صفر شدن.

۲ Dominator یک است ⟵ کدام ها صفر می‌شوند ⟵ (۱,۲) چون یک قبل‌تر که نداریم

۳ Dominator یک است ⟵ کدام‌ها صفر می‌شوند ⟵ (۱,۹) (۱,۸) (۱,۷) (۱,۶) (۱,۴) (۱,۳)

همه آنها dominator اش یک است فقط یک عدد می‌شوند.

به جز ۵ که dominator آن دو است. پس هم (۲,۵) (۱,۵) صفر می‌شوند. رنگی‌ها را که مشخص کردیم صفر می‌شوند.

گام دو $T_2 G$ را باید بگویید.

۴ Reverse dominator ، ۸ است . کدام‌ها صفر می‌شوند. (۴,۹) (۴,۸) یا

مثلاً برای ۷ ⟵ (۷,۹) (۷,۸)

یا برای ۲ ⟵ (۲,۹) چون dominatorیک است اینهای که رنگی شدن صفر می‌شوند.

کجاها بعد از انجام این دو گام هنوز یک مانده است (۲,۶) (۳,۶) (۵,۶) (۳,۷) (۳,۸)

آنهایی که یک مانده است را به هم وصل کنید. این گراف به دست آمده Complexity Graph می‌باشد.
از این T_1 و T_2 یکسری از خانه صفر می‌شدند آنهایی که صفر نمی‌شوند را به هم وصل می‌کنید.
مینیمم node cover این شبکه را محاسبه کنید. می‌شود $\mu(G)$ گفتم یکی از مسائل NP-hard است.
فرض کنید این بردارها خیابان هستند ، گره‌ها تقاطع هستند. مثلاً نیروی انتظامی می‌خواهد در این تقاطع‌ها ایستگاه پلیس به صورت سیار دایر کند. فوقش می‌خواهد با کمترین تعداد کانکس ولی همه خیابان‌ها را پوشش بدهد. پس هدف مینیمم کردن سرمایه‌گذاری در این موضوع است ولی از آن سمت پوشش همه خیابان‌ها است حالا اگر این نقشه شهر باشد. با یک کانکس می‌توان همه را مستقیم پوشش دهد نه با دو تا چی ؟ خیر ۳ و ۶ . با سه تا چی ؟ در این مثال با ۳ تا مثلاً (۶,۳,۴) یا (۶,۳,۷) یا (۶,۸,۷) می‌توانید مهم تعداد است که سه تا می‌باشد پس مینیمم کردن node cover این شبکه سه است.که $\mu(G) = ۳$
مسائلی که NP-hard هستند مسائلی که حل آنها زمانبر است حتی از نظر مفهومی سخت هم نیست. این مسأله minmum node cover ، NP-hard است و روش‌های حل مختلفی دارد. یکی از آنها را که در کلاس می‌توان عنوان کرد روش فورد و فاکرسن است این روش به این دلیل انتخاب شده است که نیاز به پیش‌نیازهای عجیب و غریب ندارد.
بهترین روش روش‌های دیگر است ولی چون نیاز به پیش‌نیاز دارد اینجا مورد بحث قرار نمی‌گیرد.
در سه بحث قبلی من اول الگوریتم را گفتم (گام‌هایش را) بعد مثال حل کردیم. اما الان در این نمی‌شود این کار را کرد. به خاطر اینکه روش فورد، فاکرسن بخواهیم ریز کنیم خیلی طولانی است. اگر از این بالا شروع کنیم خط به خط بخوانیم تا آخر بعد بخواهیم مثال حل کنیم وقتی به آخر این الگوریتم می‌رسیم اولش یادمان می‌رود که چی بود نتیجه اینکه من این را به ۴ قسمت کردم
اول : به یک data structure نیاز داریم، یک ماتریس ، چه ماتریسی ؟ این ماتریس آخر را یادتان هست چی بود؟ ماتریسی که از Complexity Graph درآوردید یک‌ها را وصل کردید این ماتریس را آنهایی که یک دارند و بردار دارند را یادتان بماند. این خانه‌هایی که یک دارند به رنگ سفید هستند را به اسم admissible cell می‌شناسیم خانه‌هایی که در یک گراف Complexity Graph بردار دارند، آنها به اصطلاح سلول‌های admissible یا شدنی هستند. برای خانه‌های که با رنگ زرد مشخص است inadmissible یا نشدنی‌اند. پس شروع با یک ماتریسی است که هیچی نوشته نشده است یعنی خالی است n در n . ولی علیرغم اینکه خالی است با هم فرق می‌کنند. یکسری admissible هستند و یک سری inadmissible . کدام ها admissible هستند؟ آنهایی که بردار داشتند مثل این که یک بودند خانه‌های (۵,۶) (۴,۷) (۳,۸) (۳,۷) (۳,۶) (۲,۶) اینها خانه‌های admissible هستند. که با رنگ جدا کردیم.
حال الگوریتم‌ها را شروع می‌کنیم.
گام اول : برای ردیف‌های ۱ تا n و برای ستون‌های ۱ تا n اگر سلول (i,j) شدن بود admissible مجاز است یک در آن خانه قرار بده و سپس سطر i و ستون (ردیف) j را حذف کن.
نکته : کلمه delete در این دستور به معنی حذف فیزیکی نمی‌باشد. یعنی از بررسی کردن کنار بگذار.
سوال : اول ردیف بعد ستون مطرح شد چه فرقی می‌کند ؟ فرق می‌کند.
اول ردیف‌ها می‌اید بعد ستون‌ها یعنی باید ردیفی scan کند اما اگر ستون می‌گفت اول باید ستون یک را ، بعد ستون دو را، درهر حالت همه خانه‌ها scan می‌شوند ولی به چه ترتیبی مهم است. وقتی اول ردیف و بعد ستون، اول ردیف‌ها را پر می‌کنیم و بعد ستون‌ها

این خانه را چک می‌کنیم admissible نیست بعدی و بعدی هم نمی‌باشد تا می‌رسیم به خانه‌ای که admissible است یک ، ۱ در آن خانه قرار می‌دهید خانه بعدی کجا می شود؟ بیایید پایین و دیگر آن را ادامه ندهید چرا ردیف ۲ و ستون ۶ را در حافظه تان نگه دارید که اینها دیگر باید بررسی نشود.

Admissible نیست ، می‌بریم و بازی نمی‌دیم ستون ۶ ،ستون ۶ جذب شد. اینجا admissible است پس یک ، یک هم اینجا می‌گذاریم دیگه ادامه ندهید. ردیف هم دیگه حذف شد. پس ردیف‌های ۲ و ۳ و ستون‌های ۶ و ۷ را دیگه بازی ندهید. Admissible نیست. این ۲ تا رد می‌شویم admissible نیست، نیست ، رد می‌شویم دیگه تا آخر نیست.

تا اینجا کار خیلی ساده بود.

حالا قسمت دوم دستور Label with dashes

یه مجموعه به اسم label-set درست کنید که هیچی هم در آن نوشته نشده باشد برای ردیف‌های ۱ تا n چک کنید اگر هیچ یکی در ردیف i نیست پس

نکته: نشانه جمع در این دستور به معنی جمع ریاضی نیست. به معنی add کردن است آن ردیف را که شمارش را به ان لیستی که خالی است اضافه کن . متوجه شدید.

بعد label بزنید همان ردیف اضافه کردیم را با dash

ردیف‌های که یک ندارند را کنارش یک تیک بزنید آن مجموعه را اسمش را بنویسید.

کدام ردیف‌ها یک ندارند ؟ ردیف‌های ۲ و ۳ یک دارند و بقیه ردیف‌ها ندارند {۱,۴,۵,۶,۷,۸,۹} یک ندارند. Label می‌خورند یک مجموعه هم این سمت درست کنید. این مجموعه شماره ردیف‌هایی هستند که یک ندارند.{۱,۴,۵,۶,۷,۸,۹} به لیست اضافه می‌کنیم.

فاز سوم :

تا اینجا کار initialize را انجام دادید و label with dash هم انجام گردید. حال در قسمت labeling هستید.

تا زمانی که labeled-set تهی نشده است (الان تهی نیست دیگه) چی‌ها نوشته شده است. یک، چهار ، پنج ، شش ، هفت، هشت و نه. تا زمانی که تهی نشده انتخاب کنید ردیفی از آن مجموعه مثلاً شماره یک را ، پس i=۱ شد بعد منها یعنی اینکه پاک کنید یعنی از آن مجموعه برش دارید. آن ردیف یا i انتخابی را از مجموعه پاک کنید پس یک الان در مجموعه نمی‌باشد. خوب حالا در همان ردیف یک ستون‌ها را چک کنید. اگر سلول (i,j) ، admissible هست و ستونش label نخورده پس label بزنید آن ستون را با شمارهٔ ردیف یعنی i اگر این ستون یک ندارد یک برنامه به اسم break through انجام می‌شود اگر یک دارد label بزنید آن ردیف را با j و دوباره این ردیف را به label-set اضافه کنید.

دستور یکبار خوانده شد حالا با مثال انجام می‌دهیم.

تا زمانی که label-set تهی نیست یک ردیف انتخاب کنید و آن را hide پاک کنید یک را انتخاب کنید از مجموعه پاک کنید پس i=۱ ، بعد در آن ردیف ستون‌ها را چک کنید اگر سلول admissible هست و label نخورده است که در اینجا نداریم یعنی کاری انجام نمی‌شود.

بعدی ، label بعدی ۴ را انتخاب کنید و پاک کنید حالا ردیف ۴ هستیم خانه‌ها را Scan کنید تا می‌رسید به خانه‌ای که admissible است و هم ستونش label ندارد. (خانه ۲ و ۶) خانه سفید بود سپس ستون j را با شماره ردیف labelبزنید. یعنی در این قسمت (پایین) شماره ۴ را می‌نویسید. (شماره ردیف می‌شود label این ستون) اگر ستون ۷ یک ندارد یه برنامه به اسم break through انجام شود در غیر اینصورت (اگر یک دارد) label بزنید آن ردیف

را ردیفی را که یک دارد. با j و دوباره آن ردیف را به label-set اضافه کنید. در این ستون یک دارد. (ستون ۷)در ردیف ۳، یک ، ۱ موجود است. این ردیف را که یک دارد label ستون بزنید شماره ۷ می‌شود وقتی که label خورد به ته این لیست این ۳ را اضافه کنید. اگر درست شد که هیچی اگر نه باید دوباره این کار انجام شود. چرا ؟ چون گفته است تا زمانی که label-set تهی نشده است این کار را انجام دهید. دوباره انجام می‌شود.

۵ را از لیست انتخاب کنید و پاک کنید خانه‌های ردیف ۵ را Scan کنید تا به خانه‌ای برسید که ۲ تا شرط را دارا باشد اولاً admissibleباشد و ثانیاً label نخورده باشد. ستون ۶ را label، ۵ بگذارید. اگر این ستون یک ندارد که break through اجرا می‌گردد در غیر اینصورت اگر یک دارد همان ردیفی را که یک دارد را (label ، ۶ بزنید) جلوش label ستون یعنی ۶ را بگذارید و به انتهای لیست هم ردیف ۲ را که label خورده است اضافه کنید.

۶ را از لیست انتخاب می‌شود و پاک می‌گردد خانه‌های ردیف ۶ را Scan کنید تا دو شرط داشته باشد ندارد پس بعدی را انتخاب می‌کنیم شماره‌های ۷،۸،۹ نیز دو شرط را ندارند.

بعدی ۳ را انتخاب می‌کنیم و بعد حذف می‌کنیم ردیف ۳ را scan می‌کنید و دو شرط را برای هر خانه بررسی می‌کنید. (دو شرط admissible باشد و Label نخورده باشد) اولین خانه سفید رنگ با اینکه admissible هست ولی label خورده است ، دومین خانه سفید رنگ هم admissible است و label خورده است می‌رویم سراغ سومین خانه سفید رنگ این خانه هم admissible است و هم label نخورده است پس ستون ۸ را label ، ۳ می‌زنیم اگر این ستون یک نداشته باشد break through اجرا می‌گردد. این ستون یک ندارد پس از این قسمت با قبلی‌ها متفاوت است.

خانه (۳،۸) یعنی i=۳ و j=۸ است.

خط اول : در خانه (i,j) یعنی (۳،۸) یک ، یک قرار بدهید. تا زمانی که row label i (i یادتان نرود که چند بود عدد در نظر داشته باشید) خط تیره نشده ، اگر دقت کنید row label i، عدد ۷ می‌باشد، row label را i را بگذارید j یعنی چند ؟ i=۳ که بود label آن ۷ بود حالا که j=۸ است الان ۷ می‌شود یعنی j عوض می شود j که ۸ بود دیگه ۸ نیست. می‌شود ۷ . i=row label (i=۳) ، ۷ می‌باشد، آن ۷ شده ۲.

حذف کنید یک را از خانه (column label , row label) این خانه الان یک، j هم شده ۷، از خانه column label و row labelیک را پاک کنید، (یک را از خانه (۳،۷) پاک کنید) یعنی خانه (۳،۷) را پاک کنید. بعدش Column label j را i بگذارید یعنی column label j می‌باشد labelستون هفت ، ۴ است پس i=۳ بود الان i=۴ می‌باشد. بعد قرار بدهید در خانه (۴،۷) را یک ، یک بگذارید.

(i,j)

پاک کنید همه labelها را برگردید به اینجا ، در واقع یک حلقه وجود دارد تا کی باید این حلقه تکرار شود ؟ تا زمانی که break through لازم نشود اگر لازم نشد توقف اگر به ستونی برسد که یک ندارد یعنی break through لازم شود بعد از break through باید دوباره از آن بالا شروع کنید. اما اگر break throughلازم نیست دیگه ادامه ندهید تمام ، توقف.

	1	2	3	4	5	6	7	8	9	*	Labelled Set
1										-	2
2						1				6	
3							1	1		7	
4							1			-	
5										-	
6										-	
7										-	
8										-	
9										-	
*						5	4	3			

i *j*

4 7

	1	2	3	4	5	6	7	8	9	*	Labelled Set
1										-	1
2						1					5
3								1			6
4							1				7
5										-	8
6										-	9
7										-	
8										-	
9										-	
*											

	1	2	3	4	5	6	7	8	9	*	Labelled Set
1										-	
2						1				6	
3								1			6
4							1				7
5										-	8
6										-	9
7										-	2
8										-	
9										-	
*						5					

چی کار کنم ؟ ردیف‌هایی که یک ندارند را خط تیره بزنید و به این لیست اضافه کنید بعد labeling را انجام دهید.
یک را انتخاب کنیم پاک کنید ، ردیف یک را scan کنید هیچی.

بعدی ۵ انتخاب و را پاک کنیم ردیف ۵ را scan کنیم خانه‌ای که admissible باشد و ستون اش label نداشته باشد هست ستون ۶ را که label ندارد را label ۵ بزنید ستون ، ۱ دارد یا ندارد اگر یک دارد چی کار کنید ؟ این ردیف را که ۱ دارد را شمارش ۶، label بزنید و این ردیف را به انتهای لیست اضافه کنید .

بعدی ۶ که هیچی

بعدی ۷ که هیچی

بعدی ۸ که هیچی

بعدی ۹ که هیچی

بعدی ۲ ، ردیف ۲ را Scan کنید. خانه‌ای که admissible باشد و ستونش label نداشته باشد. نداریم ، اون یکی که admissible هست ولی ستونش label دارد.

بعدی دیگه ندارد. Break through لازم نشد تمام توقف یعنی از حلقه اومدیم بیرون

خط آخر : تعداد ردیف‌هایی که Labelندارند با ضافه تعداد ستون‌هایی که label دارند می‌شود جواب آخر .

یکبار دیگه انجام بدهید. یک ، ۱ آنجا بگذارید و مقدار j را تغییر بدهید که قبلاً ۸ بود الان ۷ شد. بعد از خانه (column label, row label)یعنی خانه (۳,۷)یک را پاک کنید i را هم تغییر بدهید به ۴ . یک ، ۱ در این خانه بنویسید. الان همه Label ها را پاک کنید یک ها را نه ، label ها را پاک کنید. دوباره از آن بالا یعنی label with dash شروع کنید.

جواب آخر چند شد ؟

تعداد ردیف هایی که label ندارند (۲تا) با ستون‌هایی که label دارد (یکی) جمع می‌شود .

۲+۱=۳

$$\mu(G) = ۳$$

جلسه چهارم

شبکه‌های AON (Activity and Node)

تعریف اول (Cut) :

اگر گره‌های شبکه را پارتیشن کنید یعنی به دو قسمت X و Y تکفیک کنید برش یا به اصطلاح Cut می‌باشد. پس Cut تقسیم گره‌های شبکه به دو قسمت X,Y می‌باشد.

مثلاً این خط قرمز شبکه را Cut کرده است، گره‌ها را به دو قسمت تقسیم کرده است . کدام‌ها X هستند ؟

۱ و ۴ و آنهایی که آن طرف هستند y

تعریف دوم (۱,n,cut) :

یک Cut ، (۱,n,cut) نامیده می‌شود اگر یک جزء X باشد و n جزء Y باشد یعنی ۱و n نباید توی یک گروه بیافتند در دو گروه جدا قرار بگیرند. ۱,n,cut حالت خاصی از Cut است.

مثال :

یک خط افقی می‌کشیم که یک Cut ایجاد می‌شود کدام X می‌شود ؟ ۴ و آنهایی که بالای خط هستند Y می‌باشند. ولی اگر دقت کنید Cut هستند ولی ۱,n,cut نیست. ۱,n,cut این بود که یک باید در X باشد و ۶ در Y . اما در این Cut ایی که زده شد یک و ۶ هر دو در Y هستند و این دیگه ۱,n,cut نیست فقط Cut است.

تعریف سوم : UDC

یک ۱,n,cut یک UDC نامیده می‌شود (unique for great cut) اگر ۱,n,cut ، *اگر* (y, x) *تهی باشد.*

مثال : الان این یک *Cut* است و *۱,n,cut* نیز می‌باشد.

اگر خوب توجه کنید کدام بردارها برش خوردند یعنی *Cut* شدند؟

$A_۱, A_۲, A_۶, A_۸$ این بردارهایی که برش خوردند .

جهت فلش ها از *x* به *y* است یا از *y* به *x* ؟

$A_۱, A_۲, A_۸$ از *x* به *y* است و $A_۶$ از *y* به *x*

$A_۳$ اصلاً برش نخورده است.

پس یک *Cut* ، *UDC* است اگر بردارهای از *y* به *x* تهی باشد یعنی نباید هیچ برداری از *y* به *x* باشد.

نتیجه : *cut* و *۱,n,cut* است ولی *UDC* نیست.

مثال دیگر :

این هم *cut* است و هم *۱,n,cut* می‌باشد. کدام بردارها برش خورده‌اند ؟

$A_۵, A_۶, A_۷, A_۳$ همه جهت فلش‌ها از *x* (آن است یک دارد) به سمت *y* است

نتیجه : UDC می‌باشد.

تعریف چهارم :

$earliest\ cutest$ یک فعالیت تعریف می‌شود به عنوان UDCایی که کمترین شماره را دارد و شامل فعالیت هم است.

UDC که این فعالیت را Cut کرده باشد و کمترین شماره را هم داشته باشد.

مثال :

این خطها Cut هستند و $۱,n,cut$ و UDC هم هست.

نکته : هر شبکه تعداد محدودیUDC دارد یعنی UDC ها تعدادشان بی‌نهایت نیست.

مثلاً در این شبکه ۶ تا UDC وجود دارد.

یه مجموعه به اسم x و یه مجموعه به اسم y درست کنید فقط دقت کنید حتماً یک در x و n در y باشد.

مثلاً دقت کنید Cut اول به نام $C_۱$ است کدام‌ها در x هستند ؟ فقط یک و بقیه کلاً در y هستند پس یک $cut,\ ۱,n,cut\ ,\ UDC$ می‌باشد.

دفعه بعد یک باید باشد و دو را هم کنارش اضافه کنید پس ۲ , ۱ در x است و بقیه در y است ولی چک کنید که حتماً UDC هم باشد.

چرا ۳ ,۱ را ننوشت ؟ چون اگر به صورتی cut بزنیم که ۱,۳ باشد حالتی پیش می‌آید که UDC نمی‌شود.

یک باشد و هر بار یک دانه ، دو تا ، بعد سه تا بعد همه حالت‌ها را اضافه کنید. بعد چک کنید که UDC هم باشد.

خلاصه به این ترتیب ۱,۲,۳,۴,۵,۶ ⟵ x و ۶ ⟵ y

$earliest\ cutest$ ، UDC که این فعالیت را داشته باشد و کمترین شماره را هم داشته باشد.

از روی شکل در فعالیت $A_۷$ ، $earliest\ cutest$ کدام است ؟ $C_۳$ UDC که این فعالیت را داشته باشد و کمترین شماره را داشته باشد و $C_۳$ و $C_۴$ فعالیت را بریده و کمترین شماره را دارد $(C_۳)$

از روی جدول : اولین جایی که $A_۷$ وجود دارد آن UDC می‌شود و کمترین شماره را دارد.

AOA ⟵ بردارها به معنی فعالیت، گره‌ها به معنی $event$ یا رویداد یا مایلستون هستند.

AON ⟵ گره‌ها به معنی فعالیت و بردارها فقط پیش نیازی هستند در این جا مایلستون نداریم.

اصطلاح مایلستون فقط در شبکه‌های AOA می‌باشد.

سوال : کدام شبکه‌ها AOA یا AON بهترند ؟!

جواب : AOA چون مایلستون در آن مفهوم دارد.

در شبکه AOA شش ویژگی وجود دارد که اگر بعضی از این ویژگی‌ها وجود نداشته باشد ناچار از گره‌های مجازی استفاده می‌کنیم که این ویژگی را به وجود آوریم.

به سه دلیل ممکن است از مجازی استفاده کنیم :

۱- بردارها اگر موازی هم باشند یعنی بین دو گره پیش از یک بردار باشد با مجازی درست می‌کنیم.

۲- چندتا شروع و چندتا پایان داشته باشد از مجازی استفاده می‌کنیم که شروع را یکی و پایان را یکی بکنیم.

۳- برای بیان صحیح پیش نیازها هم لازم می‌شد از مجازی استفاده کنیم.

اما در *AON* فقط به یک دلیل از مجازی استفاده می‌کنیم آن هم وقتی که گره شروع و پایان بیش از یکی باشد می‌توانیم با مجازی درست کنیم. دلیل دیگری وجود ندارد.

سوال : کدام شبکه‌ها *AOA* یا *AON* بهترند ؟!

جواب : *AON* چون از مجازی کمتری استفاده می‌کنید .

پس تا اینجا یک یک شد.

در *AOA* ها پیش نیازها چه مفهومی داشته ؟ پیش‌نیازها *Binary* هستند یا پیش نیاز هستند یعنی کامل است یعنی این فعالیت تمام شود که بعدی را شروع کنیم یا مطلقاً وجود ندارد. پیش‌نیاز های *Finish-start* هستند در واقع صفر و یک هستند.

اما در این شبکه‌ها تنوع پیش‌نیازی داریم که به اصطلاح *Generalize Precedence Relations* به طور خلاصه *GPR* یا تعمیم یافته می‌گوییم.

یعنی در این جا ما چندین مدل پیش نیاز می‌توانیم داشته باشیم نه فقط *Finish to Start*

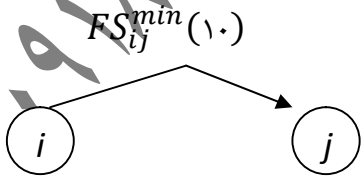
Start to start

Start to Finish

Finish to Finish

که هر کدام از پیش‌نیازهای بالا دو حالت دارد یعنی می‌تواند *Min* یا *Max* باشد پس در کل ۸ حالت پیش‌نیازی می‌تواند مطرح باشد.

بنابراین وقتی می‌گویید فعالیت *i* پیش نیاز فعالیت *j* است الان کسی که این را می‌خواند نمی‌تواند منظور شما را بفهمد باید بالای بردار یکی از این ۸ حالت پیش‌نیازی را بنویسید. مثلا بنویسید ۱۰ *FS min (i,j)* یعنی *F* مربوط به *i* و *S* مربوط به *j* است. *i* که تمام شد حداقل ۱۰ روز بعد می‌توانید *j* را شروع کنید.



i که تمام شد حداقل ۱۰ روز بعد *j* شروع شود

Lag ⟶ وقفه بین دو کار

مثال : این دیوار باید اول سفید کاری شود تا نقاشی شود. پس سفیدکاری پیش نیاز نقاشی است. این سفید کاری تمام شد باید حداقل ۱۰ روز بگذرد تا این خشک شود تا بتوانید نقاشی را شروع کنید به این ۱۰ روز *Time Lag* می‌گویند (وقفه زمانی)

Lag با *Duration* فرق می‌کند :

Lag : مدت زمان وقفه بین دو کار است

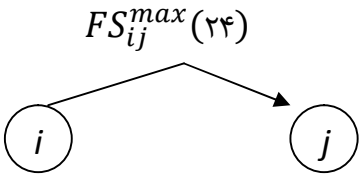
Duration: مدت زمان انجام کار است.

مثال دیگر :

وقتی بتن‌ریزی در هوای گرم انجام می‌شود باید به بتن آب زد تا ترک نخورد چه زمانی باید این آب را بزنیم ؟ حداکثر ۲۴ ساعت

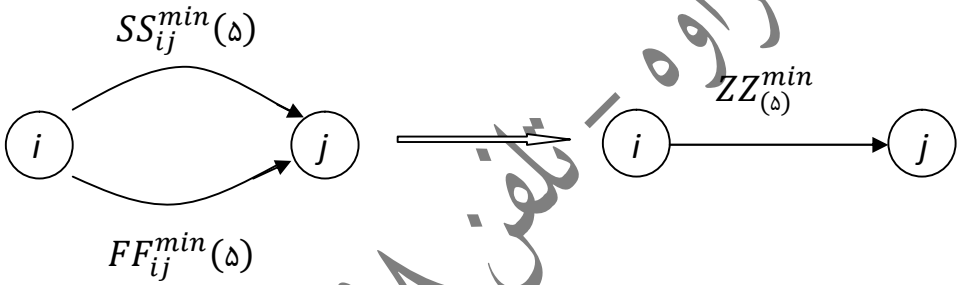
این زمان ممکن است با شدت هوای گرم کم و زیاد شود.

Max time lag = ۲۴*h*



i که تمام شد حداکثر ۲۴ ساعت بگذرد *j* شروع شود

مثال دیگر :



حداقل ۵ روز بعد از شروع فعالیت *i*، فعالیت *j* شروع شود.

اجازه نمی‌دهد فعالیت بعدی با فعالیت جاری تداخل پیدا کند. ⟵ حداقل ۵ روز بعد از پایان فعالیت *i* فعالیت *j* تمام شود.

یک مسیر به طول ۵۰ کیلومتر باید زیرسازی ، شن‌ریزی ، آسفالت گردد.

یعنی پیش نیاز هم هستند . حال چه پیش نیازی ؟

Start to Start از نوع *min*، ۵ روز یعنی ۵ روز از زیرسازی که انجام شد بعد از ۵ روز می‌توانید فعالیت بعدی را شروع کنید.

مشکل : فعالیت دومی ممکن است سرعت اش زیاد باشد چه مشکلی پیش می‌آید دو ، سه روز دیگه به کار می‌رسدو تداخل ایجاد شود.

نتیجه : پایانش را هم محدود می‌کنیم یعنی می‌گوییم ۵ روز حداقل فاصله داشته باشد.

۵ *Finish to Finish* اجازه نمی‌ده که تیمی که از پشت سر می‌آید خیلی زود به این فعالیت برسد.

این شکلی که رسم کردم چه مفهمومی دارد؟

در هفته پیش *AOA* یک شرط نوشته بودیم که بین دو تا گره چند تا بردار وجود نداشته باشد. اما در این جا ظاهراً این طوری نیست می‌تواند بین دو گره چند تا برداری وجود داشته باشد. اشکالی هم ندارد. نیازی هم نیست با مجازی درست کنیم این مجاز است. البته در بعضی از پروژه‌ها، مقاله‌ها، قرارداد ما این دو تا را پاک می‌کنیم به جاش می‌نویسیم دو تا فعالیت *i* و *j* بین اینها از اول تا آخر

نباید اینها به هم نزدیک شوند. باید فاصله ۵ روزی را با جلویی‌اش حفظ کند

این هم جای شروع را هم پایان را. این هم *ss* و *ff* است برای اینکه شلوغ نشه اینطوری هم می‌توانید استفاده کنید .که الزامی نمی‌باشد.

$SS_{ij}^{min}(۵)$ و *FF* ، *Lag* اش ۱۶ یا ۶ یا ۷ باشد دیگر نمی‌توانید استفاده کنید باید *Lag* ها با هم مساوی باشند. پس از این نظر شبکه *AON* بهتر است.

$\mathrm{SS}_{\mathrm{ij}}^{\mathrm{min}}(\mathrm{x})$ و $\mathrm{FF}_{\mathrm{ij}}^{\mathrm{min}}(\mathrm{x})$ را معمولاً $zz_{\mathrm{ij}}^{\mathrm{min}}(\mathrm{x})$ نشان می‌دهند که به این رابطه پیش نیاز ترکیبی می‌گویند.

Max هم اگر باشد همین طور $SS_{\mathrm{ij}}^{\mathrm{max}}(\mathrm{x})$ و $\mathrm{FF}_{\mathrm{ij}}^{\mathrm{max}}(\mathrm{x})$ را به جاش می‌نویسند $ZZ_{\mathrm{ij}}^{\mathrm{max}}(\mathrm{x})$

سوال : در شبکه می‌تواند حلقه وجود داشته باشد؟

جواب : در هفته گذشته در *AOA* ها گفته شد که خیر ولی در این مبحث تحت یه شرایط می‌تواند حلقه وجود داشته باشد یعنی می‌تواند *i* پیش نیاز *j* و *j* پیش نیاز *k* و *k* پیش نیاز *i* باشد. ممکن است اتفاق بیافتد منظور این است که قوانین ۶ تایی که صحبت کردیم آنها مخصوص *AOA* ها هستند ولی در این جا صحت ندارد. مثلاًبین دو تا گره چند تا بردار وجود داشته باشد اشکالی ندارد.

نکته : در شبکه کلی از این علامت‌ها بنویسید شما در *AOA* ها کار خیلی راحت است. وقتی می‌نویسید *i* پیش نیاز *j* است یک معنی بیشتر ندارد ولی در شبکه *AON* شفاف نیست که منظور چیست باید بالای آن یکی از آن ۸ حالت را بنویسید تا شفاف گردد که آن چه پیش نیازی است آن وقت هفته‌های بعد می‌خواهیم برنامه‌ریزی کنیم مشکل به وجود می‌آید.

سوال : چه مشکلی ؟

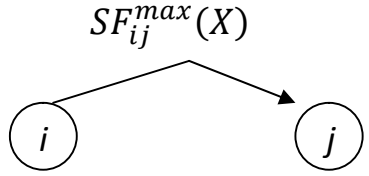
هر کدام از این بردارها معنی خاص خود را دارد یعنی ۸ معنی متفاوت دارد برای اینکه بتوانیم با اینها کنار بیام همه پیش‌نیازها را تبدیل به یک حالت استاندارد می‌کنیم که SS_{min} را معمولاً به عنوان حالت استاندارد در نظر می‌گیرند. و آن ۷ تا حالت را به SS_{min} تبدیل می‌کنند یعنی چی ؟

یعنی به صورت معادل باید به SS_{min} تبدیل کنیم این تبدیل در گام ذیل انجام می‌شود:

۱- آن چهارتای که *max* هست را تبدیل به *min* می‌کنیم که همه بشوند *min*

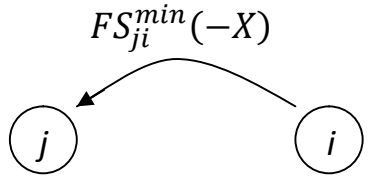
چطور مرحله پیش نیاز *max* را پیش نیاز *min* تبدیل کرد؟

در این شکل *i* پیش نیاز *j* است و پیش نیاز از نوع $\mathrm{SF}_{\mathrm{ij}}^{\max}(\mathrm{x})$ است یعنی *i* که شروع شد حداکثر *X* روز بعد *j* باید تمام شود. اگر شروع *i* را با S_i و پایان *j* را با F_j نشان دهیم همین جمله را به صورت ریاضی بنویسید.



i که شروع شد حداکثر *x* روز بعدش، *j* پایان پیدا کند.

$$\mathrm{S_i} + X \geq F_j \xrightarrow{\times(-)} -\mathrm{S_i} - X \leq -F_j \Longrightarrow -\mathrm{F_j} - X \leq -S_i$$



حالا اگر *min* بود چی ؟ برعکس می‌شد *F> S+X*

منظورم اینکه این ۸ تا جمله را می‌شود به صورت ریاضی هم نوشت.

حالا اگر *max* را بخواهید به *min* تبدیل کنید یعنی چی ؟ یعنی جهت نامساوی را برگردانید . چطوری می‌شود جهت نامساوی را برگرداند یک منفی در دو طرف آن ضرب می‌کنید. چی می‌شود آن وقت ؟!

$$-S_i - X \leq F_j$$

یک منفی اضافه کردم و از نظر مفهومی هیچ تغییری در معادله انجام نشد.

اگر *F* را بیارید این سمت و *S* را ببرید آن سمت حسنش این است که این منفی ها را از بین برده‌اید.

حالا اگر خوب دقت کنید

این چه پیش‌نیازی است ؟

اولاًاین *SF* بود این برعکس است *FS* و این از *i* به سمت *j* بود این یکی از *j* به *i* پس

الان این بالایی و پایینی با هم فرقی نمی‌کند

$\mathrm{SF}_{\mathrm{ij}}^{\max}(\mathrm{x})$ می‌شود نوشت $FS_{\mathrm{ji}}^{\min}(-\mathrm{x})$ تنها با دو تغییر:

- *X* شده قرنیه‌اش
- *ij* شده *ji*
- جهت نامساوی هم برگشته است.

$\mathrm{FS}_{\mathrm{ij}}^{\max}(۱۰)$ را چی می‌توان نوشت ؟

جای *FS* را عوض کنید $\mathrm{SF}_{\mathrm{ji}}^{\min}(-۱۰)$

آن یکی‌ها اثباتش را انجام نمی‌دهیم چون تکراری است در جدول آمده است

$$\mathrm{FF}_{\mathrm{ji}}^{\min}(-\mathrm{x}) \longleftarrow \mathrm{FF}_{\mathrm{ij}}^{\max}(\mathrm{x})$$

الان از ۸ حالت به ۴ حالت کاهش پیدا کرد.

نکته : همه *min*ها باید بشوند به صورت SS^{min} (حالت استاندارد)

i پیش نیاز *j* است و با بالای آن نوشته شده $\mathrm{FS}_{\mathrm{ij}}^{\min}(۱۱)$

فرض کنید *Duration* فعالیت *i* به عنوان مثال ۱۷ روز است و *Duration* فعالیت *j*، ۵ روز می‌باشد.

۱۷ و ۵ *Duration* است ۱۱ یعنی *Lag* از نظر مفهومی یعنی چی ؟

یعنی *i* که تمام شد حداقل ۱۱ روز بعد *j* می‌تواند شروع شود اگر بخواهیم این را به صورت گانت چارتی نشان دهیم.

در این مسأله ۱۱= *Lag* بین کدام ؟ بین پایان *i* تا شروع *j*، این فاصله را حداقل ۱۱ روز باشد. اصلاً نیاز به حفظ فرمول و اثبات عجیب و قریب نیست. اگر پایان این تا شروع آن ۱۱ روز باشد . سوال؟ شروع چقدر می‌شود.

این که ۱۷ است اگر بخواهد آن ۱۱ باشد جمعاً میشود ۲۸ روز دیگه پس حداقل ۲۸ روز باید باشد.

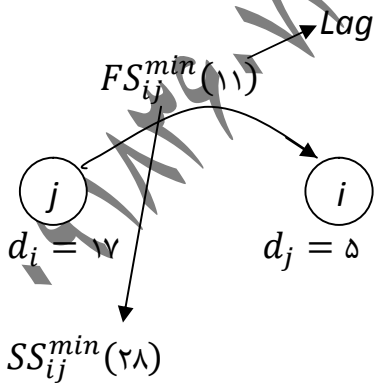
نتیجه این را خط بزنید و بنویسید $\mathrm{SS}_{\mathrm{ij}}^{\min}(۲۸)$ یعنی $\mathrm{FS}_{\mathrm{ij}}^{\min}(۱۱)$ را تبدیل $\mathrm{SS}_{\mathrm{ij}}^{\min}(۲۸)$ کردیم.

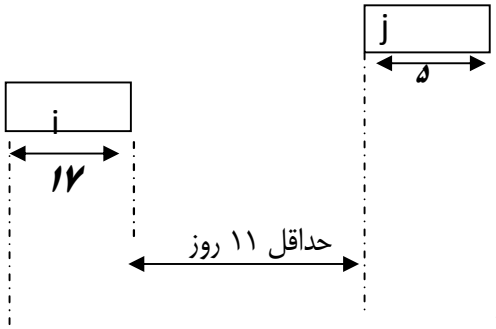
در واقع *FS* چه جوری به *SS* تبدیل شد.

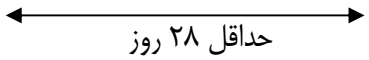
Lag + duration i = SS

پس الان فرمول استخراجی این شد که

$$\mathrm{SS}_{\mathrm{ij}} = FS_{ij} + d_i$$

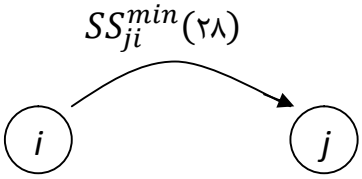






$FS \longrightarrow SS \longleftarrow Lag_{FS} + d_i$

تبدیل *FS* به *SS*



$$SS_{ij} = FS_{ij} + d_i$$

فرض کنید صورت مسأله از اول این باشد.

FF_{ij}^{min}(۱۱) را می‌خواهیم تبدیل به SS_{ij}^{min} بکنیم یعنی چیزی که شما الان دارید فاصله ***finish to finish*** اینها است به اصطلاح این فاصله حداقل ۱۱ است .

یک منطق ساده می‌توانید بگویید اگر ***finish to finish*** =۱۱ باشد پس ***Start to start*** آن چند می‌شود؟

حداقل ۲۳ . از کجا آمد ؟ این که ۵ است . این هم ۱۱ است پس ۶=۵-۱۱ و ۲۳=۶+۱۷

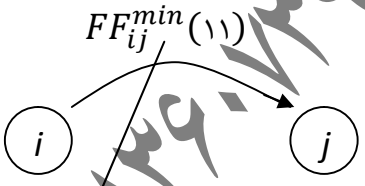
یعنی به جای FF_{ij}^{min}(۱۱) بگویید SS_{ij}^{min}(۲۳)

فرمول بسازید

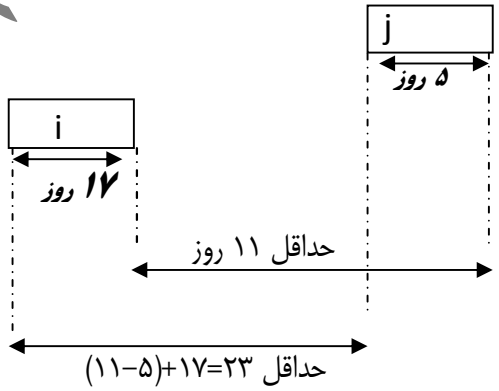
$$SS_{ij} = FF_{ij} - (duration\ j) + (duration\ i)$$

تمرین

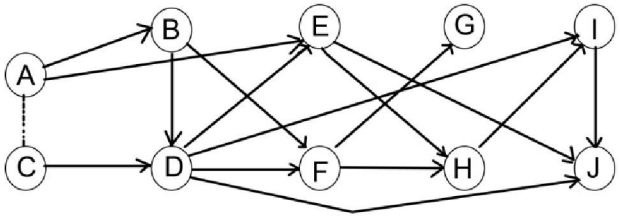
شبکه ***AON*** برای اش رسم کنید.



SS_{ij}^{min}(۲۳)



$$SS_{ij} = FF_{ij} - (d_i) + (d_i)$$



فعالیت	پیش نیاز
A	–
B	A
C	–
D	B,C
E	D,A
F	D,B
G	F
H	E,F
I	D,H
J	E,I,D

چند ثانیه بیشتر زمان نمی خواهد برای رسم . ۱۰ تا دایره بکشید و داخل آن بنویسید a تا j

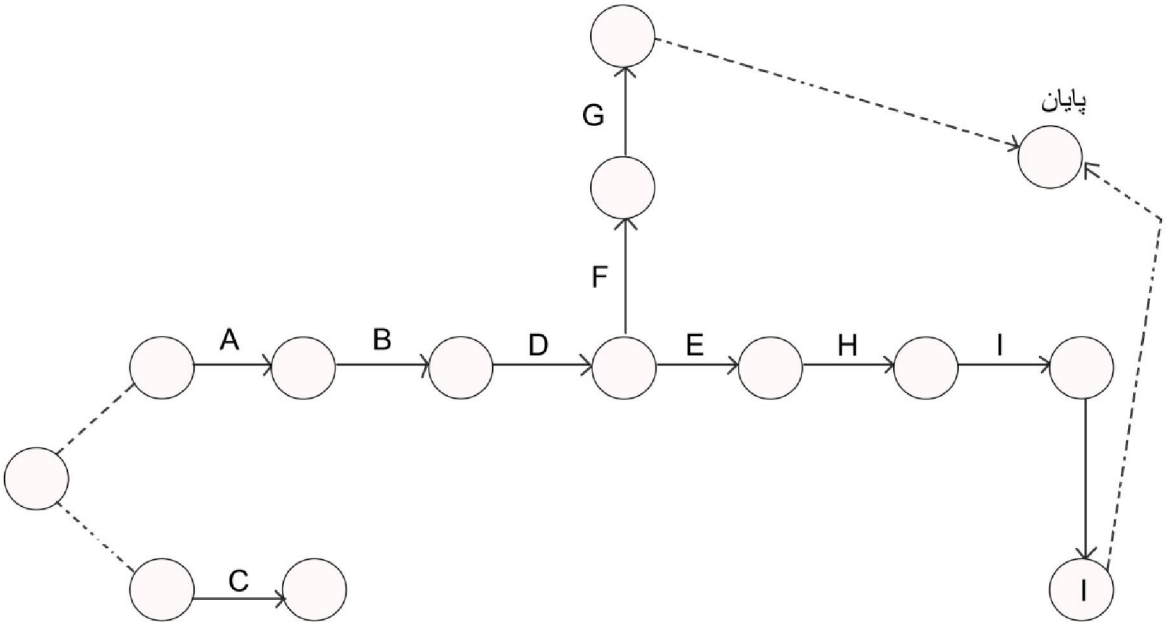
از کجا بردار وصل کنید از

a b b,c d d,a e f g

رسم *AON* خیلی ساده است به تعداد فعالیت‌ها گره ، داخل گره‌ها اسم‌هاشون را می‌نویسیم طبق پیش‌نیازی بردارها را رسم کنید.

خوب اگر درست رسم کرده باشید. جواب همه باید عین هم باشد.

سوال بعد :



همین پروژه را *AOA* رسم کنید یعنی شبکه برداری اش را رسم کنید. یعنی بردار ها فعالیت باشند و گره ها مایلستون هستند البته با رعایت این پیش‌نیازها

در غیر مستقیم‌ها که پیش نیاز هست نمی‌خواهد.

خوب نتیجه کسی هست که کامل رسم کرده باشد.

AON راحت رسم می‌شود و همه جواب‌های افراد با هم یکی است یعنی عین هم می‌باشد.

AOA راحت رسم نمی‌شود و همه جواب‌های افراد با هم فرق می‌کند چون ممکن است یک نفر با ۳ تا مجازی رسم کرده باشد و یک نفر دیگه با ۵ تا مجازی یا یک شخص با ۱۰ مایلستون و دیگری با ۱۵ تا مایلستون

نتیجه *AOA* را دستی رسم نمی‌کنیم بلکه *AON* را دستی رسم می‌کنیم و با یک الگوریتم آن را به *AOA* تبدیل می‌کنیم . با یک مکانیزم. این مکانیزم به شما تضمین می‌دهد که جوابی که بدست می‌آید کمترین دامی و کمترین مایلستون را دارد. جواب بهینه

P(v) : مجموعه *Immediate predecessor* (پیش نیازهای فوری) ، مثلاً در همین مثال *immediate predecessor G* کدام است ؟

جواب : *F* چون بلافاصله پیش نیاز است.

ولی اگر دقت کنید تنها پیش نیاز *F* نیست.

خود *F* پیش نیاز *B,D* و *B* پیش نیاز *A* و خود *D* پیش نیاز *B,C* است

نتیجه پیش‌نیاز های *G* عبارتند از *A,B,C,D,F*

در کل ۵ تا پیش نیاز داریم کل این پیش‌نیاز ها را با $P^*(V)$ نشان می‌دهیم و اونی که *immediate* هست را با *P(v)*نشان می‌دهیم

درخصوص پس نیاز ها هم همین طور می‌باشد کل پس نیاز ها را با $S^*(V)$ و اونی که بلافاصله پس نیاز است را با *S(v)* نمایش می‌دهیم.

فعالیت	پیش نیاز
A	–
B	A
C	–
D	B,C
E	D,A
F	D,B
G	F
H	E,F
I	
J	

وقتی یک شبکه *AON* را می‌خواهیم تبدیل به *AOA* کنیم یعنی یک فعالیتی را که قبلاً با گره نشان می‌دهیم الان به جای یک گره یک بردار باید باشد. این بردار یک مایلستون شروع و یک مایلستون پایان دارد. حالا به شرطی این تبدیل درست است که همه پیش‌نیاز و پس نیازهای در شبکه *AON* دقیقاً بدون اضافه و کم در شبکه *AOA* هم بیاید و اگر کم و زیاد بکنیم داریم مسأله را اشتباه حل می‌کنیم.

برای اینکه پیش‌نیازها معادل باشد باید این ۴ تا گام رعایت شود.

تمام پیش‌نیازهای *V* باید برابر با پیش‌نیازهای S_v باشد. یعنی تمام فعالیت‌هایی که در *AON* پیش نیاز *V* بودند در *AOA* که می‌خواهیم بسازیم باید پیش‌نیازهای مایلستون شروع *V* باشد و همین طور پایینی

تمام فعالیت‌هایی که پس نیاز *V* بودند حالا باید پس نیازهای t_v باشد .

ببینید S_v و t_v مایلستون هستند ما آن جا مایلستون نداشیم آنجا فقط فعالیت داشتیم . پیش‌نیازهای یک فعالیت باید پیش‌نیازهای شروع باشند. پس نیازهای یک فعالیت باید پس‌نیازهای پایان باشد.

دقت کنید پیش نیازهای S_v را گفتید پس پس‌نیازهای چی ؟

پس نیازها S_v باید از این فرمول حساب شود . خواندن این فرمول کمی مشکل است.

اول باید چی کار کنیم ؟

P_v چی بود ؟ پیش‌نیاز فوری *V* را پیدا کنید به اسم u

برای این پیش‌نیازها کل پس‌نیازها را پیدا کنید بین آنها اشتراک بگیرید.

با مثال بیشتر توضیح می‌دهم

پیش‌نیازهای t_v برعکس بالا اول پس نیازهای *V* را پیدا کنید و بین پیش‌نیازها اشتراک بگیرید.

خلاصه این ۴ تا فرمول چی کار می‌کنند.

پیش نیاز S_v

پس نیاز S_v

پیش نیاز t_v

پس نیاز t_v

الگوریتم ، این ۴ تا فرمول است. این الگوریتم زیاد طولانی نیست.

این مثال *AON* است که می‌خواهد به *AOA* تبدیل شود.

۸ تا فعالیت دارد . شروع و پایان دارد مجازی است. شروع و پایان نمی‌تواند ۴ تا باشد. برای محاسبه شبکه قبلاً انجام دادم که از وقت کلاس گرفته نشود.

ستون اول اسم فعالیت‌ها است

۴ تا ستون بعدی طبق این ۴ تا فرمول حساب شده است.

فعالیت *E* چطور این ردیفش پر شده است.

$$P^*(S_v) = P^*(v)$$

فعالیت E ⟵ پیش‌نیاز A,B,C

$P^*(S_v)$ ⟵ ستون آخر A, B, C

پس نیازهای t_v پس نیازهای *v*

فعالیت *E* پس نیازش کدام است. ؟ ندارد تهی است. Φ

ستون یک که فعالیت‌ها بودند هیچی

ستون دو و پنج را طبقه این فرمول گفتم
اما ستون‌های $S^*(S_v)$ و $P^*(t_v)$ که طبق این فرمول‌ها هستند که یکم بیشتر توضیح داده ، این توضیح را باید بنویسم.

پیش‌نیاز $P(E) = \{A, B, C\}$

برای تک تک آنها پس نیاز

$$\begin{cases} S^*(A) = \{E\} \\ S^*(B) = \{E, F, G\} \\ S^*(C) = \{E, G\} \end{cases}$$

فعالیت آخر

پس‌نیاز فوری E $= \{\ \}$ تهی

پس نیاز آن $= \{A, B, C, D, E, F, G, H\}$

فعالیت E طبق این رابطه $S^*(S_v)$ را حساب کنیم .

اول پیش‌نیازهای فوری E کدامند؟ $P(E) \longrightarrow A,B,C$

چرا E چون راجع به E صحبت می‌کنیم.

حالا برای تک تک این پیش‌نیازها، پس نیازهای آن را بنویسید.

پس نیاز A $\longleftarrow$ E $\longleftarrow$ $S^*(\mathrm{A})$

پس نیاز B $\longleftarrow$ E,F,G $\longleftarrow$ $S^*(B)$

پس نیاز C $\longleftarrow$ E,G $\longleftarrow$ $S^*(\mathrm{C})$

اشتراکشان کدام است ؟ فقط E می‌باشد.

ردیف E ، $P^*(t_v)$ طبق این فرمول باید

اول پس نیاز فوری E کدامند ؟ تهی است یعنی همان مجازی است.

برای همان فعالیت مجازی که اسمش W است پیش‌نیازهایش کدامند ؟ همه فعالیت‌ها

$\{A, B, C, D, E, F, G, H\}$

چرا اشتراک نگرفتیم ؟ چون یک مجموعه بیشتر نبود

این گام اول بود.

اصلش همین بود گام‌های بعدی آسانتر است.

گام دوم :

چند تا زوج مرتب مجزا *(Distinct page)* یعنی زوج مرتب‌هایی را بگویید که تکراری نباشد *(distinct)* یعنی مجزا . از روی جدول از روی ستون‌های ۲ و ۳ یعنی $P^*(S_v)$, $S^*(S_v)$, $P^*(t_v)$, $S^*(t_v)$ چند تا زوج مرتب غیر تکراری دارد.

من یکی را می‌گم بقیه‌اش را شما بنویسید. اولیش این است.

$P^*(S_v)$ تهی می‌باشد.

$S^*(t_v)$= A,B,C,D,F,G,H

(۱) ← $A, B, C, D, E, F, G, H, \emptyset$

بعدی دیگه نشمارید. نشمارید . نشمارید تا اینجا یکی بود. تکراری ها را نمی‌شماریم

بعدی A,B,C,E و E ← (۲)

بعدی E,F,G,B ← (۳)

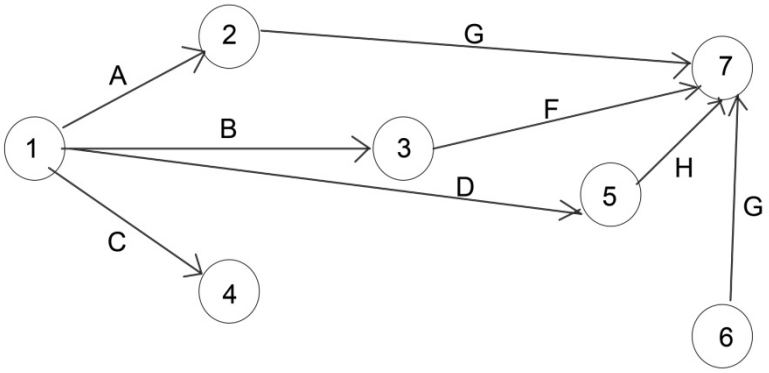
بعدی G,E,C,B ← (۴)

بعدی G,H,D ← (۵)

بعدی G,D,C,B ← (۶)

بعدی $\emptyset, H, G, F, E, D, C, B, A$ ← (۷)

پس شد ۷ تا ، ۷ تا مایلستون آماده کنید.



Min milston= 7

این هفت تا مایلستون تعداد گره‌های شما هستند. طبق این روش مینیم تعداد گره‌هایی که می‌شد این شبکه را رسم کرد ۷ تا مایلستون بود. اما اگر دستی می‌خواستید رسم کنید ممکن بود از، ۹ تا و یا ۱۲ مایلستون استفاده کنید.

حالا که شما مایلستون را ۷ تا شمردید بردارها را چطور رسم می‌کنید؟

شروع و پایان هر فعالیت را بگویید کدام است . هر فعالیت از کدام مایلستون باید شروع بشوند و به کدام مایلستون باید ختم شود.

شروع یک فعالیت این طور تعیین می‌شود مثلاً در فعالیت A، $P^*(S_v)$ اش با مولفه اول کدام گره برابر است $P^*(S_v)$ تهی است.

مولفه اول کدام گره تهی است ؟ گره یک هم تهی است. ← شروع با یک است

اما پایانش به کدام مربوط است.
$S^*(t_v)$ با مولفه دوم کدام برابر است آن پایان می‌شود.
$S^*(t_v)$ یعنی E
کدام گره مولفه دومش E است. ⟵ گره ۲
نتیجه : فعالیت A باید از یک به دو وصل شود.
فعالیت B از کجا به کجا ختم می‌شود. از ۱ به ۳
فعالیت C ⟵ از ۱ به ۴
فعالیت D ⟵ از ۱ به ۵
فعالیت E ⟵ از ۲ به ۷ (مولفه تهی)
فعالیت F ⟵ از ۳ به ۷

گام آخر : مجازی‌ها
اول : مجازی‌ها لازم است یا نه ؟ تشخیص آن که مجازی لازم است یا نه باید شبکه AON را اول مسأله داده بود رو بیارید جلوی چشمتان و با این مقایسه کنید آیا همه اونهایی که آنجا پیش‌نیاز بودند در اینجا هم پیش‌نیاز هستند اگر هست تمام ، نمی‌خواهد ولی اگر نبود با مجازی درست کنید
A پیش نیاز E است
B پیش نیاز E است ولی در اینجا نیست پس از ۳ به ۲ یه مجازی رسم کنید.
B پیش نیاز F است ⟵درست است
B پیش نیاز G است ولی در اینجا نیست پس از ۳ به ۶ یه مجازی رسم کنید.
C پیش نیاز E است ولی در اینجا نیست پس از ۴ به ۲ یه مجازی رسم کنید.
C پیش نیاز G است ولی در اینجا نیست پس از ۴ به ۶ یه مجازی رسم کنید.
D پیش نیاز G است ولی در اینجا نیست پس از ۵ به ۶ یه مجازی رسم کنید.
D پیش نیاز H است ⟵ درست است
نتیجه : این مجازی‌ها را که رسم کنید مسأله به طور کل تمام شده است. (۵ تا مجازی رسم کردیم)
در واقع شبکه AON را به AOA تبدیل کردیم.
نکته : فرض کنید این ۳ به ۲ نبود و ۳ به ۴ در اینجا اون وقت لازم نبود همه این ۵ تا را رسم کنید.
چرا ؟
اگر ۳ پیش نیاز ۴ باشد و ۴ پیش نیاز ۶ باشد عملاً ۳ را پیش نیاز ۶ هم کردید. دیگه ۳ به ۶ جدا لازم نیست ۲ بار دیگه لازم نیست ۳ به ۶ پیش نیاز شود چون به واسطه ۴ پیش نیاز شده است به طور غیرمستقیم
قبل از رسم باید چک شود قبل از اینکه ۳ به ۲ مجازی رسم کنیم، آیا گرهی مثل K وجود دارد به عنوان واسطه بتواند پیش‌نیازی ۳ به ۲ را ایجاد کند، یعنی از ۳ پیش نیاز K شود و K پیش نیاز ۲ شود ، اگر داشته باشد ادامه نمی‌دهیم و ۲ بار رسم نمی‌کنیم. مثلاً از ۳ به ۶ چی ، ۴ را ساختیم ،۴ نقش K را بازی می‌کنیم، $(۴,۳)$ و $(۶,۴)$را داریم.
در این مثال این اتفاق نیافته و مجبور شدیم که این مجازی‌ها رسم کنید.

با کمتری مجازی $AOA \longrightarrow AON$

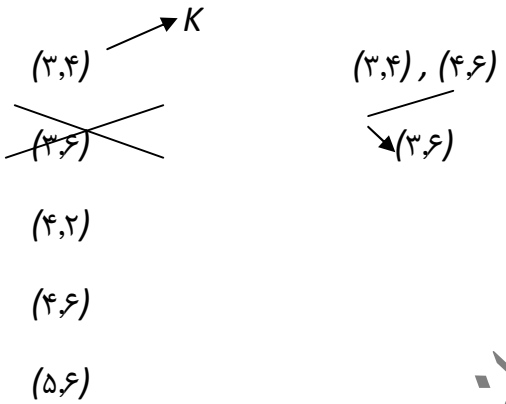
مجازی (۳,۲)

مجازی (۳,۶)

مجازی (۴,۲)

مجازی (۴,۶)

مجازی (۵,۶)



دیگه رسم نمی‌کنیم گره‌ای نقش میانی را بازی می‌کند نداریم

جلسه پنجم
طبقه بندی از مسائل زمانبندی پروژه

در ادامه سیلابس هفته گذشته بحث این جلسه در رابطه با طبقه‌بندی از مسائل زمانبندی پروژه می‌باشد. که هفته گذشته هم عرض کردم که موضوع مستقل‌ایی می‌باشد و عاری از هر گونه فرمول و الگوریتم است بیشتر در واقع توضیحی می‌باشد.

انگیزه طبقه‌بندی رشد مسائلی که در واقع (پژوهش‌هایی که) توی این فیلد زمانبندی پروژه در ۱۰ تا ۱۲ سال اخیر اتفاق افتاده است مراحل مختلفی، افراد مختلف طرح‌هایی پیشنهاد دادند که این مسائل را تقسیم‌بندی بکنند که کسی که می‌خواهد در این حوزه کار کند سر در گم نباشد بداند بین مسائل چه نظمی وجود دارد و از این نظم برای کار کردن استفاده کند.

اولین بار در کنفرانس سال ۹۷ در واقع ***start*** خورد، ولی در واقع چندین مرحله طبقه‌بندی که آقای هرودن سال ۹۸ پیشنهاد کرد به عنوان طبقه‌بندی مبنا است ولی در واقع به طور مکرر این طبقه‌بندی توسط تیم هرودن به روز می‌شود.

طبقه‌بندی که توی این کلاس تشکیل می‌شود چند تا ویژگی دارد یعنی تا حد امکان این تقسیم‌بندی که آقای هرودن به اصطلاح از مسائل ساختند سعی کرده که به اصطلاح خودش هم ***Flexible*** و هم ***Workable*** باشد. یعنی هم انعطاف‌پذیر باشد و هم قابل استفاده باشد. مثلاً مسائلی عنوان‌هایی که بسیار طولانی دارند را از نظر اختصاری به صورت نماد تعریف کرده است.

مسائل ***OR*** ⟵ ***RCPSP*** ⟵ زمانبندی پروژه با محدودیت منابع می‌گویند که خلاصه واژه ***(Resource Constrained project Scheduling Problem.)***

MRCPSP ⟵ ***M*** به خاطر ***multi mode*** بودن است.

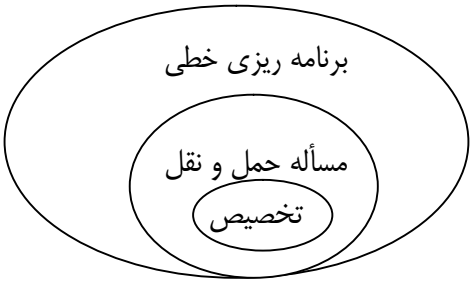
PRCSP ⟵ اگر ***P*** اول ***RCSP*** باشد می‌شود ***Preemtive***

اگر این ***RCPSP*** با پیش‌نیازی ساده همراه باشد همان است ولی اگر با پیش‌نیازی تعمیم یافته باشد بعد از ***RCPSP*** ، واژه ***GPR*** می‌آورند . این یعنی مسائل زمانبندی پروژه با محدودیت منابع و پیش‌نیازی‌ها از نوع تعمیم یافته است.

ویژگی دوم :

بعضی از مسائل ***Sub problem*** یک مسئله دیگه است یعنی زیر مسئله یک مسئله دیگر است.

مثلاً برنامه‌ریزی خطی یک مسئله، حمل و نقل هم یک مسئله ، تخصیص هم یک مسئله است. بین این ها چه رابطه‌ای وجود دارد؟



پاسخ : تخصیص یک جوری مسئله حمل و نقل بود و حمل و نقل یک جور از برنامه‌ریزی خطی کلی بود ، بعد مسئله حمل و نقل یک حالت خاصی از برنامه‌ریزی خطی بود و تخصیص هم حالت خاصی از حمل و نقل بود.

حالا در این زمانبندی پروژه هم طبقه‌بندی که اینجا هم هست سعی شده این زیر مسائل نسبت به هم معلوم باشد که کدام *Sub Problem* مسئله دیگر است .

سوال ؟ دانستن این *Sub Problem* ها چه فرقی دارد یا چه کمکی می‌کند ؟

فرض کنید شما تشخیص دادید که *A* ، *Sub problem* ، *B* می‌باشد و *B* هم *Sub Problem*، *C* است چه کمکی می‌کند؟

اگر شما این را حل کنید عملاً این دو تا را حل کردید یعنی شما یک مسئله کلی تر را حل می‌کنید یعنی *Sub Problem* های اون را نیز حل کرده‌اید.

ثانیاً اگر برعکس باشد چی ؟

یک مقاله‌ای پیدا کرده‌اید که این را حل کرده حال با دانستن این سلسله مراتب چه کمکی می‌کند ؟

در واقع می‌توانید ایده بگیرید و کمک بگیرید برای حل این بیرونی و از این کمک بگیرید برای حل بیرونی تر

حال اگر شما حل مسئله بیرونی تر را داشته باشید داخلی‌ها اصلاً حل شده است پس قرار نیست شما کاری انجام بدهید پس این که هیچی

اما اگر مسئله داخلی حل شده باشد می‌توانی ازش ایده بگیرید برای حل مسئله بیرونی

معمولاً در تعریف موضوع، مقاله، پایان نامه ، شما یک مقاله را می‌خوانید بعد توسعه (از بیرون به داخل بروید نیست) بلکه از داخل به بیرون می‌روید یعنی مسئله را تعمیم می‌دهید و وسعت می‌دهید نه اینکه کوچک‌تر کنید اگر کوچک‌تر شود که این دیگر مقاله نیست.

سوال : این مقاله در حالت *Multi Mode* حل کرده حالا من می‌خواهم همین را در حالت *Single Mode* حل کنیم ؟

پاسخ : وقتی *Multi Mode* حل شده ، *Single Mode*هم هست دیگر *Single Mode* حالت خاصی از *Multi Mode* هست.

اگر این *Single Mode*باشد و بخواهیم به *Multi Mode* تعمیم بدهیم می‌توانم به عنوان موضوع تعریف کنم چون در واقع این که الان حل شده این مسئله داخلی است من دارم یک *Level* بالاتر مسئله را دارم تعریف می‌کنم. بنابراین این می‌تواند موضوع باشد از این چه کمکی می‌گیرم ؟ ایده می‌گیرم دیگه ! نه اینکه عیناً همین رو استفاده کنم این نیست ولی میشه از نکاتی از قضایی که در آن اثبات شده به اصطلاح ایده گرفت و تعمیم داد برای مسئله بیرونی

ویژگی سوم :

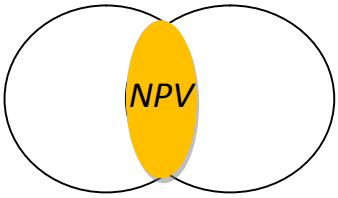
بحث سختی مسائل است هر مسئله‌ای $T(N)$ داشت . مسائل به دو دسته *P* و *N-P* تقسیم می‌شدند . *P* ها کدام بودند ؟ *N-P* ها کدام بودند ؟

آنهایی که براش الگوریتم وجود دارد که T_n اش بحث سختی مسائل

اصطلاحی گفتم *A* ، *Reduce* می‌شود به *B* یعنی چی ؟ یعنی کدام سخت تر است ؟ *B* از *A* سخت تر است.

در این ویژگی سختی‌ها را با هم مقایسه کرده است یعنی به طور مثال کتاب خانه‌ای از مسائل درست کرده ولی گفته که مثلاً هر چه قدر قفسه‌ها را بالاتر بروید مسأله سخت‌تر می‌شود و یا هر چه قدر به سمت راست بروید سخت تر می‌شود این مسأله‌ها را به این صورت تقسیم‌بندی کرده که این خودش یک حسن دارد. یعنی وقتی شما یک مسأله را دارید با این مسأله مقایسه کنید و ببینید توی راهنما که این از او سخت تر است یا راحت‌تر است یا در یک سطح است (سختی‌اش برابر است)

ویژگی چهارم :

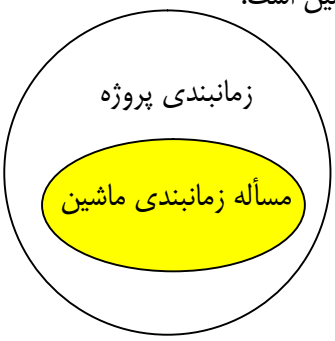


بحث انطباق روش‌های حل آنها است. که با بالایی فرق می‌کند بالایی ارتباط سختی مسائل است و این ارتباط روش‌های حل آنها است . منظور این است مثلاً این یک مسئله است و اینم یک مسئله دیگر تنها اشتراکی به با هم دارند این است که هر دو تاشون بحث NPV مطرح هستند. دو تا مسئله که همه چی شون با هم متفاوت است تنها اشتراکشون اینکه هر دو تا تابع عددشون ماکسیمم کردن NPV پروژه است نتیجه ابعاد مسئله قابل استفاده نیست چون دو تا با هم فرق می کند از یک جنس نیستند ولی از اشتراکشون می‌شود بهره‌برداری کرد یعنی از لینکی که اشتراکشون که بین مسائل وجود دارد بهره برداری کنید. شما یک مسئله را برای موضوع انتخاب کردید که کار کنید، یا بخونید یا سمینار و یا هر بحثی . خوب مثلاً می‌گم یک قسمتی داره را می‌گم که بحث *NPV* را مطرح کرده است شما اصلاً نمی‌دانید چی کار کرده حداقل می‌توانید از مقاله‌های *NPV* هست قسمت *NPV* اش را آنجا بخوانید به اصطلاح آن تیکه‌ای که اونجا *NPV* داشته را آنجا چطوری حل کرده است که بتوانید در مسئله خود استفاده کنید.

این با اون بحث دوم فرق می‌کند : آنجا بحث *Sub Problem* بود این مسئله زیر مجموعه آن بود. یا آن مسئله تعمیم یافته است ولی اینجا دو تا مسئله جدا هستند که فقط اشتراکاتی با هم دارند که می‌شود از اشتراکاتش بهره‌برداری کرد.

این ۴ تا ویژگی در این طبقه‌بندی هست وجود دارد.

به این مسائل که قرار است الان صبحت کنیم مسائل زمانبندی پروژه است ولی قبل از مسائل زمانبندی پروژه مطرح شود مسائل زمانبندی ماشین مطرح بوده است یعنی آنها بیشتر رایج بودند. مثلاً فرض کنید در دوره صنعتی شدن بیشتر بحث‌ها فقط ماشین بوده چون تولید انبوه مطرح بوده همه کارها عملیاتی بودند تا پروژه‌ای . بنابراین زمانبندی ماشین قدمتشان خیلی بیشتر از مسائل زمانبندی پروژه هست. این طبقه‌بندی که الان می‌خواهیم صحبت کنیم برای کی گفتم برای ۹۸. ولی بسیار قبل‌تر از طبقه بندی دیگری وجود داشته برای زمانبندی ماشین اما موضوع اینکه که زمانبندی ماشین و پروژه آیا دو بحث جدا از هم هستند ؟ پاسخ نه . بسیار مباحث به هم نزدیک هستند و اگر این مسائل زمانبندی ماشین باشد و این مسائل زمانبندی پروژه باشد چه ارتباطی بین شان است ؟ یعنی مسائل زمانبندی پروژه یه جورایی در تعمیم مسائل زمانبندی ماشین هستند یا مسائل زمانبندی ماشین یه جورایی *Sub Problem* های مسائل زمانبندی پروژه هستند بنابراین برای زمانبندی ماشین قبلاً یک طبقه‌بندی وجود داشته حالا آقای هرودن همان طرح را وسعت داده است. که بتواند مسائل پروژه را هم پوشش دهد در واقع این طرح از صفر نوشته نشده است بلکه در تکمیل یک طرح نوشته که قبلاً برای زمانبندی ماشین است.



زمانبندی ماشین:

در طرح زمانبندی ماشین هر مسئله با سه تا فیلد α, β, γ شناخته می‌شود یعنی برای هر مسئله سه تا مشخصه تعریف کرده است. این سه تا مشخصه بیانگر عبارت زیر است :

فیلد α : بیانگر محیط ماشین است .

یعنی در درس برنامه‌ریزی تولید محیط ماشین عبارتند از :

job shops, general shops, mixed shops ، flow shops, Parallel Machine که می‌تواند در محیط ماشین اتفاق بیافتد. (خط تولید، کارگاهی، تکنولوژی گروهی)

فیلد β : بیانگر ویژگی کارها و منابع را نشان می‌دهد و منابع غیر ماشین را چون ماشین در فیلد α آمده است.

ویژگی کارها یعنی :

- امکان بریدگی کارها وجود دارد یا نه
- محدودیت تقدم و تأخر وجود دارد یا نه
- زمان آماده سازی برای کارها داریم یا نه
- Dead line برای کارها وجود دارد یا نه
- زمان Process کارها قطعی یا احتمالی یا فازی است

و بحث‌های دیگری که می‌تواند اتفاق بیافتد.

نتیجه : α محیط ماشین است . β در این فیلد دو بحث ویژگی کارها و منابع وجود دارد.

در خود فیلد β **سه عبارت وجود دارد.** ρ, δ, λ : λ : بیانگر چند نوع منابع اضافی (منظور غیر ماشین یعنی به غیر ماشین چند منبع دیگر دارید) مثلاً نیروی انسانی یک نوع می‌شود. از این نوع منبع چند تا دارید

(resource limit) ρ برای انجام این کار چقدر به این منبع نیاز دارید که می‌شود ρ

مثلاً اگر به این سه عبارت اعداد ۱ ، ۷ ، ۳ داده شود یعنی چی ؟

یعنی یک نوع منبع داریم غیر از ماشین مثلاً کارگر و از این یک نوع منبع ۷ تا داریم (مثلاً ۷ کارگر داریم) و به ۳ تا کارگر برای انجام این فعالیت نیاز داریم.

حال اگر دو منبع باشد :

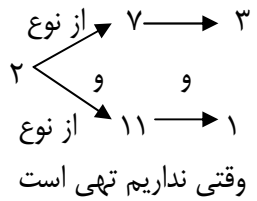
α : بیانگر محیط ماشین

β : بیانگر ویژگی‌های کارها و منابع غیر از ماشین

γ ← بیانگر نوع منابع اضافی (غیر از ماشین)	۱ یک نوع منبع
σ ← چند تا	۷ از این یک نوع ۷ تا داریم.
ρ ← چقدر نیاز داریم	۳ برای انجام این کار ۳ تا نیاز داریم.

۲ منبع داریم از نوع یک ۷ تا داریم که ۳ تا لازم داریم

از نوع دو ۱۱ تا داریم که ۱ لازم داریم.



یعنی هر چند تا λ (لاندا) داریم به تعداد آن ρ, δ (دلتا و رو) باید نوشته شود.

اگر منبع غیر از ماشین نداشتیم صفر نمی‌نویسیم تهی می‌گذاریم پس ρ, δ را هم نداریم.

فیلد γ : Optimal criteria یا همان معیار هدف (تابع هدف) مسأله شما را بیان می‌کند.

تابع هدف چی می‌تواند باشد ؟ برنامه‌ریزی تولید

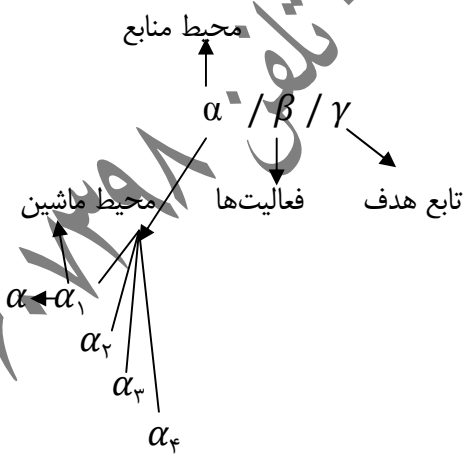
- مینیمم کردن فلوتایم ،waiting time، Flow time

هرودن در طرح پیشنهادی خود ، برای زمانبندی پروژه همان سه تا فیلد α, β, γ را بیان کرده است با کمی تفاوت :

فیلد α : در زمانبندی ماشین محیط ماشین بود ولی در زمانبندی پروژه محیط منابع می‌باشد (ماشین و غیر ماشین)

فیلد β : در زمانبندی ماشین ویژگی کارها و منابع بود ولی در زمانبندی پروژه فقط فعالیت‌ها هستند.

فیلد γ : همان معیار هدف (تابع هدف) مسأله



خود فیلد α که منابع است را به چهار قسمت تقسیم کرده است. $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ که α_1 ماشین می‌شود. نقش محیط ماشین را بازی می‌کند. در زمانبندی ماشین α محیط ماشین بود ولی در زمانبندی پروژه α_1 محیط ماشین می‌باشد.

سوال : منظور از ماشین چیست ؟ ماشین یعنی ماشین‌کاری

سوال : چه چیزهایی به عنوان ماشین‌کاری به حساب می‌آید ؟ کدام ماشین‌ها به عنوان α_1 باید ثبت شوند کدام ها به صورت سایر منابع ؟ مثلاً لودر ماشین است یا نه ؟

دستگاه پرس فقط ماشین است. اگر دستگاه ثابت است این میشه ماشین. اما اگر نه ماشین را می‌توانید بالای سر کار ببرید آن یک منبع دیگر می‌شود. پس ***Structural resource*** می‌شود منابعی که ثابت هستند. در کارگاهی (که می‌تواند روباز هم باشد) دستگاه فیکس شده ، ثابت شده است می‌شود ***Structural resource***

خود α_1 که محیط ماشین است می‌تواند چند حالت داشته باشد ؟

۱۴ حالت

حال به توضیح این ۱۴ حالت می‌پردازیم :

۱- $\alpha_1 = \emptyset$ یعنی اصلاً موضوعی به اسم ماشین ***(Structural resource)***نداریم . در پروژه زیاد از مسئله نداریم مثلاً پروژه شما نوشتن پایان نامه است ***(Structural resource)***نداریم α_1 برای پروژه‌هایی است که قطعه‌سازی صورت می‌گیرد.

۲- $\alpha_1 = P1$ یعنی ***Process*** کارها باید انجام شود روی یک ***Single Machine*** یا ***Delectated Machine*** . یعنی هر کار نیازمند فقط یک عملیات است که باید روی یک تک ماشین انجام شود تصور کنید یک کارگاهی فقط یک دستگاه پرس دارد. یعنی هر قطعه‌ای که وارد کارگاه می شود فقط برای پرس آمده است چون فقط یک دستگاه پرس وجود دارد و همچنین محیط را $P1$ می‌گویند. آن یک به خاطر این است که تنها یک ماشین وجود دارد.

خوب الان در این مسئله چه چیزهایی مجهول هستند ؟

کدام کار بهتر است اول انجام شود . در واقع توالی کارها مجهول می‌باشد.

$\alpha_1 = P1$ ساده ترین مسئله ماشین‌کاری است.

۳- $\alpha_1 = P$ یعنی ***Process*** کارها باید انجام شود روی تعدادی ***Parallel Identical Machine*** (ماشین‌های یکسان موازی)

منظور از واژه موازی در تعریف بالا : یعنی به معنای فیزیکی کنار هم هستند نمی‌باشد و یا همزمان با هم کار می‌کند نیست بلکه همه یک کار انجام می‌دهند . مثلاً ده تا ماشین وجود دارد که همه آنها ماشین تراش هستند. یک کار انجام می‌دهد. حال ممکن است این ماشین‌ها از نظر سرعت ، جنس، مدل با هم فرق داشته باشند که مهم نیست.

منظور از واژه ***Identical twin*** : یعنی کپی هم هستند. کیفیت کار، سرعت ، دقت ، مدل .... از هر نظر عین هم هستند.

کار ***i*** نیاز دارد به یک ***Single Operation*** (کار خاص) و باید ***Process*** شود روی هر کدام از این ماشین‌ها.

سوال : کار وقتی در کارگاه می‌آید کدام ماشین انجام بدهد بهتره است ؟

فرقی نمی‌کند چون همه آنها یک کار با یک سرعت و با یک کیفیت انجام می‌دهند بنابراین مهم نیست که کدام ماشین باشد.

سوال : چندتا مجهول داریم ؟

دو تا مجهول می‌باشد. ۱) توالی کارها که همیشه مجهول است. ۲) تعداد ماشین‌ها نیز مجهول می‌باشد.

سوال : تعداد ماشین‌ها زیاد باشد خوب است یا کم ؟

هیچ کدام . باید بهینه باشد. خوب اگر زیاد باشد که کلی نیاز به سرمایه‌گذاری است و اگر کم باشد که کلی از کارها طولانی می‌گردد.

آنهایی که یک m کنارش دارد یعنی Pm . که تنها با بالایی یک فرق دارد. منظور این است که m تا *Parallel Identical Machine* دارد. در واقع تعداد ماشین‌ها را به شما داده است.

پس $\alpha_1 = P_4$ یعنی یک کارگاهی داریم که ۴ تا *Parallel identical machine* دارد. پس چه چیزی مجهول می‌باشد ؟ توالی کارها مجهول است.

سوال : بین P و Pm کدام سخت تر است ؟ کدام جواب بهتری می‌دهد ؟

P چون تعداد ماشین‌ها را شما باید تعیین کنید که سخت‌تر می‌باشد. P از Pm سخت تر است.

P چون تعداد ماشین‌هارا درPm به شما تحمیل کرده است ولی در P تعداد بهینه ماشین را خودتان تعیین می‌کنید. در واقع تعداد ماشین‌ها به صورت *Optimal* می‌باشد.

$P1$ چه ربطی به اینها داشت ؟ $P1$، *Sub problem* ، P_m است یعنی اگر در مقاله‌ای خواندید که Pm را حل کرده است یعنی P_1 را هم حل کرده است. اما اگر P را حل کرده باشد یعنی چی ؟ یعنی $P1$ و Pm را حل کرده است.یعنی دقیقاً $P1$، *Sub problem* ، Pm است و Pm، *Sub problem* ، P است و به همین ترتیب مسئله سخت تر می‌شود.

۴- $\alpha_1 = Q$ وجود دارد تعداد متغییری *uniform machine parallel*

منظور از uniform: یعنی سرعت هاش با هم فرق می‌کند یعنی کارهاشان یکی است ولی با سرعت‌های متفاوت.

در قبلی identical بود یعنی از همه لحاظ سرعت‌ها یکی بود.

سوال : چند تا مجهول داریم؟

- توالی کارها
- تعداد ماشین‌ها
- تخصیص کار به ماشین

سرعت نمی‌تواند مجهول باشد چون سرعت‌ها با وجود یکسان نبودن معلوم است.

چرا در قبلی صحبت از تخصیص به میان نیامد ؟ چون در قبلی تخصیص نمی‌خواست همه ماشین‌ها یکی بود و هر کدام که خالی بود آن کار را انجام می داد ولی الان نه این سوال که کار را به کدام ماشین تخصیص بدهیم مطرح می‌باشد چون سرعت ماشین‌ها با هم فرق می‌کند با اینکه همه ماشین‌ها همان کار را انجام می‌دهند. در واقع Process timing آنها با هم فرق دارد.

آنهایی که m دارند را تقریباً نخواهم گفت.

$\alpha_1 = Qm$: Qm یعنی چی ؟ با بالایی فرق آن در این است که ماشین‌ها معلوم است پس شما تنها دو کار باید انجام بدهید. توالی کار و تخصیص کار را باید تعیین کنید و لازم نیست تعداد ماشین‌ها را تعیین کنید چون با هر سرعتی تعداد ماشین‌ها معلوم است چند تا است. مثلاً n تا ماشین

۵- $\alpha_1 = R$ ماشین‌های Unrelated parallel Machine

سرعت کار وابسته به خود کاری است که انجام می‌دهد این با قبلی چه فرقی می‌کند ؟ سرعت‌ها متفاوت است ولی سرعت‌هایش وابسته به کاری است انجام می‌شود.

به طور مثال : فردی وارد یک دفتر خدمات تایپی می‌شود می‌خواهد ۵۰ صفحه را تایپ کند. ۳ تا تایپیست وجود دارد . تایپیست اول چون مهارت زیاد دارد می‌تواند ۵۰ صفحه را عرض ۳ ساعت یعنی هر ساعت ۱۷ صفحه تایپ کند و تایپست دوم که مهارت کمتری دارد می‌تواند در هر ساعت ۱۰ صفحه تایپ کند. تایپیست سوم که تازه وارد است هر ساعت ۶ صفحه تایپ کند. تا اینجا سرعت ها با هم فرق دارد و می‌شود مسئله قبل *uniform parallel machine*. اگر بخواهیم *unrelated* آن را بگویم .

سرعت تایپست ها بستگی به فرمت تایپ‌کردن دارد . مثلاً تایپست‌ها چی تایپ کنند فارسی تایپ کنند، لاتین، فرمول ، شکل و یا نمودار . تایپست اول اگر بخواهد فارسی تایپ کند ۱۷ صفحه را در عرض یک ساعت تایپ می‌کند ولی اگر بخواهد لاتین تایپ کند سرعتش کمتر است ۱۶ صفحه یا ۱۲ صفحه می‌شود. در واقع سرعتش بستگی به کارش دارد و ثابت نیست. پس سرعت بستگی به کاری است که قرار انجام بشود.

مثال بعدی : کارگری، یا استادکاری یا یه بنایی کارش اینکه که دیوارچینی انجام دهد . اگر از او بپرسند که در طی یک روز چند متر مربع می‌تواند دیوار بچیند ؟ پاسخ می‌دهد چند سانتی ؟ با آجر، با بلوک، یا با سفال ؟ فرق می‌کند. اگر بلوک کار کند سرعت خیلی بیشتر است اگر با سفال باشد یکم کمتر می‌شود و در نهایت با آجر خیلی کمتر می‌شود چون آجر کوچکتر است سرعت کار کمتر می‌شود.

در R ماشین‌ها مجهول هستند در نسخه پایین Rm ، ماشین‌ها معلوم است

در R مجهول‌ها عبارتند از توالی کار و تعداد ماشین و تخصیص

در Rm تنها توالی و تخصیص مجهول هستند

در هر صورت کلمه تخصیص وجود دارد.

سوال : R سخت تر است یا استایل قبلی

پاسخ : R سخت تر می‌باشد چون در قبلی تنها سرعت‌ها متفاوت بود ولی اینجا نه تنها سرعت‌ها متفاوت است بلکه واسطه با کار نیز می‌باشد.

۶- $\alpha_1 = F$: محیطی به اسم $\alpha_1 = F$ است که به آن فلوشاپ می‌گوییم . همان خط تولید است.

کارها باید در یک محیط فلوشاپ انجام شود یعنی روی یک تعداد ماشین‌ها به صورت سری تمام کارها باید مسیر یکسانی داشته باشند یعنی آنها مجبور بشوند *Process* بشوند روی ماشین یک بعد روی ماشین دو و الی آخر. تعداد ماشین‌ها متغیر است که باید مشخص شود.

توالی کارها مشخص است.

اینجا یک کارگاه است ۷ تا دستگاه کنار هم قرار گرفته ، کاری که وارد کارگاه می‌شود مسیرش مشخص است اول روی دستگاه یک ، بعد دو ، بعد سه ، بعد ۴ و تا آخر و در آخر هم از کارگاه خارج می شود ولی اینکه کدام کار اول وارد شود مشخص نیست یعنی توالی کار مشخص نیست تنها مسیر مشخص است. و توالی کارها را ما باید تعیین کنیم. مجهول بعدی تعداد ماشین‌ها می‌باشد که مشخص نیست. این ۷ تا را هم خود استاد گفت. یعنی از ماشین یک چند تا داریم یا در ایستگاه ۲ چند تا ماشین وجود دارد ؟ اینها مجهول هستند و باید تعیین شوند. ولی کلمه تخصیص نداریم چون مسیر مشخص است. توالی و تعداد ماشین

خوب Fm مشخص است که ماشین‌ها معلومند فقط باید توالی کارها را تعیین کنید.

۷- *Open shop* : کارها باید *Process* بشوند در یک محیط *open shop*. شامل تعدادی متغییر ماشین. هر کار مجبور *Process* گردد روی هر ماشین. تا این قسمت شبیه فلو شاپ بوده است. هر چند برخی از این *process time* ها ممکن است صفر باشد. ممکن است روی یک ماشین کاری نداشته باشیم. تفاوت

عمده‌اش اینجا ست : وجود ندارد هیچ محدودیتی در رابطه با مسیر که کارها باید طی کند . دفعه قبل کار که می‌آمد اول روی ماشین یک ، بعد دو ، بعد سه می‌رفت ولی این کار که داخل می‌آید اول باید ۵ تا کار روی آن انجام شود ولی اولی ، دومی کار مهم نیست باید این ۵ تا کار انجام شود. به طور مثال شما روز آخر یک فرم تسویه می‌گیرید که باید چند تا امضاء و مهر شود ، کتابخانه باید بری، سایت باید بری، امور مالی باید بری و الی انتها به هم ربطی ندارد. توالی کار اختیاری است. اول، دومی مهم نیست.

پس توالی کارها، تعداد ماشین‌ها و برای اولین بار مسیر کار را ما باید تعیین کنیم در قبلی مسیر کار تعیین شده بود ولی در اینجا ما باید تعیین کنیم که اول کدام یک از این امضاءها را باید بگیریم یا اول بهتر است کدام *Process* انجام شود. کدام کار باید داخل شود (کدام بیاید مهم است) یک بحث است و به چه ترتیبی بین ماشین‌ها حرکت کند هم بحثی دیگر است و تعداد ماشین‌ها که مجهول است مسئله سختی است

Open shop m ، هم دیگه نمی‌خواهد تعداد ماشین‌ها معلوم است و توالی و مسیر کار مجهول است

۸- *Job shop* : کارها باید *Process* بشوند در یک محیط *job shop* . هر کار مسیر خاص خودش را دارد تا اینجا چه فرقی با *Open shop* دارد ؟

مسیر کار به انتخاب شما نیست مشخص شده است.

و چه فرقی با فلوشاپ دارد ؟

در فلوشاپ مسیرها مشخص است برای همه یکسان است ولی در اینجا مسیرها مشخص است ولی برای کارهای مختلف فرق می‌کند. کار یک مسیر متفاوتی از کار دو و یا کار سه دارد، تعداد توالی و ماشین مجهول است و باید مشخص شود و در Jm فقط ماشین‌ها $\alpha_1 =$

این شکل چیست ؟

Reduction Graph field α_1

گفتم α ، *Reduce* میشه به β یعنی چی ؟ یعنی β سخت تر است. جهت فلش‌ها جهت *Reduce* شدن است.

P_1، *Reduce* میشه به P_m و P_m ، *Reduce* میشه به P . P_m از P سخت تر است و Pاز P_m سخت تر است.

سوال : سخت ترین مسأله کدام است :

پاسخ *J,R*

پس بین *R* و *J* کدام سخت تر است ؟

چیزی ثابت نشده است.

شما به شرطی می‌توانید بگوید از *A* به *B* سخت تر است یا از *B* به *A* سخت تر است که از *A* به سمت *B* فلش پیدا کنید. الان از *J* به *R* یا از *R* به *J* فلشی وجود ندارد. حالا P_1 سخت تر است یا *F* ؟ مسلماً *F* چون از P_1 جهت فلش را دنبال کنید به *F*می‌رسید. چون F_m از P_1 سخت تر است *F* هم از F_m

F سخت تر است یا P_m ؟ این را نمی‌توانید بگویید . جهت فلش‌ها مهم هستند جهت فلش‌ها جهت سخت شدن است به این می‌گویند *Reduction Graph* یک مسأله

در زمانبندی پروژه

α_1 نداریم یعنی تهی است

α_2 تعداد منابع (غیر از ماشین‌ها) – لاندا به معنای چند نوع منبع غیر ماشین داریم بود در مبحث زمانبندی ماشین و حال این α_2 نقش لاندا را در زمانبندی ماشین را دارد. انواع منابع غیر ماشین :

تهی باشد یعنی منبع غیر ماشین نداریم

یک باشد یعنی یک منبع غیر ماشین داریم

m باشد یعنی m منبع غیر ماشین داریم.

تهی راحت است و اگر یک نوع منبع داشتیم سخت می‌شود و اگر m باز هم سخت‌تر

کدام *sub problem* است ؟ یک ، *Sub problem*

m رو که حل کرده باشید یک رو نیز حل کرده‌اید چون m تعمیم یافته یک است.

پس α_2 تعداد منابع غیر ماشین است.

α_3 نوع منبع است. در کنترل پروژه لیسانس چند نوع منبع داشتیم ؟ تجدید پذیر ، تجدید ناپذیر و دیگه چی ؟

۷ حالت منبع از لحاظ نوع داریم :

$\alpha_1 = 0$: تهی است یعنی وقتی منبع نداریم پس نوع هم نداریم.

$\alpha_3 = 1$: منابع باید *renewable* و تجدید پذیر هستند. یعنی منبعی که مصرفی نیست مثل : کامپیوتر ، نیروی کارگر، منبع تجدیدپذیر هستند در گیر کار هستند ولی مصرف نمی شوند.

$\alpha_3 = T$: *Non-renewable* دقیقاً برعکس قبلی است که با T نشان داده می‌شود تجدید ناپذیر هستند. مثل سیمان کلاً مواد اولیه یعنی وقتی داری از این مواد امروز استفاده می‌کنید داره ازش کم می‌شود مصرف می‌شود، فردا دیگر به مقدار امروز نداری.

$\alpha_3 = 1T$: یک تجدید پذیر بود و T تجدید ناپذیر. $1T$ چی میشه ؟ *double constrain* یعنی هم تجدید پذیر است هم تجدید ناپذیر. چطور ممکن است هم تجدید پذیر باشد تجدید ناپذیر؟

فرض کنید در محیط پروژه هر هفته چهارشنبه یک کامیون سیمان می‌آید چه جور منبع می‌شود هر هفته یک کامیون سیمان می‌آید یعنی هر هفته داره تجدید می‌شود اما در طول هفته مصرف می‌شود. که ازش کم می‌شود و دوباره شارژ می‌شود. یعنی در بازه‌های مشخصی دوباره پر می‌شود ولی در طول بازه دوباره مصرف می‌شود یعنی هم هست و هم نیست.

$\alpha_3 = V$: *Partial renewable resource* (منبع تجدیدپذیر جزئی) مثلاً کارگر تجدیدپذیر است اما کارگرها باید بین پنج شنبه و جمعه یک روز را *Off*باشند الان مسئله تجدید پذیر بودن کارگر زیر سوال می‌رود ؟ چرا ؟ چون روز پنج شنبه و جمعه این کارگر تجدید نمی‌شود این را به اصطلاح *Partial renewable resource* می‌گویند. یعنی تجدید پذیر بودنش مقطعی است یعنی یه جایی هم نیست. درخصوص *non-renewable* ها هم تواند قضیه *partial* باشد.

$\alpha_3 = x$: که با نماد X است منابع تجمعی *(cumulative resource)* است. در سیستم‌های موجودی و تولید منابع داریم به اسم منابع *cumulative* مثلاً پالت‌ها چه طور منبعی هستند ؟ تجدیدپذیر نیستند چون وقتی روش قطعه وجود دارد قابل استفاده نیست یا لیفتراک که برای حمل استفاده می‌شود اینها در طبقه بندی تجدیدپذیر و ناپذیر نمی‌گنجد باید طور دیگری به آن نگاه کرد. که اسمشون را منابع تجمعی گذاشته کلاً کالاهایی که برای حمل و نگهداری استفاده می‌شود جزء منابع تجمعی هستند. مثل لیفتراک ، پالت

$\alpha_3 = \sigma$: منابع دیگری *Special resource* است . این کلاس چه طور منبعی است . یه سری صندلی خالی در کلاس است کسی جرأت داره که در آنجا بشیند آیا کسی می‌تواند از این منبع استفاده کند یا نه ؟ منبع توسط گروهی از فعالیت‌ها (که شما هستید) فقط تخصیص داده شده تازمانی که آخرین نفر از این کلاس خارج نشده گروه بعدی نمی‌تواند ازش استفاده کند. حتی اگر این کلاس با یک یا دو نفر تشکیل شده بود با کلی صندلی خالی باز قابل استفاده برای منبع دیگر یا فعالیت دیگر نبود. چون این فعالیت در آن واحد تنها به یک گروه خاص اختصاص داده شده

است . از لحظه اولین فعالیت از این منبع استفاده می‌کند تا آخرین فعالیت این منبع را خلاص کند این درگیر است که بهش منابع ***Special*** گویند.

این گراف سختی مسائل از این جنبه است تجدیدپذیر یا تجدیدناپذیر کدام سخت تر است ؟ چیزی معلوم نیست ولی ***double constrain*** ها از هر دوی اینها سخت تر است ***partial renewable*** ها از ***double constrain*** ها سخت تر اند . ***cumulative***ها از ***partial*** ها سخت تراند. آخر ***Special resource*** از تجدیدپذیرها هم سخت تر اند. ولی بین ***special*** و ***double*** ها نمی‌دانید کدام سخت تر است.

T ، ۱ سختی‌شان برابر است را ترجمه نکنید. گفتم اثباتی برای اش وجود ندارد.

$\alpha_۴$: دسترس پذیری منبع که خودش ۷ حالت دارد:

تهی است یعنی چی ؟ $\alpha_۴ = \emptyset$ یعنی ***renewable resource*** ها در دسترس هستند به هر مقدار تایپ دلخواهی ، پس نه اینکه نداریم. منبع که هر چقدر بخواهید هست می‌شه $\alpha_۴ = \emptyset$ انتخاب اش با شما است. دسترس پذیری‌اش حتماً باید ثابت باشد. مثلاً شما به کارفرما می‌گویید که قرار است چند تا کارگر استخدام کند. میگه شما به عنوان پیمانکار هر چقدر اعلام کنید من استخدام می‌کنم. این چه منبعی است ؟ تنها تعداد بگویید. انتخابش با مسأله است یعنی جزء مجهولات مسأله است.

$\alpha_۴ = k$ ، ***renewable resource*** ها در دسترس هستند در مقدار ثابت ولی اختیاری نیست ولی مقدارش چند است ؟ K یعنی سقفش تعیین شده است. ما از این منبع حداکثر ۵۰ تا (ثابت) داریم. کمی فکر کنید کدام سخت تر است. کدام مسأله سخت تر حل می‌شود بالایی، همیشه تهی آسان نیست. این که منبع خودش را تعیین کند راحت‌تر است تا اینکه خودمان منبع را تعیین کنیم اگر ما تعیین کنیم خودش داستانی می‌شود.

$\alpha_۴ = Va$: ***partial renewable resource*** ها در دسترس اند در ، فرق اش با اولی این است که اولی مقدار ثابت بود ولی اینجا مقدار متغیر است. سخت تر است.

حالا این سه مورد را به ترتیب ***Sort***کنیم.

اول از همه K راحت‌تر، تهی سخت تر می شود و متغیر $\alpha_۴ = k$ از همه سخت‌تر می‌شود.

خوب تا آخر کلاس هر نمادی که ***Bold***شده باشد تصادفی ***(stochastic)*** و هر نمادی که بالای آن علامت مد دارد فازی است.

$\alpha_۴ = a$ یعنی دسترس‌پذیری منبع ***stochastic*** است که در طی زمان ثابت می‌ماند این با K صفحه قبل چه فرقی می‌کند. آن K معلوم و ثابت بود ولی در اینجا مقدار ثابت است ولی هنوز معلوم نشده چند است. با اولی چه فرقی می‌کند با تهی ؟ تهی هم ثابت بود ولی انتخابش با شما بود ولی در اینجا این انتخاب با شما نیست تصادفی قرار اتفاق بیافتد. مثلاً پیمانکار قرار است تعدادی استخدام کند ، اگر قرار باشه به پیمانکار بگویید چند نفر استخدام کند که میشه آن اولی ولی اگر قرار چند نفر استخدام کند و تصمیم را هم خودش می‌گیرد شما را در جریان نمی‌گذارد و تعداد استخدامی مشخص نیست احتمالی است معلوم نیست

$\alpha_۴ = \tilde{a}$: دسترس پذیری منبع فازی است، غیرقطعی می‌باشد.

$\alpha_۴ = Va$: دسترس پذیری در طول زمان متغیر است

یعنی متغیر و تصادفی است.

$\alpha_4 = V\tilde{a}$: هم منبع متغیر است و هم فازی است.

پس می تواند حالت‌های مختلف در اینجا اتفاق بیافتد گراف سختی‌اش K راحت‌تر، تهی سخت، متغیر سخت ، متغیر احتمالی از همه سخت‌تر ، احتمالی از قطعی سخت‌تر ، فازی هم سخت‌تر، فازی و احتمالی را نمی‌شه گفت کدام سخت تر است ولی متغیر و فازی اگر باشد از فقط متغیر و فقط فازی سخت تر است. نمادهایی که Bold هستند یعنی احتمالی‌اند. خوب کلاً راجع به مبحثی صحبت کردیم. α بود دیگه، منابع ، از ۴ جنبه :

۱- α_1 که فقط محیط ماشین‌کاری بود که یه بحث جدایی بود

۲- α_2 تعداد منابع

۳- α_3 نوع منابع

۴- α_4 دسترس پذیری بودن منابع را نشان می‌داد.

۱۶ ۱T ۳ ۰

۱۶ تا از آن داریم *doubly constrain* داریم سه نوع منبع ماشین نداریم

این را ترجمه کنید. تهی، سه ، یک ، T ، ۱۶

تهی یعنی محیط ماشین نیست ، سه یعنی سه نوع منبع داریم، این منابع *Doubly constrain*هستند ، یک تجدیدپذیر بود ، T تجدید ناپذیر، T , ۱ هر دو . این منابع *Doubly constrain* ، ۱۶ تا هم ازش داریم*(k)*. پس ۳ نوع منبع *Doubly constrain* داریم که از هر کدام ۱۶ تا موجود است، ماشین هم نداریم این میشه فضای پروژه از نظر منبع حالا اگر این را عوض کنید خوب مسئله عوض می‌شود. هر کدام از این‌ها را تغییر بدهید یک بحث جدید می‌شود. یه فرض جدید می‌شود.

بریم سراغ ، β چیست؟ β فعالیت می‌باشد.

خود β چند حالت دارد. α خودش ۴ حالت داشت. β را باید به ۹ قسمت کنید تا بتوانید از آن صحبت کنید.

$$\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9$$

β_1 بحث انقطاع فعالیت است که خودش سه حالت دارد.

$\beta_1 = \emptyset$: یعنی *Preemptive* مجاز نیست یعنی بریدگی فعالیت مجاز نیست.

بریدگی یعنی چی ؟ یعنی فعالیتی که شروع می‌شود پیوسته تا انتها پیش رود. مثلاً همین کلاس یک *non- preemptive* می‌باشد یعنی کلاسی که شروع کرده‌ایم تا آخر به صورت پیوسته تشکیل می‌شود حال اگر مابین کلاس یک آنتراک بدهیم *Preemptive* می‌شود یعنی فعالیت را به دو قسمت کردیم. حالا در پروژه هم یه کاری شروع شده باید تا آخر یه تیکه انجام شود و بدون وقفه انجام می‌شود *non- preemptive* است.

$\beta_1 = pmtn$: یعنی *Preemption* مجاز است و از چه نوعی می‌باشد ؟

Preempt-resume یعنی فعالیت از جایی ادامه پیدا می‌کند که از همان جا قطع شده است. مثلاً اگر کلاس را متوقف کنیم و یک آنتراک بدهیم بعد از یک ربع برگشتن از همان جایی شروع می‌کنم که تمام کرده بودم از اول شروع نمی‌کنم

$\beta_1 = pmtn - rep$: اما بعضی مواقع، بعضی فعالیت‌ها *prmpt* که می‌شوند باید از اول تکرار بشوند یعنی زمانی که بریده شد باید آن قسمت دوباره گفته شود. مثال واضح آن بتن ریزی می‌باشد. اگر بتن‌ریزی متوقف شود برای شروع دوباره باید از اول بتن‌ریزی کرد نمی شود آن را ادامه داد چون بتن جدید به بتن قبلی نمی چسبد.
سوال : کدام سخت‌تر است ؟ هیچ اثباتی از لحاظ سختی اینها نسبت به هم وجود ندارد یعنی نموداری اصلاً وجود ندارد.

β_2 چه چیزی را نشان می‌دهد؟ پیش‌نیازی *(precedence constrain)* فعالیت‌ها است. که خودش شش حالت دارد :

- $\beta_2 = \emptyset$: یعنی هیچ پیش‌نیازی بین فعالیت‌ها نداریم.
- $\beta_2 = cpm$: یعنی پیش‌نیازی‌ها فقط *finish to start* ساده هستند.
- $\beta_2 = min$: یعنی پیش‌نیازی‌ها از انوع *time lag* فقط از نوع مینیمم هستند.
- $\beta_2 = gpr$: یعنی پیش‌نیازی از نوع *time lag* ، مینیمم و ماکسیمم هستند. *(Generalize Precedence Constrain)*
- $\beta_2 = prob$: یعنی شبکه پروژه احتمالی است.
- $\beta_2 = fuzzy$: یعنی *time lag* ها فازی هستند.

پس پیش‌نیازی‌ها شش حالت داشت گراف آن هم به صورت زیر می‌باشد.
اگر پیش‌نیازی نداشته باشید مسأله خیلی راحت است. *min time lag* ها سخت‌تر، *min,max time lag* ها سخت تر ، احتمالی و فازی همیشه بالای نمودار هستند. یعنی وقتی مسأله احتمالی و فازی است یعنی بدترین حالت ممکن مسأله اتفاق می‌افتد.

β_3 : زمان *set up* یا زمان آماده‌سازی می‌باشد که خودش ۴ حالت دارد.
سوال : زمان آماده سازی چیست ؟
قبل از انجام یک کار ، مقدار زمانی که می‌خواهید مقدمات انجام آن کار را آماده کنید. یا به اصطلاح زمان آماده سازی بحث آماده کردن فضای کار است. مثلاً فرض کنید ابتدا باید قالب را عوض کنید تا بتوانید ماشین‌کاری کنید. بعضی مواقع خود آن مقدمه یک کار است مثلاً قبل از اینکه این دستگاه را نصب کنید باید فونداسیون بریزید. خود فونداسیون یک فعالیت ، یک کار است. اما اگر بخواهید نمای یک ساختمان را کار کنید نیاز به داربست است. داربست کار نیست یه چیز موقتی است که برداشته می‌شود. که در واقع این پیش نیاز، مقدمه ، ابزار آماده کردن نمای ساختمان است.

- $\beta_3 = \emptyset$: یعنی زمان آماده سازی ندارد.
- $\beta_3 = \rho_j$: یعنی قطعی است
- $\beta_3 = \rho_{jo}$: یعنی احتمالی است
- $\beta_3 = \tilde{\rho}_j$: یعنی فازی است

β_4 : زمان فعالیت (*Duration* فعالیت)
(این فعالیت چند روز طول می کشد تا انجام شود راجع به این مسأله صحبت می‌کند.)

- $\beta_4 = \emptyset$: یعنی فعالیت *Duration*عدد صحیح دلخواه دارد.

مثال : وقتی یک متن تایپی دارید که می‌خواهید تایپ شود و به تایپیست می‌دهید و سوال می‌کنید چند روز طول می‌کشد ؟ از شما سوال می‌کند ؟ چند روزه می‌خواهید ؟
یعنی چی چند روزه می‌خواهید ؟ یعنی مشخص است که می‌تواند یک روزه هم انجام دهد ، ۳ روزه ، ۷ روزه ، ده روزه هم می‌تواند انجام دهد به خاطر اینکه منابع در اختیارش می‌باشد اگر شما عجله داشه باشید به ۵ تایپست می‌دهد و یک روزه تحویل می‌دهد و در صورتی که عجله نداشته باشید به یک تایپست می‌دهد و هفته بعد تحویل می‌دهد.
یعنی کاری که زمان انجامش می‌تواند هر عددی باشد. البته اگر گسسته باشد.

- $\beta_۴ = cont$: اگر موضوع بالا پیوسته باشد زمانش ولی پیوسته است شمارش پذیر نیست.
- $\beta_۴ = d_j = d$: یعنی اگر همه فعالیت‌های پروژه شما تعداد روزهای مشخصی زمان می‌برند که خیلی بعید است چنین اتفاقی بیافتد.
- $\beta_۴ = d_{jo}$: یعنی اگر زمان همه فعالیت‌ها احتمالی است.
- $\beta_۴ = \tilde{d}_j$: یعنی اگر زمان همه فعالیت‌ها فازی است.

$d_j = ۱$: یعنی همه فعالیت‌ها یک روزه‌اند.
$d_j = d$: یعنی همه فعالیت‌ها d روزه‌اند.
$\{d, \dots, \bar{d}\}$ یعنی همه فعالیت‌ها مثلاً بین ۵ تا ۷ روز طول می‌کشد بین یه بازه‌ای طول می‌کشد. یعنی گسسته‌اند . اگر پیوسته هم که باشد . $[d, \bar{d}]$
فعالیت‌ها می‌تواند هر تعداد روزی انجام شود. اون بالا هم که احتمالی و فازی به همین ترتیب مسأله سخت تر می‌شود.

$\beta_۵$: *dead line* است که سه حالت دارد:

- $\beta_۵ = \emptyset$: یعنی اگر *dead line* ، تاریخ تحویل برای فعالیت‌ها، پروژه نداشته باشید.

اگر *dead line*، تاریخ تحویل برای فعالیت ها، پروژه داشته باشید دو حالت زیر می‌شود :

- $\beta_۵ = \delta_j$: اگر *dead line* روی همه فعالیت‌ها باشد.
- $\beta_۵ = \delta_n$: اگر *dead line* روی پروژه باشد.

Dead line را روزی که قرارداد می‌بندند در پروژه نوشته‌اند چرا که احتمالی‌اش را بحث نکرده است. چون *Dead line* احتمالی هیچ وقت نداریم. *Dead line* بین کارفرما و پیمانکار قرارداد نوشته شده که مشخص شده احتمالی نمی‌تواند باشد چیزی نیست که احتمال باشد.
مثال : پایان ترم کی است ؟ احتمالی که نیست روزی که برنامه‌ریزی می‌کردند امتحان شما مشخص شده است این چیزی نیست که به احتمال های شما واگذار کرد.. این در واقع مشخص است جز بحث‌هایی که از اول ترم برنامه‌ریزی می‌شود که امتحان شما در فلان روز است یعنی مشخص است.
اگر *dead line*نداشته باشید که مسأله آسان است و اگر داشته باشید روی پروژه یکم سخت تر می‌شود و روی فعالیت‌های سخت تر از پروژه می‌گردد. چون باید همه فعالیت‌ها را رعایت کنید اما در پروژه ، فقط پروژه را باید رعایت کنید.

$\beta_۶$: نیازمندی منابع

این فعالیت‌ها چقدر منبع نیاز دارند؟

$\beta_۶ = \emptyset$: یعنی اگر نیازمندی هر منبع هر مقدار ثابت باشد. مثلاً برای انجام این دیوار چینی چند کارگر نیاز است ؟ برای ساختن این ساختمان چند کارگر لازم است ؟ با هر تعدادی می شود این کار را انجام داد هر چقدر کارگر بیشتر باشد زودتر تمام می‌شود تعدادش چیزی نیست که مثلاً دقیقاً بگوییم ۱۶ تا ، ۱۷ تا ، ۱۰۰ تا . مقدار نیازش می‌تواند هر عددی باشد. هر چقدر زیادتر باشد، زمان بهتر است.

$\beta_۶ = k$: یعنی برای یک فعالیت حداکثر ۶ نفر نیاز است یعنی تعداد نیاز مشخص است.

$\beta_۶ = Vr$:اگر نیاز ما به فعالیت متغیر باشد، مثلاً اوایل این پروژه کارگر زیادی نمی‌خواهیم اوایل کار اسکلت و جوشکاری است که زیاد کارگر نمی‌خواهیم ولی بعد کارگر زیاد می‌خواهد و بعد کارگر کم می‌خواهد یعنی نوسان دارد.

$\beta_۶$ خیلی تنوع دارد.

$\beta_۶ = disc$: نیازمندی منابع تابعی از زمان است. این تابع می‌تواند گسسته، پیوسته، خطی، محدب، مقعر باشد.

توجه به نمودار ۹ حالت دارد.

مثلاً این دو تا هیچ سختی ای نسبت به اون یکی‌ها اثبات نشده است. اگر آن تابع خطی باشد یعنی نیازمندی و منابع تابع خطی از زمان باشد راحت تر از این که تابعی محدب یا مقعر باشد. باز تابع محدب یا مقعر بهتر از تابع دیگر است. چون حداقل می‌دانید محدب یا مقعر است ولی هر تابعی دیگری غیر از اینها باشد مسأله دوباره سخت می‌شود. اگر سقف نیازمندی مشخص باشد بهتر از این که نامشخص باشد و باز این بهتر است از اینکه متغییر باشد. نمودار دو تیکه است.

$\beta_۷$: مدهای انجام یک فعالیت است. سه حالت دارد :

در β ها کلاً راجع به فعالیت از جنبه‌های مختلف : پیش‌نیازی، بریدگی، *Duration* ، روش‌های انجامش، چقدر منبع نیاز دارد ؟ و ...

- $\beta_۷ = \emptyset$: اگر فعالیت فقط با یک روش اجرای می‌تواند انجام شود.
- $\beta_۷ = mu$: اگر فعالیت با چند مد اجرای مشخص شده انجام شود

فرض کنید یک کانالی را این بیرون حفر کنیم دو تا مد دارد یعنی هم می تواند با کارگر انجام شود هم با ماشین (بیل مکانیکی) انجام شود اگر کانال نمی‌تواند بیل مکانیکی حفر شود پس تنها یک راه دارد آن هم کارگر است. پس انجام کار فقط یک روش دارد Single mode و اگر چند روش دارد multiple

- $\beta_۷ = id$: اگر فعالیت با *mode identity* انجام شود.

فرض کنید قرار است دو سه جا کانال حفر کنید هر کدام از اینها دو مد دارد هم با کارگر هم با بیل مکانیکی می‌شود حفر کرد. اگر قرار شد یکی را با بیل مکانیکی انجام بدهید بهتر است بقیه را نیز با بیل مکانیکی انجام دهیم . چرا ؟ چون وقتی بیل مکانیکی را اجاره کردید و آوردید برای بهینه بودن در همه چیز آن یکی ها را هم با بیل مکانیکی انجام بدهید. یعنی این سه تا به اصطلاح فعالیت باید انجام شوند از یک جنس هستند، هم خانواده‌اند ، بهتر است با یک روش انجام شود درست است که برای هر کدام دو تا انتخاب دارید هر انتخابی که می‌کنید بهتر است برای همه یکسان باشد. چند تا فعالیت هم خانواده داشته باشد. مثلاً پنجره‌ها ، در یک کلاس می‌تواند گالوانیزه، آلومینیوم ، *UPVC* باشد پس سه تا راه دارد. کلاس کناری هم همین

مسأله است کل پنجره‌های ساختمان همینطوری است. آن یکی دانشگاه هم همین طور. ولی خوب از نظر منطقی، زیبایی ، معماری بهتر است همشان مثل هم باشد.
Singlemode راحت و *Multiple* سخت و *Mode identity* سخت تر است.

β_8 : بحث های مالی است.

- $\beta_8 = \emptyset$: اگر فعالیت‌ها بحث مالی نداشته باشد.

سوال : آیا می‌شود فعالیتی بحث مالی نداشته باشد ؟
پاسخ : به هر حال برای هر فعالیت باید پول خرج کرد. بحث مالی نیست نه اینکه تو کار نیست . تو مسأله شما نیست. مساله شما مینیمم کردن زمان پروژه است به پولش کار نداشته باشید. هدف مسئله مینیم کردن زمان است. مسئله شما بحث مالی مهم نیست. نه اینکه وجود خارجی ندارد تو مسئله شما آن مورد بحث نیست.

اگر فعالیت‌ها بحث مالی وجود داشته باشد خودش سه حالت دارد.

- $\beta_8 = C_j$: اگر جریان مالی مشخص باشد.
- $\beta_8 = \boldsymbol{C_j}$: اگر جریان مالی احتمالی باشد.
- $\beta_8 = C_j^+$: اگر مشخص هستند و فقط در آمد هستند
- $\beta_8 = \boldsymbol{C_j^+}$: درآمد و احتمالی هستند.
- $\beta_8 = \tilde{C}_j$: درآمدهای فازی هستند.

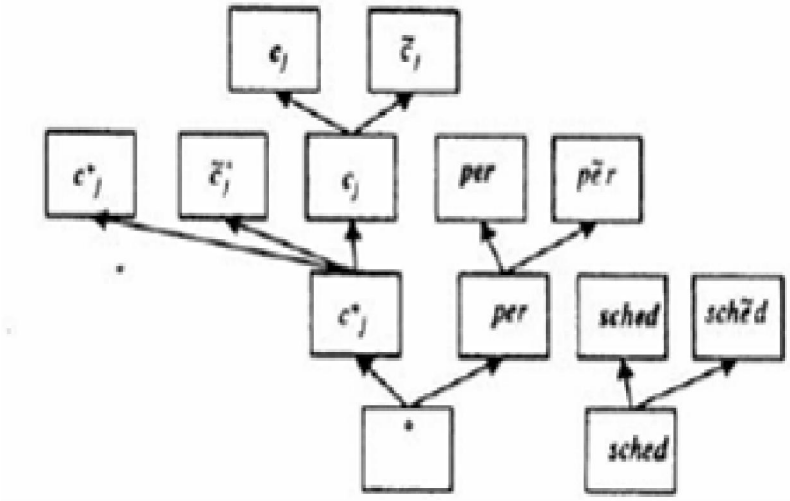
ورژن‌های مختلفی برای مسئله اتفاق بیافتد.
سوال شفاهی : این کانال را که می‌خواهید بیرون حفر کنید چقدر هزینه می‌شود طول کانال ۱۷ متر، عمق یک متر، به عرض ۷۰ سانت ؟
یه زمانی ممکن است شما حساب کرده باشید و دقیقاً می‌دانید هزینه‌اش چقدر می‌شود (C_j) . اما یه زمانی نمی‌دانید چقدر است $\boldsymbol{(C_j)}$. احتمالی است . بستگی دارد که چقدر پول بگیرد.
یک مسئله به اسم *Payment scheduling* داریم یعنی کارفرما طبق چه طرحی به پیمانکار مبلغ پروژه را پرداخت کند. خودش کلی داستان است. این *Payment scheduling* کی مشخص شده ؟ معمولاً در هنگام نوشتن قرارداد یکسری حساب کتاب‌هایی و یک بندهایی نوشته می‌شود و امضاء می‌گردد. قطعاً پیمانکار دنبال این است که این *Payment*ها را زیاد و زود باشد. هر چقدر مبلغ زیادی را زود بگیرد به نفع اش است. کارفرما دوست دارد مبالغ را در زمان دیرتر و کمتر بدهد. این دو تا نگرش که با هم تضاد دارد. معمولاً این مسئله *Payment scheduling* وقتی مطرح می‌کنند باید اولش شما بگویید از طرف کی صحبت می‌کنید یعنی به عنوان مشاور کارفرما مسئله را حل می‌کنید یا به عنوان مشاور پیمانکار چون هدف‌ها با هم فرق می‌کند چون آن می‌خواهد *Payment* ها را ماکسیمم کند و فاصله ها را کم ، اینجا برعکس است می‌خواهد *Payment* ها را مینیمم کند و فاصله‌ها را دورتر بنابراین باید شفاف شود.
در بحث *Payment*

- $\beta_8 = C_j^*$ درآمد احتمالی
- $\beta_8 = \tilde{C}_j$ درآمد فازی
- $\beta_8 = \tilde{C}_j^*$ مشخص و فازی

- $\beta_8 = per$ اگر این مبالغ *(periodic cash flows)* ها مشخص اند.
- $\beta_8 = \boldsymbol{per}$ اگر مشخص هستند ولی معلوم نیست چقدر است احتمالی است.
- $\beta_8 = \widetilde{per}$ اگر مبالغ فازی هستند.
- $\beta_8 = sched$ اگر قرار است شما مشخص کنید خودش یک سوال است یعنی ما هنوز مسئله را حل نکردیم باید تعیین کنیم

پس فرق بین *Per* و *Sched* در چیست ؟ در *Per* تصمیم گیری شده است ولی در *Sched* تازه می‌خواهیم تصمیم گیری کنیم. که خود تصمیم‌گیری قطعی، احتمالی ، فازی‌اش دو باره می‌تواند سه حالت اتفاق بیافتد.

- $\beta_8 = \boldsymbol{sched}$ اگر مبالغ *Payment scheduling* با جریان نقدی تصادفی همراه باشد.
- $\beta_8 = \widetilde{sched}$ اگر مبالغ *Payment scheduling* با جریان نقدی فازی همراه باشد.



β_9 : بحث *change over time*

Ready time(زمان آماده سازی) مثال آن داربست بود.

Change over time یه جورایی همان *setup time* یا *ready time* است ولی وابسته به توالی است.

فرض کنید دستگاه اینجاست که می‌خواهیم این قطعه را فرز کنیم. برای این قطعه را به دستگاه ببندیم تا فرز انجام شود باید ببینیم کدام قطعه قبلاً روی دستگاه بوده است. چون قطعه قبلی تقریباً شبیه قطعه جدید است از هر نظر . سوییچ کردن قطعه‌ها یا در واقع جابجا کردن قطعه‌ها زمان زیادی طول نمی‌کشد شاید چند ثانیه باشد. اما زمانی قطعه ای که روی دستگاه بوده از نظر ابعاد، سایز و ... متفاوت از این قطعه جدید می‌باشد که می خواهیم فرز کنیم پس زمان زیادی می‌برد. درنتیجه زمان تنظیم دستگاه به قطعه قبلی که روی دستگاه بوده است بستگی دارد که به اصطلاح گفته میشود *Sequence –depenendent change over time* زمان برای این قطعه به قطعه بستگی دارد یعنی به توالی بستگی دارد.

- $\beta_9 = \emptyset$ اگر اصلاً همچین چیزی ندارید.
- $\beta_9 = S_{jk}$ اگر دارید مقدارش قطعی است
- $\beta_9 = \boldsymbol{S_{jk}}$ اگر دارید احتمالی است
- $\beta_9 = \tilde{S}_{jk}$ اگر دارید فازی است.

این در واقع مفهوم *change over time* است که اگر نداشته باشید مسئله آسان‌تر و قطعی اش سخت‌تر و احتمالی و فازی اش هم سخت تر می‌شود.

فیلد آخر فیلد γ است. این فیلد مثل فیلدهای قبلی نیست که تعداد حالت داشته باشد این می‌تواند طیف وسیعی از اهداف نوشته شود اهداف مالی خودش کلی است. اهداف منبع محور، اهداف زمان محور، اهداف کیفیت محور بعد دو به دو این اهداف مسائل دو هدفه، سه به سه این‌ها مسائل سه هدفه یعنی ترکیب‌های بسیار مختلفی می‌تواند برای این قضیه اتفاق بیافتد. هر چند من از دو زاویه اینها را تقسیم‌بندی می‌کنم

به طور کلی اهداف به دو قسمت *regular* و *nonregular* (منظم و نامنظم) تقسیم میشود.

پس چه اهدافی منظم هستند: هر هدفی که در راستای تکمیل سریع فعالیت‌ها و پروژه باشد منظم است در نتیجه اهداف نامنظم عکس اهداف منظم هستند.

مثال : هدف در این پروژه این است که کیفیت انجام کار یا به اصطلاح دوباره کاری ها را مینیمم کنیم ؟

اگر هدفتان این است که دوباره‌کاری ها را مینیم کنید انگار کیفیت کار را بیشتر کنید یعنی باید حوصله کار کنید. اصرار و عجله‌ای نباید به خرج دهید. یعنی این موضوع مفهومی است کامل. وقتی که شما می‌خواهید دوباره‌کاری ها را مینیمم کنید باید کیفیت کار را بالا ببرید پس کار را با حوصله بیشتری باید انجام شود سرعت نباید به خرج دهید سرعت مهم نیست در حالی که در اهداف منظم برعکس است سرعت باید به کار داد.

مثال : هدف این است که *npv* (ارزش فعلی خالص پروژه) را ماکسیمم کنید. هر فعالیتی یکسری درآمد دارند و یکسری فعالیت‌ها هزینه دارند و در کل می‌خواهیم *npv* پروژه را ماکسیمم کنیم؟ آیا منظم است؟

خیر منظم نیست چون

نکته : کارهایی که قرار است درآمد داشته باشد را زود انجام می‌دهیم و آنهایی که هزینه دارند را دیر انجام می‌دهیم. پس فعالیتی که هزینه دارد بهتر است آخر انجام شود. یعنی هر چقدر درآمدها نزدیک و هزینه ها دور باشند باعث بهتر شدن *npv* میشود.

خوب با این نکته ماکسیمم کردن *npv* منظم است یا نامنظم ؟

فعالیت‌های که هزینه دارند را می‌خواهید دیر انجام شود در تعریف اهداف منظم گفته شده است سریع پس اینطور نیست فعالیت‌هایی که درآمد دارند سریع و آنهایی که هزینه دارند دیر در نتیجه منظم نیست.

اما یه موقع می‌گویید همه فعالیت‌های پروژه من درآمد زا هستند نتیجه منظم است.

جنبه دیگر تقسیم‌بندی اهداف پروژه عبارتند از : اهداف مالی، اهداف زمان محور، اهداف منبع محور ، اهداف کیفی

Max کردن *npv* چه هدفی است ؟ هدف مالی

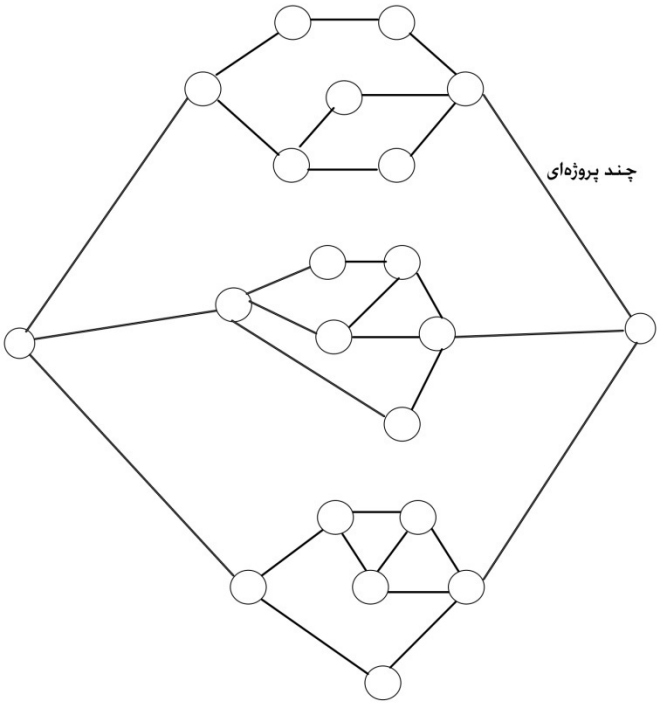
*Min*کردن دوباره‌کاری ها چه هدفی است ؟ اهداف کیفی

تسطیح کردن منابع چه هدفی است ؟ منبع محور

زود تمام کردن پروژه چه هدفی است ؟ زمان محور

حال ده‌ها هدف مالی، زمانی، منبعی، کیفی داریم که بعضی‌هاشون منظم و بعضی ها نامنظم است. اما نکته بعدی این است که ما یکسری از مسائل داریم که می‌خواهیم چند تا هدف را با هم می‌خواهیم. یعنی من هم زمان می‌خواهم زمان پروژه را *min*کنم و همزمان می‌خواهم دوباره کاری ها را مینیمم کنم این قضیه مسائل *multi objective* (چند هدفه) می‌شود.

Multi project scheduling (چند پروژه‌ای)



سه تا پروژه داریم.

مثلاً شرکتی که اجرا کننده شرکت‌های عمرانی است با ۳ تا پروژه قرارداد بسته است پس برای آن شرکت، مسئله‌ای که با آن مواجه است ***multi project*** است .

این جمله درست است یا خیر

بهتر است این سه تا پروژه را جدا کند و بگویید سه تا پروژه جدا دارم ؟

خیر چون منابعی که توی اینها استفاده میشود معمولاً منابع مشترکی می‌باشند.

مثلاً تیم ماشین حفاری که در پروژه ۱ درگیر است نمی‌تواند در پروژه ۲ و ۳ باشد برای ما همه اینها کلاً یک پروژه است درست است که سه تا قرارداد جدا دارند.

اما گاهی اوقات سه تا تیم جدا دارید این یه پروژه ***IT*** است . این پروژه راه سازی و دیگری پروژه ساختمان سازی است. این سه پروژه منابع هیچ ربطی به هم ندارند. پس می‌توانید بگویید که این سه پروژه مستقل هستند و اشتراکی ندارند.

حل اولی چطوری است ؟

شروع پروژه را با یک گره مجازی و پایان را با یک گره مجازی

الان در کل این یک پروژه است.

Multi project دو رویکرد دارد. اگر این پروژه‌ها اشتراکی ندارند جدا میشوند و سه پروژه مستقل می‌گردد. اما اگر اشتراک منابع دارد این سه تا پروژه را یک پروژه ببنید.

مینیمم کردن هزینه زود کرد ـ دیر کرد پروژه ⟵ هدف مالی است

مینیمم کردن ***lateness*** پروژه دیرکرد پروژه ⟵ زمان محور

تسطیح منبع ⟵ مینیمم کردن ***Waiting jumping resource*** (میزان پرش در استفاده از منبع یا افت در استفاده از منبع) ⟵ منبع محور

پارامترهای که ***Bold*** بودن به معنی احتمالی است . نکته : دراحتمال بودن شما نمی‌توانید بگویید ماکسیمم کردن ***npv*** یا نمی‌توانید بگویید مینیمم کردن زمان . به جای آن باید بگویید، مینیمم کردن متوسط زمان ، ماکسیمم کردن متوسط ***npv*** . در احتمالی حتماً باید با امید ریاضی بحث کنید.

$\alpha_1 = P_1$: یعنی یک تک ماشین در کارگاه

$\alpha_2, \alpha_3, \alpha_4$: هیچ چیز ننوشته است یعنی تهی است یعنی منبعی غیر از ماشین هم نداریم.

از ۹ تا β دو تا را نوشته است.

پیش‌نیازی‌ها از نوع ***Finish to start*** و زمان آماده سازی هم داریم و هدف $C_{\max}$: زمان پروژه را $C_{\max}$ می‌گویند.

این زمانبندی ماشین است نه پروژه ، چرا ؟ چون α_1 دارد ولی $\alpha_2, \alpha_3, \alpha_4$ ندارد

بین فعالیت ها پیش‌نیازی از نوع ***Finish to start*** است زمان آماده سازی قطعی و معلوم است. و هدف انجام سریع کار ها می‌باشد.

مسئله آخری چیه ؟

$P_2 \longleftarrow P_m$ یعنی ۲ تا ***Parrallel identical machine*** داریم (ماشین‌های یکسان و موازی)

بازم پیش‌نیازی دارد زمان آماده سازی هم دارد هر فعالیت یا هر کار چقدر طول می‌کشد که در دستگاه ***Process*** شود یک ساعت و هدف هم $C_{\max}$ است.

مسئله دیگر

$j_2 \longleftarrow j_m$ یعنی دو تا ماشین در کارگاه در محیط ***(job shop)*** دارید، زمان آماده سازی وجود دارد، ***duration*** فعالیت‌ها یک روز است و هدف مینیمم کردن ***Lateness*** (هدف زمان محور) دیرکرد پروژه را می‌خواهیم. مینیمم کنید بیشترین دیرکرد را $L_{\max}$ بیشترین دیرکرد است. بیشترین ***Lateness*** (تأخیر) را مینیمم کنید.

Resource scheduling

$$P1 \mid cpm, \rho_j \mid C_{max}$$
$$P2 \mid cpm, \rho_j, d_j=1 \mid C_{max}$$
$$F2 \mid pmtn, tree \mid \overline{F}$$
$$J2 \mid \rho_j, d_j=1 \mid L_{max}$$
$$P2,1,1 \mid d_j=1 \mid C_{max}$$
$$P2,1,1,1 \mid chain,\ d_j=1,1 \mid C_{max}$$
$$F2,2,1 \mid pmtn, d_j=1 \mid C_{max}$$
$$F2,1,1,1 \mid chain, d_j=1,1 \mid C_{max}$$
$$J2,1,1,1 \mid d_j=1,1 \mid C_{max}$$

Machine scheduling

$$1 \mid prec, r_j \mid C_{max}$$
$$P2 \mid prec, r_j, p_j=1 \mid C_{max}$$
$$F2 \mid pmtn, tree \mid \overline{F}$$
$$J2 \mid r_j, p_j=1 \mid L_{max}$$
$$P2 \mid res1.., p_j=1 \mid C_{max}$$
$$P2 \mid res111, chain, p_j=1 \mid C_{max}$$
$$F2 \mid pmtn, res2.., p_j=1 \mid C_{max}$$
$$F2 \mid res111, chain, p_j=1 \mid C_{max}$$
$$J2 \mid res111, p_j=1 \mid C_{max}$$

$F_{2,2,1}, |pmtn, d_j = 1|C_{max}$

یعنی فلوشاپ، دو تا ماشین است ، دو تا منبع غیر ماشین داریم که تجدید پذیر هستند. بریدگی مجاز است. *duration* آن یک است. هدف هم C_{max}

$F_2|pmtn, rest_2 \ldots, P_j = 1|C_{max}$

حالا این ظاهراً در طبقه بندی قبلی ، α_1 نداشته فقط α داشته که $\alpha = F_2$ بوده است. مثلاًبحث بریدگی وجود دارد حالا با $rest_2$ نشان داده ولی C_{max} همان است .

بیشترین تفاوت در اولی می‌باشد. آنجا فقط α داشته‌اید که الان α خودش ۴ تا است α_1 تا α_4 . که البته شما این جا سه مورد را می‌بینید. چون چهارمی تهی بود اینجا نوشته نشد. $C_j \longleftarrow C_4$

در این پارت که صبحت شد کلی اصطلاح تکرار شد. شما اگر بخواهید یک مسأله *search* کنید باید با نگرش زیر *search*کنید باید بگویید مسأله من سه تا جنبه دارد:

- منبع
- فعالیت
- هدف

حالا که *search* میکنید حتماً که اول شروع می‌کنید باید تایپ کنید *Project scheduling* چرا ؟ چون شما در مسائل زمانبندی پروژه *search* می‌کنید اگر کلمه *Project* را ننویسید کلی از مقاله‌هایی که درخصوص زمانبندی ماشین هستند را پیدا می‌کند که اصلاً به درد شما نمی‌خورد یعنی تا کلمه *Project* را نیاورید نمی‌فهمد که *search* شما درخصوص زمانبندی پروژه است و بسیاری از مقاله‌هایی که هم زمانبندی ماشین است و زمانبندی پروژه خیلی کمتر است. اگر می‌خواهید زمانبندی پروژه پیدا کنید حتماً کلمه *Project scheduling* را باید تایپ کنید. و به اضافه چه منابعی که می‌خواهید *search* کنید. α را *renewable resource* تایپ کند از نظر β چه ویژگی‌های را می‌خواهی داشته باشد . مثلاً تایپ کن *Cashflu* $\longleftarrow C_8$ ، تایپ کن *change overtime* مثلاً تایپ کن *ready time* تایپ کن *GPR* یا *min time lag* یعنی هر کدام از این ها را که تایپ کنید *search* شما محدود به فیلد مورد نظر است.

حالا توی فیلد α تایپ کند *multi objective* اگر دنبال مسائل چند هدفه هستید یا تایپ کن C_{max} یا فرض کنید تایپ کنید *resource leveling* یعنی می‌توانید با این کلید واژه‌های ۱۴ گانه ، ۱۴ زاویه متفاوت مسأله‌تان را پیدا کنید یا برعکسش یک مقاله در دست دارید می خواهید ببینید چه جوری است باید از ۳ زاویه این را کشف کنید از نظر منبع چه فرض‌هایی دارد. از نظر α ها

حالا ممکن است α یا β را ننویسید ولی ...

این بیشتر بحث نگرش است و مسئله سختی آن

جلسه ششم
زمانبندی پروژه با منابع نامحدود

زمانبندی پروژه با منابع نامحدود (بدون محدودیت منبع) از نظر علمی فرض واقع‌بینانه نیست و هر پروژه منابع آن محدود است، فرضض می‌کنیم منابع نامحدود است. زمانبندی پروژه را انجام می‌دهیم یعنی تعیین زمان شروع و پایان فعالیت‌های پروژه ، خروجی ان چند تا عدد است، زمان شروع یا پایان، یکی کافی است. چون شروع را بگوییم پایان مشخص می‌شود چون مدت زمان آن مشخص است. فعالیتی ۱۰ روز طول می‌کشد، سوم شروع می‌شود، یا سیزدهم تمام می‌شود بیشتر معمول این است که شروع فعالیت را متغیر می‌گیرند، ولی پایان را اگر متغیر بگیرید مشکلی نیست، در زمانبندی پروژه که منابع نامحدود دارند سه سوال پاسخ داده می‌شود.

۱- پروژه یا بخشی از پروژه چقدر طول می کشد یا *minimum makespan* پروژه چند روز است. این پروژه را حداقل چند روزه می‌توان تمام کرد.

۲- یه فعالیت خاص زودترین زمانی که می‌تواند شروع شود چه زمانی است.

۳- چقدر یک فعالیت می‌تواند تأخیر داشته باشد که منجر به تأخیر پروژه نشود، شناوری فعالیت

البته ما فرض می‌کنیم که منابع مطرح نیستند تنها موضوعی که باید رعایت شود تنها محدودیتی که باید رعایت شود محدودیت به اصطلاح *Precedence Constraints* یا محدودیت‌های پیش‌نیازی می‌باشد. جمله آخر اینجا در مورد چیست؟

فرض می‌کنیم که تمام *Data* های مسأله قطعاً معلوم هستند به اصطلاح احتمالات یا فازی در مسأله درگیر نیستند. برای اینکه بتوانیم این موضوع را تحلیل کنیم در ادامه موضوع را به دو قسمت تقسیم می‌کنیم.

قسمت اول : شبکه پروژه در قالب *AON* باشد.

قسمت دوم : شبکه پروژه در قالب *AOA* باشد.

درخصوص این شبکه‌ها قبلاً صحبت شده است. در شبکه‌های *AON* گره‌ها به معنی فعالیت، بردارها به معنی پیش نیازی و پیش‌نیازی بین فعالیت‌ها از نوع *Finish to Start* ساده است. پس گرافی داریم که یک گراف را با G نشان داده است V همان فعالیت‌ها است، E همان بردارها به معنی پیش‌نیازی است. چند تا نماد در اینجا تعریف کرده است که خیلی نمادهای ناآشنا نیستند و احتمالاً ذهنیتی هم در رابطه با آنها دارید.

EST_j ⟵ *(earliest start time)* یا زودترین زمان شروع فعالیت j

EFT_j ⟵ *(earliest Finish time)* یا زودترین زمان پایان فعالیت j

P_j ⟵ *(immediate predessor)* یا مجموعه پیش‌نیازی فوری فعالیت

اگر یادتون باشه همه پیش‌نیازی P_j^* نشان می‌دهیم.

d_j ⟵ هم به معنی *Duration* فعالیت j ام است.

حالا اولین باری که در کلاس تعریف می‌کنیم ولی تا آخر، در هر مقاله، یا کتاب یا متنی که راجع به زمانبندی باشد این اصطلاح معمولاً می‌باشد. حالا ممکن است در بعضی از *reference* ها *ES* را به عنوان *earliest start time* تایپ شده باشد و *T* را نداشته باشد. این قراردادهایی که است مهم است که به صورت *Unique* شما آن را تعریف کنید.

مهمترین نکته در یک متن، کتاب، مقاله ، جزوه یا هر بحثی در تعریف پارامتر که تعریف پارامتر تا آخر ثابت بماند یعنی شما نمی توانید تعریف را بعد از یک ، جلوتر عوض کنید و یه چیز دیگه را به عنوان *EST* در نظر بگیرید.

از نظر نگارش یه حرفی را که *Italic* می‌نویسید یا نمی‌نویسید یا *Bold* هست یا نیست معنی متفاوتی دارد. پس وقتی *d* را که *Italic* می‌نویسید و به عنوان *Duration* تعریف می‌کنید تا آخر همین نظم را رعایت کنید حالا این موضوع مثلاً در پایان نامه، پروپوزال تو هر متن، در هر مقاله ، در هر بحثی که نمادگذاری است باید قطعاً رعایت شود.

earliest start time و *earliest Finish time* در کنترل پروژه یک ذهنیتی دارید. از محاسباتی موسوم به محاسبات *Forward* و *Backward* که زودترین و دیرترین زمان‌ها در واقع چطور محاسبه می‌شوند. *Forward* زودترین زمان حساب می‌شود پس محاسبات *Forward* انجام می‌شود برای اینکه *earliest start time* و *earliest finish time* فعالیت‌ها محاسبه شود . یادتان می‌آید در محاسبه *Forward* چه کار می‌کردیم؟ گام یک، دو، سه ، چهار چی بود؟

سودوکد یا همان شبه کد *(pseda code)* در واقع یه جوری بیان آن الگوریتم است . البته این اولین باری نیست که داریم قبلاً هم داشتیم دیگه نه کجا داشتیم؟ محاسبه μ_j شبکه در جلسات قبل، سودوکد خیلی طولانی هم داشت که تقسیم به قسمت‌های کوچکتر تقسیم می‌کردیم و به سراغ مثال می‌رفتیم. که یک جور سودوکد بود.

سودوکد با کد چه فرقی می‌کند؟

مثلاً شما در *Matlab* یا در *VB* یا زبان *C* کد می‌نویسید یه الگوریتم را کد می‌نویسید و *Run* می‌کنید، خروجی هم می‌گیرید. در کد شما به زبان *C* با رعایت قوانین برنامه‌نویسی *C* یه برنامه‌ای را کد می‌نویسید اما عین همین را اگر کپی کنید در *C* یا هر زبان برنامه‌نویسی دیگر نمی‌توانید ازش خروجی بگیرید این در واقع یه بیان استاندارد متنی است نه کامپیوتری. نه اینکه شما وارد برنامه کنید خروجی از آن بگیرید. این فقط برای این است که در واقع یک زبان مشترک برای توضیح داشته باشیم. به زبانی شبیه زبان برنامه‌نویسی نه عیناً.

به طور مثال: در *Matlab*، دستور *For* ، := نمی خواهد فقط = است اگر =: بگذارید *error* می‌دهد یا *;* (سمیکالون) در این برنامه ندارید در صورت گذاشتن با پیغام *error* روبرو می‌شوید. این دستورها در زبان‌ها با یکدیگر متفاوت می‌باشد.

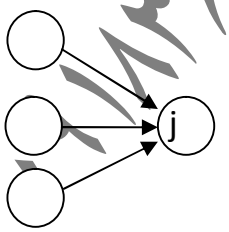
پس شبه‌کد این نیست که عیناً کپی کنید و بتوانید برای آن خروجی دریافت کنید ولی می‌توانید از آن استفاده کنید و در*Matlab* بنویسید ولی با رعایت قوانین *Matlab* ، با رعایت قوانین *C++* ، با رعایت قوانین *VB* تا بتوانید خروجی را دریافت کنید. ولی این کد است کدی که به اصطلاح سودوکد می‌گویند.

در کنترل پروژه دستور *Forward* و *Backward* داشته‌اید ولی این را نداشتید پس چه جوری آنجا این را انجام می‌دادید؟ الگوریتمی گفته می‌شود دارای گام‌هایی بود مثلاً گام یک : یک جمله بود. گام یک : برای اولین فعالیت زودترین زمان شروع و دیرترین زمان شروع را مساوی صفر قرار دهید.

گام دو : به ترتیب برای فعالیت‌های از ۲ تا n ، مثلاً این کارها را انجام دهید. گام سه، گام چهار، گام پنج به این نقطه که رسیدید تمام می‌شود. یعنی در قالب گام‌هایی که بصورت *level* پایین استفاده می‌شود. معمولاً در نوشتن مقاله این شیوه نوشتاری را به عنوان یک الگوریتم یک توضیح نمی‌پذیرند مخصوصاً اگر این مثلاً گام‌هایش بسیار طولانی باشد. به طور مثال نمی‌توانید ۱۷ تا گام راجع به یک الگوریتم بنویسید ولی با زبان سودوکد می‌شود که یک طوری استاندارد ارائه الگوریتم است.

Initialisation : $EST_1 = EFT_1 = 0$;
for $j := 2$ **to n do**
begin
$EST_j := \max\{EFT_i | i \in P_j\}$
$EFT_j := EST_j + d_j$;
end;

در سودوکد Forward، فاز شروع یا Initialisation این است که زودترین زمان شروع فعالیت یک و زودترین زمان پایان فعالیت یک را برابر صفر قرار دهید فعالیت یک چون مجازی است شروع و پایان آن صفر است . حالا چرا صفر چرا ده نیست ؟ چون زودترین زمان شروع می‌شود همین الان است. پس مبنا باید اول کار، لحظه صفر باشد. بعد به ترتیب این دستور For برای j اندیس فعالیت، برای فعالیت‌ها از ۲ تا n یعنی اول ۲، بعد ۳، بعد ۴، بعد تا n به خاطر همین Forward می‌گویند. شروع کنید به انجام این کار زودترین زمان شروع فعالیت J (یعنی ۲) ، ماکسیمم دیرترین زمان پایان i را بگیرید که i خودش پیش نیاز j است . مثلا اگر بخواهید از فعالیت j صحبت کنید فعالیت j تعدادی پیش‌نیاز دارید اگر شما بخواهید زودترین زمان شروع j را تعیین کنید باید در پیش‌نیازهای فوری باید ببینید کدامش Max ، EFT را دارد. ماکسیمم EFT پیش‌نیازها می شود EST



حالا که EST_j را حساب کردید $(EST_j + d_j)$ EST را با Duration همان فعالیت جمع کنید که نهایتاً EFT_jبدست می‌آید. بعد حلقه می‌باشد و این مسئله برای n فعالیت انجام می‌شود و فعالیت آخر را که انجام دادید این مسئله در واقع تمام شده است.

در محاسبات Backward که اسمش دقیقاً روی نام آن است یعنی از انتها به ابتدا محاسبات انجام شود . در این محاسبات Lateness start time و Lateness finish time محاسبه می‌شود. یعنی دیرترین زمان شروع و پایان فعالیت‌ها قرار است محاسبه گردد.

$\text{Initialisation} : \text{LFT}_n = \text{LST}_n = T;$

for $j: n - ۱$ **downto** ۱ **do**

begin

$\text{LFT}_j := \min\{\text{LST}_i | i \in S_j\};$

$\text{LST}_j := \text{LFT}_j - d_j ;$

end;

که برای محاسبه دیرترین زمان شروع و پایان برخلاف قبلی از آخر کار باید شروع کرد. دیرترین زمان پایان و دیرترین زمان شروع فعالیت n یعنی فعالیت آخر را چند است ؟ با T نشان داده شده است. $(\text{LST}_n = T)$

سوال : سه تا سوال بود a,b,c

a: یک پروژه چقدر طول می‌کشد ؟

b : زودترین زمان شروع فعالیت‌ها چه زمان است ؟

c : شناوری فعالیت‌ها چقدر است ؟

الان در محاسبات رفت به کدام یک از سوال‌های بالا جواب داده شد ؟

سوال یک جواب داده شده است. وقتی شما فعالیت n را انجام می‌دهید EST_n و برای EFT_n بدست می‌آید. (محاسبات رفت)

EST_n : زودترین زمان پایان فعالیت n

n : یعنی فعالیت آخر

فعالیت آخر همان پایان پروژه است. نتیجه اینکه سوال یک در محاسبات Forward جواب داده شده است. محاسبات Forward به انتها رسید آخرین فعالیت آن زمانی براش بدست می‌آید همان make aspan پروژه است که در متن با T نشان داده شده است. پس اینکه محاسبات برگشت را با T شروع می‌کنیم T همان make aspan پروژه است.

برای فعالیت‌ها n-۱ ((downto) یعنی شمارش کاهشی به طور مثال ۹، ۸، ۷، ۶، تا یک) به صورت معکوس تا یک این کار را انجام دهید. این کار عکس کاری است که در Forward بوده است. یعنی Lateness finish time فعالیتی، می‌شود مینیمم Lateness start time پس‌نیازهایش $(\text{LFT}_j := \min\{\text{LST}_i | i \in S_j\})$.

در واقع شما با پیش‌نیازها کاری ندارید. برای عدد i، چند تا پس نیاز دارد. مثلاً ۳ تا ، ۴ تا ، ۱۰ تا پس نیاز فوری دارد این پس نیازها Lateness start time ها را ببنید چند است ؟ آن Lateness start time ایی که کمتر است می‌شود Lateness finish time.

این Lateness finish time را منهای Durationکنید که Lateness start time بدست می‌آید.

$$LST_j := LFT_j - d_j$$

برای محاسبات Backward به پس نیازها ، نیاز داریم. برای فعالیت‌ها به صورت معکوس انجام می‌دهید تا به یک برسید که نهایتاً محاسبات Backward به انتها می‌رسد.

الان تا این لحظه چند تا عدد حساب کردیم ؟ ۴ تا ، دو تا در برگشت و دو تا در رفت. (LST,LFT در برگشت و EST,EFT در رفت) پس ما برای هر فعالیتی ۴ تا عدد داریم.

به سادگی می‌شود بررسی کرد که Overall time complexity محاسبات Forward و Backward هست $O(n^2)$.

$O(n^2)$ یعنی چه ؟

در پارت ۲ مسائل به دو دسته Pو NP تقسیم می‌شدند که P آسان بود و NP سخت بود. ویژگی P: اگر برای مسئله الگوریتمی وجود داشته باشد که Time complexity, (O) چند جمله‌ای باشد آسان است.

مسئله Forward و Backward ، الگوریتمش $O(n^2)$ است . n^2 دو جمله‌ای است یعنی آسان است یعنی مسئله Forward و Backward ، مسئله‌ای آسان است مسئله NP نیست مسئله P است.

حال اگر n^3 بود یا اگر n^{1000} چه اتفاقی می‌افتاد ؟ فرقی نمی‌کرد n به توان هر عددی یک چند جمله‌ای است مهم نیست پس مسئله‌ای که الگوریتمش تابع پیچیدگی n به توان یک عدد باشد چند جمله‌ای است آسان است.

جمله آخر خلاصه‌اش به صورت زیر است : مسئله‌اش در کلاس P است آسان است.

اما اگر مثلاً به فرض $O(2^n)$ بود یعنی n در توان بود تابع نمایی می‌گردید و آن وقت مسئله NP می‌شد.

در پاسخ به سوال سوم : یک فعالیت چقدر می‌تواند به تأخیر بیافتد تا پروژه به تأخیر نیافتد اصطلاحی به اسم شناوری بیان می‌گردد.

شناوری Float (slack)

سه نوع شناوری در پروژه‌ها تعریف می‌گردد.

۱- شناوری کل Total float (slack) که به صورت اختصاری با TS_j نشان داده می‌شود.

۲- شناوری آزاد Free float (slack) که به صورت اختصاری با FS_j نشان داده می‌شود.

۳- شناوری امن Safety float (slack) که به صورت اختصاری با SS_j نشان داده می‌شود.

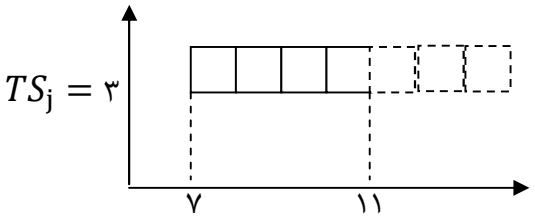
Total float (slack):

نحوه محاسبه به صورت ذیل می‌باشد:

$$TS_j = LST_j - EST_j = LFT_j - EFT_j$$

دیرترین زمان منهای زودترین زمان شروع یا دیرترین زمان پایان منهای زودترین زمان پایان
برای محاسبات شناوری حتماً باید محاسبات رفت و برگشت را انجام شده باشد.

مفهوم شناوری کل :



تمدید شدن delay prolong

بدون تأثیر در ختم پروژه

سوال : اگر شناوری کل ۰، ۳ و یا ۷ باشد یعنی چی ؟ مثلاً شناوری مربوط به یک فعالیت مثلاً j ، سه روز است $TS_j = ۳$ یعنی چه ؟

یعنی اینکه این فعالیت که مثلاً ۴ روز است و امروز در تاریخ ۷ قرار است شروع شود و در یازدهم به اتمام برسد می‌تواند به جای تاریخ ۷ در تاریخ‌های ۸، یا ۹ و یا ۱۰ هم اتفاق بیافتد که تأثیری در کل پروژه ندارد. پس شروعش تا سه روز می‌تواند عقب بیافتد.

پس یا یک فعالیت به عقب می‌افتد Delay (تأخیر) دارد به جای اینکه ۷ ام شروع بشود، ۸ ام شروع شده و یا این که به موقع شروع شده است ولی بیش از آن حدی که باید طول کشیده است Prolong شدن یا تمدید شدن فعالیت.

خلاصه : شناوری کل مفهوم‌اش این است که یک فعالیت می‌تواند با چند روز تأخیر شروع شود یا تمدید شود بدون اینکه تأثیری در زمان ختم پروژه داشته باشد.

نکته : می‌تواند هم دیر شروع شود و هم طولانی گردد.

اگر $TS_j = ۰$ باشد یعنی این فعالیت Critical (بحرانی) است.

بحرانی است یعنی امکان تأخیر یا تمدید ندارد و اگر بخواهد تأخیر یا تمدید داشته باشد باعث دیرکرد پروژه می‌شود.

حالا اگر زنجیره از این فعالیت‌ها را در پروژه پشت سر هم از ۱ تا n در نظر بگیرید به اصطلاح مسیر بحرانی می‌گوییم.

مسیر بحرانی عملاً طولانی ترین مسیر پروژه است که در واقع همان T تعیین می‌کند. مسیر بحرانی ، جمع duration هاش، همون ، این‌ها می‌شه واژه‌هایی که شما با آن آشنا هستید.

شناسایی مسیر بحرانی دارای چه حسنی است ؟

اگر بخواهیم پروژه را فشرده کنیم یعنی زمان پروژه را کوتاه کنیم باید مسیر بحرانی را کوتاه کنیم.

مفهوم کنترل پروژه : برای اینکه پروژه به موقع تمام شود روی بعضی از فعالیت‌ها باید تمرکز کنید. در واقع به پیمانکار فشار بیاورید، منابع‌اش را به موقع تأمین کنید که کار را On time تمام کند. کدام فعالیت‌ها ؟ فعالیت‌هایی که بحرانی هستند. چون آنها تعیین کننده هستند. در اولویت بعدی یعنی بعد از فعالیت های بحرانی، فعالیت‌هایی که شناوری کمی دارند حائز اهیمت هستند. مثل فعالیت‌هایی که شناوری یک روزه دارند. به طور مثال ممکن است به دلیل

بارندگی در آن روز فعالیت متوقف گردد. نتیجه اینکه از شناوری یک روزه استفاده کرده و فردا آن فعالیت هم جزء فعالیت‌های است که بحرانی هستند به حساب می‌آید. بحرانی بودن این نیست که شما یک لیست از فعالیت داشته باشید و تا آخر هم همان لیست باشد لیست بحرانی یک لیست پویایی است که دائماً می‌تواند استفاده شود.

سوال : آیا فعالیت‌هایی که بحرانی بودند می‌تواند از حالت بحرانی بودن خارج شود.

بله ممکن است با تزریق منابع زیاد، فشرده کردن و سرعت دادن به کار بتوانید محدوده‌ای که بحرانی بوده است را از وضعیت بحرانی خارج سازید.

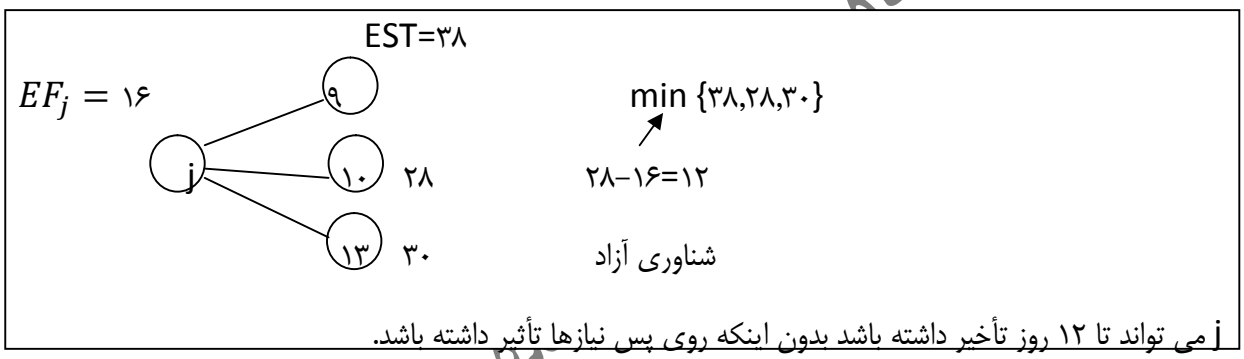
شناوری آزاد Free Float(slack)

شناوری آزاد را که با FS_j نشان می‌دهند به صورت زیر محاسبه می‌گردد:

$$\mathrm{FS_j} = min_{i \in S_j}\{EST_j\} - EFT_j$$

مینیم earliest start time پس نیاز های یک فعالیت منهای earliest finish time همان فعالیت.

مثال :



j می تواند تا ۱۲ روز تأخیر داشته باشد بدون اینکه روی پس نیازها تأثیر داشته باشد.

فعالیت j دارای سه پس نیاز است . فعالیت ۹، ۱۰، ۱۳ پس نیاز j است که دارای مقدارهای زیر می‌باشد.

$EST_۹ = ۳۷$
$EST_{۱۰} = ۲۸$
$EST_{۱۳} = ۳۱$
$EFT_j = ۱۶$

بین پس نیازهای آنهایی که مینیم EST دارد فعالیت ده می‌باشد یعنی ۲۸ پس

$۲۸ - ۶ = ۱۲$ ⟵ FF

این فعالیت ۱۲ روز شناوری آزاد دارد. یعنی این فعالیت می‌تواند تا ۱۲ روز تأخیر داشته باشد بدون اینکه رو پس نیازهایش اثر بگذارد.

مقایسه شناوری کل با شناوری آزاد :

در شناوری کل چقدر می‌تواند تأخیر داشته باشد تا پروژه عقب نیافتد.

شناوری آزاد چقدر فعالیت می‌تواند تأخیر داشته باشد که فعالیت‌های پس نیازهایش جابجا نشوند و یا تغییر نکنند.

در محاسبه شناوری آزاد اگر دقت کنید براساس زودترین زمان‌ها EST,EFT در محاسبه لازم است و LST,LFT لازم نیست.

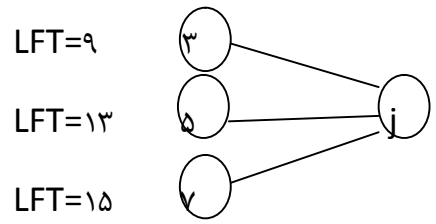
شناوری اطمینان Safety Float (Slack)

یک فعالیت می‌تواند شناوری اطمینانش از فرمول زیر محاسبه گردد.

$$SS_j = LST_j - Max_{i \in P_j}\{LFT_i\}$$

LST این فعالیت منهای ماکسیمم LFT پیش نیازهایش یعنی موضوع با پیش‌نیازهایش مرتبط است نه با پس‌نیازها.

به طور مثال : اگر فعالیت J سه پیش‌نیاز دارد فعالیت ۳، ۵، ۷ پیش نیاز هستند.



$$SS_j = ۲۱ - ۱۵ = ۶$$

فعالیت j می‌تواند ۶ روز زودتر شروع شود یا تمدید شود که روی پیش نیازها تأثیر نگذارد.

بین فعالیت‌ها آنی که ماکسیمم پیش نیاز LFT دارد فعالیت ۷ یعنی ۱۵ می‌باشد.

$LFT_۳ = ۹$
$LFT_۵ = ۱۳$ } max
$LFT_۷ = ۱۵$
$LST_j = ۲۱$
$SS_j = ۲۱ - ۱۵ = ۶$

یک فعالیت می‌تواند چقدر تمدید شود ، تمدیدش از این سمت است نمی‌توانید دیرتر شروع کنید، چون محاسبه‌اش می‌بینید براساس LST در دیرترین زمان دارد شروع میشود. بنابراین دیرتر نمی‌تواند شروع شود. چقدر اگر قرار است تمدید شود از کدام قسمت تمدید می شود ؟ مثلاً فرض کنید ساعت این کلاس از ساعت ۱۵ تا ۱۷:۱۵ است و ۱۷:۱۵ هم دانشگاه تعطیل می‌شود پس اگر بخواهیم کلاس را در مدت زمان بیشتری تشکیل دهیم از شروع‌اش باید زودتر شروع کنیم. در واقع تمدیدش از شروع می‌باشد و چقدر می‌تواند از این سمت شروع شود که به کلاس‌های قبلی تداخل نکند. یک فعالیت چقدر می‌تواند تمدید شود که روی پیش‌نیازهایش اثر نگذارد. ولی روی پس نیازها نباید صحبت کنید. چون محاسبه در دیرترین زمان ممکن دارد حساب می‌شود بنابراین با پس‌نیازها امکان تأخیر روش مطرح نیست.

سوال : برای هر فعالیتی چند تا عدد محاسبه شد ؟

هفت عدد محاسبه کردیم ۲ تا در محاسبات رفت، ۲ تا در محاسبات برگشت، ۳ تا هم در شناوری . الان برای هر فعالیت ۷ تا عدد داریم که هر کدام معنای خاص خودش را دارد. EST,EFT,LST,LFT و سه تا هم شناوری TS,FS,SS .

مثال :

۸ تا فعالیت در نظر گرفته شده است. فعالیت ۱ و ۸ مجازی‌اند بنابراین ۶ تا فعالیت داریم .

$\mathrm{EST}_1 = 0 \qquad EST + d = EFT \qquad 0 + 0 = 0$

فعالیت ۲ $\longrightarrow \mathrm{EFT}_1 = \mathrm{EST}_2 = 0 \longrightarrow \mathrm{EST}_2 + \mathrm{d} = \mathrm{EFT} \longrightarrow 0 + 2 = 2$

فعالیت ۵ $\longrightarrow \mathrm{EFT}_2 = \mathrm{EST}_5 = 2 \longrightarrow 2 + \mathrm{d} = 2 + 1 = 3$

فعالیت ۶ $\longrightarrow$ سه تا پیش نیاز (۳،۱،۴) دارد max=۴ $\longrightarrow \mathrm{EFT}_4 + \mathrm{d}_6 = 4+2=6$

$\mathrm{EFT}_5 = 3 \quad \mathrm{EFT}_3 = 1 \quad \mathrm{EFT}_4 = 4$

فعالیت ۸ $\longrightarrow$ دو تا پیش نیاز (۷و۶) دارد max=۱۱ $\longrightarrow \mathrm{EFT}_7 + \mathrm{d}_8 = 11 + 0 = 11$

$\mathrm{EFT}_6 = 6 \quad \mathrm{EFT}_7 = 11$

۱۱ همان Tاست

این پروژه ۱۱ روز طول می‌کشد محاسبات *Backward* همین ۱۱ به عنوان *LST* و *LFT* همین فعالیت نوشته می‌شود بعد باید برخلاف *Forward* مینیمم بگیریم و منهای *Duration* کنیم.

فعالیت ۶ $\longrightarrow$ پس نیازش ۸ است $\longrightarrow LST_8 = 11 \longrightarrow LST_8 = LFT_6 = 11 \longrightarrow LFT_6 - d_6 = 11 - 2 = 9$

فعالیت ۵ $\longrightarrow LST_6 = 9 \longrightarrow LST_6 = LFT_5 = 9 \longrightarrow LFT_5 - d_5 = 9 - 1 = 8$

فعالیت ۴ $\longrightarrow$ دو پس نیاز (۶ و ۷) دارد. $\longrightarrow min\ LST : LST_7 = 4 \longrightarrow LFT_4 - d_4 = 4 - 4 = 0$

$LST_6 = 9 \quad LST_7 = 4$

نکته : فعالیتی که چند پس نیاز داشته باشد از بین آنها مینیم آنها را محاسبات وارد می‌کنیم.

این مسیری که پررنگ تر از بقیه است نشان دهنده مسیر بحرانی است چون شناوری فعالیت‌های این مسیر صفر است.

نحوه محاسبه شناوری کل *LS-ES=LF-EF* برابر صفر است یعنی ۰=۱۱-۱۱=۴-۴

به این شکل نمودار گانت می‌گویند. نمودار گانت بیان تصویری از زمانبندی است در واقع برای نمایش گرافیکی استفاده می‌شود محور افقی به معنی زمان، محور عمودی فعلاً مفهومی ندارد چون منبع نداریم ولی بعداً محور عمودی به عنوان منبع استفاده میشود. بعداً که محدودیت منابع داشته باشیم محور عمودی به معنی منبع است الان فعالیت ۳ بالاتر است و فعالیت ۵ پایین‌تر ولی هیچ مفهومی ندارد چون محور عمودی در این جا تعریف خاصی ندارد.

تفاوت در این دو گانت چارت :

ESS احتمالاً مخفف *Earliest Start scheduling* (زمانبندی در زودترین زمان)

LSS احتمالاً مخفف *Lateness Start Scheduling* (زمانبندی در دیرترین زمان)

در شکل قبلی که نمودار داشتیم محاسبات هفت‌گانه‌اش در این جدول می‌باشد. همه فعالیت‌هایی که در این جا می‌بینید *ES,EF,LS,LF,TS,FS,SS* آن محاسبه گردیده است.

کدام‌ها بحرانی هستند ؟ آنهایی که شناوری کل آن صفر باشد. فعالیت‌های ۱،۴،۷،۸ بحرانی هستند. در واقع *LF,EF* و*LS,ES* آنها باهم برابر است.

خوب که به *ESS*دقت کنید فعالیت‌ها کی شروع شده‌اند. در گانت سمت راست شروع فعالیت‌ها نشان داده شده است. مثلاً

فعالیت یک که مجازی است هیچی

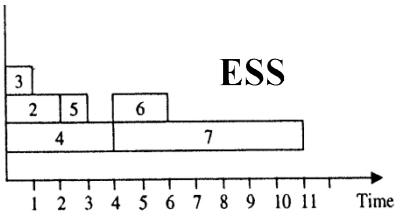
فعالیت‌های ۳،۴،۲ ⟵ $EST_۲ = ۰ \quad EST_۳ = ۰ \quad EST_۴ = ۰$

فعالیت‌های ۵ ⟵ $EST_۵ = ۲$

فعالیت ۶ ⟵ $EST_۶ = ۴$

فعالیت ۷ ⟵ $EST_۷ = ۴$

فعالیت ۸ (مجازی) ⟵ $EST_۸ = ۱۱$



اما در قسمت *LSS* شروع فعالیت‌ها از

فعالیت ۴ ⟵ $LST_۴ = ۰$

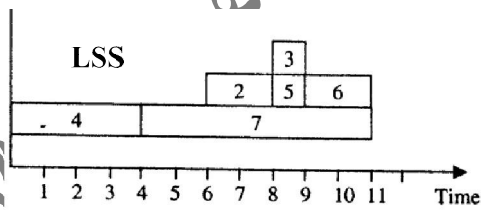
فعالیت ۲ ⟵ $LST_۲ = ۶$

فعالیت ۳ ⟵ $LST_۳ = ۸$

فعالیت ۷ ⟵ $LST_۷ = ۴$

فعالیت ۵ ⟵ $LST_۵ = ۸$

فعالیت ۶ ⟵ $LST_۶ = ۹$



دو گانت را با هم مقایسه کنید :

فعالیت ۴ و فعالیت ۷ هیچ تفاوتی با هم ندارند چون بحرانی هستند فعالیت‌هایی که بحرانی هستند در دو گانت *Earliest start scheduling* و *Lateness start scheduling* ثابت هستند. پس اگر جدول پایین را نداشته باشید از روی گانت هم می‌توانید مشخص کنید که کدام‌ها بحرانی هستند. آنهایی که در *ESS* و *LSS* جایشان ثابت است.

فعالیت ۵ را در دو گانت مقایسه کنید:

فعالیت ۵ در *ESS* چه زمانی شروع شده است ؟ شروع آن ۲ می‌باشد.

فعالیت ۵ در *LSS* چه زمانی شروع شده است ؟ شروع آن ۸ می‌باشد.

نتیجه : شش روز شناوری دارد.

اگر جدول پایین را به طور کل پاک کنید می‌توانید از روی گانت مجدداً آن را پر کنید.

شناوری آزاد از روی گانت *ESS* مشخص می‌شود. مثلاً

فعالیت ۵ پس نیازش ۶ می‌باشد در گانت بین ۵ و ۶ چقدر فضای خالی است ؟ یک روز پس شناوری آزاد آن یک روز می‌باشد. $(FS_۵ = ۱)$

فعالیت ۲ پس نیازش ۵ می‌باشد در گانت بین ۲ و ۵ چقدر فضای خالی است ؟ هیچ فاصله‌ای ندارد پس شناوری آزاد ندارد. $(FS_۲ = ۰)$

شناوری اطمینان از گانت *LSS* مشخص میشود و با پیش نیاز سنجیده می‌شود.

فعالیت ۵ پیش نیازش ۲ می‌باشد هیچ فضایی بین ۲ و ۵ نیست پس شناوری اطمینان صفر است $(SS_۵ = ۰)$

فعالیت ۶ در گانت *ESS* شروع آن ۴ می‌باشد و در گانت *LSS* شروع آن ۹ می‌باشد پس اختلاف آن ۵ روز است که نشان دهنده شناوری کل است است پس بحرانی نیست $(TS_۶ = ۵)$

شناوری آزاد از گانت *ESS* بود پس نیازش فعالیت ۸ می‌باشد که مجازی است و در گانت مشخص نیست. یعنی یک نقطه است در ۱۱ پس ۵=۶-۱۱ پس شناوری آزاد فعالیت ۶ ، پنج روز است. $(FS_۶ = ۵)$

شناوری اطمینان از گانت *LSS* و با پیش نیازش سنجیده می‌شود. فعالیت ۶ دارای سه پیش نیاز است فعالیت ۴،۳،۵ می‌باشد. با فعالیت ۴ به مدت ۵ روز، با فعالیت ۳ صفر می‌باشد. وقتی با یکی صفر است آن یکی ها مهم نیست یعنی مینیمم فاصله صفر است پس شناوری اطمینان صفر است $(SS_۶ = ۰)$

سوال : دو زمانبندی بدست آوردیم یکی از فعالیت‌ها در *Earliest start scheduling* و دیگری در *Lateness start scheduling* . کدام زمانبندی بهینه است ؟ *ESS* یا *LSS*؟

پاسخ : بستگی به تابع هدف دارد.

در جلسه قبل تقسیم‌بندی γ وβ و α را داشتید. γ تابع هدف است.

شما اصلاً به دنبال چه چیزی هستید ؟ یعنی هدفتان از زمانبندی فعالیت‌ها چه چیزی می‌باشد ؟ مینیمم کردن زمان، مینیمم کردن هزینه، ماکسیمم کردن سود، بهتر کردن کیفیت ، کم کردن دوباره کاری‌ها ، مینیمم کردن دیر کرد و یا تسطیح منبع است و صدها هدف دیگر پس معیار هدف تعیین کننده است که کدام بهتر است.

در فیلد γ اهداف به دو دسته منظم و نامنظم تقسیم می‌شد.

اهدف منظم : هر هدفی که در راستای زود تمام شدن کارها و پروژه باشد منظم است.

اگر تابع هدف منظم باشد *ESS* بهتر است چون *ESS* در زودترین زمان فعالیت‌ها را انجام می‌دهد.

نکته : اگر معیار هدف منظم باشد زمانبندی *ESS* بهینه است.

سوال : نکته بالا آیا به این معنی است که زمانبندی که هدف‌اش نامنظم باشد *LSS* بهینه است ؟ خیر، عکس آن درست نیست.

شبکه‌های *AOA*

همان سه سوال مطرح است :

- پروژه چند روز طول می‌کشد ؟
- زودترین زمانی که می‌شود شروع کرد چه زمانی است ؟
- چقدر فعالیت می‌تواند تأخیر داشته باشد که یک پروژه را به تأخیر نیافتد ؟

فرمت شبکه به صورت *Activity ARC* یا برداری است. تفاوت این شبکه *AOA* در این است که گره‌ها به معنی مایلستون هستند و بردارها به معنی فعالیت می‌باشد. پس موضوع با شبکه *AON* فرق کرده است در نتیجه محاسبات رفت و برگشت نیز در این شبکه متفاوت است در شبکه *AON* چهار عدد محاسبه شد ولی در اینجا باید دو عدد محاسبه شود که با نمادهای زیر مشخص می شود:

$t_j(E)$: زودترین زمان تحقق مایلستون *(earliest realization time)*

$t_j(L)$: دیرترین زمان تحقق مایلستون *(latest realization time)*

مایلستون شروع و پایان ندارد. حادثه‌ای است اتفاق افتادنی است.

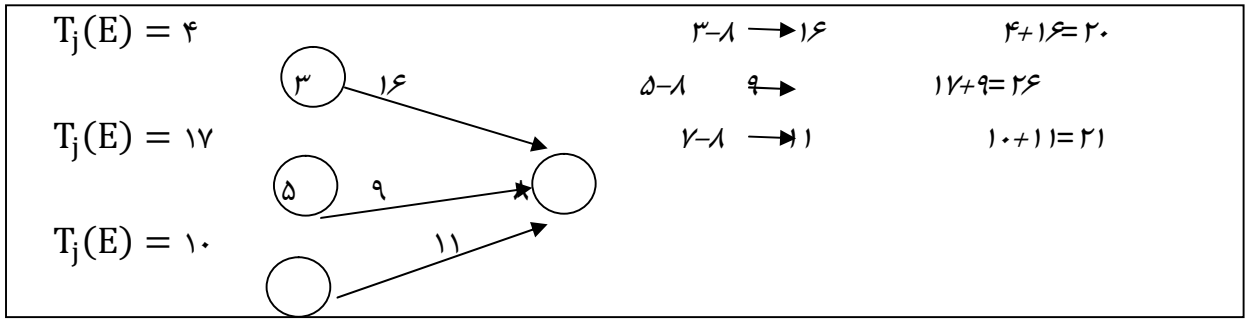
پس به واژه‌ها باید دقت کرد چون صبحت از *Finish , start* نیست صحبت از تحقق یافتن است.

سودوکد *forward* و *backward* شبکه‌های *AOA* است اول *Forward* را باید انجام بدهید تا بتوانید *backward* را محاسبه کنید. در *Forward* از گره یک (مایلستون یک) شروع میشود. زودترین زمان اتفاق افتادن مایلستون یک، همین الان یعنی صفر است. بعد به ترتیب برای مایلستون‌های ۲ تا n به ترتیب کار زیر انجام گردد.

$$t_1(E) = 0$$
$$\textbf{for } j := 2 \textbf{ to } n \textbf{ do}$$
$$t_j(E) = \max_{i \in P_j} \{t_i(E) + y_{ij}\}$$

y_{ij} : همان *Duration* است که در شبکه *AON* با d_i نشان داده می‌شد ولی چون در شبکه *AOA* فعالیت‌ها به صورت دو اندیس شناخته می‌شوند به این صورت نوشته شده است.

مثال :



در شکل بالا سه تا فعالیت داریم فعالیت *(۵,۸) , (۷,۸) , (۳,۸)*

برای اینکه زودترین زمان گره ۸ را حساب کنید فرض کنید زودترین زمان تحقق اینها را حساب شده است به صورت ذیل می‌باشد.

فعالیت *(۳,۸)* به مدت ۱۶ روز طول می‌کشد. ⟵ زودترین زمان تحقق آن ۴ می‌باشد $T_j(E)$

فعالیت *(۵,۸)* به مدت ۹ روز طول می‌کشد. ⟵ و زودترین زمان تحقق آن ۱۷ می‌باشد $T_j(E)$

فعالیت *(۷,۸)* به مدت ۱۱ روز طول می‌کشد. ⟵ زودترین زمان تحقق آن ۱۰ می‌باشد $T_j(E)$

۴+۱۶=۲۰
۱۷+۹=۲۶ } *max*=۲۶
۱۰+۱۱=۲۱

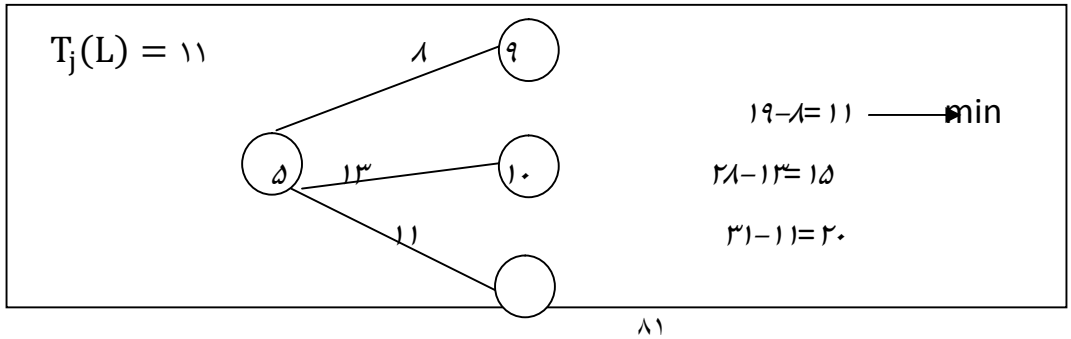
نتیجه : زودترین زمان تحقق فعالیت مایلستون ۸ ماکسیمم ۲۶ است یعنی زودترین زمانی که این مایلستون می تواند حادث شود روز ۲۶ می‌باشد.

در محاسبات رفت که در آخر انجام دادید یک T به دست می‌آید یعنی برای آخرین فعالیت جوابی که به دست می‌آید همان T است. T= تحقق آخرین فعالیت . در واقع پاسخ سوال یک همان T یا زمان پروژه است.

در محاسبات برگشت را با T باید انجام داد یعنی دیرترین زمان تحقق مایلستون آخر که همان Tاست یعنی دقیقاً عکس قبلی است. از شماره‌ها از آخر به اول تک به تک چک کنید برای گره‌ها از آخر به اول . دیرترین زمان هر مایلستون را به صورت زیر محاسبه کنید. مینیمم (این بار دیرترین زمان منهای *Duration* پس نیازها) و هر کدام که کمتر شد.

$$t_n(L) = t_n(E) = T$$
$$\textbf{for } j := n\text{-}1 \textbf{ downto } 1 \textbf{ do}$$
$$t_j(L) = \min_{k \in S_j} \{ t_k(L) - y_{jk} \}$$

مثلاًدر شکل :



برای فعالیت ۵، $T_j(L)$ را محاسبه کنید.

برای فعالیت ۵ دیرترین زمان

۱۹-۸=۱۱
۲۸-۱۳=۱۵ } min=۱۱
۳۱-۱۱=۲۰

مینیمم آنها عدد ۱۱ است پس دیرترین زمانی که می‌تواند اتفاق بیافتد روز یازدهم است درنتیجه در محاسبات *backward* با پس‌نیازها درگیر هستیم و در محاسبات *Forward* با پیش نیازها

در هر حال در محاسبات رفت و برگشت دو تفاوت وجود دارد:

- این محاسبات برای مایلستون‌ها می‌باشد.
- دو عدد هم بیشتر مطرح نیست. در حالی که در *AON* ها ۴ عدد محاسبه در رفت و برگشت می‌شد.

در شبکه‌های *AOA* هم الگوریتم *Forward* و *Backward* هم باز $O(n^2)$ است یعنی جزء مسائل آسان است.

پس شبکه *AOA* باشد یا *AON* محاسبه رفت و برگشت از نظر سختی فرقی نمی‌کند.

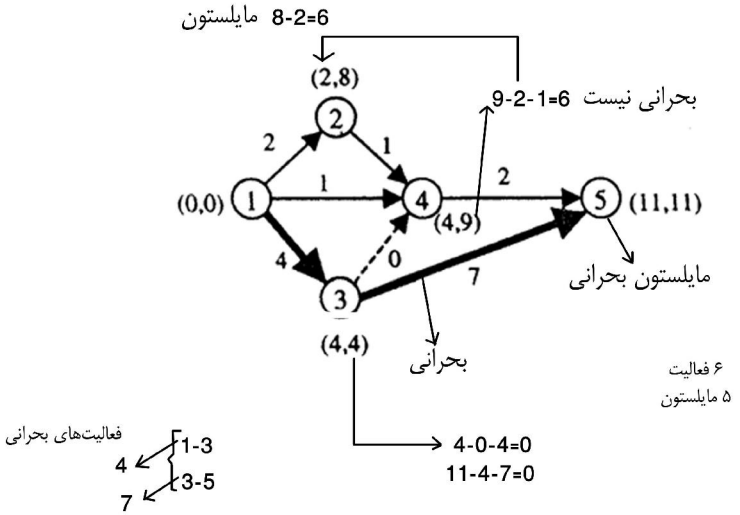
برای اینکه یک مسأله حل شده باشد این مثال همان مثال قبلی در شبکه *AON* است که در شبکه *AOA* حل می‌گردد.

چند تا فعالیت دارد ؟ بردارها فعالیت هستند باید تعداد بردارها را بشمارید. ۶ تا فعالیت و یکی مجازی که مهم نیست.

چند تا مایلستون دارد ؟ گره‌ها مایلستون هستند ۵ تا مایلستون دارد.

اعدادی که روی بردارها نوشته شده است نشان دهنده *Duration* است .

محاسبه رفت اولی صفر



فعالیت‌های بحرانی
4 ← 1-3
7 ← 3-5

۰+۴=۴

این سه تا پیش نیاز دارد ۲ با یک ، صفر با یک ، ۴ با صفر . ماکسیمم ۴ است. ۴ با ۲ ⟵ ۴ با ۷ ← ماکسیمم ۱۱

۱۱ منهای ۲ ← ۹ و ۹ منهای ۱۱ ⟵ ۸ و ۱۱ منهای ۷ ← ۹ و ۹ منهای صفر ← مینیمم ۴

۴ منهای ۴ ⟵ و ۹ منهای ۱ ⟵ و ۸ منهای ۲ مینیمم ⟵ صفر

نکته : مسیر پرنگ مسیر بحرانی است کدام فعالیت‌ها بحرانی هستند ؟ *(۱,۳)* و *(۳,۵)* بحرانی هستند باید به صورت برداری گفته شود.

در شبکه *AON* فعالیت ۴ و ۷ را می‌گفتیم بحرانی هستند ولی در شبکه *AOA* ظاهراً *(۱,۳)* همان فعالیت ۴ است و *(۳,۵)*همان فعالیت ۷ می‌باشد.

برای تشخیص بحرانی بودن باید شناوری کل صفر باشد.

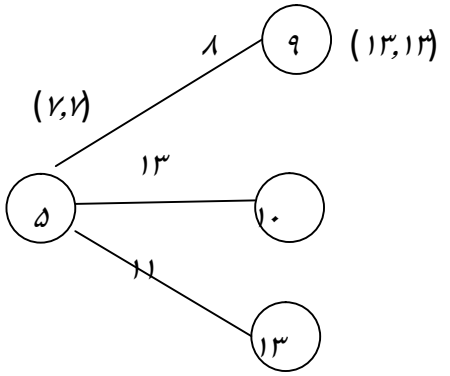
شناوری کل در شبکه‌های *AOA* محاسبه آن فرق می‌کند. مفهوم یکی است ولی فرمول‌ها را باید دقت کنید.

نحوه محاسبه شناوری کل در شبکه *AOA* *(Total Float)*

شناوری کل = دیرترین زمان وقوع مایلستون پایان منهای زودترین زمان مایلستون شروع منهای *Duration*

$$TS_{ij} = t_j(L) - t_i(E) - y_{ij}$$

به طور مثال : فعالیت *(۵,۹)* شناوری کل را به صورت فرمول بگید.



اگر فعالیت *i-j* بحرانی باشد حتماً گره *i* و گره *j* بحرانی است ولی عکس این موضوع لزوماً درست نیست.

میشه دیرترین زمان اتفاق افتادن این مایلستون پایان *(j)* منهای زودترین زمان مایلستون شروع *(i)* منهای *Duration*

وقتی میگوید *j-i* یعنی *j* انتهای بردار ، *i* ابتدای بردار است چون دو اندیس است فعالیت *ij* یعنی بردار از *i* به سمت *j* است.

شناوری آزاد و شناوری اطمینان فرمول‌های شبیه به هم دارد که تفاوت بسیار جزئی است.

شناوری آزاد *Free Flaot* $$FS_{ij} = t_j(E) - t_i(E) - y_{ij}$$

زودترین زمان وقوع مایلستون پایان *(j)* منهای زودترین زمان مایلستون شروع *(i)* منهای *duration*

شناوری اطمینان *Safety Flaot* $SS_{ij} = t_j(L) - t_i(L) - y_{ij}$

دیرترین زمان وقوع مایلستون پایان ***(j)*** منهای دیرترین زمان مایلستون شروع ***(i)*** منهای ***duration***

به هر حال سه عدد با هم فرق خواهد کرد فرمول‌ها خیلی به هم نزدیک است. فعالیت‌هایی که شناوری کل آنها صفر است بحرانی است.

در شکل مسیر پر رنگ نشان دهنده مسیر بحرانی است.

فعالیت ***(۱,۳)*** را شناوری کل را محاسبه کنید.

$TS_{۱۳} = ۴ - ۰ - ۴ = ۰$

دیرترین زودترین ***duration*** ⟹ بحرانی

$TS_{۳۵} = ۱۱ - ۴ - ۷ = ۰$

$TS_{۲۴} = ۹ - ۲ - ۱ = ۶$ ⟶ شناوری اش شش می‌باشد پس بحرانی نیست

گره ***node slack*** یا شناوری مایلستون در شبکه ***AOA***

در شبکه ***AON*** سه تا شناوری (کل ، آزاد ، اطمینان) داشت ولی در شبکه ***AOA*** جدا از این سه تا شناوری یک شناوری دیگری به نام ***node slack*** یا شناوری مایلستون داریم که با حروف اختصاری NS_j نشان داده می‌شود که از طریق فرمول زیر محاسبه می‌گردد.

$$NS_j = t_j(L) - t_j(E)$$

دیرترین منهای زودترین زمان همان مایلستون را ***Node slack*** یا شناوری مایلستون می‌گویند.

یعنی یک مایلستون هم می‌تواند بحرانی باشد در مثال قبلی

شناوری مایلستون (۴) چند روز است ۵=۴-۹ ، پنج روز است. یا شناوری مایلستون (۲) ، ۶=۲-۸ شش روز می‌باشد.

سه مایلستون دیگر بحرانی اند چون اختلاف زودترین و دیرترین آن صفر است.

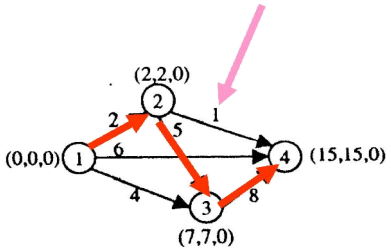
نکته :

مایلستون ۵ بحرانی است یعنی زودترین اش ۷ و دیرترینش هم ۷ می‌باشد. مایلستون ۹ هم زودترینش ۱۳ و دیرترین اش هم ۱۳ است پس هر دو مایلستون بحرانی اند و لزوماً فعالیت ***(۵,۹)*** بحرانی نیست.

مایلستون ***i*** و مایلستون ***j*** بحرانی باشند لزوماً فعالیت ***(i,j)*** بحرانی نیست ولی اگر فعالیت ***(i,j)*** بحرانی باشد حتماً (گره) مایلستون ***i*** و(گره) مایلستون ***j*** بحرانی اند. ولی عکس موضوع لزوماً صحیح نیست.

مثال :

در شکل ۶ تا فعالیت و ۴ تا مایلستون وجود دارد. محاسبات هم انجام شده است. پرانتز دارای سه عدد شامل زودترین، دیرترین و تفاضل آنها یعنی *node slack* می باشد.



سوال : کدام مایلستون‌ها بحرانی هستند ؟

همه مایلستون‌ها بحرانی اند ولی لزوماً همه فعالیت‌ها بحرانی نیستند. *(۴,۳,۲,۱)* بحرانی هستند.

مثلاً فعالیت *(۲,۴)* شناوری‌اش ۱۲ است. $TS_{۲۴} = ۱۵ - ۲ - ۱ = ۱۲$ درست است که مایلستون ۲ و مایلستون ۴ بحرانی هستند ولی فعالیت *(۲,۴)* بحرانی نیست.

به نظر تضاد دارد. چه طور ممکن است مایلستون اول و آخر بحرانی است ولی فعالیت بحرانی نیست، چه طور ممکن است این اتفاق بیافتد ؟

$TS_{۱۲} = ۲ - ۰ - ۲ = ۰$

$TS_{۲۳} = ۷ - ۲ - ۵ = ۰$

$TS_{۳۴} = ۱۵ - ۷ - ۸ = ۰$

هر سه فعالیت *(۳,۴) – (۲,۳) – (۱,۲)* بحرانی هستند و این مسیر بحرانی است و اگر دقت کنید می‌بینید که همه مایلستون‌ها در این مسیر هستند پس همه مایلستون‌ها بحرانی شده‌اند. پس اگر مایلستون ۲ و ۴ بحرانی شده‌اند نه به خاطر فعالیت *(۲,۴)* بلکه به خاطر اینکه در مسیر بحرانی بودند. پس بحرانی بودن را از جای دیگر نشأت گرفته است. (مسیر لزوماً فعالیت ۲ و ۴ را شامل نمی‌شود)

در جلسه قبل مثالی داده شد که خواسته بود که *AON* را به *AOA* تبدیل کنید.

شبکه *AON*راحت تر و *AOA* سخت تر بود. جواب شبکه *AON* ، *unique* بود یعنی تمام جواب‌های شما یکسان و عین هم هستند ولی در شبکه *AOA* جواب ها متنوع بود .

سوال : در *AON* که مشکلی نیست چون یک جواب و یک شبکه بیشتر وجود ندارد. اما در *AOA* که شبکه‌ها با هم فرق می کند محاسبات، شناوری‌ها عین هم می‌شود ؟ اگر مثل هم نشود که غلط است ! چون یک مسأله که بیشتر نیست پس چرا جواب‌هاش فرق می‌کند ؟ یک مسأله است فقط نحوه نمایش آن فرق می‌کند. منطقاً نباید جوابش فرق کند. هر چقدر ظاهر شبکه‌ها با هم متفاوت است تعداد مایلستون‌ها و مجازی‌ها متفاوت است مثلاًاگر بگوییم شناوری فعالیت *F* چقدر است باید همه یک عدد را بگویند.

این مثال چه نوع شبکه‌ای است ؟

این شبکه *AON* است ؟ چرا

AO را محکم می‌گید آخرش را آروم *AON* چرا ؟ چون اون بالا نوشته است.

چون *duration*ها روی گره‌ها نوشته است چون اگر برداری باشد باید *duration* روی بردارها باشد.

گره اول که البته فعالیت مجازی است آخری هم مجازی است. عملاً ۱۰ تا فعالیت دارد و ۲ تا هم مجازی مسیر پررنگ نشان دهنده مسیر بحرانی است. یعنی این پروژه چند روز طول می‌کشد ؟ ***duration*** ها را با هم جمع بزنید می‌شود ۳۰ روز. در مسیر بحرانی فعالیت ***C , i*** بحرانی اند. ۰+۲۰+۱۰+۰=۳۰

کدام‌ها شناوری صفر است ؟ ***C,i*** ⟵ ***TS***

حال اگر شبکه ***AOA*** مثال بالا را رسم کنید جواب‌های شما از لحاظ ظاهر با یکدیگر فرق می‌کند به خاطر اینکه از تعداد مجازی‌ها وتعداد مایلستون‌های متفاوتی برای رسم شبکه ***AOA*** استفاده شده است. ولی پیش‌نیازها به درستی رسم شده است. دو نمونه از شبکه ***AOA*** را که با پیش‌نیازی درست رسم شده است را انتخاب می‌کنیم. همان سوال، همان شبکه است فقط نسخه ***AOA*** است یکی از این شبکه‌ها با ۹ تا مایلستون و ۶ تا مجازی رسم شده است و دیگری با ۱۱ تا مایلستون و مجازی بیشتری رسم شده است. و هر دو شبکه از لحاظ پیش‌نیازی درست است.

محاسبات شناوری اینها در سه ستون داریم و نتایج را بررسی می‌کنیم.

به طور واضح مشاهده می‌کنید ***Total slack*** ها با هم یکسان است. جواب‌هایی که در شبکه ***AON***بود در این دو شبکه ***AOA*** هم یکی است.

اشکال اساسی : ***Free slack*** ها تضاد وجود دارد ؟

Free slack برای سه تا از فعالیت‌ها همخوانی ندارد. فعالیت ***F,E,D*** شناوری آزاد ۵,۱۲,۳ باید باشد. اما در شبکه ***AOA*** رسم شده به جای ۵,۱۲,۳ صفر و به جای ۱۲,۳ صفر است. ***(۰,۰,۰)*** و ***(۵,۰,۰)*** جواب‌ها تضاد دارد. این تضاد در شناوری اطمینان هم می‌باشد در حالی که سه تا فعالیت آخر را چک می‌کنید ۸ و ۰ و ۴ شناوری اطمینان است برای این شبکه ***AOA*** جواب‌های متفاوتی بدست آمده است یک ابهام ایجاد می‌کند که این جواب‌ها فاقد اعتبار است. ***(۰,۰,۰)*** و ***(۴,۰,۰)***

چه کار می‌توان کرد ؟

اگر در مسئله تنها با شناوری کل کار می‌کنید و شناوری آزاد و اطمینان مطرح نیست که مشکلی نمی‌باشد چون ***Total slack*** ها ثابت هستند. ابهامی ندارد. اما اگر با آن دو تا شناوری ***Free slack*** و ***Safety slack*** سرکار دارید دو راه کار موجود است :

۱- اصلاً از شبکه ***AOA*** استفاده نکنید چون تضاد دارد. ولی بعضاً اینکار را نمی‌توان انجام داد چون گاهاً به مایلستون‌ها نیاز است. این راه پاک کردن صورت مسئله است.

۲- ریشه‌یابی کنیم و ببینیم این اختلاف از کجا است.

۳- مقصر رفتار وابسته به جواب ها در شبکه‌های ***AOA*** ، ***in-dummy nodes*** (گره‌هایی که تمام بردارهای وارد شونده‌اش مجازی هستند) و ***Out-dummy nodes*** (گره‌هایی که تمام بردارهای خارج شونده‌اش مجازی هستند) می‌باشد.

پس ***AOA*** را استفاده کنید که ***in-dummy*** و ***out-dummy*** نداشته باشد. صحبت از تعداد مجازی کم یا زیاد نیست .

در دو مثال قبلی از شبکه‌های *AOA* در یکی ۳ و ۴ و در دیگری ۳،۴، ۷، ۵ ***out dummy*** هستند.

مثلاً ۵ ⟵ همه بردارهایی که از ۵ بیرون آمده است مجازی است

۳ ⟵ همه برداریی که از ۳ بیرون آمده است مجازی اند

یا برعکس

۸ ⟵ دو بردار ***in dummy*** به آن وارد شده است برای ۹ و ۱۰ هم دو بردار ***in dummy*** وجود دارد.

دراین مثال به ۸ یک بردار مجازی وارد شده است. ***in dummy***

پس راه دوم از شبکه *AOA* ایی استفاده کنیم که کلاً ***in-dummy , out dummy*** نداشته باشد.

نکته : بعضاً نمی‌توانید شبکه‌ای*AOA* را رسم کنید که ***in-dummy***و ***out-dummy*** نداشته باشد. اگر بتوان رسم کرد دیگر تضاد نداریم این تضاد پیش نمی‌آید. بعضاً اجباری است حتی اگر یکی از ***in-dummy , out dummy*** را هم داشته باشد آن تضاد در شناوری ها را ایجاد می‌کند. حذف مطلق ***dummy*** ها مفید است. نمی‌توان رسم کرد حتی یکی هم اگر باشد تضاد داریم.

– یک فعالیت مثل *v* حتماً ختم خواهد شد به یک گره ***out-dummy*** اگر $P^*(W) \neq \cap_{y\in S(V)} P^*(y)$ اتفاق بیافتد. منظور این جمله این ایست که چه زمان شما به یک ***out-dummy*** نیاز دارید.

کافی است همین فرمول را بفهمید که این دو فرمول درک یکسانی دارد هر جا که *P* است شده *S* و هر جا *S* است شده *P*

اول از همه ***S(V)*** را پیدا کنید. ***S(V)*** یعنی پس نیازهای مستقیم *V* که اسمش را *W* گذاشته پس *W* پس نیاز *V* است برای این *W*، P^* همه پیش‌نیازهایش را لیست کنید یعنی $P^*(W)$ یعنی مجموعه همه پیش‌نیازهای ***(W)*** که خود *W* پس نیاز *V* است. با این ترتیب سمت چپ نامساوزی ساخته شد.

سمت راست ***S(V)*** پس نیازهای *V* را پیدا کنید که اسم آن را *y*گذاشته است برای این پس نیازها همه پیش‌نیازها را پیدا کنید و بین آنها اشتراک بگیرید.

حال اگر این دو تا یعنی عبارت سمت راست و چپ با هم برابر نباشند به این معنی است که شما حتماً به یک ***Out-dummy*** نیاز دارید اما اگر مساوی باشند یعنی نیاز ندارید.

– یک فعالیت مثل *v* شروع می‌شود از یک گره ***in-dummy*** اگر $S^*(x) \neq \cap_{x\in P(V)} S^*(a)$ اتفاق بیافتد. منظور این جمله این ایست که چه زمان شما به یک ***in-dummy*** نیاز دارید.

مثال :

در مورد فعالیت *e* صحبت می شود.

پس نیازهای *e* کدام است $S(e) = \{g, h, i, j\}$

حالا پیش‌نیازهای تک تک آنها را لیست کنید.

$$P^*(g)=\{a, b, d, e\}$$

$$P^*(h)=\{b, c, d, e, f\}$$

$$P^*(i)=\{b, c, d, e, f\}$$

$$P^*(j)=\{b, c, e\}$$

حال اشتراک پیش‌نیازهای بالا را بنویسید.

$$P^*(g) \cap P^*(h) \cap P^*(i) \cap P^*(j)=\{b, e\}$$

کدام این P^* ها با آن اشتراک برابر است ؟ کدام این ۴ تا با بالایی برابر است ؟ هیچ کدام برابر نیست. نتیجه فعالیت e باید به یک *out-dummy* ختم شود اگر در این دو مثال دقت کنید بردار e کجا آمده است به ۳ و ۳ یک گره‌ای است که دو تا *out-dummy* دارد.

این در دو مورد شبکه *AOA* بود بقیه جواب‌ها چه می‌شود ؟ همه به همین صورت است جواب‌های درست برای این شبکه بدون استثنا فعالیت e را به *out-dummy* می‌رساند علت این است که اگر اینها هیچکدام با اشتراک برابر نباشند بدون شک *out-dummy* می‌خواهد. سعی کردن برای حذف *Out-dummy* و *in-dummy* شاید ممکن نباشد.

برای رفع این مشکل آقای المغربی و کامبوروسکی روش و الگوریتمی برای محاسبات شبکه‌های که *in-dummy , out-dummy* دارند، پیشنهاد کردند.

سودوکد محاسبات رفت و برگشت است و زمانی که شبکه *AOA* است و دارای *in-dummy , out-dummy* است استفاده می‌شود. و این سودوکد زمانی استفاده می‌شود که شما با شناوری آزاد و اطمینان سرکار دارید.

این روش یه جورایی اصلاح همان روش رفت و برگشت است در رفت و برگشت شما دو بار مسئله را *Scan* می‌کردید. یک رفت و یک برگشت ولی در اینجا خواهید دید که شبکه سه بار *Scan* می‌شود رفت، برگشت ، رفت.

رفت اول کاملاً شبیه قبل است . اگر دقت کنید این سه خط تا اینجا کپی محاسبات *Forward* است محاسبات برگشت نیز مجدداً همان است.

قسمت جدید این سودوکد از این مرحله است در محاسبات برگشت اگر گره j ، *Out-dummy* است پس زودترین زمان گره ، کدام گره ؟ گره *Out-dummy*را بنویسید مینیمم زودترین زمان تحقق پس نیازش.

اگر *Out-dummy* بود چک می‌کندی یک عدد است که باید عوض کنید

اینجا تمام نمی‌شود اگر دقت کنید برای بار دوم شبکه را *Forward* ، *Scan* می‌کند از ۲ تا n یکبار دیگه چی کار کن؟ عکس آن است. اگر گره j ، *in-dummy* (قبلی *Out-dummy* را اصلاح کرد) اگر j ، *In-dummy* است چی کار کن؟ دیرترین زمان آن گره را بنویس ماکسیمم دیرترین زمان پیش‌نیازهایش. پس محاسبات رفت کپی، محاسبات برگشت و یک *Scan* دیگه لازم است که بتوانید این را *Run* کنید.

مثال دیگر

چرا لازم است که این را با روش اصلاح شده حل کنیم به خاطر اینکه ۳ و ۴ *out-dummy* هستند و ۸ ، *In-dummy* است ناچار هستیم از این روش استفاده کنیم البته به شرط نیاز داشتن شناوری آزاد و اطمینان لازم داشته باشیم.

خوب شروع که کپی است.

محاسبات رفت.

۰+۳=۳

۳+۱۱=۱۴

۳+۲=۵

این دو تا پیش نیاز دارد ؟؟؟؟؟؟؟؟

۳+۱۱=۱۴ ⎫
⎬→ *Max*→۱۴ ۱۴+۳=۱۷
۵+۰=۵ ⎭

۳۰ همین جا نوشته شود

۳۰−۷=۲۳ *Out-dummy* نیست → ۶ برگشت

سوال آیا این *Out dummy* است ؟ خیر پس ادامه می‌دهیم یعنی هر بار که انجام می‌دهید چک می‌کنید که *out-dummy* است یا نه اگر هست که باید یک کاری انجام دهید .

۳۰−۲=۲۸ *Out-dummy* نیست ۸

۳۰−۱۰=۲۰ ⎫ *min*=۲۰ → *Out-dummy* نیست → ۷
۲۸−۰=۲۸ ⎭

۳۰−۶=۲۴ ⎫ *min* =۲۰ → *Out-dummy* نیست → ۵
۲۰−۰=۲۰ ⎭

۲۳−۰=۲۳ ⎫ *min* =۲۰ → *Out-dummy* است → ۴
۲۰−۰=۲۰ ⎭

حال که *out-dummy* است چی کار کنید؟

زودترین زمان که ۱۴ است را خط بزنید به جاش چی بنویسید ؟ به پس‌نیازهایش دقت کنید زودترین زمان این ۱۷ است و زودترین زمان این ۲۰ است بین ۱۷ و ۲۰ ،مینیمم ۱۷ است این ۱۴ پاک می‌شود به جاش ۱۷ نوشته می‌شود.

چون *Out-dummy*بود این اتفاق برایش می‌افتد.

حالا سه را شما بگویید. قبلش یک *out-dummy* انجام بدهید.

۲۰-۰=۲۰ است *Out-dummy*

Min ←۲۰

۵ را پاک کن به جاش مینیمم ۱۷ و ۲۰ که ۱۷ است را بنویس. به جای ۱۴ ← ۱۷ می‌نویسیم.

۲۰-۲=۱۸
۲۰-۱۱=۹
۲۰-۱۲=۸

min =۸ ←*Out-dummy* نیست

یکبار دیگه محاسبات رفت را برای بار دوم انجام دهید تنها باید چک کنید که *in-dummy* است یا نه ؟ همه را که چک کنید به ۸ می‌رسید که *in-dummy* است برعکس قبلی، دومی باید اصلاح شود یعنی *Latest* باید اصلاح شود. ۲۸ را پاک کنید به جاش پیش نیاز ۲۰ را بنویسید. اگر دو تا پیش نیاز داشت ماکسیمم آن را می‌نویسید.

تبدیل

۲۰ ←۲۸ در گره ۸

بعد از انجام اصلاحات حالا محاسبات شناوری را دوباره انجام دهید و الان این قضیه را دقت بفرمایید که تضاد دیگه برطرف گردید.

خلاصه بحث اگر *AOA* را استفاده می‌کنید در هر صورت که جواب‌هایتان متفاوت است ولی آخر از همه باید جواب‌ها یکسان باشد. اگر جوابها یکسان نباشد دو علت دارد یا شبکه را غلط رسم کردید. یا شبکه شما *in-dummy , out-dummy nodes* دارد. اگر *in-dummy , out-dummy nodes* دارد باعث اشکال شده که آن وقت باید محاسبات رفت و برگشت را طبق این کد آخر اصلاح کنید تا جواب‌ها یکسان گردند.

جلسه هفتم

پارت ۶ راجع به زمانبندی پروژه‌هایی که محدودیت منابع نامحدود با فرض پیش‌نیازی های ***finish to start*** از نوع ساده بود. این جلسه تعمیم موضوع به پیش‌نیازهایی که به اصطلاح ***GPR*** هستند، می‌باشد. در جلسات قبل گفته شد که پیش‌نیازی های ***time lag*** را که ۸ حالت مختلف داشت***FF, FS, SF,SS*** که هر کدام به دو حالت ***Max , min*** تقسیم می‌شد. این ۸ حالت را پیش‌نیازی ***GPR*** می‌گویند.

موضوع از نظر تیتر خیلی ساده است. تعمیم پیش‌نیازی های ***Finish to start*** به حالت کلی یعنی ***GPR*** می‌باشد.

قبل از شروع شبکه‌های ***GPR*** ، اولین کاری که باید انجام گردد استاندارد کردن کل شبکه می‌باشد. یعنی تمام ۸ حالت (پیش‌نیازی‌ها) به یک صورت شوند. SS^{min} را به عنوان استاندارد پذیرفتیم و ۷ حالت دیگر را باید به SS^{min} تبدیل شود.

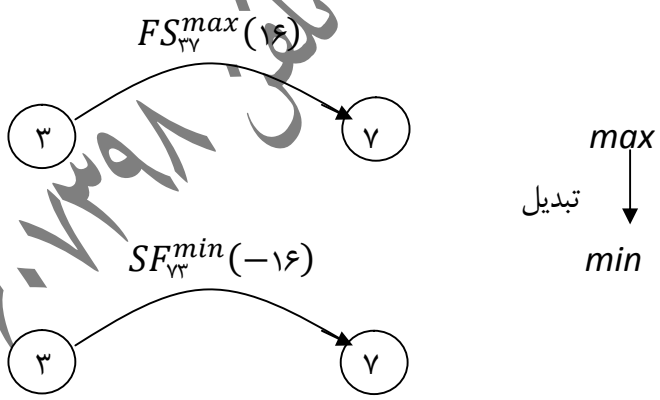
نحوه تبدیل این ۷ حالت به SS^{min} به چه صورت بود.

یک روش آن استفاده از فرمول بود (نیاز نیست که فرمول‌ها حفظ شود خیلی راحت می‌توانید آنها را به هم دیگه تبدیل کنید.)

گام اول : ***Max*** را به ***min*** برمی‌گردانیم.

گام دوم : ***min*** ها را SS^{min}

نحوه تبدیل پیش‌نیازی ***max*** به ***min*** :



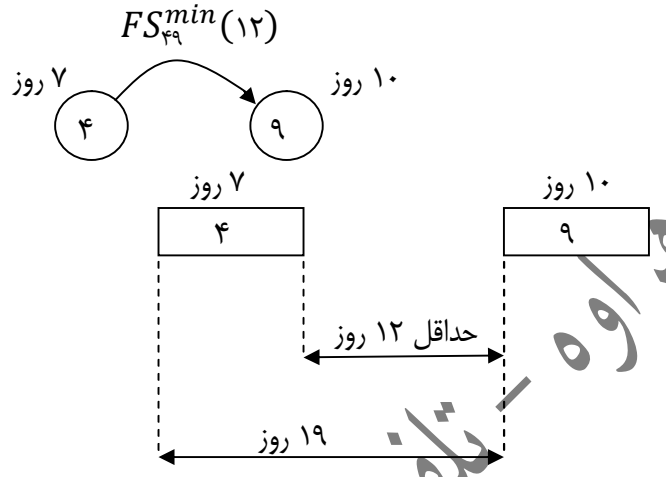
به طور مثال فرض کنید ۳ پیش نیاز ۷ باشد که بالای آن نوشته شده است $FS_{۳۷}^{max}(۱۶)$ یعنی فعالیت ۳ که تمام شود حداکثر ۱۶ روز بعد باید فعالیت ۷ شروع شود ***max*** را به ***min*** تبدیل می‌کنیم.

- اول جهت فلش را باید برعکس کنیم یعنی ***i*** به ***j*** به صورت ***j*** به ***i*** گردد.
- جای آن ***FS*** را باید عوض کنید اگر ***FS*** است بشود ***SF***
- بعد ***Max*** به ***Min*** نوشته شود.
- ***Lag*** هم باید قرینه شود یعنی ۱۶ می‌شود ۱۶-

الان به جای ۸ تا پیش‌نیازی ، ۴ تا پیش نیازی داریم. همه *max* ها تبدیل به *min* می‌شوند. حالا خود این *Max* ها هم باید به SS^{min} تبدیل شود.

گام دوم : که برای این حالت هم می‌توانید هم از فرمول استفاده کنید و هم آنها را بسازید. (همه *min* شدند و باید تبدیل به SS^{min} شوند)

مثال : فرض کنید فعالیت ۴ پیش نیاز فعالیت ۹ است و بالای آن $FS_{۴۹}^{min}(۱۲)$ و فعالیت ۴ ، ۷ روز طول می‌کشد و فعالیت ۹ ، ۱۰ روز طول می‌کشد. حالا این پیش‌نیازی به چه معنی است ؟ $FS_{۴۹}^{min}$ یعنی پایان ۴ تا شروع ۹ حداقل ۱۲ روز می‌باشد. اگر پایان تا شروع ۱۲ روز باشد شروع تا شروع اش چقدر می‌شود ؟ با توجه به اینکه فعالیت ۴ خودش ۷ روز است. مشخص است شروع تا شروع آن $lag_{FS} + d_{۴} = ۱۲ + ۷ = ۱۹$



$$f_i + FS_{ij}^{min} \leq S_j \qquad S_i + L_{ij} \leq S_j$$

FS^{min} به چه صورت تبدیل به SS^{min} می‌شود. باید *FS* را با *duration i* جمع کنید که *Lag SS* به دست می‌آید که با L_{ij} می‌توانید نشان دهید.

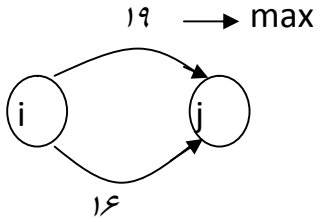
خلاصه اینکه: اگر یک شبکه شلوغ و درهم برهمی داشته باشیم بعد از استاندارد شدن، شبکه یک مقدار خلوت‌تر می‌شود. این دو تا شبکه با هم کاملاً یکی هستند. فقط اعدادی که در روی بردارها نوشته شده است یعنی *Lag* استاندارد شده است یعنی ۱۹

سوال : پس چرا نوشتم SS^{min} ؟ چون دیگه نیازی نیست همه آنها یکی هستند . قبلش یک توافقی کردیم که SS^{min} استاندارد باشد و دیگه نیاز به گفتنش نیست. تمام اینهایی که روی بردار می‌بینید SS^{min} *Lag* است.

ولی در شبکه اصلی هر کدام یک معنی داشت. اینجا *Lag*چقدر است ؟ (۲) اینجا چی ؟ (۲) هردو (۲) می‌باشد. ولی اون با این یک معنی دیگر دارد. ترجمه آن ۲ با این ۲ یکی نیست چون FS^{min} است این یکی FF^{min} است. فرض کنید که اینجا ۳ داریم و اینجا هم یک ۳ دیگه ، این یکی FF^{max} است این یکی FS^{max} است این معنی‌هاش یکسان نیست حال که استاندارد گردید اعدادی که اینجا می‌بینید *Lag* است. ۲ است اینم ۲ است. خوب باشد هر دو یکی است.

نکته : اگر بین دو گره‌ای بیش از یک *Lag* وجود داشت باید ماکسیمم *lag* را بنویسید.

فرض کنید بین i و j دارای دو Lag ، ۱۹ و ۱۶ باشد. کدام یکی را می‌نویسید ؟ ۱۹ یا ۱۶ یا هر دو ؟
بین i و j حداقل ۱۶ روز فاصله باشد
بین i و j حداقل ۱۹ روز فاصله باشد.



$i \rightarrow ۴$ $\quad L_{ij} = ۱ \quad [S_۴ + ۱, S_۴ + ۳]$
$j \rightarrow j$ $\quad L_{ji} = -۳ \quad ۴$ به ۶ time

از دو جمله بالا اگر ۱۹ را بپذیرید یعنی ۱۶ را هم پذیرفته‌اید. چون وقتی فاصله i و j بالای ۱۹ روز باشد بالای ۱۶ هم هست.
پس اگر بین دو گره بیش از یک بردار بود ماکسیمم را می‌نویسیم این ماکسیمم کار بقیه را نیز انجام می‌دهد.
در شبکه‌های GPR ممکن است یک فعالیت از چند جهت به همدیگر پیش نیاز باشند. مثلاً فعالیت a پیش نیاز b است از نوع FF^{max} . همین دو فعالیت می‌تواند پیش نیاز هم باشند از نوع SS^{min} یا FS^{max} یعنی می‌تواند بیش از یک پیش‌نیازی بین دو فعالیت وجود داشته باشد. ولی موقع استاندارد کردن به اصطلاح فقط ماکسیمم را می‌نویسیم.
فاصله (بازه) $[S_i + L_{ij}, S_i - L_{ji}]$ را $Time\ window$ ، S_j نسبت به S_i
$S_i + L_{ij}$ حد پایین و $S_i - L_{ji}$ حد بالا
L_{ij} با L_{ji} با هم فرق می‌کند ؟ فرق آن دو در این است که جهت فلش‌ها برعکس هستند.
مثلاً بین ۴ و ۶ ، $time\ window$ را بنویسید؟

$i = ۴$
$j = ۶$
$L_{ij} = ۱$
$L_{ji} = -۳$
$[S_۴ + L_{۴۶}, S_۴ - L_{ji}] \longrightarrow [S_۴ + ۱, S_۴ - (-۳)] \longrightarrow [S_۴ + ۱, S_۴ + ۳]$

این را می‌گوییم $time\ window$ ، ۶ نسبت به ۴
مفهوم : ۶ چه زمانی می‌تواند شروع شود ؟ ۶ باید حداقل یک روز تا حداکثر ۳ روز بعد از ۴ شروع شود یعنی فاصله شروع ۴ و ۶ باید بین ۱ تا ۳ روز باشد نه بیشتر و نه کمتر.

مثلاً بین ۲ و ۸ ، *time window* را بنویسید؟ ۸ نسبت به ۲ کی می‌تواند شروع شود.

$i = ۲$
$j = ۸$
$L_{ij} = ۶$
$L_{ji} =$ ندارد
$\left[S_۲ + L_{۲۸}, \text{ندارد}\right] \longrightarrow [S_۲ + ۶, \infty]$

وقتی جهت برعکس فلش ندارد جهت باز است و نامحدود است $[S_۲ + ۶, S_۲ + \infty]$ بین ۲ و ۸

حد بالای آن چیزی ندارد یعنی سقفی ندارد پس ۸ می‌تواند ۶ روز بعد از ۲ شروع شود تا بی‌نهایت. وقتی جهت برعکس فلش ندارد یعنی حد بالای آن مهم نیست بازه نامحدود است.
این مسیر پر رنگ ، مسیر بحرانی می‌باشد در این مثال ۴ مسیر بحرانی وجود دارد.
تعریف مسیر بحرانی : مسیری که فعالیت‌هایش بحرانی هستند و طولانی‌ترین مسیر پروژه است. طولانی ترین است یعنی چی ؟ زمان اش طولانی‌تر است صفر است منظور این است طول یک مسیر چه طور حساب می‌شود ؟ جمع *duration* ها . اینجا به این صورت است نیست. طول یک مسیر جمع *Lag* ها می‌باشد پس طولانی‌ترین مسیر، مسیری است که *Lag* هاش از همه بیشتر است. پس اولین تفاوت از همین جا شروع شد.
نکته ۱: در شبکه های *GPR*، طول مسیر از مجموعه *Lag* ها بدست می‌آید.
سوال : در این مثال این پروژه چند روز طول می‌کشد تا انجام شود ؟ ۱۶ روز
۴ تا مسیر ۱۶ دارد که همه بحرانی هستند. (جمع *Lag* ها)

۲+۶+۲+۶=۱۶
۲+۶+۱+۱+۶+=۱۶
۲+۶+۲+۱+۵=۱۶
۲+۶+۱+۱+۱+۵=۱۶

اما بقیه مسیرها از ۱۶ کمتر هستند.
در شبکه‌های *GPR* وجود حلقه در شبکه‌ها مجاز است.
در شبکه‌های *finish to start* حلقه مجاز نبود اما در این شبکه با *GPR* حلقه مجاز است.
حلقه‌های این شبکه عبارتند از :

۴ ← ۶ ← ۴
۹ ← ۶ ← ۴
۲ ← ۴ ← ۲
۷ ← ۸ ← ۱۰ ← ۷ (۷ پیش نیاز ۸ ، ۸ پیش نیاز ۱۰ و ۱۰ هم پیش نیاز ۷ می‌باشد)
۲ ← ۸ ← ۹ ← ۴ ← ۲

جهت فلش‌ها اگر برگردد به سر جای اول ایجاد یک حلقه می‌کند.

نکته ۲ : در شبکه با GPR وجود حلقه مجاز است به شرطی که طول حلقه غیر مثبت باشد یعنی منفی یا صفر باشد.

اگر مثبت باشد صورت مسئله تان غلط است. در این مثال همه حلقه‌ها غیر مثبت هستند یعنی یا صفر یا منفی هستند.

حلقه اول : ۲- =۱+۳-

حلقه دوم : ۳- =۹-۲+۱

حلقه سوم : ۰ =۲-۲

حلقه چهارم : ۱- =۶+۱+۸-

حلقه پنجم : ۴- =۲-۹-۱+۶

هیچ کدام مثبت نیستند.

فعالیت شروع و پایان پروژه

کدام فعالیت شروع پروژه است ؟

در شبکه‌های Finish to start فعالیتی شروع پروژه بود که پیش نیاز نداشته باشد و فعالیتی ، فعالیت پایان پروژه بود که پس نیاز نداشته باشد. این تعریف برای شبکه‌های GPR صحیح نمی‌باشد.

در این مثال فعالیت ۱۰ که به نظر فعالیت پایان است یک پس نیاز دارد که آن هم ۷ می‌باشد. پس در شبکه‌های GPR پیش‌نیاز دارد یا خیر برای شروع و پایان مهم نیست.

نکته ۳: گره یک در شبکه‌های GPR به شرطی شروع پروژه است که از گره یک به همه گره‌ها حداقل یک مسیر با طول نامنفی وجود داشته باشد. اگر به برخی از گره‌ها چنین مسیری وجود نداشت یک بردار از یک به ان گره با طول صفر اضافه می‌کنیم.

در این مثال بررسی کنید که آیا گره یک می‌تواند شروع پروژه باشد یا خیر ؟

از ۱ به ۲ یک مسیر به طول صفر

از ۱ به ۳ یک مسیر به طول ۲

از ۱ به ۴ یک مسیر به طول ۲

از ۱ به ۵ یک مسیر به طول ۸

از ۱ به ۶ یک مسیر به طول ۳

از ۱ به ۷ یک مسیر به طول ۹

از ۱ به ۸ یک مسیر به طول ۶

از ۱ به ۹ یک مسیر به طول ۷

از ۱ به ۱۰ یک مسیر به طول ۱۲

پس حداقل یک مسیر داریم که مثبت است.

حال فرض کنید که بردار ۲ به ۴ در این مثال نبود. از ۱ به ۴ تنها یک مسیر وجود دارد با طول منفی (۲-=۹-+۱-+۶+۰) چون منفی است یک بردار از یک به ۴ اضافه می‌کنیم با طول صفر . در واقع این بردار یک پیش نیاز مجازی است.

نکته ۴: قبلاً گرهی که پس نیاز نداشته پایانی بود ولی گره n به شرطی گره پایان است که از هر گره i به آن حداقل یک مسیر با طول d_i (منظور همان duration i) وجود داشته باشد اگر چنین مسیری وجود نداشت باید از i به n یک بردار با طول d_i اضافه می‌کنیم. یک شروع و ۱۰ پایان می‌باشد.

حال در این مثال بررسی کنید که گره ۱۰ پایان پروژه است یا خیر ؟ از همه گره‌ها به ۱۰ یک مسیر حداقل با طول d_i باشد.

از ۹ به ۱۰ یک مسیر حداقل به طول (حداقل باید باشد) $d_i = ۴$ وجود دارد (۵) هست

از ۸ به ۱۰ یک مسیر حداقل به طول (حداقل باید باشد) $d_i = ۶$ وجود دارد (۶) هست

از ۷ به ۱۰ یک مسیر حداقل به طول (حداقل باید باشد) $d_i = ۵$ وجود دارد (۶+۱=۷) هست

از ۶ به ۱۰ یک مسیر حداقل به طول (حداقل باید باشد) $d_i = ۴$ وجود دارد (۲+۵=۷) هست

از ۵ به ۱۰ یک مسیر حداقل به طول (حداقل باید باشد) $d_i = ۴$ وجود دارد (۲+۶=۸) هست

از ۴ به ۱۰ یک مسیر حداقل به طول (حداقل باید باشد) $d_i = ۳$ وجود دارد (۱+۲+۵=۸) هست

از ۳ به ۱۰ یک مسیر حداقل به طول(حداقل باید باشد) $d_i = ۷$ وجود دارد(۴-=۶+۱۰-) , (۱۴=۶+۲+۶) هست.

نکته با اینکه دو مسیر وجود دارد که یکی از مسیرها منفی است اما در تعریف داشتیم حداقل یک مسیر مثبت کافی است.

از ۲ به ۱۰ یک مسیر حداقل به طول (حداقل باید باشد) $d_i = ۲$ وجود دارد (۱۲=۶+۶)

از ۱ به ۱۰ یک مسیر حداقل به طول(حداقل باید باشد) $d_i = ۰$ وجود دارد (۱۲=۶+۶+۰)

خلاصه اینکه در این مثال یک واقعاً شروع پروژه و ۱۰ پایان پروژه می‌باشد.

نکته ۵: مفهوم پیش‌نیازی در شبکه GPR

هنگامی که a پیش نیاز b است منظور این است که فعالیت a باید زودتر انجام شود اما در این شبکه GPR لزوماً اینطوری نیست.

در این مثال ۴ پیش نیاز ۶ است یا خیر ؟ بله به خاطر اینکه از ۴ به ۶ یک بردار دارید .

سوال : آیا ۶ پیش نیاز ۴ است یا خیر ؟ چه طور می‌شود که فعالیت ۴ پیش نیاز ۶ باشد و برعکس آن یعنی ۶ هم پیش نیاز ۶ باشد اتفاق بیافتد.

اگر از i به j مسیر داشته باشیم یعنی i پیش نیاز j باشد و اگر جمع Lag های این مسیر مثبت باشد یعنی پیش نیاز واقعی است و اگر منفی شد پیش نیاز کاذب است.

دو نوع پیش نیاز در شبکه GPR است : پیش نیاز واقعی ، پیش نیاز کاذب

اگر طولانی تری مسیر از i به j نامنفی باشد Real Predecessors i یا پیش نیاز واقعی نامیده می‌شود i پیش نیاز واقعی j و j هم پس نیاز واقعی خواهد شد. در غیر اینصور Fictitious یا پیش نیازی کاذب گفته می‌شود.

سوال : اگر در این حلقه هر دوی اینها مثبت بودند چی می شد؟ چرا حلقه مثبت داشته باشیم شبکه غلط است؟

چون معنی‌اش از نظر منطقی بهم می‌ریزد اگر هر دو مثبت شود یعنی ۴ واقعاً پیش نیاز ۶ است و باید زودتر انجام شود و ۶ هم پیش نیاز ۴ است و باید زودتر انجام گردد. پس نمی‌شود که دو فعالیت همزمان پیش نیاز باشند و بخواهند زودتر انجام شوند که یک تضاد ایجاد می کند پس حلقه مثبت ممنوع است یعنی با منطق قضیه نمی‌خواند ، نمی‌شود که دو فعالیت هر دو پیش نیاز هم باشند.

نتیجه :

۴ تفاوت بین شبکه‌های قبلی و شبکه GPR به صورت ذیل می‌باشد:

- عدم مطرح بودن حلقه در شبکه‌های قبلی و وجود حلقه در شبکه GPR به شرطی که طولش مثبت نباشد.
- تفاوت تعریف شروع و پایان در این شبکه
- مسیر بحرانی (طول مسیر) جمع Lagها می‌باشد نه duration
- تعریف پیش نیازی در این شبکه GPR (واقعی و کاذب) . اگر i پیش نیاز j است یعنی بردار دارید لزوماً به معنی زودتر شروع شدن نیست.

زمانبندی

چه طور این پروژه‌ها زمانبندی می‌شوند ؟ اگر بخواهیم زمانبندی را به صورت عددی گزارش کنیم ، زمانبندی یعنی چی؟

زمانبندی یعنی تعیین کنید که فعالیت‌های چه زمان شروع شوند و یا چه زمان تمام شوند. که یکی را هم بگویید کافی است. یک زمانبندی را معمولاًبا یک بردار n تایی نشان می‌دهید بردار سطری n عضوی که این مولفه‌ها چه چیزی را نشان می دهند. S_1 شروع یک را نشان می‌دهد. S_2 شروع دو را نشان می‌دهد. S_n شروع گره پایان را نشان می‌دهد یعنی ختم پروژه.

در مثال قبلی : یک بردار ۱۰ تا عدد است (۰,۴,۹,۱۱,۶,۱۵,۱۰,۷,۶,۳۰) یک زمانبندی است.

به چه شرطی این یک زمانبندی Feasible(موجه) است ؟

این را گفتم ۴، این را گفتم ۷، این جواب درست است ؟ این را ۴ می‌خواهم شروع کنم و این را ۷ . غلط است چون ۶ به چه معنی است ؟ ۲ که شروع شد حداقل ۶ روز بگذرد من این را ۴ ام شروع کردم این را نمی‌توانم ۷ ام شروع کنم چون حداقل ۶ روز نشده است پس زمانبندی هر بردار عددی زمانبندی است.ولی بحث feasible بودن‌اش است. برای اینکه Feasible باشد اول از همه اعدادی که نوشته می‌شوند باید مثبت باشد.

سوال : چرا مثبت باشند ؟ چون زمان است پس باید مثبت باشد.

ثانیاً Lag را هم در نظر بگیرید.

الان بین ۲ و ۸ مشکل کجا است ؟ lag، ۶ روزه را رعایت نکرده است. حالا اینجا بین i و j باید یک lag_{ij} را رعایت کنید.

$Start_j$ باید حداقل L_{ij} تا از $Start_i$ بیشتر باشد. L_{ij} آنجا عدد بود اینجا پارامتر است. پس هر جوابی که ، هر sequence ای از اعدادی که این دو تا شرط را رعایت کند time feasible می‌باشد. یعنی از نظر زمان موجه است.

از نظر Resource چی ؟ آیا resource feasible هم می‌باشد ؟

در این مبحث فرض بر این شده است که منابع نامحدود است بعداً باید resource feasible هم باشد. شما زمانبندی کردید ولی منبع ندارید که این بحث دیگری است که بعداً در رابطه با آن صحبت خواهد شد.

پس هر زمانبندی که این دو شرط را رعایت کند feasible می‌شود برای بهینه بودن آن بستگی به تابع هدف دارد.

در مباحث گذشته سه عنوان γ و β و α مطرح گردید. که γ تابع هدف بود. بهینگی یعنی باید یک شاخصی را بهینه کند. کدام شاخص را ؟ بستگی به این دارد که شما چه شاخصی را تعریف کرده‌اید. هدف شما مینیمم کردن زمان است، یا ماکسیمم کردن سود یا مینیمم کردم انرژی، یا ماکسیمم کردن کیفیت است که همه اینها فرق می‌کند که ما معمولاً در ساده‌ترین حالت فرض می‌کنیم که هدف ما مینیمم کردن make span است معمولاًاین هدف زمان است که با C_{max}نشان می‌دهیم پس فرض می‌کنیم هدف C_{max} استC_{max} یعنی مینیمم کردن زمان پروژه

S_n پایان پروژه است یعنی زمان شروع فعالیت n ام (آخر) . خوب فعالیت آخر که مجازی است شروع و پایانش یکی است و وقتی فعالیت پایان را مینیمم کنید یعنی عملاً پروژه را مینیمم کرده‌اید. پروژه کی تمام می‌شود؟ وقتی که آن فعالیت آخر تمام شود.

پس تابع هدف به زبان ریاضی ← مینیمم کردن S_n است.S_n همان C_{max} است . اسم‌ها را عوض کنید مهم نیست. مفهوم مهم است زمان پروژه یعنی min S_n

با توجه به این محدودیت time lag ها باید S عدد صحیح مثبت باشد. آیا باید عدد صحیح باشد ؟ آیا می‌تواند پیوسته هم باشد ؟ می‌تواند پیوسته هم باشد، اعشاری باشد. چون ما صنایعی فکر می‌کنیم یعنی پروژه‌هایی که در ذهن ما وجود دارد پروژه‌های صنعتی، عمرانی است چندین ماه و چندین سال طول می‌کشد صدم ثانیه ، صدم روز برای ما مهم نمی‌باشد. اگر یک کاری ۳ روز و ۰.۴۵ طول بکشد به این صورت عنوان نمی کنیم روندش می‌کنیم و می‌گوییم ۴ روز . در واقع در یک پروژه چند ماهه ، چند ساله وارد ساعت و دقیق نمی‌شویم.

اما گاهی اوقات پروژه‌ای نرم‌افزاری داریم که می‌خواهید یک برنامه را در سیستم‌تان run کنید و بدانید چقدر طول می‌کشد که این را حل کند ؟ چند صدم ثانیه ؟ چند هزارم ثانیه ؟ که در اینجا می‌گوییم S_i متعلق به R (اعداد حقیقی) است یعنی می‌تواند اعداد اعشاری هم باشد.

خوب عموماًدر پروژه این فرض مطرح می‌شود که عدد صحیح فرض کنید والا می‌توانید اعشاری هم باشد ولی عدد صحیح حل می‌کنیم.

یک مدل ریاضی را الان داریم در این مدل چند تا محدودیت و چند تا متغییر داریم ؟

n تا متغیر داریم (۱۰ تا) متغیرها چی هستند S_1, S_2, S_3, S_n مجهول هستند.

چه محدودیت‌هایی داریم ؟ lag ها چند تا است ؟

هر بردار یک Lagدارد (به تعداد بردارها) ۱۷ تا

به تعداد گره‌ها متغیر و به تعداد بردارها محدودیت داریم.

در این مثال اگر بخواهید مدل ریاضی‌اش را بنویسید باید ۱۰ تا متغیر با ۱۷ تا محدودیت برای آن تعریف کنید. حال چه طوری می‌توان مسئله را حل کرد؟ با توجه به اینکه عدد صحیح هستند. در لیسانس تحقیق ۲ برنامه‌ریزی عدد صحیح ، شاخه و کران صفحات برشی یادتان می‌آید ، باید یادتان بیاد. شاخه و کران جزء روش‌هایی است که ما در هفته آتی توضیح خواهم داد. استثناً اینجا می‌شود این را راحت‌تر حل کرد. یعنی می‌شود بدون شاخه و کران این مسئله را حل کرد. حل این مسئله به لطف تابع هدف‌اش می‌باشد. تابع هدف در این مسئله ، Min کردن make span است و make span یک هدف منظم است ، هدف شما این است که پروژه زود تمام شود پس کارها هر چقدر زودتر انجام شود بهتر است درنتیجه منظم است. اگر هدف منظم باشد بهترین زمانبندی ESS (earliest start scheduling) است. ESS یعنی فعالیت‌ها در زودترین زمان شروع کنید. بهینه است پس نیازی به روش شاخه و کران نیست چون یک راه میانبر برای حل مسئله وجود دارد البته به خاطر هدف‌اش می‌باشد. چون منظم است. اگر هدف آن نامنظم بود ESS بهینه نبود آن وقت حتماً باید از روش شاخه و کران حل می‌کردید.

چه طور ESS حساب می‌شود. محاسبات رفت و (برگشت).

برای اینکه earliest start scheduling را حساب کنیم محاسبات forward را انجام دادیم تا EST و EFT زودترین زمان. بعد از انجام محاسبات در این مثال که ESS به دست می‌آید یعنی اینها کجا شروع شدند در EST هایشان. EST با محاسبات Forward انجام می‌شد.

محاسبات Forward برای شبکه‌هایی بود که پیش‌نیازی از نوع Finish to start بود ولی در این شبکه‌های GPR پیش‌نیازیها ۸ مدل می‌باشد. پس محاسبات Forward فقط در شبکه‌های CPM درست است که پیش‌نیازی ها از نوع Finish to start است.

در شبکه GPR ، ESS با روش دیگر محاسبه می‌شود. این روش Floyd-warshall می‌باشد. پس در شبکه GPR با Forward , backward کاری نداریم.

الگوریتم Floyd-warshall ، $O(n^3)$ time complexity می‌باشد . n به توان می‌شود یک مسئله آسان است. این مسئله هم جزء مسائل آسان است ولی نسبت به Forward,backward سخت تر است چون $O(n^2)$ بود. پس بین آسان و آسان‌تر ، هر دو آسان هستند این یکم سخت تر است. سخت یعنی زمانبر است سخت به معنی درک مسئله نیست.

روش Floyd-warshall براساس برنامه‌ریزی پویا است.

در این روش یک ماتریسی به اسم $\Pi^{(1)}$ آماده کنید این ماتریس $\Pi^{(1)}$ ، n×n است مثلاً در این مثال ۱۰×۱۰ است. در این ماتریس ۱۰×۱۰ چه چیز نوشته می‌شود ؟ اگر i=j یعنی قطر اصلی صفر است. در ردیف دوم به ازای هر (i,j)

متعلق به E (مجموعه بردارها هستند) lag را بنویسید. به زبان ساده‌تر اگر از i به j مستقیماً بردار داشتید مقدار lag را بنویسید و در بقیه خانه‌ها $-\infty$ قرار بدهید.

در این ماتریس

از ۱به ۲ ⟵ صفر ⟵ از یک به دو بردار داریم lag=۰

از ۱ به ۳ ⟵ ۲

ما فقط از ۱ به ۲ و از ۱ به ۳ مستقیماً بردار داریم که در این خانه‌ها lag را می‌نویسیم اما از ۱ به ۹،۱۰،۴،۵،۶،۷،۸ بردار نداریم $-\infty$ می‌نویسیم.

از ۲ به ۱ و از ۲ به ۳ بردار نداریم ⟵ $-\infty$

از ۲ به ۴ ⟵ ۲

از ۲ به ۵ ⟵ نداریم ⟵ $-\infty$

از ۲ به ۸ ⟵ ۶

خلاصه این ماتریس ۱۰۰ تا خانه دارد که باید پر شود چون ۱۰×۱۰ است پس این پر کردن آن کاری ندارد.

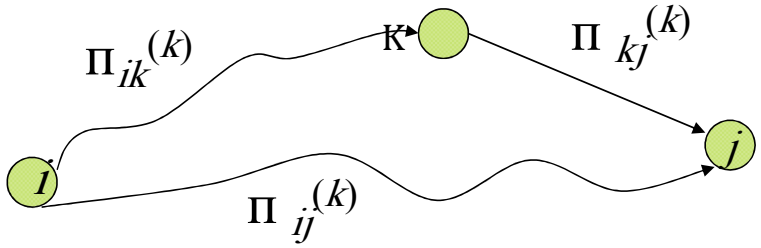
$$\Pi^{(1)} = \begin{array}{c} \\ ۱ \\ ۲ \\ ۳ \\ ۴ \\ ۵ \\ ۶ \\ ۷ \\ ۸ \\ ۹ \\ ۱۰ \end{array} \begin{array}{c} \begin{array}{cccccccccc} ۱ & ۲ & ۳ & ۴ & ۵ & ۶ & ۷ & ۸ & ۹ & ۱۰ \end{array} \\ \begin{pmatrix} ۰ & ۰ & ۲ & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty & -\infty \\ -\infty & ۰ & -\infty & ۲ & -\infty & -\infty & -\infty & ۶ & -\infty & -\infty \\ -\infty & & ۰ & & & & & & & \\ & & & ۰ & & & & & & \\ & & & & ۰ & & & & & \\ & & & & & ۰ & & & & \\ & & & & & & ۰ & & & \\ & & & & & & & ۰ & & \\ & & & & & & & & ۰ & \\ & & & & & & & & & ۰ \end{pmatrix} \end{array}$$

۱۰ × ۱۰

شروع Floyd-warshall این است که ابتدا یک ماتریس ۱۰×۱۰ آماده کنید حالا از روی این ماتریس $\Pi^{(1)}$ یک ماتریس دیگری به اسم $\Pi^{(2)}$ درست کنید که آن هم ۱۰×۱۰ است فعلاً چیزی در خانه‌های این ماتریس نوشته نشده است.

ماتریس دوم از روی ماتریس اول طبق این فرمول درست می شود.

K یک اندیس است که در این جا برابر یک است. از روی ماتریس بالامثلاًمی‌خواهد خانه i به j را حساب کنید. که به صورت عددی توضیح داده می شود.



مثلاًخانه ۳ به ۷ را می‌خواهید حساب کنید چه عددی باید نوشته شود پاسخ براساس این فرمول یا شکل فرق نمی‌کند. باید از ماتریس بالایی بفهمید که خانه ۳ به ۷ چند است. ماتریس بالا ردیف ۳ ، ستون ۷ را پیدا کنید از ۳ به ۷ بردار داریم ؟ خیر پس خانه ۳ به ۷ در این ماتریس $-\infty$ است. حال خانه ۳ به ۱ به اضافه خانه ۱ به ۷ چند می‌شود ؟ خانه ۳ به ۱ ⟵ $-\infty$ و خانهٔ ۱ به ۷⟵ $-\infty$ (چون بردار نداریم) حالا کدام بیشتر است $-\infty$ یا $-\infty$ باید max را بگویید . $-\infty$

بنابراین در این مثال این خانه $-\infty$ نوشته می‌شود حالا من کدام خانه را گفتم . ۳ به ۷

شما کل این ۱۰۰ خانه را طبق این قاعده پر کنید. max مسیری که قبلاً بوده و مسیری که از ۱ عبور می‌کند. چون k=۱ است. به این ترتیب ماتریس $\Pi^{(۲)}$ پر می شود.

$$\Pi^{۲}= \begin{matrix} \\ ۳ \end{matrix} \begin{bmatrix} & & & & & & ۷ \\ & & & & & & \\ & & & & & & -\infty \\ & & & & & & \end{bmatrix}$$

حال می‌رویم سراغ ماتریس بعدی که ماتریس $\Pi^{(۳)}$ می‌باشد. تفاوت این ماتریس در این است که K=۲ می‌باشد با K=۳ است ماتریس پایه ۴ طبق همان قاعده به دست می‌آید. با K=۴ است ماتریس $\Pi^{(۵)}$ طبق همان قاعده به دست می‌آید. با k که ۵ است ماتریس $\Pi^{(۶)}$ طبق همان قاعده به دست می‌آید. تا آخرین K =n که در این مثال k=۱۰ است با k که ۱۰ است ماتریس پایه ۱۱ به دست می‌آید. آخرین ماتریس یعنی باید ۱۱ تا ماتریس ۱۰× ۱۰ را حساب کنید.

حساب که کردید و تمام شد ردیف اول آخرین ماتریس ESS می‌شود. چون محاسبات خیلی طولانی بود آخرین ماتریس را اینجا آورده‌ام یعنی باید ۱۱۰۰ عدد حساب کنید تا به ماتریس آخر برسید. ردیف اول آخرین ماتریس ESS است. خلاصه این پروژه ۱۶ روز طول می‌کشد.

زمانبندی کارها را چه زمان شروع کنیم

فعالیت‌های ۱ و ۲ روز صفرام

فعالیت ۳ و ۴ روز ۲ ام، فعالیت ۵ روز ۸ام ، فعالیت ۶ روز ۳ ام، فعالیت ۷ روز ۹ ام، فعالیت ۸ روز ۱۰ ام، فعالیت ۹ روز ۱۱ ام فعالیت ۱۰ روز ۱۶ ام .←که می‌شود پایان پروژه

روش بالا روش Floyd-warshall است که به جای روش Forward , backward است.

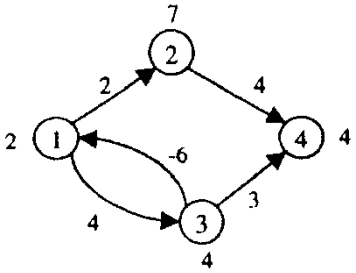
نکته : قطر اصلی در ماتریس $\Pi^{(1)}$ صفر است و باید تا آخر صفر باقی بماند. یعنی اگر از صفر خارج بشود و حتی یکی از آنها مثبت شود به این معنی است که شبکه دارای حلقه مثبت است.

پس روش Floyd-warshall دو کار زیر را انجام می‌دهد:

- ESS را محاسبه می‌کند
- وجود حلقه مثبت در شبکه را بررسی می‌کند. (با چک کردن قطر اصلی که باید تا آخر محاسبه صفر باشد)

مثال بالا خیلی طولانی بود پس با مثال زیر که کوتاه تر است حل می‌کنیم.

$$\Pi^{1}=\begin{bmatrix} 0 & 2 & 4 & -\infty \\ -\infty & 0 & -\infty & 4 \\ -6 & -\infty & 0 & 3 \\ -\infty & -\infty & -\infty & 0 \end{bmatrix}$$



در مثال بالا ۴ تا فعالیت و ۵ تا پیش‌نیازی داریم.

نکته : در AON ما فقط GPR داریم در AOA ، GPR نداریم . پس در GPR ها گره ها فعالیت هستند. بردارها پیش‌نیازی می‌باشد.

ماتریس $\Pi^{(1)}$

قطر اصلی صفر است.

از ۱ به ۲ بردار داریم پس مقدار lag را که ۲ است می‌گذاریم.

از ۱ به ۳ بردار داریم پس مقدار lag را که ۴ است می‌گذاریم.

از ۱ به ۴ بردار نداریم پس $-\infty$ می‌گذاریم.

از ۲ به ۱ بردار نداریم پس $-\infty$ می‌گذاریم.

از ۲ به ۳ بردار نداریم پس $-\infty$ می‌گذاریم.

از ۲ به ۴ بردار داریم پس مقدار lag را که ۴ است می‌گذاریم.

از ۳ به ۱ بردار داریم پس مقدار lag را که ۶- است می‌گذاریم.

از ۳ به ۲ بردار نداریم پس $-\infty$ می‌گذاریم.

از ۳ به ۴ بردار داریم پس مقدار lag را که ۳ است می‌گذاریم.

از ۴ به ۱ و ۴ به ۲ و ۴ به ۳ بردار نداریم پس $-\infty$ می‌گذاریم.

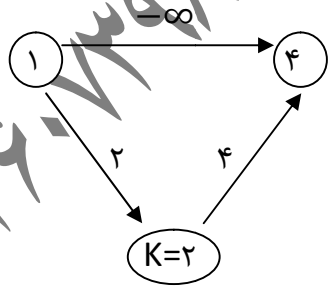
ماتریس $\Pi^{(2)}$:

K را برابر یک قرار می‌دهیم و مسیرهایی را که از یک عبور می‌کند را ببینیم آنها طول بیشتری دارند یا این که الان است. در این مثال $\Pi^{(2)}$با $\Pi^{(1)}$ فرقی نکرد. که در این حالت هیچ اتفاقی نیفتاده است. یعنی هر بار که ماکسیمم گرفتیم پایینی ماکسیمم شد.

$$\Pi^1=\begin{bmatrix} 0 & 2 & 4 & -\infty \\ -\infty & 0 & -\infty & 4 \\ -6 & -\infty & 0 & 3 \\ -\infty & -\infty & -\infty & 0 \end{bmatrix} \Rightarrow \Pi^2=\begin{bmatrix} 0 & 2 & 4 & -\infty \\ -\infty & 0 & -\infty & 4 \\ -6 & -\infty & 0 & 3 \\ -\infty & -\infty & -\infty & 0 \end{bmatrix}$$

ماتریس $\Pi^{(3)}$:

K را برابر ۲ قرار می دهیم. (K=۲) مثلاًخانه (۱,۴) چند می‌شود از ۱ به ۴ در ماتریس $\Pi^{(2)}$، $-\infty$ است ، K=۲ است یعنی از ۱ به ۲ برابر ۲ است و از ۲ به ۴ برابر ۴ است این دو را با هم جمع می‌کنیم ۶=۴+۲. خانه (۱,۴) قبلاً $-\infty$ بوده است و الان مقدار ۶ گرفته است ۶ بزرگتر است و در ماتریس $\Pi^{(3)}$، ۶ جایگزین $-\infty$ می‌کنیم. بقیه خانه‌ها هم به همین ترتیب باید بررسی شود.



$$\Pi^3=\begin{bmatrix} 0 & 2 & 4 & 6 \\ -\infty & 0 & -\infty & 4 \\ -6 & -\infty & 0 & 3 \\ -\infty & -\infty & -\infty & 0 \end{bmatrix} \Leftarrow \Pi^2=\begin{bmatrix} 0 & 2 & 4 & -\infty \\ -\infty & 0 & -\infty & 4 \\ -6 & -\infty & 0 & 3 \\ -\infty & -\infty & -\infty & 0 \end{bmatrix}$$

ماتریس $\Pi^{(۴)}$:

K را برابر ۳ قرار می‌دهیم یعنی هر بار که ماتریس را کامل کردیم یک عدد به K اضافه می‌کنیم.

اگر K را ۳ بگیریم از ۱ به ۳ و از ۳ به ۴ را با هم جمع می‌کنیم در ماتریس $\Pi^{(۳)}$ خانه از ۱ به ۴ مقدار ۶ است. از ۱ به ۳ مقدار ۴ و از ۳ به ۴ مقدار ۳ است پس ۷= ۳+۴ در نتیجه ۷ را جایگزین ۶ قرار می‌دهیم.

$$\Pi^3=\begin{bmatrix} 0 & 2 & 4 & 6 \\ -\infty & 0 & -\infty & 4 \\ -6 & -\infty & 0 & 3 \\ -\infty & -\infty & -\infty & 0 \end{bmatrix} \Rightarrow \Pi^4=\begin{bmatrix} 0 & 2 & 4 & 7 \\ -\infty & 0 & -\infty & 4 \\ -6 & -\infty & 0 & 3 \\ -\infty & -\infty & -\infty & 0 \end{bmatrix}$$

ماتریس $\Pi^{(۵)}$:

K را برابر ۴ قرار می‌دهیم. که البته در این مثال هیچ کدام از اعداد تغییر نمی‌کنند.

$$\Pi^5=\begin{bmatrix} 0 & 2 & 4 & 7 \\ -\infty & 0 & -\infty & 4 \\ -6 & -\infty & 0 & 3 \\ -\infty & -\infty & -\infty & 0 \end{bmatrix} \Leftarrow \Pi^4=\begin{bmatrix} 0 & 2 & 4 & 7 \\ -\infty & 0 & -\infty & 4 \\ -6 & -\infty & 0 & 3 \\ -\infty & -\infty & -\infty & 0 \end{bmatrix}$$

نتیجه آخر :

$S_۱ = ۰\,,\ S_۲ = ۲,\quad S_۳ = ۴\,,\quad S_۴ = ۷$

ردیف اول از ماتریس آخر نشان دهنده ESS است . قطر اصلی هم صفر می‌باشد .

این مثال چون کوچک بود راحت می‌شد فهمید که یک حلقه بیشتر نیست جمع‌اش هم منفی است. اما در مسائل ۴۰۰ تا ۵۰۰ تا فعالیت نمی‌توانید سریع چک کنید. چون مسیر حلقه ممکن است از یک جاهایی باشد که نشود آن را به راحتی پیدا کرد.

پس آن وقت است که در روش Floyd-warshall حتماً لازم دارید که قطر اصلی را امتحان کنید. از نظر محاسباتی روش Floyd-warshall روش خوبی نمی‌باشد.

در تحقیق عملیات یک یک Simplexبود و یک Simplex اصلاح شده داشتید تفاوت بین این دو در چه چیزی است ؟

روش حل همان بود متغیر ورودی انتخاب کنید ، خروجی انتخاب کنید منطق‌اش همان بود هر دو تا یک جواب دارد. اعدادی که نیاز نبود را حساب نمی‌کرد و فقط اعدادی را در جدول حساب می‌کرد که لازم است محاسبات الکی انجام نمی‌گردید در واقع کل موضوع این بود که در جدول Simplex اعدادی را که لازم بود حساب می‌کرد همین.

در این ماتریسی که حساب شد به چه اعدادی نیاز است ؟ بقیه را برای چی حساب شدند ؟ درخصوص این سوالات یک روش دیگری به Modiefied label correcting Algorithm وجود دارد که راحت تر و موثرتر می‌تواند ESS را محاسبه کند در واقع همان کار را با محاسبات بسیار کم انجام می‌دهد.

Modiefied label correcting algorithm (الگوریتم اصلاح Label) ، time complexity اش $O(|V||E|)$ است . اولی تعداد فعالیت‌ها است و دومی تعداد بردار ها می‌باشد. تعداد فعالیت‌ها ضرب در تعداد بردارها اما در روش Floyd-warshall ، Time complexityاش $O(n^3)$ بود.

به طور مثال :

در مثالی که ۱۰ تا فعالیت دارد با روش Floyd-warshall محاسبات $۱۰^۳ = ۱۰۰۰$ خواهد بود ولی در روش Modiefied label correcting algorithm ، ۱۰ تا فعالیت با ۱۸ بردار، محاسبات $۱۰ \times ۱۸ = ۱۸۰$ انجام می‌شود. پس محاسبات کمتری در این روش جدید وجود دارد. اولی با محاسبه ۱۰۰۰ تا به جواب می‌رسد ولی دومی با ۱۸۰ تا به جواب می‌رسد پس آسانتر است.

روش Modiefied label correcting algorithm پنج گام دارد :

۱- برای گره ها به صورت زیر distance label (Label فاصله) بزنید. چطوری آقای چناری Label می‌زنید؟ $\Pi^{(1)}$ را صفر می‌گیرد Π ها همان Label است بقیه را $-\infty$ می‌نویسید سپس لیستی درست می‌کند و در آن یک را می‌نویسد. به طور خلاصه فعالیت یک Labelصفر می‌گیرد(یعنی کنارش شماره صفر می‌نویسد) بقیه را $-\infty$ شماره می‌زند و در لیستی که درست میکند یک را در آن می‌نویسد.

۲- اگر لیست تهی است برو به گام ۵ یعنی تمام شد. اما اگر لیست تهی نیست که در اینجا در این لیست یک ، وجود دارد پس انتخاب کن اولین گره i را از لیست و آن را پاک کن. یعنی یک شماره از لیست به اسم i انتخاب کن که البته این i فعلاً یک است چون در این لیست تنها یک عدد پاکش کن از لیست. پس i=۱ است.

۳- اگر i هیچ پس نیاز اصلاح نشده (uncorrected) ندارد یعنی پس‌نیازی که label اش اصلاح نشده است ندارد. برگرد به گام دو یعنی دوباره چک کن ببین لیست تهی است اگر تهی که تمام و اگر تهی نیست یکی دیگر را انتخاب کن. اگر پس نیاز دارد انتخاب کن یکی از پس نیازهایش اصلاح نشده را به دلخواه انتخاب کن یعنی از بین پس‌نیازهایش یکی را به دلخواه انتخاب کن مهم نیست که کدام را انتخاب کنی. اسم این پس نیاز را j گذاشته است پس j پس نیاز i است . i از لیست انتخاب شده است.

۴- تنها گامی است که فرمول دارد. label j را بنویس ماکسیمم label ایی که الان دارد و label i را باضافه lagبین آنها . $\Pi_j = Max\{\Pi_j, \Pi_i + l_{ij}\}$ هر کدام بیشتر است. label فعلی بیشتر است یا نه label i +lag بیشتر است. اگر label j تغییر کرد آن را به لیست اضافه کن و برگرد به گام ۳

۵- گام پنجم پایان می‌باشد.

مثال :

گام یک : فعالیت یک =۰ Label ، $(\Pi_۱ = ۰)$ و بقیه را $-\infty$ می‌گذاریم. $(\Pi_i = -\infty)$ یک لیست هم درست کن در آن یک را بنویس

گام دوم : اگر لیست تهی است که تمام و اگر نیست i را از لیست که یک است انتخاب کن و پاک کن

گام سوم : این i یعنی ۱ دو تا پس نیاز مستقیم اصلاح نشده دارد که فعالیت های ۲ و ۳ می‌باشد فرقی نمیکند یکی را انتخاب کنید مثلاً ۲ را انتخاب کردیم. J=۲

گام چهار : Label=۲ ، $\Pi_۲$ را که انتخاب کردید Labelاش چند است ؟ $-\infty$

$۲ = ۲ + ۰ = \Pi_۱ + l_{ij}$ این ۲ بیشتر است یا $-\infty$ معلوم است ۲ . پس آن Labelرا پاک کن و به جاش ۲

$label_{\,i} + \text{lag}_{\,ij}$

را بنویس چون Label تغییر کرده است ۲ را به لیست اضافه کن.

$$\Pi_j = max(\Pi_۲, \Pi_۱ + L_{۱۲})$$

i=۱ و j=۳ است حالا چرا j ، ۳ است چون پس نیاز دو تا داشت. یکی را انجام دادید و یکی دیگری اش مانده است. بین ۱ و ۳ همین را بگویید label ، ۳ چند است ؟

$$\Pi_j = max(\Pi_۱, \Pi_۳ + L_{۳۱})$$

$label_{\,i} + \text{Lag} = ۴$ وقتی مقایسه می‌کنید ۴ بیشتر است، ۴ آنجا به جای ***label*** قرار می‌گیرد. فعالیت ۳ که ***label*** اش عوض شد می‌آید اینجا اضافه می‌شود. پس ۳ هم به لیست اضافه شود حال ***i*** چند است ؟ ***i***=۱ ، خوب ***j*** بعدی ؟ ***j*** بعدی نداریم . برگردیم به گام ۲ یعنی چی ؟ لیست اگر تهی است تمام اگر نه ***i*** بعدی ، ۲ را انتخاب کنید، حذف. ***J*** پس نیاز ۲ است پس ***j***=۴ است. ۶=۴+۲ به جای $-\infty$ ، ۶ را می‌نویسیم چون فعالیت ۴ ، ***label*** اش تغییر کرد به لیست اضافه می شود. ***I***=۲ است و ***j***=۴ می‌باشد. حالا بعدی ، ***i*** چند است ؟ چون ***j*** دیگری نداریم پس می‌رویم سراغ ***i*** بعدی. ***i*** از لیست ۳ است وقتی از لیست انتخاب می‌کنید حتماً پاکش کنید. ***i***=۳ و ***j*** را هم ۱ و هم ۴ را می‌توانید بنویسید چون ۳ دارای دو پس نیاز ۱ و ۴ است. فعلاً با ۱ شروع می‌کنیم. یعنی ***j***=۱است. ۴ با ۶- می‌شود ۲- و وقتی با صفر مقایسه می شود صفر بیشتر است. پس تغییر نمی‌کند. چون تغییر نمی‌کند بنابراین یک دیگر به لیست اضافه نمی‌شود. پس ***i*** همان ۳ است ***j*** بعدی ۴ است. خوب ۴ با ۳ می‌شود ۷ در مقایسه با آن ۶، هفت بیشتر است. ۴ به لیست اضافه می‌شود چون بود دیگه اضافه نکردیم. ***i***=۳ دیگه نداریم. چون پس نیاز ندارد می‌رویم سراغ ***i*** بعدی ، ***i***=۴ است. ۴ که انتخاب شد از لیست پاک می‌شود پس نیاز ندارد، می‌رویم سراغ فعالیت بعدی چون دیگر نداریم لیست تهی می‌گردد.

گام پنج : تمام شد.

نتیجه : ***label*** ها ← ***(۰,۲,۴,۷)*** ← این مجموعه ***ESS*** است.

همین مثال را با روش ***Floyd-warshall***حل کردیم که با ۵ ماتریس و هر کدام ۱۶ عدد داشت یعنی ۸۰ تا محاسبه انجام گردید. ولی در این روش اینطوری نبود. اینجا ۴ تا اول نوشته بودم اینجا اصلا اصلاح نشد. کلاًبا ۸ تا محاسبه انجام شد.

اشکال روش Modiefied label correcting algorithm : تشخیص حلقه در این روش وجود ندارد.

شما خودتان باید بررسی کنید که حلقه مثبت در روش Modiefied label correcting algorithm وجود دارد یا ندارد چون این روش پیدا کردن حلقه مثبت را برای شما انجام نمی‌دهد.

سوال : از بین روش Floyd-warshall و Modiefied label correcting algorithm کدام یک بهتر است ؟ Floyd-warshall بهتر است.

پاسخ : وقتی مسئله کوچک است مثل مثال بالا (که دارای ۴ فعالیت بود) می‌توان از روش Modiefied label correcting algorithm استفاده کرد و تشخیص حلقه را بصری خودمان انجام دهیم ولی اگر به جای ۴ تا فعالیت ۴۰۰ تا فعالیت داشتیم. حتی اگر ساعت‌ها وقت بگذاریم برای پیدا کردن حلقه ممکن است در شناسایی حلقه‌های مثبت اشتباه کنیم اینجا است که روش Floyd-warshall به کمک شما می‌آید. و یادمان نرود که روش Floy-warshall ، $O(n^3)$ بود یعنی نسبت به این محاسبه‌اش زیاد است در کل زیاد نیست چون n^3 یک چند جمله‌ای است و سخت به حساب نمی‌آید.

تفسیر بحث‌های بحرانی بودن در شبکه‌های GPR

فعالیت‌هایی که بحرانی هستند را چطور می‌توان می‌فهمید که بحرانی هستند ؟ شناوری آنها صفر است. یعنی روی مسیر بردار هستند.

مشکل فعالیت بحرانی در چیست ؟ چون اگر تأخیر یا تمدید در زمان فعالیت داشته باشید پروژه عقب می‌افتد.

تأخیر یعنی به طور مثال این کلاس ساعت ۱۵ شروع می‌شود در ساعت ۱۵:۱۵ شروع شود. (delay) تمدید یعنی استاندارد این کلاس ۲ ساعت ۱۵ دقیقه است ولی ۲ ساعت و ۴۰ دقیقه طول بکشد. یعنی زمان اش از آنی که باید باشد بیشتر گردیده است. به این می‌گویند Prolong یا تمدید شدن.

در تأخیر شما دیر کار را شروع کرده‌اید جدای از اینکه که کی تمام کنید مهم نیست ممکن است حتی زود هم تمام کرده باشید ولی دیر شروع کردید این می شود تأخیر ولی تمدید یعنی شما از آن زمان duration فراتر رفته‌اید.

فعالیت‌هایی که بحرانی هستند چه تأخیر و چه تمدید فرقی نمی‌کند هر دو باعث تأخیر پروژه می‌شود.

نکته ۱) : در شبکه‌های GPR جمله بالا درست نیست در چنین شبکه‌هایی تأخیر در یک فعالیت بحرانی قطعاً یا همیشه منجر به افزایش پروژه می‌شود. منظور duration increase همان تمدید لزوماً منجر به این قضیه نمی‌شود. تأخیر در فعالیت‌های بحرانی در شبکه GPR حتماً باعث تأخیر در پروژه می‌شود ولی تمدید لزوماً باعث تأخیر نمی‌گردد ولی ممکن است هم باعث تأخیر بشود یا نشود ولی لزوماً نیست، ممکن است باعث تأخیر پروژه بشود یا نشود. تفسیر فعالیت‌های بحرانی در شبکه‌های GPR فرق می‌کند.

نکته ۲): در شبکه‌های GPR مسیر بحرانی می‌تواند شامل حلقه باشد به شرطی که جمع Lag های حلقه (طول حلقه) صفر باشد.

همان طور که می‌دانید طول حلقه مطلقاً مثبت نیست . صفر است یا منفی. اما در مسیر بحرانی آنهایی که صفر هستند می‌تواند باشد یعنی هیچ وقت یک فعالیتی که حلقه‌ای که جمع‌اش منفی است در مسیر بحرانی قرار نمی گیرد.

از نظر بحرانی بودن در شبکه‌های GPR دو مدل فعالیت بحرانی داریم:

- Start – Critical : شروع‌اش بحرانی است یعنی تأخیر در شروع‌اش باعث تأخیر پروژه می‌شود.
- Finish-Critical : پایانش بحرانی است یعنی تأخیر در پایان اش باعث تأخیر پروژه می‌شود.

در این مثال که می بینید ۵ از کدام نوع است ؟

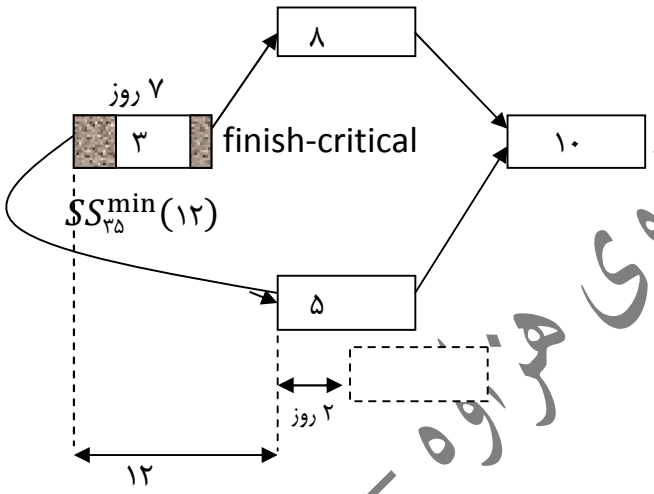
پاسخ : ۵ اصلاًبحرانی نیست اصلاًاول باید در مسیر بحرانی باشد بعد از نوع آن صحبت باید کرد. فعالیتی که اصلاً بحرانی نیست نمی‌تواند در تقسیم‌بندی بالا جا داشته باشد.

تقسیم‌بندی بحرانی ها از زاویه دیگر

- Forward- Critical: یعنی فعالیتی Forward-Critical است که دارای دو شرط باشد اولاً Start-critical باشد ثانیاً تمدید زمان‌اش (افزایش زمان فعالیت) باعث تأخیر در پروژه شود. (افزایش زمان پروژه)

- Backward-Critical : یعنی فعالیتی Backward-Critical است که دارای دو شرط باشد اولاً Finish-critical باشد ثانیاً وقتی که زمان فعالیت کاهش پیدا کند زمان پروژه افزایش می‌یابد.

مثال :

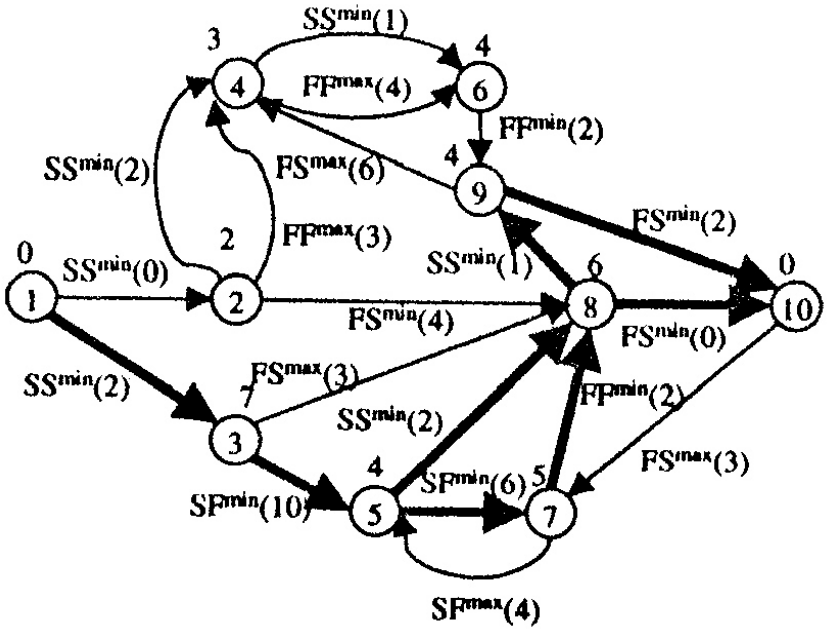


یک پروژه داریم که دارای ۴ فعالیت است (فعالیت های ۳، ۵، ۸، ۱۰) که فعالیت ۳ ، ۷ روز است. فعالیت ۱۰ پایان پروژه است. فعالیت ۳ ، Finish-critical است یعنی پایان‌اش نباید عقب بیافتد اگر عقب بیافتد پروژه را به تأخیر می‌اندازد. فعالیت ۳ به این علت که پیش نیاز ۸ است و ۸ هم پیش نیاز ۱۰ است و اگر فعالیت ۳ عقب بیافتد فعالیت ۸ عقب می‌افتد و بعد فعالیت ۱۰ عقب می‌افتد در نتیجه پروژه به تأخیر می‌افتد پس به این دلیل از نوع Finish-critical است. فرض کنید فعالیت ۵ پیش نیاز فعالیت ۱۰ است و بین فعالیت ۳ و ۵ رابطه $SS_{۳۵}^{\min}(۱۲)$ وجود دارد.

فعالیت ۳ که شروع شد باید حداقل ۱۲ روز بعد فعالیت ۵ را شروع کنیم. فرض کنید در زمانبندی این فاصله چند روز است ؟ همان ۱۲ روز است. حالا من ۳ را نمی‌توانم عقب بندازم چون پایانش بسته است. بحرانی است ولی می‌خواهیم به جای ۷ روز فشرده انجام بدهم و در ۵ روز آن را انجام بدهم . این که اشکال ندارد. می‌خواهم دو روز دیرتر شروع کنم ولی سریعتر انجام بدهم یعنی اجازه نمی‌دهم که به تأخیر بیافتد. پایانش را دست کاری نمی‌کند به جای این نقطه شروع کنم اینجا شروع می‌کنم ۲ روز دیرتر شروع می‌کنم و On-time تمام می کنم. ظاهراً همه چیز درست است ولی پایانش دست نخورده است ولی از این ور شروع‌اش به شروع ۵ که ۱۲ روز lagدارد . شما وقتی شروع این را دو روز جابجا کردید یعنی عملاً ۵ را باید دو روز جلوتر شروع کنید. اگر ۵ بره جلو، ۱۰ هم می‌تواند جابجا شود. فعالیت را فشرده کردید پروژه عقب افتاده است. این قضیه به خاطر lag می‌باشد.

- Bi-Critical : فعالیتی که شروع و پایانش بحرانی است و اضافه و کم کردن duration هر دو باعث تأخیر در پروژه می‌گردد.

وجود حلقه مثبت در این شبکه‌ها به شرطی که جمع lag ها صفر یا منفی باشد مجاز است.
وقتی فعالیتی را که duration را دست‌کاری می‌کنید چه مشکلی ایجاد می‌گردد؟
وقتی شما این فعالیت را دست کاری کردید ۷ روز را به ۵ روز کردید تا الان که این ۷ روز بود طول حلقه منفی بود مثلاً ۱- بود حال که دست کاری کردید ممکن است حلقه‌ای که درست بود طولش منفی بود مثبت شود. شما با این کار شبکه را ناموجه کرده باشید.
فعالیتی که دست‌کاری duration اش باعث ایجاد حلقه مثبت نشود را فعالیت flexibility (انعطاف‌پذیر) میگویند.
فعالیتی که دست‌کاری duration اش باعث ایجاد حلقه مثبت شود را فعالیت inflexibile (غیر انعطاف‌پذیر) میگویند.
این قضیه به بحرانی بودن ربطی ندارد راجع به حلقه مثبت است.
اگر اضافه کردن duration فعالیت باعث حلقه مثبت شود فعالیت Forward inflexible است.
اگر کوتاه کردن duration فعالیت باعث حلقه مثبت شود فعالیت Backward inflexible است.
اگر هر دو مورد بالا باشد یعنی از هر دو طرف انعطاف ناپذیر باشد فعالیت Bi-inflexible است.
مثال ۵ چه فعالیتی است.



فعالیت ۵ که در حلقه‌ها هیچ دخالتی ندارد . حلقه‌ها عبارتند از : (۶,۴) (۴,۲) (۱۰,۸,۷) (۱۰,۹,۸,۷) است آنهایی که حلقه دارند هیچکدام فعالیت ۵ در آنها دخالت ندارد پس فعالیت ۵ یک فعالیت flexibile است یعنی فعالیت ۵ که ۴ روز است را اگر ۴۰ روز یا حتی یک روزه هم انجام شود تأثیری در ایجاد حلقه مثبت ندارد. اما بین فعالیت ۴ و ۶ حلقه وجود دارد و جمع Lag ها منفی است . فعالیت ۴ که ۳ روز طول کشیده است اگر ۳ را دست کاری کنید یعنی اینها را دست‌کاری کرده‌اید گفته بودیم duration ها در دل Lagها هستند وقتی duration ها را دست کاری می‌کنید عملاً lag ها را تغییر داده اید

بعضی از این فرمول ها durationدر آن‌ها ضریب منفی دارد یعنی اگر یک پیش نیازی SS داشته باشید durationرا زیاد کنید این اشکالی ندارد چون durationرا زیاد می‌کند و چون ضریبش منفی است خوب منفی تر هم می‌شود این اشکالی ندارد ما نگران این هستیم که مثبت نشود.

تشخیص خود این forward , backward باید ببیند که کدام رابطه را دست کاری می‌کنید فعالیت ۴ و ۶ از چه نوع است ؟ ماکسیمم یا مینیمم است ؟ SS یا FS است ؟ طبق این فرمول‌ها اثرش را مشاهده کنید.

مثلاً بین ۴ و ۶ جمع Lag ها ۲- است حلقه‌ای که بین ۴ و ۶ داریم رابطه FS^{min} باشد اگر duration فعالیت ۴ را ۳ روز اضافه کنیم با این فرمول باعث می‌شود صفریش مثبت است که lag، ۳ روز اضافه شود به ۲- ، ۳ اضافه می‌شود نهایتاً ۱+ خواهد شد. در نتیجه فعالیت ۴ Forward inflexible است چون اضافه کردن زمان باعث اختلال می‌شود اما فعالیت ۶ در محاسبه duration j دخالتی ندارد یعنی مهم نیست.

فرض کنید بین ۴ و ۶ رابطه FF^{max} باشد. این الان یک عدد منفی است چه جوری ممکن است مثبت شود ؟ فعالیت j یعنی ۶ می‌شود یعنی اضافه کردن اش باعث مشکل می‌شود یعنی فعالیت ۶ می‌شود Forward – inflexible و فعالیت ۴ برعکس یعنی کم شدن اش باعث مشکل است است می‌شود backward- inflexible.

آن فعالیت‌های که در حلقه نیست خیال ما راحت است که آنها در این قضیه نیستند. Flexible هستند.

جلسه هشتم

پارت هشتم طبق سیلابسی که قبلاً داشتیم این دو مدل باید صحبت شود.

زمانبندی با پروژه Start time dependent costs یا هزینه‌های وابسته به زمان شروع

زمانبندی پروژه با هدف ماکسیمم کردن NPV

که به طور واضح هر دو تا اهداف مالی هستند اهداف هزینه و سود برخلاف دو جلسه قبلی که اهداف زمان (C_{max}) را به عنوان هدف داشتیم اینجا اهداف مالی هستند.

زمانبندی پروژه با هزینه‌های وابسته به زمان شروع :

هزینه وابسته به زمان شروع یعنی چه ؟

به طور مثال وقتی شما کاری را امروز انجام بدهید یا هفته بعد یا یک ماه دیگر، هزینه‌اش می‌تواند به خاطر منبع گران‌تر یا ارزان تر شود فرق می‌کند. یا فرض کنید هوا سردتر می‌شود انجام این کار هزینه‌اش بالاتر می‌رود شما اگر دو روز پیش می‌خواستید بتن‌ریزی انجام بدهید یا اینکه بخواهید امروز انجام بدهید به خاطر شرایطی که از نظر آب و هوایی تأثیر در این کار دارد. هزینه‌اش یکسان نیست ولی واضح‌ترین اش تغییر قیمت منبع است.

فرض می‌کنیم که یک شبکه AON داریم.

Duration هر فعالیت عدد صحیح ثابت است هر فعالیت i متحمل می‌شود یک هزینه W_{it} را اگر فعالیت شروع شود در زمان t. پس W_{it} هزینه انجام فعالیت i است اگر در زمان t شروع شود. که می‌تواند امروز ۱۰۰ هزار تومان و فردا ۱۲۰ هزار تومان باشد. به خاطر اینکه ممکن است منبع گران‌تر باشد دسترسی‌پذیراش کمتر باشد.

فرض می‌شود یک upper bound به اسم T داریم. یک dead line به اسمT داریم برای زمان پروژه .

سوال : چرا در اهداف C_{max} برای پروژه dead line تعریف نکردیم؟ چرا اهداف مالی باید dead line داشته باشد؟

در هر دو حالت منبع نداریم. از این لحاظ یکسان هستند.

اهداف به دو دسته منظم و نامنظم تقیسم می‌شوند:

اهداف منظم اهدافی است که ما دنبال کاهش زمان و سرعت دادن به کار داریم نتیجه که دیگر dead line نمی‌خواهد یعنی خود هدف کششی است یعنی دنبال این است که پروژه را زود تکمیل بکند. فعالیت‌ها را به سمت صفر می‌کشد بنابراین دیگر dead line نمی‌خواهد.

اما اهداف مالی اهداف نامنظم هستند چون نامنظم هستند اگر dead line نذارید اصلا این کار انجام نمی‌شود.

به طور مثال اگر تمرینی به شما داده شود که آن را انجام بدهید ولی زمان مشخص نباشد به طور آن را اصلاً انجام نمی‌دهید چون نامنظم است. هیچ انگیزه‌ای یا Force ایی یا الزامی برای انجام آن نمی‌باشد. ولی اگر dead line

داشته باشیم یا جریمه داشته باشد خلاصه به نوعی محدودیت داشته باشد به خاطر آن محدودیت ناچارید که آن را انجام دهید. چون هدف یک هدف نامنظم (واگرا) است dead line نداریم پروژه هیچ وقت انجام نمی شود.
هر وقت هدف نامنظم بود حتماً dead line می‌خواهد واگر منظم بود نمی‌خواهد.
فرض می‌کنیم فعالیت‌ها پیش‌نیازی اش از نوع چی باشد ؟
قبلاً هم Finish to start و هم از GPR را در هدف C_{max} صحبت شد. در این جا هم همین طور است هم می‌تواند Finish to start باشد و هم GPR باشد. چون دو تا مدل داریم مدل اول را با GPR حل می‌کنیم و مدل دوم را با Finish to start حل می‌کنیم. تا به نوعی هر دو را اینجا تمرین کرده باشیم.
پس در مدل اول Start time dependent costs را می‌خواهیم Minimize کنیم پیش‌نیازی ها از نوع GPR است .
در رابطه با GPR قبلاً صحبت کردیم هشت نوع پیش‌نیازی داریم که البته باید استاندارد کنیم و به حالت S^{min} تبدیل کنیم. خط ۲ مشخص است، هدف این مسئله زمانبندی فعالیت های پروژه است که این هزینه‌ها را مینیمم کنیم. قبلاً از انجام این کار باید مسئله را مدل‌سازی کنیم . اول باید مدل مسئله را داشته باشیم. نحوه مدل‌سازی را در درس تحقیق خوانده‌اید؟
گام‌های مدل‌سازی در تحقیق

- تعریف متغیر تصمیم
- تابع هدف
- محدودیت

متغیر: خوب برای اینکه مسئله را مدل کنیم متغیرش این پایین نوشته شده اگر zero-one variable های X_{it} متغیرهای صفر و یک یا باینری X_{it} را به این صورت تعریف کنیم.
X_{it} : چه زمان یک می‌شود ؟
$X_{it} = ۱$ وقتی فعالیت i شروع در زمان t باشد و اگر شروع نشود صفر است $X_{it} = ۰$
پس متغیرهای این مسئله متغیرهای باینری (صفر و یک) X_{it} هستند. اگر فعالیت i در زمان t شروع شود می‌شود یک و اگرفعالیت i در زمان t شروع نشود صفر می‌شود.
قبلاً مسئله باینری (مدل صفر و یک) دیدید دیگه نه ؟ در تحقیق یک در لیسانس مسئله تخصیص یادتان هست؟
X_{ij} اگر کار i را به ماشین j بدهیم یک می‌شود و اگر ندهیم صفر می‌شود. یک مسئله باینتری محض بود. اگر کار (فعالیت) i را در زمان t شروع کنید چقدر هزینه دارد ؟ در اسلاید قبلی اسمش W_{it} بود. پس در کل این پروژه چقدر هزینه دارد؟ Sumation همه فعالیت‌ها و کل زمان پروژه $\sum_i \sum_t W_{it}X_{it}$ می‌شود. مجموعه هزینه‌ها یعنی تابع هدف. ما می‌خواهیم این را مینیمم کنیم پس X_{it} به تنهایی متغیر بود اما این عبارت الان تابع هدف این مسئله است که می‌خواهیم مینیمم کنیم.
اگر $X_{it} = ۱$ شود یعنی فعالیت i در زمان t شروع بشود پس هزینه‌اش هم به ضربدر یک به حساب می آید و اگر $X_{it} = ۰$ شود یعنی فعالیت i در زمان t شروع نشده است پس هزینه به روز آن حساب پروژه نوشته نمی‌شود
$\sum_i$ یعنی باید برای همه فعالیت ها جمع بزند و $\sum_t$ یعنی باید برای همه زمان‌ها آن را جمع بزنید پس این $\sum_i \sum_t W_{it}X_{it}$ می‌شود تابع هدف که باید مینیمم شود چه محدودیت‌هایی داریم ؟
این مدل را آقای مهرین پیشنهاد کرد
Subject to چه چیزی را نشان می‌دهد ؟

$\sum_t X_{it} = ۱$: X_{it} این بود که فعالیت i شروع بشود یا نه ؟ چرا باید $\sum_t X_{it} = ۱$ باشد ؟ برای اینکه یک فعالیت فقط یک جا می تواند شروع شود شما نمی توانید یک فعالیت را ۳ بار شروع کنید این محدودیت نشان می دهد که هر فعالیتی مثل i باید فقط یکبار شروع شود اگر این قسمت را ننویسید چه اتفاقی می‌افتد؟

مثلاً برای فعالیت ۳ ممکن است $X_{۲} = ۱$ و هم $X_{۷} = ۱$ و زمانی که بخواهید اجرا کنید. ابهام به وجود می‌آید. مثلاً اگر به شما گفته شود که شروع ترم ۷ است و بعد دوباره بگویند ترم ۹ ام شروع می‌شود ؟ سوال پیش می‌آید که آیا ۷ ام شروع می‌شود یا ۹ ام . باید یکی بگویم. خلاصه باید یک نقطه شروع برای فعالیت داشته باشد اما وقتی این را می‌نویسیم این را چطور اعمال می‌کند.

$X_{۳۱}, X_{۳۲}, X_{۳۳}, X_{۳۴}, \ldots X_{۳t} = ۱$ یعنی فعالیت ۳ از روز یکم تا آخر پروژه فقط می‌تواند یکی ۱ داشته باشد. یک نقطه شروع داشته باشد. البته من این را فقط برای فعالیت ۳ نوشتم.

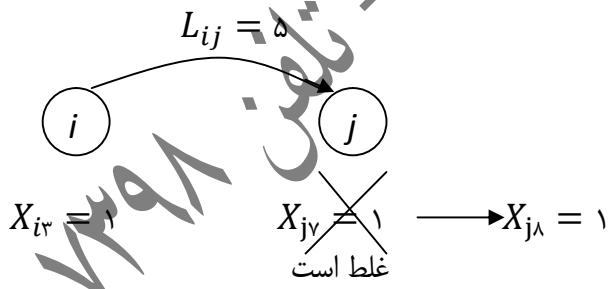
حالا شما حساب کن این را باید چند بنویسید؟ این جلوش نوشته است i اندیس است.

i متعلق به V است V مجموعه گره‌ها است . حالا شما چند تا گره دارید ؟ مثلاً ۲۰ تا، ۳۰ تا ،۱۰۰ تا یعنی هر چند تا فعالیت دارید باید برای هرکدام یک خط بنویسید. درست شد. پس این یک خط یک محدودیت نیست n تا یا ۵۰ تا ، ۱۰۰ تا محدودیت به تعداد فعالیت است. فرض کنیم n باشد. یک محدودیت دیگری هم داریم .

L_{ij} همان Lag یعنی پیش‌نیازی است. اگر L_{ij} را تشخیص ندهید خلاصه این که چون مسئله منبع ندارد تنها محدودیت موجود پیش‌نیازی است این باید محدودیت پیش‌نیاز باشد محدودیت پیش‌نیازی چه طور کار می‌کند؟

i پیش نیاز j است و بالای آن نوشته شده است $L_{ij} = ۵$ یعنی i که شروع شد حداقل ۵ روز باید بگذرد که j را شروع کنید. $start\ to\ start$ است.

فرض کنید $X_{i۳} = ۱$ و $X_{j۷} = ۱$ چه اشکالی دارد ؟

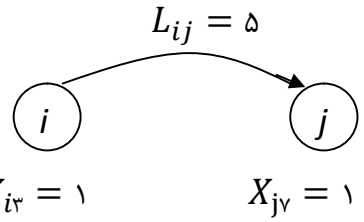


Lag بین این دو فعالیت چهار روز می‌باشد در صورتی که گفته شده است ۵ روز فاصله تا شروع j باشد. پس غلط است اگر ۸ یا ۹ بود درست بود. در محدودیت شبیه این مسئله یعنی این محدودیت قرار است جلوی چنین جواب‌هایی را بگیرد. ما محدودیت پیش‌نیازی را برای این می‌نویسیم که آن lag ها را به اصطلاح رعایت کند یعنی جواب‌هایی را بدهد که lag ها در آن رعایت شده باشد پس نمونه یک جواب غلط بود.

$q \longleftarrow \sum X_{iq}$ خودش یک اندیس است یک شمارنده می‌باشد. که از t تا T که همان $dead\ line$ است.

$$\sum_{q=t}^{T} X_{iq} + \sum_{q=۰}^{t+l_{g+۱}} X_{jq} \leq ۱ \qquad (i,j) \in t\,, \quad t = ۰,۱,\ldots,T$$

T=۲۰ t=۳



$۱ + ۱ \nleq ۱$

$$(X_{i۳} + X_{i۴} + \cdots + X_{i۲۰}) + (X_{j۰} + X_{j۱} + \cdots + X_{j۷}) \leq ۱ \qquad ۱ + ۱ \leq ۱/$$

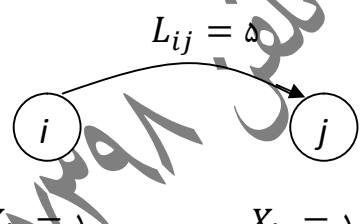
فرض می‌کنیم $T=۲۰$ یعنی *dead line* پروژه مساوی ، ۲۰ روز است و $t=۳$ امروز سوم است . X_{iq} که q از ۳ تا ۲۰ است.

$(X_{i۳} + X_{i۴} + \cdots + X_{i۲۰})$ باضافه تیکه دوم X_j که q از صفر است تا $t=۳$ به اضافه *lag* می‌شود ۵ منهای یک ، ۴ $(X_{j۰} + X_{j۱} + \cdots + X_{j۷})$ کمتر مساوی یک است.

$$(X_{i۳} + X_{i۴} + \cdots + X_{i۲۰}) + (X_{j۰} + X_{j۱} + \cdots + X_{j۷}) \leq ۱ \qquad ۱ + ۱ \leq ۱/$$

البته عددی نوشتم فرض کردم dead line، ۲۰ باشد و t=۳ باشد این عبارت فقط با این اعداد است اگر اعداد را عوض کنید ، یه طور دیگری نوشته می شود این عبارت چه طوری جواب غلط بالا را مانع می شود؟

T=۲۰ t=۳



$۱ + ۱ \nleq ۱$

$X_{۳۲} = ۱$

$X_{۳۷} = ۱$

$X_{۳۱} + X_{۳۲} + X_{۳۳} + \cdots + X_{۳۷} + \cdots + X_T = ۱$

$X_{۴۱} + X_{۴T} + \cdots$

$X_{۴۲} = ۱$

اینها نباید هر دو یک بشوند؟ $X_{i۳}$ اینجا است $X_{j۷}$ هم اینجا است اگر هر دوی اینها قرار باشد یک باشند یک با یک می‌شود ۲ که کمتر از یک نیست . نتیجه : بین این دو تا فقط یکی می‌تواند یک باشد هر دو نمی‌تواند یک باشد

بنابراین جواب غلط دیگر اتفاق نمی‌افتد. جواب‌های غلط دیگر چی ؟ جواب‌های دیگری هم داریم که آنها غلط هستند. من برای t=۳ نوشتم برای تمام tها از صفر تا t

برای تمام پیش‌نیازی‌ها و برای تمام زمان‌ها این را باید بنویسید پس این یک خط نیست برای هر روزی و برای هر دو تا i,j باید بنویسید من برای i,j خاص در روز سوم نوشتم. والا جواب‌های دیگری هم هستند که غلط هستند باید براشون نوشته شود در کل به اصطلاح جواب‌های را بدهد که lag ها در آن رعایت شده باشد.

سوال : این مدل چند متغیر و چند محدودیت دارد؟

این کلمه i اندیس است یک شمارنده است i فعالیت‌های شما است از ۱تا n است.

n×T ← متغیر در این مدل داریم.

X_{it} i تا n است و t هم تا T است.

چند تا محدودیت داریم ؟ n تا است چون به ازای هر i باید نوشته شود i=۱,...,n این چند تا است ؟ این t درست است این هم هست دیگه، چند تا بردار داریم ؟ تعداد بردارها را من با این نشان دادم E مجموعه بردارها و این می‌شود $|E|$ تعداد بردارها پس n×T متغیر داریم n باضافه تعداد بردارهاضرب در T هم محدودیت داریم ← $n + |E| \times T$

این می‌شود به اصطلاح ابعاد این مدل و همه متغیرهایش صفر و یک است.

سوال دوم : آیا تنها راه مدل‌سازی این مسئله فقط راه حل بالا است ؟

خیر ، کلاً مدل‌های عدد صحیح صفر و یک را می‌شود متنوع و چند جور مدل کرد، مثلاً همین مثالی را که آقای مهرینگ پیش‌نیازی‌هایش را گفته می‌تواند اینطوری هم بنویسید.

یعنی به جای اینکه این قبلی باشه گفتند با همان متغیرها تابع هدف پیش نیازی‌اش را می شود اینطوری نوشت.

شما می‌توانید پیش‌نیازی را چند حالت بنویسید.

حالا این بهتر است یا قبلی ؟ چرا این بهتر است ؟ تعدادش کمتر است

متغیرهاش که همان است اینم که همان n تا است و فقط این تعدادش به تعداد بردارها است. دیگه ضرب در T را ندارد. یعنی این قسمت اش که ضرب در T می‌شد در این مدل جدید وجود ندارد. بهتر شد یا بدتر ؟ بهتر شده است چون کمتر شده است .

$$\sum t(X_{j} - X$$

متغیر ← $n \times T$

محدودیت ← $n + |E|$

کلاً ویژگی‌های مدل خوب عبارتند از :

- متغیر کم داشته باشد
- محدودیت کم داشته باشد.
- تونوک بودن ماتریس ضرایب

$X_{i۴} - X_{i۷} \leq ۱$ باشد. ولی این محدودیت با این محدودیت موقع حل کردن اثرش با هم برابر نیست تعداد این یکی است و اینم یکی ولی این تونوک است به اصطلاح، یعنی اکثر متغیرها ضریب صفر هستند فقط دو تا متغیر در آن است ولی بالایی متغیر بیشتری در آن است.

همین مثال، همین پیش‌نیازی را اینجا یکبار نوشتیم چند تا شده است ؟ چند تا متغیر اینجا می‌بینید؟
از ۳ تا ۲۰ ⟵ ۱۸ تا و اینم ۸ تا جمعاً ۲۶ تا یعنی در این محدودیت ۲۶ تا متغیر وجود دارد.
حالا شما فرض کنید همین پیش‌نیازی را با این بنویسید. این الان ۲۶ تا متغیر دارد اما اگر طبق این فرمول بنویسید همین محدودیت چند تا ضریب خواهد داشت؟ زیگما است t از صفر تا ۲۰ است . آن وقت هر زیگمایی در آن ۲ تا متغیر دارد از ۰ تا ۲۰ می‌شود ۲۱، ۲۱×۲=۴۲ یعنی درست است که تعداد کمتر است وماتریس اش شلوغ‌تر است. بنابراین نمی‌شود بلافاصله گفت که این بهتر از قبل است . ظاهراً محدودیتش کمتر است آره درست است ولی محدودهایت کمتر ولی شلوغ‌تر ، شلوغ‌تر بودنش ممکن است کار را خراب کند آره اگه کمتر بود و ماتریس‌اش هم خلوت تر بود خوب بود ولی این شلوغ تر است. معمولاً اگر بخواهید مدل ها را مقایسه کنید باید مثال با آن حل کنید یعنی چی؟
۲۰ تا، ۳۰ تا ،حداقل ۳۰ تا مسئله را با این مدل را با یکی از نرم افزارها حل کنید و سپس همین ۳۰ تا مسئله را با این یکی مدل حل کنید. ببینید CPU time (زمان محاسباتی) در اولی متوسط چند ثانیه شد در دومی چند ثانیه گردید. هر کدام که CPU کمتری داشت آن مدل در کل سریعتر جواب داده است بهتر است. منظور از بهتر در اینجا سریعتر است ولی از نظر دقت هر دو مدل مدل دقیقی هستند و جواب را به شما می‌دهد. ولی مدلی که سریعتر با آن به جواب برسید بهتر است.
در نرم افزار هر سه دخالت دارند ولی معمولاً محدودیت کمتر مهمتر از دو تا دیگری است تو اولویت اول، معمولاً محدودیت است. این دو تا حالا بستگی به مسئله دارد محدودیت‌ها، تعداد محدودیت‌ها شاخص اصلی است ولی تنها شاخص نیست. به طور مثال یک مدل ۵ تا محدودیت و مدل دیگر ۶ تا محدودیت دارد در نگاه اول بلافاصله نمی‌توانید بگویید که کدام یک بهتر است با اینکه مدل اول محدودیت کمتری دارد. کم بودن محدویت شاخص است ولی تنها شاخص نمی‌باشد.
پس دو مدل برای یک مسئله داریم مدل سوم . اسلایدهایی که داریم که این مسئله را ۳ بار مدل کرده است. در این مدل ، متغیر را یک طور دیگر تعریف کرده است. آقای مهرین سال ۲۰۰۰ یک مدل عدد صحیح دیگری را با استفاده از این متغیر پیشنهاد داد. متغیر Z_{it}

$$\min \sum_i \sum_t \overline{w}_{it} z_{it}$$

subject to

$$z_{iT} = 1 \qquad i \in V$$

$$z_{it} - z_{i,t+1} \le 0 \qquad i \in V, t = 0,1,...,T$$

$$z_{j,t+l_{ij}} - z_{it} \le 0 \qquad (i,j) \in E, t = 0,1,...,T$$

$$z_{it} \in \{0,1\} \qquad i \in V, t = 0,1,...,T$$

متغیر Z_{it} چه زمان یک می‌شود؟
اگر فعالیت i در زمان T یا زودتر شروع شود یک می‌شود و اگر شروع نشود صفر می‌گردد. در قبلی کلمه زودتر نداشت. به صورت زیر تعریف شد .(اگر فعالیت i در زمان t شروع شود یک و اگر شروع نشود صفر می‌شود.) با این تعریف Z که با X قبلی، کلاً تعریفش محتوایی فرق می‌کند دوباره تابع هدف نوشته است محدودیت‌ها و فقط محدودیت‌هاش براساس متغیر ، چون متغیر ماهیتاً عوض شده ، تعریف‌ها و نوشتن محدودیت‌هاش ماهیتاً فرق می‌کند دقت کنید در

قبلی $\sum_t X_{it} = ۱$ یعنی هر فعالیت یک جا باید شروع شود چون در آنجا شروع شدن در نقطه متغیر بود ولی الان شروع شدن یا زودتر متغیر است به جای آن نوشته شده است $Z_{iT} = ۱$ که T در اینجا منظور dead line است.
تعریف Z : اگر فعالیت i در زمان t یا زودتر شروع شود یک می‌شود. Z_{iT} باید برای همه فعالیت‌ها یک باشد.
سوال اگر $Z_{iT} = ۱$ نوشته نشود چه اتفاقی می‌افتد ؟
به طور مثال باید همه فعالیت‌ها قبل از ۲۰ روز شروع شده باشد وقتی می‌گویند dead line سه ماهه است یعنی همه فعالیت‌ها قبل از ۳ ماه یا زودتر شروع شده باشند. اگر نوشته نشود یه جوابی می‌دهد مثلا $Z_{j۲۳} = ۱$ چه اشکالی دارد؟ اشکالش این است که dead line ، ۲۰ روز است ولی این ۲۳ شروع می شود. ۲۳ به درد نمی‌خورد باید تا ۲۰ کار تمام شود. بنابراین نباید فعالیتی به بعد ۲۰ام بماند. همه فعالیت‌ها باید قبل ۲۰ ، قبل T ، قبل dead line، Start خورده باشد.
خوب بعدی چه می‌گوید؟ البته اگر بخواهید بعدی را ترجمه کنید بهتر است Z را به آن سمت انتقال دهید.

$$Z_{i,t+۱} \geq Z_{it}$$

به طور مثال : امروز ۲۱ ام است $Z_{i\,۲۱} = ۱$ یعنی فعالیت i امروز یا قبلاً شروع شده است یعنی فعالیت در حال انجام است یا امروز شروع شده یا قبلاً. به هر حال این فعالیت Start خورده است. حالا فردا اگر گفته شود $Z_{i\,۲۲} = ۰$ است یعنی فعالیت تا این لحظه هنوز شروع نشده است آن وقت با قبلی تضاد دارد یا خیر ؟ بله تضاد دارد اگر روز ۲۱ ام فعالیت شروع شده بوده است بنابراین ۲۲ ام هم شروع شده است نمی‌شود که روز ۲۱ کار را انجام شده بدانید ولی روز ۲۲ام بگید هنوز شروع نشده است تضاد دارد. چون روز قبل‌اش گفته شده شروع شده است اون قطعاً باید از این بیشتر باشد یا حداقل مساوی باشد ولی کمتر از آن نمی‌تواند باشد.

۱- همه فعالیت‌ها قبل dead line باید شروع شده باشد.

۲- $Z_{it} \leq Z_{i,(t+۱)}$

$Z_{i\,۲۱} = ۱$ و $Z_{i\,۲۲} = ۰$

۳- $Z_{j,t} + l_y - Z_{it} \leq ۰$ ⟵ پیش‌نیازی

چون روز قبل‌اش گفته شده شروع شده است اون قطعاً باید از این بیشتر باشد یا حداقل مساوی باشد ولی کمتر از آن نمی‌تواند باشد. بعدی هم که l_{ij} دارد قابل حدس است این که l_{ij} دارد یعنی پیش‌نیازی را عنوان می‌کند.
سه مدل برای یک مسئله به اسم start time dependent costs ارائه شده است هر سه تا مدل ، مدل‌های صفر و یک خالص هستند.
دو مدل صفر و یک داریم :

- Mix integer یعنی یکسری متغیر عدد صحیح دارد و یکسری هم ندارد یعنی درهم است
- Pure integer یعنی خالص‌اند یعنی تمام متغیرها عدد صحیح هستند متغیر غیر صحیح نداریم.

پس این مسئله صفر و یک خالص است. در تحقیق ۲ بیان شد که با روش شاخه و کران به اسم بالاس یا بالاش حل می‌شود.
Linear programming relaxation این مسئله عدد صحیح ، integral(عدد صحیح) است یعنی LPR این مسئله عدد صحیح است. Relax کردن یک محدودیت یعنی چی ؟ آزادسازی یک محدودیت یعنی چی ؟
آزاد سازی یک محدودیت یعنی که محدودیتی که هست را بذارید کنار و به آن توجه نکنید.
آزاد سازی LP یعنی چی ؟ یعنی مسئله شما IP است (عدد صحیح) است، Relax کنید به صورت LP حل کنید یعنی با سیمپلکس LP را حل کنید. یعنی مسئله‌ای که به صورت IP بود را IP اش را کنار بگذارید و Relax کن و فرض

کن که عدد صحیح نیست با LPحل کن. آن وقت در ادامه LPR مسئله IP ، Intergral (عدد صحیح) است عدد صحیح بودنش را توجه نکردن و با Simplex حل کردی ولی بازم جواب عدد صحیح شد. این ویژگی خوب است یا بد؟ مسئله شما ذاتاً عدد صحیح است باید با روش‌های سخت عدد صحیح مثل بالاس حل کنید ولی این جمله بسیار امیدوارکننده می‌گه این مسئله یک ویژگی خوب دارد. ویژگی است این است که اگر ما عدد صحیح بودن را توجه نکنید و با LPحل کنی بازم جواب عدد صحیح می‌شود.

نتیجه : روش بالاس لازم نیست با Simplexحل کنید (به خاطر داشتن ویژگی بالا)

سوال : چه مسئله‌ای در تحقیق ۱ می‌شناسید که همین ویژگی را داشته باشد ؟

پاسخ : مسئله حمل و نقل و مسئله تخصیص . این مسائل عدد صحیح هستند ولی ما آنجا این مسائل را با شاخه و کران حل نمی‌کردیم اصلاً ما در تحقیق یک روش شاخه و کران نمی‌دانستیم. با همان تکنیک‌های LP حل می‌کردیم و در آخر باز هم جواب عدد صحیح به دست می‌آمد. این نکته را داشتند که LPRشان Integral (یک عدد صحیح) بود مدل یک فقط مدل را یافتیم حل نکردیم چون حل آن با Simplex انجام می‌شود.

سوال : از نظر تصویری (گرافی) یا از نظر فضای جواب می‌توانید توضیح دهید چه ویژگی‌هایی دارند ؟ در واقع چه ویژگی‌های دارند با اینکه با Simplex حل می‌کنی ولی بازم عدد صحیح در می‌آید. چون Simplex معمولاً عدد اعشاری می‌دهد و این مسئله را عدد صحیح می‌دهد. این مسئله شکلش، فضای حلش چه ویژگی‌های دارد ؟

در Simplex فضای جواب یک چند ضلعی محدب بود وجواب بهینه گوشه‌ها است. یعنی وقتی با simplex حل می‌کنید یک گوشه به دست می‌آید چه طوری این مسئله با Simplexحل می‌شود ولی عدد صحیح است؟

معلومه که گوشه‌های این مسئله همه عدد صحیح هستند. گوشه‌های این مسئله مختصات عدد صحیح دارند. به خاطر همین است که با Simplex هم که حل می‌کنید عدد صحیح است چون simplex همین گوشه ها را می‌دهد.

خلاصه : هر مسئله‌ای که گوشه‌اش مختصات عدد صحیح داشته باشند می‌توانیم عدد صحیح بودنش را کنار بگذاریم و با Simplexحل کنیم و خیالتان راحت باشد که جواب عدد صحیح می‌گیرید گوشه‌ها را فقط می‌دهد که عدد صحیح است.

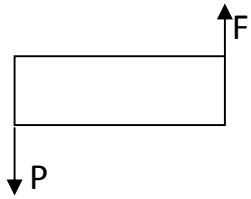
مدل یک به اتمام رسید.

مسئله دوم ، مسئله Max NPV (مسئله ماکیسیمم کردن Net present value) یک پروژه است. هدف مالی مثبت سود است می‌خواهیم NPV پروژه، ارزش فعلی خالص را حداکثر کنیم.

اقتصاد مهندسی

اگر این یه برهه زمانی باشد شما در ابتدای این برهه زمانی مبلغی مثل P را در پروژه‌ای، در حساب بانکی، طرح اقتصادی سرمایه‌گذاری کنید بعد از پایان دوره مبلغی را که کسر می‌کنید به این می‌گوییم present value پولی که الان دارید ولی پولی را که بعداً می‌گیرید که به اصطلاح سود حاصل از آن پروژه است هم Future Valueمی‌گویند که با F نشان می‌دهند که معمولاً اگر i نرخ مثبتی باشد سودآور باشد. F از P بیشتر است.

نحوه محاسبه P را ضرب در یک باضافه نرخ بهره‌برداری یکسال ، اگر n سال است به توان n و برعکس می‌توانید از رویF هم P را حساب کنید. فقط این فرمول توانش برعکس است منفی است.



$$F = P(1+i)^n = 1000000 + (1+0.2)^1 = 1200000$$
$$P = F(1+i)^{-n}$$

۲۰٪ ⟵ اسمی		۱۲۰۰
۲۱٪ ⟵ موثر		۱۲۱۰

پس می‌توانید از روی ارزش فعلی ، ارزش آتی را حساب کنید یا برعکس از ارزش آتی ، ارزش فعلی را حساب کنید. ارزش فعلی را در این نرخ به آن فاکتور می‌گفتند.

مثلاً یک میلیون تومان را با نرخ ۲۰٪ توی بانک سرمایه‌گذاری کنید بعد از یکسال چقدر دریافت خواهید کرد.

$1000000 \times (1+0.2)^1 = 1200000$ یک میلیون اصل پول و ۲۰۰ هزار تومان سود سرمایه‌گذاری است.

این فرمول‌ها با این فرض است که بهره آخر دوره به حساب شما واریز شود یعنی اگر شما اول فرودین حساب باز کرده باشید ۲ فرودین همان یک میلیون تومان در حساب است. اول اردیبهشت، مهر، آبان ، آذر ، بهمن ، اسفند همان یک میلیون تومان است ۲۹ اسفند که شد ۱۲۰۰۰۰۰ در حساب خواهد بود. سود آخر دوره فرض می‌شود که واریز شود اما ممکن است این طور نباشد شما در تبلیغات بانکی، سرمایه‌گذاری متوجه سود روز شمار، یا پرداخت سود هر سه ماه یکبار شده‌اید. این چه تفاوتی با قبلی می‌کند؟ مثلاً فرض کنید پرداخت سود هر ۶ ماه یکبار باشد. بنابراین اگر یک میلیون سرمایه‌گذاری کرده باشید بعد از ۶ ماه اول یعنی آخر شهریور، یک میلیون که داشتید اگر i برابر ۲۰ ٪ باشد برای ۶ ماه ۱۰ درصد می‌شود، ۱۰ درصد یک میلیون یعنی ۱۰۰ هزار تومان به حساب شما واریز می‌شود یعنی در طول شش ماه یعنی از اول مهر به بعد حساب شما ۱,۱۰۰,۰۰۰شده است. این ۱۰۰ هزار تومان بابت ۱۰٪ سود شش ماه اولیه است آن وقت آخر سال ، آخر اسفند ده درصد ۱,۱۰۰,۰۰۰ می‌شود ۱,۲۱۰,۰۰۰ . در کل قضیه چه فرقی کرده است؟

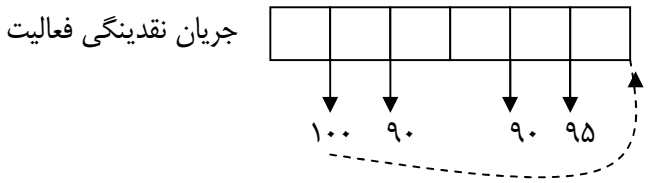
در کل قضیه فرقش این بود که آخر سال بهره سال می‌کرد شد و ۱,۲۰۰,۰۰۰ ولی الان ۱,۲۱۰,۰۰۰شده است. این ۱۰ تومان اضافی بابت سود سود است به سود ۶ ماه اول یک سودی در شش ماه دوم تعلق گرفته است نرخ بهره در هر دو این مثال ۲۰٪ است ولی عملاً ۲۱٪ در دومی سود گرفته‌اید. پس اون ۲۰٪ چی بود ؟ آن اسمی و موثر است. اسماً ۲۰٪ است ولی آخر شش ماه داره سود می‌دهد موثرش بیشتر می‌شود که در اینجا ۲۱٪ شد. اما اگر آخر سال حساب کنید دیگه اسمی و موثرش هر دو یکی است. اسمی و موثر ۲۰٪ است. ولی در این مثال ۲۱٪ سود دارد.

حال اگر به جای ۶ ماه بازه را کوچکتر کنیم یعنی ۳ ماه ،یا یک ماه یا یک هفته یا یک روز کنیم تا کجا می‌توانیم این بازه را کمتر کنیم limit اش لحظه ای، Compounding پیوسته (مرکب شدن پیوسته) یعنی به صورت لحظه‌ای محاسبه می‌شود. این سود برای کسی که سرمایه‌گذاری می‌کند بهترین Option این است که به صورت لحظه‌ای سود محاسبه شود آن وقت اگر Compounding به صورت پیوسته باشد فرمول‌ها به صورت زیر می‌گردد:

سود لحظه‌ای
$$\begin{cases} F = Pe^{\propto t} \\ P = Fe^{-\propto t} \end{cases}$$

۳۰۰

۶۰



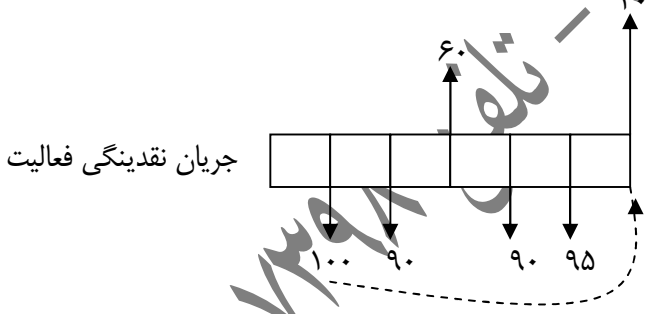
$$C_i = -۱۰۰e^{۵\propto} - ۹۰e^{۴\propto} + ۶۰e^{۳\propto} - ۹۰e^{۲\propto} - ۹۵e^{\propto} + ۳۰۰$$

C_i سود دهی اگر مثبت باشد

تفاوت این فرمول‌ها با فرمول های قبلی در این است که فرمول‌های الان برای این است که [illegible] شود اما قبلی‌ها سود در آخر دوره محاسبه می‌گردید. در ادامه بحث با فرمول‌های سود لحظه‌ای سر و کار داریم. این فرمول‌ها بیشتر رایج هستند.

یک فعالیتی که ۶ هفته طول می‌کشد که انجام شود هر هفته که کار انجام می‌شود باید هزینه کنید این کار چه طور پیش می‌رود ؟ با تزریق منبع و سرمایه باید هزینه کنید که این کار جلو برود. ممکن است به طور مقطعی درآمدی ازش بدست بیاورید مثلاً فاز فلان پروژه بهره‌برداری شود از منفعتی هم بگیرید ولی عمدتاً در طول پروژه هزینه و در آخر پروژه درآمد (بهره‌برداری) نهایتاً این روند را جریان نقدینگی می‌گویند. در طول یک فعالیت در طی روزهای مختلف هزینه یا بعضاً مبالغی درآمد ازش بدست می‌آید.

در کل اگر این ۱۰۰ باشد، این ۹۰ باشد و این ۹۵ . این ۶۰ باشد و این ۳۰۰ در کل این پروژه منفعت دارد یا هزینه ؟



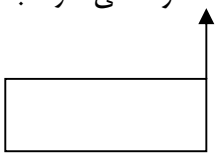
اگر همین‌طوری جمع بزنید و بالا و پایین را مقایسه کنید یعنی بویی از اقتصاد مهندسی نبردید این مقایسه کلی نمی‌تواند باشد چون این مبلغ‌ها در time های مختلفی داره هزینه می‌شود بنابراین قابل قیاس نیست مگر این که همه را طبق این فاکتورها به یک نقطه واحد منتقل کنیم نقطه واحد دیگر ، مهم نیست همه را اول پروژه ، وسط پروژه یا آخر پروژه منتقل کنیم مهم نقطه واحد بودن است. تمام جریان‌های نقدی با فاکتورهای که آنجا خواندید به یک نقطه واحد منتقل می‌کنید آن وقت برآیند مثبت و منفی مبنای قضاوت است که اقتصادی است یا نه ، که معمول اش یا ابتدا است یا انتها.

در این جا به انتها منتقل می‌کنیم. این ۱۰۰ واحد هزینه هفته اول را ۵ هفته به آینده جابه جا می‌کنیم. یعنی P را به F تبدیل می‌کنیم وقتی به آینده انتقال پیدا میکند یعنی باید F را برایش محاسبه کنید که به صورت زیر نوشته می‌شود.

$$C_i = -۱۰۰e^{۵\propto} - ۹۰e^{۴\propto} + ۶۰e^{۳\propto} - ۹۰e^{۲\propto} - ۹۵e^{\propto} + ۳۰۰$$

۳۰۰ دیگر نیاز به $\propto$ ندارد چون خودش در انتها می‌باشد و نیاز به جابجایی ندارد.
که در کل C_i به دست می‌آید که اگر مثبت باشد یعنی این فعالیت اقتصادی بوده است و یک سودی داشته است و اگر منفی باشد خوب به سمت پایین رد می‌کنیم. هزینه پس خلاصه جریانات نقدی فعالیت را به انتها انتقال می دهیم انتهای یک فعالیت یک عدد می‌دهیم در کل این فعالیت توی این مدت انجام شود اینقدر هزینه و اینقدر درآمد داشته است.

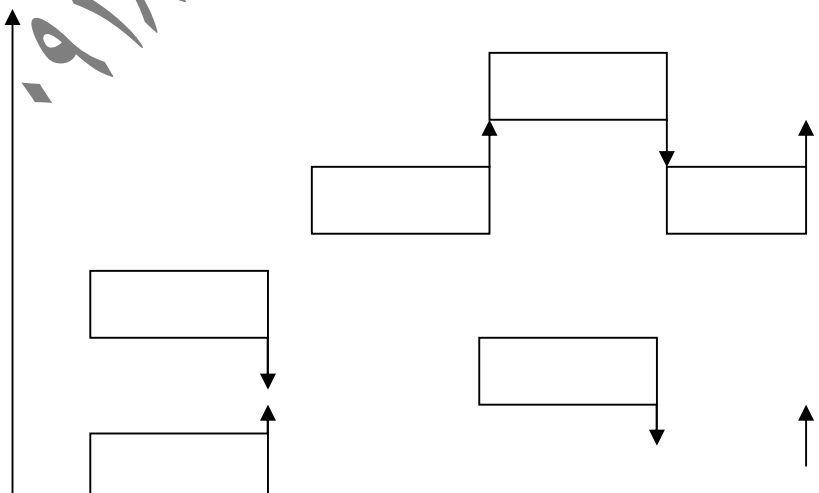
C_i سود دهی اگر مثبت باشد



یک پروژه ساخت این ساختمان است یکی از کارهایی که باید انجام شود مثلاً بتن‌ریزی است که باید برای آن پول خرج کنید تا انجام شود. بتن‌ریزی سه هفته طول می‌کشد توی این سه هفته باید هزینه کنی این هزینه‌ها در آن واحد ممکن است پرداخت نشود چون فعالیت زمانبر است یک مدتی طول می‌کشد، هر روز قسمتی از این هزینه صرف می‌شود.
همین که رو مثال صحبت کردیم را به صورت فرمول کلی نوشته شده است این g_{it} در واقع بده ، بستون‌های روزانه پروژه هستند که شما چطوری به انتها منتقل کردید.
اینها را ضرب کردید در $e^{\propto}$. ۴ $\propto$ و ($\propto$ ۲, $\propto$ ۳, $\propto$ ۴, $\propto$ ۵) یعنی چه ؟ به خاطر اینکه بستگی دارد به این t نسبت به آخر پروژه چند تا فاصله دارد .مثلاً این ۱۰۰ را به روز ۶ ببریم ۵ روز است ۵=۱-۶ است که ۶ ، duration است یک هم این t ایی که شما در آن هستید اما وقتی ۹۰ را می خواستیم منتقل کنید چند روز جابجا کردید. ۴=۲-۶ ، ۴ روز آخری سرجایش هست پس این C_i از روز اول تا آخر روز آن فعالیت جریانات نقدینگی (نقدی) روزانه ضربدر $e^{\propto}$ و مدت زمانی که تا به انتهای فعالیت برسیم.

$$c_i = \sum_{t=1}^{d_i} g_{it} e^{\alpha(d_i - t)}$$

t ⟵ روزی است که شما در آن هستید.
d-t ⟵ روزهایی که دارید جابجا می‌کنید.



این را برای فعالیت مثل i انجام داده شد. باید برای همه فعالیت‌ها این کار انجام شود وقتی برای همه فعالیت ها این کار انجام شد الان شما یک پروژه دارید که این پروژه شامل تعداد زیادی فعالیت که هر فعالیت برای خودش C_i دارد که طبق این فرمول حساب شده است . C_1, C_2, C_3, C_4

برای فعالیت یک برای فعالیت دو

برای فعالیت سه برای فعالیت ۴ و الی آخر

سوال : کل پروژه اقتصادی است یا خیر ؟ باید این C_i ها را دوباره جمع کنید بیارید کجا ؟ صفر که NPVبشود.

NPV (net present value) باید به لحظه صفر که به لحظه

ما تا این لحظه داشتیم برای یک فعالیت برآیند مالی را حساب می‌کردیم که این C_i شده است. الان می خواهیم برآیند مالی کل پروژه را حساب کنیم که تک تک C_i ها بیایند به لحظه صفر، با کدام فرمول ؟ اینجا دو تا فرمول است فرمول پایینی .

NPV پروژه چطور محاسبه می‌شود $C_i \times e^{-\propto(\overbrace{S_i+d_i}^{F_i})}$، به جای t چه چیزی باید بنویسیم. این C_1 را می‌خواهم انتقال بدهم به لحظه صفر $C_1 \times e^{-\propto(S_1+d_1)}$ شما C را در پایان فعالیت داشتید فعالیت یک کی تمام شده است ؟ اسمش را F_i می‌گذارید. یا می‌خواهید شروع اش را بگید می‌شه شروع + duration پس $e^{-\propto(S_i+d_i)}$ یا به جای $S_i + d_i$ می‌توانید بنویسید F_i. چون در بحث قبلی ما قرارداد کردیم این برآیند مالی را کجا حساب کنیم آخر فعالیت بنابراین پایان فعالیت F_i مدت زمانی است که باید توی این فرمول به جای t استفاده شود.

این برای یک فعالیت بود برای کل پروژه چطور می‌شود. یک زیگما به پشت این فرمول اضافه می‌کنیم. که در اسلاید بعدی این تابع هدف شما است. چرا باید ماکسیمم بشود ؟ چون NPV است، برآیند سود شما است.

$$\max \sum_{i=2}^{n-1} c_i e^{-\alpha(\overbrace{s_i+d_i}^{F_i})}$$

$$s_i + d_i \leq s_j$$

می‌توان بر حسب F هم نوشت قبلاً توافق کردیم زمانبندی یعنی تعیین زمان شروع پایان فرقی نمی‌کند می‌توان مساله را با شروع مدل کنی یعنی S متغیر باشد و یا می‌توان مسئله را با پایان مدل کنی فرق نمی‌کند هر دو تاشون درست است چون S را حساب کنید F به دست می‌آید و F را حساب کنید S به دست می‌آید اینها به هم وابسته هستند.

چرا از ۲ شروع شده است ؟ L-۱ می‌شود حدس زد؟ i شماره فعالیت است چرا از ۲ شروع شده تا n منهای یک است قابل حدس می‌باشد. فعالیت یک و n مجازی هستند فرض شده است فعالیت شروع و پایان (۱ و n) مجازی هستند.
با توجه به محدودیت‌ها چی هستند؟ محدودیت‌های مدل قبل GPR ایی بودند این محدودیت‌های (پیش‌نیازی ها) از نوع Finish to start ایی است البته اشکال تایپی هم دارد فقط این باضافه d_i این $S_i + d_i$ است.

$S_1 = 0$ ⟵ چرا صفر است ؟ تابع هدف نامنظم است خود پروژه ممکن است مایل نباشد همین الان شروع شود ولی ما داریم این را Force می‌کنید که پروژه باید لحظه صفر شروع شود هفته پیش نداشتیم چون آنجا هدف منظم بود خود تابع هدف دنبال این بود که زود کار را انجام دهد

$S_n \leq \delta_n$ ⟵ این موضوع هر کدام مال یک کتاب، یک مقاله است نمادگذاری ها یک فرق می‌کند شروع که کردیم T همان dead line، ۲۰ بود این همان است . δ_n همان T (dead line) است چون این مقاله این مدل مال یک مقاله دیگری است اسم‌ها ، نمادهای همان مقاله استفاده شده است چون اینها لزوماً اسم است شما می‌توانید هر اسم دیگری به جای dead line استفاده کنید δ همان T است.

$S_i \in N$ ⟵ S_i همان مجهول های شما است و متعلق به N(عدد صحیح) اند.

پس این شد مدل سازی مسئله چه چیزهایی متغیر هستند ؟ S_i . تابع هدف و محدودیت‌ها هم که صحبت شد فقط پیش‌نیازی‌هایش از نوع Finish to start است می‌بینید که Lag (l_{ij}) در آن وجود ندارد اگر پیش نیازی ها GPR بودند باید این پیش‌نیازی را چه چیزی نوشت ؟ $S_i + l_{ij} \leq S_j$ یعنی باید پیش‌نیازی را اینطوری نوشت ولی این مسئله GPR نیست از نوع Finish to start است.

حل مسئله پس از مدل‌سازی

قبلی‌ها را حل کردیم خوش شانس بودیم چون مدل LPR عدد صحیح می‌شد گفتیم که حل لازم نیست و با Simplex حل می‌شود و جواب حتماً عدد صحیح است چون ویژگی فوق‌العاده‌ای داشت. اما در این مورد چون این ویژگی را ندارد با Simplex حل نمی‌شود و باید آن را حل کنیم. چرا با Simplexحل نمی‌شود ؟ چون تابع هدف خطی نیست به تابع هدف دقت کنید e به توان مدل غیر خطی است با Simplex مطلقاً قابل حل نیست غیرخطی و عدد صحیح است.

در برنامه‌ریزی خطی یا همان LP تحقیق یک ما چند تا ویژگی خیلی خوب بلدیم.

- اولاً می‌دانیم در LP فضای جواب محدب است. اینکه بدانیم فضای جواب محدب است چه حسنی دارد ؟ الان در این مدل فضای جوابش محدب نیست. مسئله max- NPV فضای جواب نامحدب است آنجا که محدب بود چه حسنی داشت ؟

  تعریف محدب : دو نقطه از فضا را به هم وصل می‌کنید پاره خط در آن فضا باشد.

  به گوشه‌ها نیست یعنی می‌تواند فضا نامحدب باشد ولی جواب در گوشه‌ها باشد. اتفاقاً در این مثال همین‌طوری است در LP جواب بهینه در گوشه است. در max- NPV هم جواب بهتر در گوشه است اینجا هم اتفاقاً همین است خوب محدب بودن چه حسنی دارد ؟

  Local Optimum یعنی بهینه محلی و Golobal Optimum یعنی بهینه کلی . وقتی فضای جواب محدب است هر Local optimum ای یک golobal optimum نیز می‌باشد. این چه حسنی دارد؟ این کلاس مکعب است ۸ تا نقطه گوشه‌ای دارد پس فضای جواب این کلاس ۸ تا نقطه گوشه ای است این گوشه که این بالا است فرض کنید تابع هدف ماکسیمم سازی است Z (سود) این ۵۰ است این کناری ۳۷ است اون یکی کناری ۴۱ است و اون کناری انتها هم ۲۹ ، خلاصه این نقطه گوشه‌ای Local optimum

است چرا ؟ چون از سه گوشه مجاور کناری خودش بهتر است اون دو تا گوشه اون چی ؟ یا اون گوشهٔ پایین ؟ این گوشه‌ها بررسی نشده است اما چون فضای جواب محدب است نتیجه می‌گیریم که این جواب بهینه کل مسئله است (۵۰) هر جوابی که در فضای محدب Locl optimum باشد global optimum هم است لازم نیست اون گوشه‌ها را چک کنید همین که از گوشه‌های کناری اش بهتر است تمام.

اگر فضای جواب محدب باشد هر نقطه بهینه محلی، بهینه مطلق (کلی) نیز هست. ما در simplex از این موضوع بهره‌برداری می‌کردیم و همه گوشه‌ها را چک نمی‌کردیم. اما اینجا بدشانسی این موضوع اتفاق نیافتده است.

نکته بعد در مسئله max-NPV مدل غیرخطی و عدد صحیح است و فضای جواب آن نامحدب است پس این نامحدب بودنش یک پون منفی به حساب می‌آید هر جواب Local ایی که پیدا کردید بلافاصله نمی‌توانید بگویید که بهینه است . باید همه فضا را بگردیم.

اما نکته مثبتی هم دارد در این مدل جواب بهینه در یکی از نقاط گوشه‌ای (رأس) قرار دارند و ضمناً نقاط رأسی این مدل عدد صحیح هستند. عین مدل قبلی است نقاط رأسی عدد صحیح بود

فقط آنجا خوش‌شانسی این بود که مدل خطی بود و با simplex حل می‌شد ولی این خطی نیست پس نامحدب است و با simplexحل نمی‌شود خوب فعلاً مسئله را حل نکردیم ولی تا حدی سر نخ‌هایی را داریم مطمئن هستیم کجا را باید search کنیم گوشه‌ها را فقط باید هوشیار باشیم که جواب‌های Local لزوماً بهینه کل مسئله نیستند.

خوب ۵ تا اسلاید داریم که در چند ثانیه ازش رد می‌شویم که فاقد ارزش است.

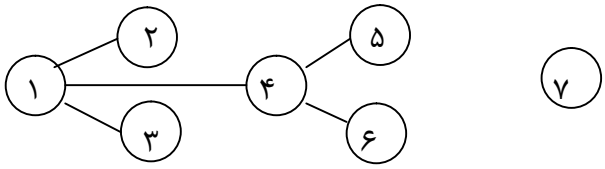
آقای گری‌نولد گفته است که می‌شود مدل غیر خطی را خطی کرد. اگر بتوانیم این کار را بکنیم به ظاهر خوب می‌آید ولی اینجا باشید.

$e^{-\propto(S_i+d_i)}$ ⟵ را به اسم y_i نشان داده است آن وقت تابع هدف می‌شود ، $\sum c_i y_i$ ظاهراً داره خطی می‌شود. پس این را گرفته y_i و از دو طرفش Logaritm گرفته است، خروجی‌اش این است این مدل غیر خطی (مدل اصلی) الان خطی شده است فوقش خطی شده است که جوابش به درد شما نمی‌خورد . چرا ؟ چون y به شما می‌دهد و y ها اعشاری هستند درحالی که مسئله شما یک مسئله عدد صحیح است ظاهراً خطی شده ولی خطی عدد صحیح نیست خطی اعشاری می‌باشد. به دلیل اینکه جوابش اعشاری است به درد نمی‌خورد.

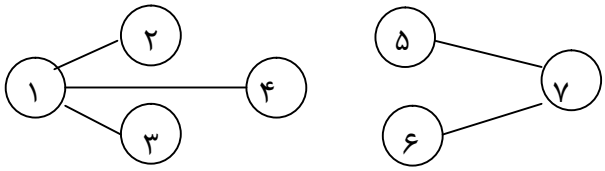
فرض کنید که حل کردید ازش چی بدست می‌آید ؟ y ها به دست می‌آید آن وقت Sچطوری محاسبه می‌شود ؟ S که یادتان نرفته چی بود اگر شما y ها را در این فرمول قرار دهید عدد صحیح نمی‌شود. $S_i = \frac{Lny_i}{\propto} - d_i$ اعشاری می‌شود پس آن قسمتش نا امید کننده است هر چند ظاهر این قضیه خطی شده ولی فایده‌ای ندارد. پس دنبال راه حل دیگری برای حل آن باید باشیم.

(تعریف Tree : به گرافی یک گراف tree می‌گویند که همبند و بدون دور باشد. Tree عبارت است یک گراف همبند بدون دور یا حلقه) Tree در هر گرافی از جمله TSP,VRP، پروژه و location هر مسئله‌ای که بحث گراف داریم این تعریف درست است.

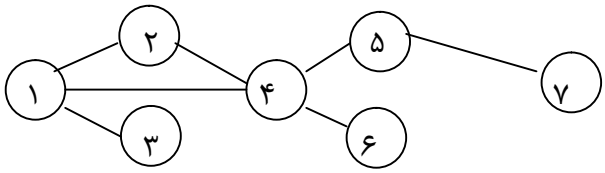
این tree است یا خیر ؟ خیر چون همبند نیست یعنی همه گره‌ها به همدیگر وصل نیست.



این یکی آیا tree است ؟ خیر بازم همبند نیست چون گراف ۲ تیکه ، ۳ تیکه شده است.



همبند یعنی کلاً گراف باید یک تیکه باشد الان این گراف همبند است بدون دور است یا خیر ؟ بدون دور یعنی باید مسیر بسته نداشته باشد مثلاً می‌گم من این را وصل می‌کنم به اینجا چی اتفاقی می‌افتد ؟ اینجا را بستم اینجا که محصور می‌شود ایجاد دور می‌کند به اصطلاح بازم tree به حساب نمی‌آید.

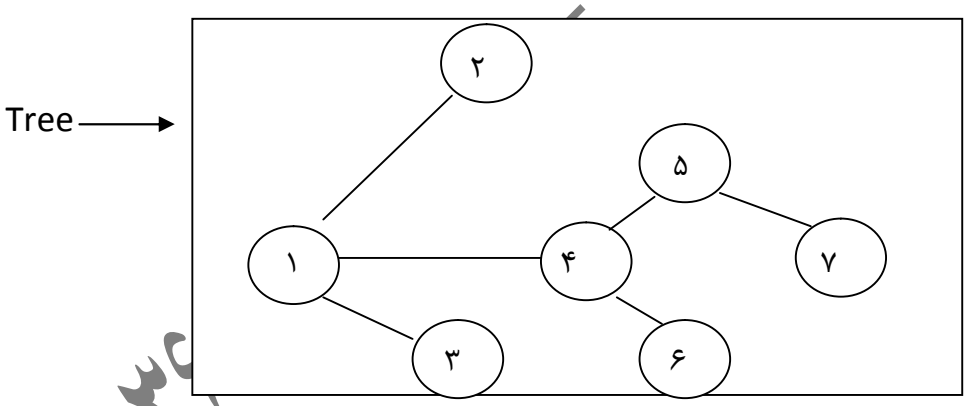


چند تا گره ، چند تا بردار دارد ؟ همیشه در treeبردارها یکی از گره‌ها کمتر است.

نکته : در یک tree همواره تعداد بردارها ۱−n است.

۷ گره

۶ بردار



هر برداری که اضافه کنید ایجاد دور می‌کند و دیگر tree به حساب نمی‌آید.

زمانبندی پروژه

در زمانبندی پروژه یک بردار مثل (۴,۶) عضو Tree است اگر ۴ پیش نیاز ۶ است به هم مماس باشند یعنی لحظه‌ای که ۴ تمام شده همان لحظه ۶ شروع شود اما اگر بین (۴,۶) فاصله (گپ) باشد این بردار عضور tree نیست.

در یک زمانبندی هر بردار $(i,j)\in E$ عضو Tree است اگر فعالیت i و j در زمانبندی به هم مماس باشند یعنی پایان i با شروع j منطبق (برابر) باشد به عبارت بهتر

شروع j پایان j

$F_i = S_j$

کل بردارها

یک بردار

$\overbrace{F_i}$

$$\begin{cases} S_i + d_i = S_j \\ \text{یا} \\ S_j - S_i = d_i \end{cases}$$

مفهوم این است این دو فعالیت پشت سر هم و بدون وقفه انجام شده باشد که می‌گویند این بردار عضو Treeاست.

Extreme Point (نقطه گوشه‌ای)

هر نقطه گوشه‌ای معادل با یک tree است یعنی هر زمانبندی که ساختار tree دارد دقیقاً معادل نقطهٔ گوشه‌ای است. جواب بهینه کجا می‌باشد ؟

گوشه‌ها . الان در تعریف بالا هر گوشه معادل یک tree است یعنی عملاً جستجوی گوشه‌ها، به جاش بگویید جستجوی tree . فقط جواب‌هایی را search کنید که ساختار tree دارند چون هر tree معادل با یک گوشه است کلمه گوشه برای انتقال بحث است والا ما اینجا تصویر یا نقشه گرافیکی نداریم از جواب. در تحقیق یک گوشه‌ها را چی می‌گفتید؟ در هنگام رسم جدول ، هر جدول معادل یک گوشه بود ولی چون شکلی نداشتید که رسم کنید می‌گفتید جواب (موجه یا شدنی، محدودیت ها را راضی کند می‌تواند گوشه‌ای نباشد ولی موجه باشد. آنهایی که گوشه‌ای هستند گفتیم جوابهای پایه،ای (متغیر وارد پایه می‌شود از پایه خارج می‌شود) این تعریف جواب پایه‌ای می‌گفتند هر جواب گوشه‌ای یعنی یک جواب پایه‌ای. حالا خودش تعریف مفصلی بود) حالا اینجا هم دقیقاً همین است ما تصویر گرافیک از فضای حل نداریم معادل‌اش باید با این اصطلاح پیش رویم.

بنابراین برای یافتن جواب بهینه کافی است فقط جواب‌ها با ساختار tree جستجو شود.

هنوز هم مسئله را حل نکردیم ولی سرنخ‌ها بسیار بهتر و بیشتر شد آقای هرولن و بعد آقای ون‌هاک یک روشی را پیشنهاد دادند ولی مبنای روششون این قضایایی که الان نوشتید یعنی اول اثبات‌هایی که الان گفتید و شما نوشتید را اثبات کردند که جواب‌ها ساختار tree دارند و جواب‌ها در گوشه‌ها هستند و در آخر سر کار راحت است و تنها دنبال جواب‌هایی که tree هستند بگردید چون هر tree یعنی یک گوشه.

این روش یک روش بازگشتی دقیق یعنی جواب بهینه را به صورت دقیق پیدا می‌کند جواب تقریبی نیست چون یکسری روش های تقریبی هم داریم تقریبی‌ها چی کار می‌کنند اسم‌اش روش است یه جواب خوب می‌دهند شما در طرح‌ریزی تجهیزات صنعتی روش‌های جانمایی، کورلپ داشتید به بعضی از آنها روش‌های تقریبی بودند و بعضی هاشون دقیق بودند craftدقیق بود ولی آل دپ تقریبی بود نسبتاً جواب خوبی می‌داد ولی قطعی نبود پس این روش دقیق مسئله است.

این روش سه گام دارد.

- گام یک : (ساخت) پیدا کردن جواب اولیه (early tree). جواب یعنی tree چون هر جواب یعنی گوشه ، گوشه هم یعنی tree
- گام دو: ساخت یک جواب فعلی (current tree)
- گام سه : یک جستجوی recursive بر روی current tree

سیمپلکس recursive است یا خیر ؟ چه طور روش‌هایی را recursive می‌گویند ؟ بازگشتی یعنی چه ؟ تکراری . یعنی یک کاری را دائماً تکرار می‌کنید. در سیمپلکس جدول ۲ با جدول ۱۲ از نظر روش محاسبات فرق نداشت اعداد عوض می‌شد ولی روش فرق نداشت یعنی اگر جدول ۲ را می‌توانستی حل کنی بقیه را تا آخر نیز حل می‌کردی

روش مجارستانی و جانمایی روش‌های بازگشتی‌اند.

بازگشتی‌ها شرط توقف دارند یک کار ثابتی دائماً تکرار می‌شود تا فلان اتفاق بیافتد. این ردیف‌ها مثبت شد تمام. روش بازگشتی روشی است که تا یک نقطه‌ای که به یک شرطی برسند باید انجام بدهید.

گام اول و دوم فقط یکبار انجام می‌شود ولی گام سوم recursive است.

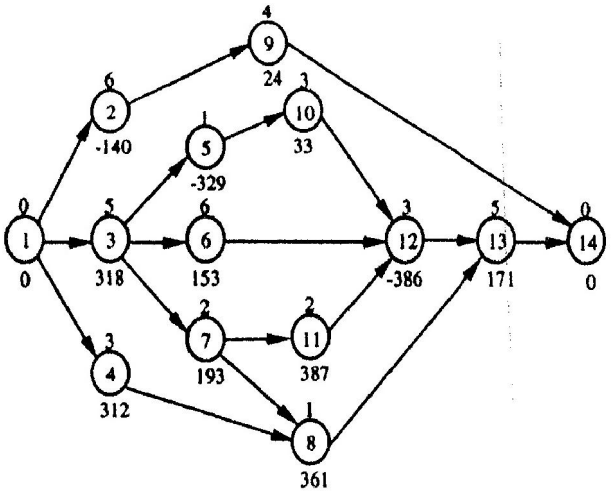
گام اول : شما قبلاً محاسبات forward را انجام دادید محاسبات forward زودترین زمان شروع و پایان. بدون اینکه متن را بخوانیم قضیه همین است فعالیت‌ها را در زودترین زمان ممکن بچینید.

فعالیت ها را با محاسبات forward در زودترین زمان ممکن زمانبندی کنید.

مثال :

۱۴ تا فعالیت داریم که فعالیت ۱ و ۱۴ مجازی هستند. چون duration آنها صفر است. پس ۱۲ تا واقعاً فعالیت داریم عددی که بالای گره‌ها است duration است.

عددی که پایین نوشته است cash flue یا C_i برآیند مالی آن فعالیت است این مسئله درخصوص Max است آنهایی که مثبت هستند مثل فعالیت ۱۱ ، ۳۸۷ واحد سود دارد و آنهایی که منفی هستند مثل فعالیت ۵، ۳۲۹ واحد هزینه دارد. برآیندش منفی است به هر حال این می‌شود یک مثال dead line ، ۴۴ روز است و نرخ تنظیم (بهره) منظور همان α است ۱٪ است.



برای گراف روبرو می‌خواهیم گام یک را انجام دهیم.

گام یک باید ۱۴ عدد از F_1 تا F_{14} را بنویسید یعنی بگویید که هر فعالیت زودترین زمان کی می‌تواند تمام شود . چرا شروع را نمی‌نویسیم فرقی نمی‌کند یا شروع یا پایان. (F یا S)

$F_1 = ۰$ $F_2 = ۶$ $F_3 = ۵$ $F_4 = ۳$

$F_5 = ۶$ $F_6 = ۱۱$ $F_7 = ۷$ $F_8 = ۸$

$F_9 = ۱۰$ $F_{10} = ۹$ $F_{11} = ۹$ $F_{12} = ۱۴$

$F_{13} = ۱۹$ $F_{14} = ۴۴$

فعالیت یک صفر است مجازی است و duration آن صفر است همین الان شروع و همین الان تمام می‌شود.

فعالیت ۲ پیش نیازش یک است (مجازی) یعنی هیچی یعنی می‌تواند همین الان شروع شود و کی تمام شود ۶ روز طول می‌کشد تا تمام شود $F_2 = ۶$ امروز شروع شود روز ششم تمام می‌شود.

بردار (۱,۲) عضو tree است یا خیر؟ یک کی تمام شد صفرم ، دو کی شروع شد ؟ آن هم صفرم. یعنی دقیقاً لحظه‌ای که یک تمام شد ۲ را شروع کردیم یعنی این بردار چند ثانیه دیگر قرار عضو tree باشد.

فعالیت ۳ صفرم شروع، ۵ تمام می‌شود این هم عضو tree است.

فعالیت ۴ صفرم شروع و ۳ ام پایان

فعالیت ۵ پیش‌نیازش ۳ است. ۳ کی تمام شده است ؟ ۵ ام تمام شده است اگر بلافاصله ۵ را شروع کنید یک روز هم خود ۵ طول می‌کشد می‌شود ۶ روز و البته یادمان نرود که ۳ و ۵ هم عضو tree خواهد بود. چون لحظه که ۳ تمام شده است ۵ را شروع کرده‌ایم.

فعالیت ۶ این ۵ ام بلافاصله ۶ را شروع کنید که خودش هم ۶ روز است ۱۱=۶+۵

فعالیت ۷، ۵ ام تمام بلافاصله شروع ۲ روز هم این است می‌شود ۷=۲+۵

تا اینجا همه بردار عضو treeبودند

فعالیت ۸ ، کمی فرق می‌کند به خاطر اینکه دو تا پیش نیازی دارد. محاسبات forwardمی‌گفت که باید هر دو پیش نیازی بررسی شود و ماکسیمم نوشته شود سپس ماکسیمم عضو tree می‌شود نه هر دو تا پیش نیازی

۴ کی تمام شده است ، ۷ ام اینم یک روز طول می‌کشد پس ۸=۱+۷ این مسیر ۸ است

۴ کی تمام شده است ، ۳ ام با یک می‌شود ۴=۱+۳

بین ۴ و ۸ ماکسیمم ۸ است پس بردار (۷,۸)فقط عضو tree است.

الان بردار (۴,۸)بینشان یک فاصله‌ای است ۴ سوم ام تمام شده است و اینم ۷ ام شروع شده است که بین آنها ۳ تا ۴ روز فاصله است.

فعالیت ۹ ۱۰=۴+۶

فعالیت ۱۰ که ۶ ام بود با ۳ می‌شود ۹

فعالیت ۱۱ ، ۹ ام

فعالیت ۱۲ سه تا پیش نیازی دارد

- ۱۰ ⟵ ۹ ام با ۳ ⟵ ۱۲
- ۶ ⟵ ۱۱ ام با ۳ ⟵ ۱۴ مسیر ۶ به ۱۰ است ۱۲ عضو tree خواهد بود. max
- ۱۱ ⟵ ۱۹ ام با ۳ ⟵ ۱۲ طولانی‌ترین مسیر

فعالیت ۱۳ دو تا پیش نیاز دارد ۱۴ با ۱۵ می‌شود ۱۹ و ۱۹ و ۱۳ ماکسیمم ۱۹

- ۱۲ ⟵ ۱۴ ام با ۵ ⟵ ۱۹
- ۸ ⟵ ۱۳ ام با ۵ ⟵ ۱۳

Max=۱۹

استثنا : محاسبات forward یک استثنا دارد. برای آخرین فعالیت را ، dead line پروژه به عنوان F_i ثبت می شود. و یک بردار مجازی از فعالیت پایان به شروع که عضو tree هست اضافه می‌کنیم.

$F_۱ = ۰$ $F_۲ = ۶$ $F_۳ = ۵$ $F_۴ = ۳$

$F_۵ = ۶$ $F_۶ = ۱۱$ $F_۷ = ۷$ $F_۸ = ۸$

$F_۹ = ۱۰$ $F_{۱۰} = ۹$ $F_{۱۱} = ۹$ $F_{۱۲} = ۱۴$

$F_{۱۳} = ۱۹$ $F_{۱۴} = ۴۴$

نتیجه : الان چی ها را حساب کردیم آن F_i ها هر چند در اسلاید بود. کدام بردار ها عضو tree هستند ؟ آن مسیرهایی که ماکسیمم می‌شود آخر سر یک بردار از ۱۴ به یک از فعالیت پایان به شروع وصل می‌کنید که اینم باید عضو tree باشد این برای کنترل dead line است این سر و ته پروژه را دارد چی کار می‌کند به هم متصل نگه می‌دارد که فاصله اینها از ۴۴ بیشتر نشود. کنترلی dead line

گام ۲ ساخت current tree :

در این گام الگوریتم می‌سازد current tree با به تأخیر انداختن تمام فعالیت‌هایی که delaying cash flue دارند می‌سازد یعنی جواب C_i منفی است و هیچ پس نیازی روی tree ندارد.

ساخت Current tree : تمام فعالیت‌های با C_i منفی که در روی tree پس نیاز ندارد تا حد امکان به تأخیر انداخته می‌شوند.

مثال

کدام C_i منفی دارند ؟ (۲,۵,۱۲) C_i منفی دارد.

دو شرط بود یکی داشتن C_i منفی و دیگری پس نیاز هم نداشته باشد.

فعالیت ۲، C_i منفی دارد ولی بعد از ۲ ، ۹ مماس با ۲ است چون بردارش عضو tree است . بردارهایی که ضرب در خورده‌اند عضو tree نیستند ولی آن یکی ها عضو tree هستند. وقتی ۲ پشت سرش دقیقاً ۹ مماس است یعنی ۲ امکان جابجایی ندارد پس هیچی ما می‌خواستیم ۲ هل بدیم جلوتر ولی ۹ مانع شده است.

فعالیت ۵ با داشتن C_i منفی ولی پشت سرش ۱۰ است

فعالیت ۱۲ با داشتن C_i منفی ولی پشت سرش ۱۳ است

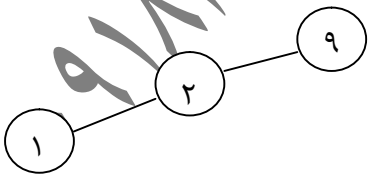
هیچکدام از فعالیت‌هایی که منفی بودند امکان جابجایی ندارند چون همه آنها پس نیاز دارند.

حال به فرض برای اینکه همچنین حالتی را درست کنیم برای چند ثانیه فرض می‌کنیم ۴ این ۳۱۲- باشد اگر منفی بود ۴ را می‌توان جابجا کرد ۴ منفی است بعد از ۴ هم ۸ پس نیازش است ولی عضو treeنیست یعنی بین ۴ و ۸ یک فاصله چند روزه وجود دارد و به همان میزان یک ، دو ، سه حالا هر چقدر جا است اضافه می‌کنیم تا این مماس شود ولی این اتفاق در اینجا نیفتاده است.

چرا منفی‌ها را جابجا می‌کنیم.؟معمولاًدنبال این هستید که بدهکاریها را تا حد امکان دیرتر پرداخت کنید و طلب‌کارهای تان را زودتر دریافت کنید ارزش زمانی پول می‌گید اگر این ۱۰۰۰ تومان را الان داشته باشم بهتر است از اینکه یکسال دیگه داشته باشم. بنابراین مثبت‌ها را می‌آوریم نزدیک و منفی‌ها را دور می‌کنیم.

در گام اول در زودترین زمان درهم بود مثبت و منفی‌ها را به سمت صفر می‌بردید و در گام دوم منفی‌هایی که بشود به سمت جلو هل می‌دهید.

گام سوم : منفی‌هایی که پشت سرشان مثبت هست اگر نرود منفی بیشتر باشد هر دو را منفی حتی به قیمت عقب فرستادن مثبت جابجا کنیم. اگر زور مثبت بیشتر بود همان جا بماند. مثلاً ۲ ، ۱۴۰- ولی ۹ ، ۲۳ این خیلی منفی و اون یکی مقدار مثبت پس احتمالاً وزن این خیلی بیشتر باشد بین ۲ و ۹ برآیند منفی است، نتیجه ۲ و ۹ باید با هم جابجا شوند. یعنی از یک tree برویم به tree دیگر.



وقتی ۲ و ۹ کنده می‌شوند از ۱ یعنی این بردار عضو tree نیست به جای آن این ۲ تا از اون طرف به یکی دیگر می‌چسبد این بردار از tree کنده می‌شود به جای آن بردار به tree اضافه می شود. یعنی از این نقطه گوشه‌ای داریم به گوشه‌ای دیگر می رویم و از یک tree به treeدیگر یک tree بهتر ، حالا بهینه است یا نه ، چک می‌کنیم اگر بهینه بود که تمام و اگر نه بعدی، بعدی اگر هیچ جابه جایی مقرون به صرفه نبود مسئله کلاً تمام می‌شود

جلسه نهم

یادآوری جلسه هشتم:

دو تا مسئله را معرفی کردیم که وجه اشتراک اش در این بود که هر دو تا دارای تابع هدف مالی بودند. اولی مینیمم کردن start time dependent costs (هزینه‌های وابسته به زمان شروع) بود هر فعالیت بسته به این که کی شروع می‌شود هزینه‌اش متفاوت بود. که این مسئله سه بار مدل شد یعنی ۳ تا مدل‌سازی صفر و یک داشت که البته ما حل نکردیم. به این دلیل حل نکردیم چون که LP Relaxiation اش عدد صحیح می‌شد گفتیم که با simplex هم حل شود عدد صحیح بدست می‌آید. مسئله دوم، مسئله‌ای بود که تا نصف حل کردیم به است max-NPV ، ماکسیمم کردن NPV پروژه یا سود پروژه با لحاظ ارزش زمانی پول مدل ریاضی‌اش (تابع هدف‌اش) غیر خطی بود و فضای جواب آن نامحدب بود. که این دو مورد عیب آن بود بنابراین حل کردنش با simplex منتفی بود چون اصلاً خطی نبود. اما دارای دو حسن بود اولاً جواب بهینه‌اش در گوشه‌ها بود، گوشه‌ها هم ویژگی عدد صحیح داشتند. یک نکته‌ای دیگری که صحبت شد این بود که گوشه‌ها ساختار tree داشتند. اگر بردار (i,j) مماس با هم زمان بندی داشتند می‌گفتیم که این بردار عضو tree است بعد گفته شد اگر همه گوشه‌ها ساختار tree دارند و جواب هم در گوشه‌ها است. یعنی به زمان بهتر فقط باید این tree ها را search می‌کردیم. چون هر tree یعنی یک گوشه. بعد یک الگوریتم سه مرحله‌ای را آقای ون هاگ پیشنهاد کرده بود که دو گام اش انجام شد.

گام اول : فعالیت‌ها را طبق محاسبات forward در زودترین زمان می‌چینیم.

گام دوم : آنهایی که cash flue منفی داشتند تا جایی که می‌شد به تأخیر می‌انداخت (البته با نداشتن پس نیاز) این دو گام انجام شد.

$F_۱ = ۰$ $F_۲ = ۶$ $F_۳ = ۵$ $F_۴ = ۳$

$F_۵ = ۶$ $F_۶ = ۱۱$ $F_۷ = ۷$ $F_۸ = ۸$

$F_۹ = ۱۰$ $F_{۱۰} = ۹$ $F_{۱۱} = ۹$ $F_{۱۲} = ۱۴$

$F_{۱۳} = ۱۹$ $F_{۱۴} = ۴۴$

محاسباتش forward یک کمی فرق داشت و اونم آخری بود که dead line را می‌نوشتیم.

گام سوم :

گام سوم به اصطلاح باید بعد از گام دوم انجام شود یعنی این مسئله را با هم صورت مسئله‌اش را دیدیم گام اول و دوم را هم انجام دادیم. این الان اسمش Current tree است حالا باید روی این باید گام سوم را انجام دهیم . در گام سوم به دنبال چی هستیم ؟ در گام سوم می‌خواهیم منفی‌ها را البته با آن مثبتی که پشت سرش هست هر دو را با هم

به اصطلاح جابجا کنیم به آینده موکول کنیم . البته به شرط اینکه برآیند مالی منفی باشد. مثبت که باشد خوب باید همین جا بماند. یعنی زودترین زمان، اقتصادی مهندسی دیگه. آنهایی که مثبت‌اند هر چقدر زودتر آنهایی که منفی اند هر چقدر دیرتر اگر مثبت باشد همین جا بهتر است باشد. اگر منفی شد این را به تأخیر بندازیم.

برای اینکه بتوانیم از این گام‌ها صبحت کنیم من سه تا پارامتر را تعریف می‌کنم این را ترجیحاً در این اسلاید بنویسید. اولین بار که این را می‌بینید در این اسلاید است.

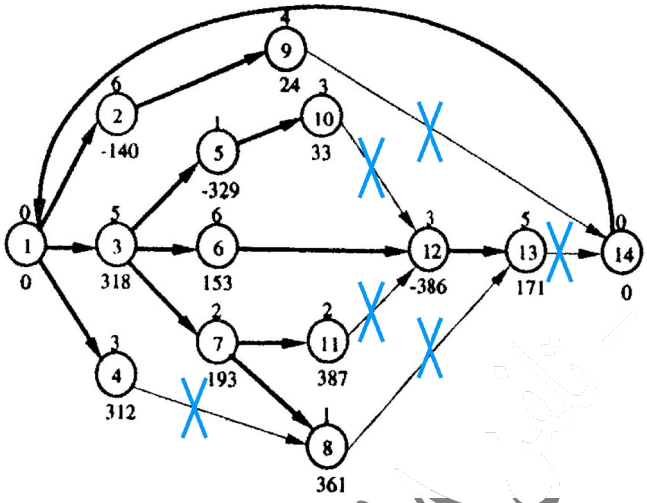
SA : مجموعه فعالیت‌های در حال بررسی

CA : مجموعه فعالیت‌های بررسی شده (از اول تا اینجا هر چی بررسی کردیم)

DC : برآیند مالی مجموعه SA است (discount cash flue)

ایشون در واقع این سه تا نماد را معرفی کرده است که بعد راحتر صحبت کند هر بار نگویید مجموعه فعالیت‌های در حال بررسی بگوید SA . در واقع با این کار، نگارش آن کوتاه می‌شود. عرض کنم که این گام سوم (Recursive)به اصطلاح recursion ها یعنی (تکراری‌هاش) از کجا شروع می‌شود ؟ از گره یک

شروع پروژه به اصطلاح ریشه آن tree است .



Recursion ۱ پس کدام بررسی می‌شود ؟ از گره یک

RECURSION (۱)

SA={۱}, *CA*={۱} and *DC*=۰.۰۰ خط تشریفاتی

تا الان کدام بررسی شد ؟ خوب تازه شروع کردیم دیگه . فقط یک .

برآیند مالی فعالیت یک چقدر است ؟ خوب خیلی ساده به اصطلاح این ردیف گفتم یک خط تشریفاتی است یعنی عملاً نوشتن و ننوشتن آن خیلی فرقی نمی‌کند چون فعالیت یک مجازی است چون وجود خارجی ندارد. پس حالا این را در این الگوریتم نوشته و ما هم نوشتیم ولی عملاً فعالیت یک مجازی است. این هیچی نیست بعدی کدام است ؟ این الگوریتم به ترتیب شماره‌ها نیست. اول ۱ ، بعد ۲، بعد ۳، بعد ۴. اول ۱ بعد یکی از پس‌نیازهای ۱ . پس نیازهای ۱ کدام است ؟ ۲,۳,۴ حالا کدام اول ۲ به ترتیب شماره ؟ ۲ ، پس ببیند اگر ۲ پس نیاز یک نبود اصلاً بررسی نمی‌شد. پس نیاز باید اول باشد بعدش به ترتیب شماره اول نگاه کنید ببینید پس نیازش هست یا نه بعد به ترتیب شماره‌اش کنید و کدام پس‌نیازها؟ پس نیازهایی که روی tree هستند یعنی اگر پس نیاز باشد ولی روی tree نباشد آن هم دوباره بازی داده نمی‌شود. یعنی گام سه فقط روی tree باید صحبت کنید درست شد recursion چند ؟

RECURSION (۲): successor node ۲

SA={۲}, *CA*={۱,۲} and *DC*=−۱۳۱.۸۵

recursion ۲ چرا ۲؟ چون اولین پس نیاز یک است روی tree. SA={۲} تا الان کدام‌ها بررسی شدند ؟ CA={۱,۲} . برآیند مالی ۲ چقدر می‌شود ؟ ۱۳۱.۸۵− از کجا آمده است ؟ آن هفته از فرمول‌های اقتصاد مهندسی صحبت شد. NPV چه طوری برای یک فعالیت محاسبه می‌شود این یکی را کنارش فلش بزنید. $-۱۴۰\ e^{-۰.۰۱\ (۶)}$ فعالیت ۲ کی تمام شده است$F_۲ = (۶)$ پس این۱۳۵.۸۵− این عبارت است در واقع .

$$F_۲ = (۶)$$

$$-۱۴۰\ e^{-۰.۰۱\ (۶)} = -۱۳۱.۸۵$$

α

ارزش فعلی۱۴۰

پایان ۲ هم که روز ۶ است به اصطلاح ۱۳۱.۸۵− ارزش فعلی ۱۴۰ است خوب منفی است یا مثبت ؟ منفی پس بهتر است که چی بشود ؟ بهتر است که جابجا شود منفی مگر نیست فوقش مشکل کار کجاست مشکل کار این است که فعالیتی که بعدش است این مانع می‌شود فقط مانع شدن را می‌گم مماس است دیگر. نمی‌گذارد جابجا شود اگر ۲ بخواهد جابجا شود باید ۹ هم جابجا شود. (۲ و ۹ باید با هم جابجا شود)

RECURSION (۹): successor node ۹

SA={۹}, *CA*={۱,۲,۹} and *DC*=۲۱.۷۲

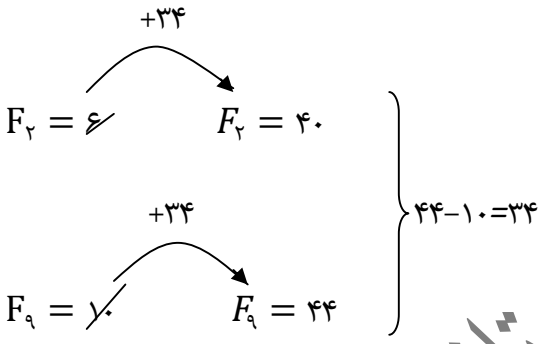
SA'={۲,۹} and *DC'*=−۱۱۰.۱۳

پس recursion بعدی کدام می‌شود؟ ۹ ، ببینید اینطوری تصور کنید یک درختی را تصور کنید ریشه‌اش صفر است وارد یک شاخه می شوید تا آخر شاخه را باید بررسی کنید. تمام فعالیت‌های شاخه تا آخر شاخه را باید بررسی کنید. از این شاخه به شاخه دیگر، فعلاً منتقل نشوید. خود شاخه را تا آخر بروید. من ۱ و ۲ را گفتم. حالا ۹ recursion ، پس SA={۹} تا الان discount cash flue, CA={۱,۲,۹} هم که دیگه گفتن نمی‌خواهد. از کجا آمده است DC=۲۱.۷۲ ؟ $۲۴e^{-\propto(F_۹=۱۰)}$ پس $۲۴\ e^{-(۰.۰۱)\times۱۰} = ۲۱.۷۲$ دیگه ادامه ندارد ؟ چرا ؟ این شاخه تمام شده است. DC=۲۱.۷۲ برداری که ضرب در خورده است دیگر عضو Tree نیست دیگر جدا شد. این شاخه را تا آخر آمدید. حالا دقت کنید در این شاخه که بررسی کردید کدام منفی بود ؟ ۲، آنهایی که مثبت است نباید جابه جا شود فقط منفی‌ها. ۲ اگر بخواهد جابه جا شود باید با کدام جابه جا شود ؟ با ۹ . آن وقت برآیند ۲ و ۹ می‌شود.

پس الان ۲ و ۹ را با هم می بینیم برآیندش این از کجا آمد ؟ ۱۱۰.۱۳− از کجا آمد ؟ جدا جدا حساب کرده بودید . ۱۳۱.۸۵−۲۱.۷۲=−۱۱۰.۱۳ . نتیجه : چون برآیند منفی است باید جابه جا شود ۲ و ۹ باید با هم جا به جا شود. شاخه‌های دیگه چی ؟ فعلاً اینها را جابجا می‌کنیم. شاخه‌های دیگر را هم بررسی می‌کنیم پس فعلاً به محض اینکه به نقطه‌ای رسیدم که جابجایی توجیح دارد ؟ این جابجایی را انجام می‌دهیم دوباره از اول reset می‌کنیم. تکرار دیگه ، گام سه باید بارها و بارها تکرار شود.

حالا چقدر باید جابجا شود ؟ یک روز، دو روز، سه روز، ده روز ؟ تا کجا باید جابجا شود ؟

جواب : تا آنجایی که می‌توانیم باید جابجا کنیم. در شکل ۲ و ۹ تا کجا می توانند جابجا شوند تا جایی که به ۱۴ برسند. دقت کنید ۲ و ۹ با هم مماس اند ولی ۹ و ۱۴ . این فاصله خالی است من می‌توانم از این ۹ تا به ۱۴ برسم جابه‌جاش کنم فاصله ۹ تا ۱۴ چقدر می‌باشد ؟ این فاصله خیلی راحت محاسبه می‌شود. ۹ کی تمام شده است ؟ ۱۰ و ۱۴ کی شروع شده است ؟ ۴۴ و از ۱۰ ام تا ۴۴ می‌شود ۳۴ روز پس چی کار کنم الان ؟ اینها را ۳۴ روز اضافه می‌کنم یعنی به تأخیر می‌اندازم. پس ۶ می‌شود ۴۰ و ۱۰ می‌شود ۴۴ این ۳۴ را از کجا آوردم فاصله ۹ تا ۱۴ شد ۳۴ روز به اینها یعنی فقط به ۲ و ۹ ، SA منفی شد. این ۳۴ روز اضافه می‌شود. چون اینها توجیه داشتند دیگر در tree چه اتفاقی می‌افتد ؟ ۹ و ۱۴ قبلاً فاصله داشتند الان ۹ به ۱۴ خورد این بردار عضو tree گردید و از آن طرف ۲ ایی که با ۱ مماس بود جلو آمد. از یک به اصطلاح جدا شد. یعنی مماس بودن برعکس است. ۲ رفت جلو و کنده شده به اصطلاح. فاصله افتاد یعنی یک بردار که عضو tree بود الان دیگر عضو tree نیست و برداری دیگری که عضو tree نبود الان برعکس، عضو tree گردیده است. از نظر گرافیکی چه اتفاقی افتاده است ؟ یعنی از این گوشه‌ای که بودیم رفتیم به گوشه بهتر از فضای جواب یعنی جواب دیگر بدست آوردیم. تمام. در گراف فقط tree ها را هایلات کردم ضرب در زدم والا خود زمانبندی را این ور نوشتم.



گام سه تمام شد مسئله که تمام نشده است. دوباره این کار را انجام دهید ولی حرف جدید زیادی برای گفتن نیست چون همین داستان دوباره شود.

تکرار بعد همان گام ۳ ولی یک تکرار دیگر. از کجا شروع می‌شود reset کردیم از اول دیگه .

Recursion (۱)

RECURSION(1)
***SA*={1}, *CA*={1} and DC= 0.00**

خط تشریفاتی بعد

Recursion (۳)

بعد از یک ۳ است درست است پیش نیاز ۲، یک است ولی عضو tree نیست، عضو tree هم ۳ و ۴ هستند که اول ۳ را بررسی می‌کنیم پس recursion (۳)

RECURSION(3) : successor node 3
***SA*={3}, *CA*={1,3} and *DC*=302.49**

بعد از ۳ مثبت است پس هیچی، بعد قرار شد شاخه را تا آخر برویم. ۵، ۶، ۷ را می‌توانید بگویید اما قرار شد به ترتیب بروید پس ۵ همین شاخه، شاخه ۱،۳،۵ SA={۵} , CA={۱،۳،۵}, DC=-۳۰۹.۸۴ این مثبت شد. DC=۳۰۲.۴۹ این منفی . DC=-۳۰۹.۸۴ مثبت که هیچی ولی

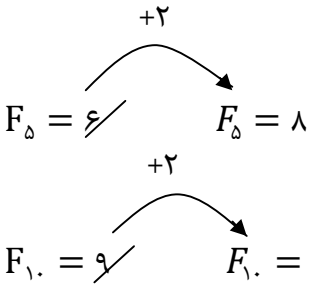
RECURSION(5) : successor node 5
***SA*={5}, *CA*={1,3,5} and *DC*=-309.84**

منفی را می‌خواهم جابجا کنم البته این شاخه هنوز تمام نشده است

۱۰ هم را بگویید.

RECURSION(10) : successor node 10
***SA*={10}, *CA*={1,3,5,10} and *DC*=30.16**

این شاخه تمام شد من از ۱,۳,۵به ۱۰ رسیدم این شاخه تمام شد. آیا در این شاخه منفی داشتیم ؟ بله منفی بود. همه حرف‌ها تکراری است. کدام منفی شد؟ ۵ چون منفی است بهتر است جابه‌جا شود به شرطی که برآیندش با ۱۰ منفی شود چون جابجایی ۵، جابجایی ۱۰ را به دنبال دارد بنابراین باید دید که ۵ و ۱۰ برآیندش چند می شود. ۵ شده بود ۳۰۹/۸۴− و ۱۰ شده بود ۳۰/۱۶+ پس برآیندش ۲۷۹/۶۸− می‌شود. منفی است پس باید جابجا شود. کدام ؟ ۵ . چقدر ؟ به اندازه‌ای که ۱۰ به ۱۲ بخورد الان بین ۱۰ و ۱۲ یک فاصله (گپ) هست. ۱۰ کی تمام شده است ؟ ۹ ام . ۱۲ کی شروع شده است ؟ شروع اش سه روز قبل یعنی ۱۱ ام است. از ۹ تا ۱۱ می‌شود ۲ روز یعنی بین اینها ۲ روز جا است. پس فعالیت‌های ۵ و ۱۰ را ۲ روز اضافه می‌کنیم یعنی ۶ می‌شود ۸ و ۹ می‌شود ۱۱ و روی tree هم کاملاً قابل حدس است که چی کار باید بکنید ۵ رفت جلوتر. از ۳ کنده شد این دیگر عضو tree نیست و از آن سمت ۱۰ به ۱۲ چسبید و این بردار عضو tree شد. پس دوباره به یک جواب بهتر رسیدیم.



بعدی را شما بگویید چون تکرار است هر چند توی این تکرار سوم یک نکته ریزی هست که در قبلی‌ها نبود. از کجا شروع کنم یک، صفر هیچی بعدی ۳ ، مثبت است $۳۱۸e^{(?)}$ بعدی ۶ ، مثبت $۱۵۳e^{(?)}$ یک عدد مثبتی می‌شود. ۱۲ منفی است $-۳۸۶e^{(?)}$ یک عدد منفی است و ۱۳ مثبت است $۱۷۱e^{(?)}$ پس ۱۳,۱۲,۶,۳,۱ را رفتم فقط ۱۲ توی این مسیر منفی بود حالا باید جابجا شود یا نه ؟ برآیندش با ۱۳ مهم است.

STEP 3. Set *CA*=∅. The algorithm again performs a recursive search starting with node 1.
RECURSION(1)
***SA*={1}, *CA*={1} and *DC*= 0.00**
RECURSION(3) : successor node 3
***SA*={3}, *CA*={1,3} and *DC*=302.49**
RECURSION(6) : successor node 6
***SA*={6}, *CA*={1,3,6} and DC=137.06**
RECURSION(12) : successor node 12
***SA*={12}, *CA*={1,3,6,12} and *DC*=-335.57**
RECURSION(13) : successor node 13
***SA*={13}, *CA*={1,3,6,12,13} and *DC*=141.41**
***SA* ={12,13} and *DC*=-194.16**
RECURSION(10) : predecessor node 10

اسلاید بعدی را می‌آورم که ببیند برایند با ۱۳ چند شده است ؟ ۱۲ و ۱۳ برآیندش ۶ ، ۱۹۴- پس باید جابجا شود.

$SA'=\{12,13\}$ and $DC'=-194.16$

نکته : اگر جابجا شود چه اتفاقی در tree می‌افتد اگر ۱۲ و ۱۳ جابه‌جا شوند باید به ۱۴ بخورند این برداری می‌شود عضو tree و برداری که عضو tree نبود می‌شود عضو tree. این ور چه اتفاقی می‌افتد کدام کنده می‌شود ؟ تفاوت‌ها اینجاست دقت کنید اگر ۱۲ و ۱۳ را هل بدهید جلوتر، ۱۲ هم به ۱۰ وصل است هم به ۶ ، ۱۱ عضو tree نیست و ضربدر باید خورده باشد که اشتباه شده روی شکل یعنی اگر جلو برود دو تا بردار از tree پاک می‌شود ولی یکی اضافه می‌شود نتیجه دیگر tree نمی‌شود. tree یک گراف همبند، بدون دور است و تعداد بردارهایش حتماً یکی از گره‌ها کمتر است. الان شما دارید چی کار می‌کنید ؟ یک بردار اضافه می‌کنید، ۲ تا پاک می‌کنید نتیجه این معادله بهم می‌خورد دیگر ساختار tree ندارد از نظر گرافی شما از این گوشه می‌خواهی به جای اینکه بروید گوشه دیگر داری چی کار می‌کنی ؟ ببینید این فضای جواب شما،از این گوشه به گوشه دیگر رفتید و از این گوشه به گوشه دیگر. الان اینجا هستید به جای اینکه برید به گوشه دیگر، دارید چی کار می‌کنید دارید داخل فضای جواب می‌شود و ما مطمئن هستیم که نمی‌تواند بهینه باشد چون جواب بهینه‌اش حتماً نقاط گوشه است پس شما اگر این جابجایی را انجام بدهید اشکال فنی ناموجه ندارد یعنی جواب موجه است ولی دیگر tree نخواهد بود در حالی که قرار شده فقط tree ها را search کنیم . نتیجه : این جابجایی انجام نمی‌شود. شما علیرغم توجیه ۱۲ و ۱۳ نباید اینها را جابجا کنید مگر اینکه که بین این دو تا یکی را قطع کنید یعنی ۵ و ۱۰ باید دنبالش بکشید. که این بردار باید متصل بماند فقط این را قطع کنید. یک بردار قطع می‌شود یک بردار اضافه می شود این اشکال ندارد. این tree است.

سوال : آیا توجیهی وجود دارد که ۵ و ۱۰ را دنبال این بکشید ؟ حساب می‌کنیم این دو تا توجیه داشت. حسابش را هم دیدیم حال ۵ و ۱۰ را حساب می‌کنیم. برآیند ۱۰ ← ۲۹/۵۶+ و برآیند ۵ ← ۳۰۳/۷۰- پس ۵ و ۱۰ ← ۲۷۴/۸۴- آن دو تای قبلی ۱۹۴/۱۶- بود پس ۵ و ۱۰ و ۱۲ و ۱۳ ⟵ ۴۶۸/۳۱- قطعاً الان توجیه شدم که جابجا کنیم و tree بودنش را به هم نریزم ۵ و ۱۰ و ۱۲ و ۱۳ با هم جابجا می‌کنیم که به این ترتیب این بردار از دست می‌دهیم اشکال ندارد به جایش این بردار اضافه می‌شود بنابراین ساختار tree حفظ می‌شود. خوب حالا باید چند روز جابجا شویم ؟ چقدر جلوتر برویم ؟ فاصله ۱۳ تا ۱۴ ، ۱۹ تا ۴۴ چقدر می‌شود ۲۵ روز کدام‌ها ؟ ۵ و ۱۰ و ۱۲ و ۱۳ را ۲۵ روز اضافه کنید این ۸ می‌شود ۳۳، این ۱۱ می‌شود ۳۶، این ۱۴ می شود ۳۹ ، این ۱۹ می‌شود ۴۴ روی tree هم واضح است باید چی کار کنیم نه ؟ خوب من اینها را دیگر ضرب در نزدم bold شده این بردار به tree اضافه شده و این از tree حذف شده است. پایان ۱۳ را با شروع ۴۴ بسنجیم این ۱۹ پایان ۱۳ است پایان تا شروع فعالیت آخر مجازی است پس شروع و پایانش یکی است.

$F_۵ = ۶$ $F_۵ = ۸$ $F_۵ = ۳۳$

$F_{۱۰} = ۹$ ✓ $F_{۱۰} = ۱۷$ ✓ $F_{۱۰} = ۳۶$

$F_{۱۲} = ۱۴$ ✓ $F_{۱۴} = ۳۹$

$F_{۱۳} = ۱۹$ ✓ $F_{۱۳} = ۴۴$

$F_{۱۳} = ۱۹$

$F_{۱۴} = ۴۴$ } ۲۵

تا به اینجا سه بار انجام شد الان بار چهارم است.

Recursion (۱)

بعد ۳ مثبت است، ۶ عضو Tree و مثبت است اول این شاخه را بگید فوقش مثبت است و منفی در مسیر نیست که هیچ ، ۷ مثبت است بعدی ۱۱ و ۸ مثبت است این شاخه هم مثبت است همه شاخه‌هایی که مانده‌اند تا آخر مثبت است و دیگر منفی ندارد نتیجه : به اتمام رسید و جواب بهینه است.

آن زمانبندی دیگر بهینه است جواب بهینه این مسئله است فقط tree را search کردیم همان گوشه‌ها هستند.

این ایده آقای ون هاگ داره برای مسئله Max-NPV است البته بدون داشتن محدودیت منابع . اول همه را در زودترین چیدیم بعد منفی ها را سعی می‌کنیم به تأخیر بندازیم و تا آنجایی که می شود به تأخیر انداختیم دیگر بیشتر از این توجیح ندارد. این مثالی که حل کردیم اکثرش مثبت بود سه تا منفی بود حال فرض کنید یک پروژه‌ای دارید که اکثر فعالیت‌هایش منفی است و بعضی‌ها مثبت اند بهتر است این پروژه را چطوری حل کنیم ؟ برعکس این عمل کنید اول همه را در دیرترین زمان ببریم و بعد سعی کنیم مثبت‌ها را نزدیک ببریم این پیشنهاد ایشون است که در ادامه اسلایدها است که اسمی هم برایش گذاشته است. در همچین مواردی Procedure شروع می شود با (به جای early tree می‌گه late tree بهش) یعنی فعالیت‌ها در دیرترین زمان بچینید مثبت‌ها را بیارید عقب. فوقش این مثال که ما حل کردیم اینطوری نبود. این مثال اکثرش مثبت بود و سه تا منفی داشت. ولی خوب اگر عکس بود. پس خلاصه کدام را استفاده کنیم ؟ در زودترین زمان بچینیم و منفی را عقب بندازیم یا نه در دیرترین بچینیم و و مثبت‌ها را جلو ببریم. نگاه کنید ببینید که اکثرش مثبت است یا منفی . اگر اکثرش مثبت بود همین کاری که الان انجام دادید را انجام دهید اما اگر اکثر منفی است برعکس. هر دو تاشان یک مسئله است. هر دو تا NPV را حساب می‌کند. تعداد مهم است تعداد تکرارها بستگی به تعداد اینهایی که مثبت یا منفی هستند دارد.

$F_{۱} = ۰$

$F_{۲} = ۶$ ✓ $F_{۲} = ۴۰$

$F_{۳} = ۵$

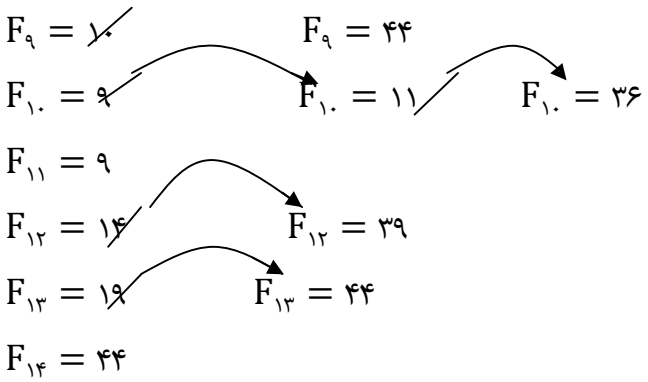
$F_{۴} = ۳$

$F_{۵} = ۶$ ✓ $F_{۵} = ۸$ ✓ $F_{۵} = ۳۳$

$F_{۶} = ۱۱$

$F_{۷} = ۷$

$F_{۸} = ۸$



پارت ۹

پارت ۹ این مسئله باید صحبت شود مینیمم کردن جریمه‌های زودکرد و دیرکرد پروژه

Minimizing weighted earliness-tardiness penlty costs.

یعنی زودکرد یعنی دیرکرد

در خط ۲ یک اصطلاحی به اسم Just in time (JIT) است یعنی تولید به هنگام یا تحویل به هنگام یعنی بهتر است فعالیت‌ها دقیقاً در یک تاریخ مشخص هر کدام انجام شود اگر شما زودتر یا دیرتر انجام دهید جریمه می‌شوید به این محیط محیط Just in Timeمی‌گویند مثالی که می‌شود زد مثلاً فرض کنید چهارم آذر ماه می‌خواهیم بتن‌ریزی یک ساختمان را انجام بدهید بنابراین مصالح مثلاًسیمان کی باید بیاید؟ یک روز قبلش حالا اگر یک هفته قبل سیمان وارد شده باشد چه اتفاقی می‌افتد ؟ می‌ماند زیر باران . آن وقتی باید کلی کارگر به جای اینکه در بخش‌های مختلف پروژه کار کنند باید به کار گرفته شوند تا این مصالح را به سوله جابجا کنند. نتیجه کار را زود انجام داده‌اید ولی کار مثبتی نبوده است. یعنی لزوماً زود انجام دادن کار حسن نیست. در واقع این سیمان را تا ۳ ام آذر ماه نمی‌خواستیم و شما چون یک هفته زودتر این مصالح را آوردید فقط دردسر درست کرده‌اید. این یک earliness بی‌فایده و زیان بار می‌باشد. حالا دیرکردش هم همین طور است . مثلاً این کار باید فلان تاریخ شروع می‌شده شما مصالح را نیاوردید و باعث شدید که کل تیم، کل پروژه فعالیت نداشته باشند. پروژه به خاطر عدم داشتن مصالح خوابیده است. یعنی on-time بودن خیلی از کارها در واقع به ضرورت است و دیر یا زودش مشمول جریمه است.

اینجا این مسئله را (WETPSP) گفته است که ابتدای حروف اختصاری Weighted Earliness-tardiness Penlty Costs است . کلمه weighted به چه معنا است ؟ وزن دهی .

وزن دهی یعنی چه ؟ خود زودکرد و دیرکرد ممکن است جریمه‌هایش یکسان نباشد. ما یک روز زودتر سیمان را آوردید باعث ۲ ساعت معطلی کارگر شده است که اینها را جابجا کردند اما اگر دیر می‌آوردید چی ؟ کل پروژه تعطیل می‌شد. منظور این است که یک روز زودتر و یا یک روز دیرتر لزوماً جریمه‌اش مساوی نیست. وزن‌اش ممکن است سبک و سنگین باشد. کلمه weighted به این خاطر آورده شده است. چه طور هدفی است ؟ این مسئله تابع هدفش منظم است یا نامنظم؟ همان اول در شروع گفته است نامنظم است . آن هفته نوشتیم که اهداف مالی نامنظم است حالا می‌خواهد NPV باشد ، earliness یا tardiness یا Started time dependent costs باشد فرقی نمی‌کند پس این اهداف نامنظم هستند.

خوب یک پروژه‌ای داریم که شبکه اش AON است و پیش نیازی‌هایش از نوع Finish to Start است فعالیت‌ها از یک تا n شماره گذاری شده است و فعالیت ۱ , n مجازی هستند.

d_i یعنی duration است. F_i یعنی Finish time است. و مجهول است.

اما پارامتری را که تا الان نداشتیم h_i است. هر فعالیتی یک h_i دارد . h_i همان موعد تحویل است. این که گفتم مصالح را باید سوم آذر می‌آوردید آن سوم آذر می‌شود Due date است. (موعد تحویل) و واحدش زمان است. (معلوم است)

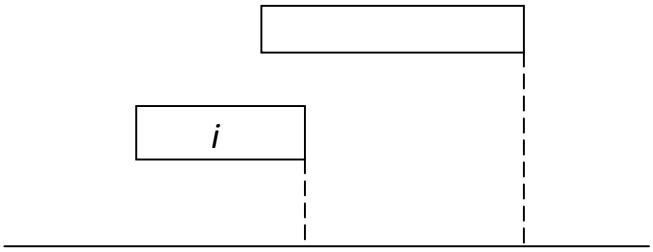
سوال : F_i بهتر است برابر با چند باشد ؟

جواب : بهتر است که پایان فعالیت همان تاریخی باشد که به اسم Due date است. یعنی $F_i = h_i$ در واقع به این معنی است که شما نه زودکرد و نه دیرکرد داشته‌اید و جریمه نداشته‌ایم.

اگر $f_i > h_i$ باشد یعنی دیرکرد داشته‌اید و اگر $f_i < h_i$ باشد یعنی زودکرد داشته‌اید که در هر دو صورت مشمول جریمه می‌شود.

سوال : h_i معلوم است یا مجهول ؟

معلوم است . چون در قرارداد پروژه بین پیمانکار و کارفرما این تاریخ‌ها ثبت شده است که هر مرحله از کار زمانبندی شده است پس h_i جزء معلومات مسئله است. به طور خلاصه چی مجهول است ؟ F_i ها مجهول هستند. F_i ها را باید طوری حساب کنید که تا حد امکان F_i ها مساوی h_i ها شوند که جریمه ندهید.



$f_i = ۶ \quad h_i = ۱۰ \quad f_i = ۱۳$

موعد تحویل

۴ روز زودکرد

زودکرد $E_i = \max(۰, h_i - f_i)$

دیرکرد $T_i = \max(۰, f_i - h_i)$

E_i , T_i مجهول هستند

فرض کنید $h_i = ۱۰$ یعنی موعد تحویل این فعالیت ۱۰ ام می‌باشد. حالا فرض کنید که این فعالیت i در زمانبندی شما $f_i = ۶$ باشد یعنی ۶ ام تمام شده است. چه اتفاقی افتاده است ؟ ۴ روز زود کرد انجام شده است . این ۴ از کجا آمد ؟ $h_i - f_i = E_i \Longrightarrow$ ۱۰-۶=۴ که در واقع E_i همان زودکرد است اگر مثل این مثال $f_i < h_i$ باشد در نتیجه $h_i - F_i = E_i$ اما حال فرض کنید که فعالیت i این طوری تمام شود یعنی یعنی $F_i = ۱۳$ است. آن وقت $h_i - f_i$ منفی می‌شود یعنی زودکرد نداریم. به طور خلاصه $E_i \neq h_i - f_i$ اگر $h_i - f_i$ مثبت باشد آره اگر مثبت باشد زودکرد و اگر منفی باشد زودکرد صفر است خلاصه اگر بخواهید فرمول بگویید $E_i = max(۰, h_i - f_i)$ نه صفر است و نه $h_i - F_i$ است هر کدام بیشتر است. اگر $h_i - f_i$ منفی باشد. زودکرد نداریم آن دیرکرد است. دیرکرد برعکس است $T_i = max(۰, f_i - h_i)$ پس E_i و f_i را به صورت بالا تعریف می‌شود میزان زودکرد i و میزان دیرکرد i پس $h_i - F_i$ زود کرد نیست، صفر هم زودکرد نیست ماکسیسم اینها زودکرد می‌شود برای T_i هم اینطور است.

سوال : E_i و T_i جزء معلومات مسئله‌اند یا باید تعیین بشوند ؟

E_i و T_i بستگی به F_i دارند چون F_i را ندارید پس E_i و T_i هم مجهول‌اند باید تعیین شوند.
دو پارامتر دیگر هم وجود دارد که با حروف کوچک نشان داده شده‌اند (e_i و t_i)، اینها جریمه‌های زودکرد و دیرکرد هستند هر روز که زودکرد داشته باشید e_i جریمه و هرروز که دیرکرد داشته باشید t_i جریمه دارد. اینها هم جزء معلومات مسئله هستند. جریمه‌ها در قراردادها نوشته می‌شود. پس E_i و T_i مجهول‌اند و e_i و t_i معلوم‌اند.
الان خلاصه یک فعالیت چقدر جریمه دارد؟ چند روز زودکرد داشته E_i به ازای هر روزش e_i و چند روز دیرکرد داشته T_i به ازای هر روزش t_i جریمه . پس $e_i E_i + t_i T_i$ می‌شود هزینه یک فعالیت. حال هزینه کل پروژه با افزودن یک *summation* (زیگما) $\sum e_i E_i + t_i T_i$ می‌شود.
تابع هدف را خلاصه حساب کردیم $min \sum_{i=2}^{n} e_i E_i + t_i T_i$ ، حالا از ۲ شروع کرده است چون یک مجازی است هر چند یک را هم بنویسید چون مجازی است مشکلی ندارد.
هدف منظم است یا نامنظم ؟ هدف نامنظم است بنابراین پروژه خودش لحظه صفر شروع نمی‌شود. هدف منظم هدفی است که خود پروژه دنبال این است که کارها زود انجام شود ولی هدف نامنظم اینطوری نیست یعنی شما باید به زور بگید که $F_1 = 0$ است اما اگر هدف منظم باشد من نمی گم $F_1 = 0$ خودش سعی‌اش بر این است. دیگر شما نمی‌خواهد اجبار کنید پس بدون استثناء هر هدف نامنظم باید بهش ملزوم کنیم چه کار کند ؟ پس شروع پروژه صفر را باید بهش بگید و گرنه در لحظه صفر شروع نمی‌شود چون منظم نباشد تابع هدف اصطلاح خاصیت کششی ندارد. که این را به سمت صفر بکشید پس محدودیت ۴ ام را اول گفتم این چی می‌گه رابطه پیش نیاز است.
$f_i \leq f_j - d_j \longleftarrow f_j - d_j$ یعنی چه ؟ پایان فعالیت منهای *duration* فعالیت می شود شروع فعالیت. این می‌گه شروع فعالیت j باید بزرگتر باشد از پایان فعالیت i همان *finish to start* شروع j باید بعد پایان i باشد. پس این رابطه پیش نیازی *finish to start* است.
بعدی‌ها چی می‌گن ؟ جملات زیر با هم چه تفاوتی دارند؟
فرض کنید از شما میان ترم گرفته شده است و گزارشی که از نتایج امتحان داده می شود به صورت زیر است.

- ماکسیمم نمره کلاس ۱۷ است.
- یا همه نمره‌ها کمتر مساوی ۱۷ است
- یا هیچ نمره‌ای بالاتر از ۱۷ نداریم.

هر سه جمله بالا یک مفهوم را می‌رساند و هیچ فرقی با یکدیگر ندارند. حال اگر بگویید ماکسیمم این دو تا E_i یا بگید E_i از هر دو آنها بزرگتر است فرق نمی‌کند. اینجا $E_i = max\{0, h_i - f_i\}$ و در مدل نوشته شده است $E_i \geq h_i - f_i$ یعنی E_i هم از $h_i - f_i$ بزرگتر است و هم از صفر . برای آن یکی هم همین طور به جای اینکه بگویم $T_i = max\{0, f_i - h_i\}$ گفتم T_i از هر دو اینها بزرگتر است. در فرمول‌ها کلمه *max* به این دلیل برداشته شد که مسئله خطی شود. تمام. این مدل نهایی مسئله است.

$$\min \sum_{i=2}^{n} (e_i E_i + t_i T_i) \qquad (1)$$

s.t :

$$f_i \leq f_j - d_j \qquad \forall (i,j) \in A \qquad (2) \longrightarrow \text{به ازای هر بردار}$$

$$E_i \geq h_i - f_i \qquad \forall i \in N \qquad (3) \longrightarrow \text{به ازای هر فعالیت}$$

$$T_i \geq f_i - h_i \qquad \forall i \in N \qquad (4) \longrightarrow \text{به ازای هر فعالیت}$$

$$f_1 = 0 \qquad (5)$$

$$f_i \geq 0 \ , E_i \geq 0 \ , T_i \geq 0 \qquad \forall i \in N \qquad (6)$$

یکی ⟵ عدد صحیح *Integer*

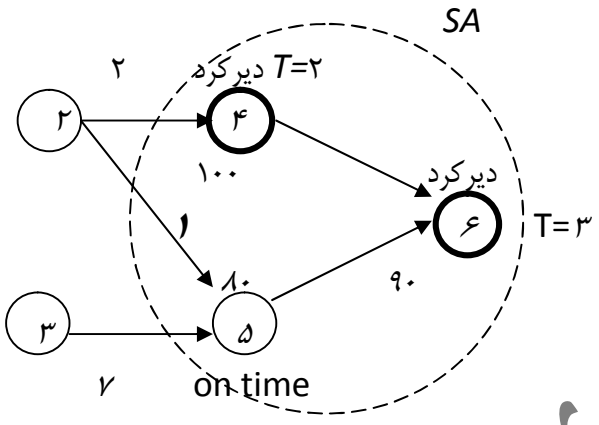
کلمه *max* را چطور می‌توان به شاخه و کران حالی کنی ؟ ولی این یکی را می‌توانید یا در صفحات برشی این را راحت می‌توان بنویسید. پس تعدادی مهم نیست تعداد محدودیت‌ها، فرمت اش هم مهم است خطی یا غیرخطی بودنش هم مهم است. الان برای نوشتن یک مدل یک مسئله چه تعداد متغییر و محدودیت باید تعریف کنیم ؟
این مدل چند تا متغیر و چندتا محدودیت دارد ؟
مثلاً یک پروژه ۵۰ تا فعالیت دارد. می‌خواهم مدل ریاضی‌اش را بنویسم و هدف مینیمم کردن همین جریمه‌های زودکرد و دیرکرد است این مدل که می نویسم چند تا متغییر دارد و چند تا محدودیت می‌خواهد. اول متغییرهایش را بگم که(۵۰×۳)، ۱۵۰ است چرا ؟ *F,E,T* مجهول اند هر فعالیتی یک *finish time*، یک *earliness* و یک *tardiness* دارد. بنابراین وقتی ۵۰ تا فعالیت دارید ۱۵۰ تا متغیر باید تعریف کنید. به طور کل اگر n تا فعالیت دارید $3n$ تا متغیر باید تعریف کنید. چند تا محدودیت باید بنویسید ؟ بشمارید دیگه این چند تا است ؟ اینها چند تا است ؟ این (۵) یکی است. (۲) این به ازای هر فعالیت نیست این پیش‌نیازی مگر نبود . این به ازای هر بردار است i پیش نیاز j باید یک خط بنویسید. این را ببینید اسم بردارها را با نماد A نشان داده است. اگر A مجموعه بردارها باشد، جلوش اصلاً خودش گفته است خیلی ساده است به ازای هر i,j باید نوشته شود به ازای هر برداری باید نوشته شود اما خوب بعدی‌ها (۳) به ازای هر فعالیت و ما وقتی n تا فعالیت داریم اینم n تا است اینم n تا (۴) می‌شود پس $2n$ البته این یکی هم است. ⟵ به ازای هر بردار
$|A| + 2n + 1$ خلاصه اگر می‌خواهید یک مدل روی کاغذ بنویسید مثلاً برای ۵۰ تا فعالیت باید ۱۵۰ تا متغیر تعریف کنید و ۱۰۱ به اضافه تعداد بردارها هم محدودیت.
حذف کنی چی می‌شود فکر کنی ؟ پس مدلت دیگر نمی‌داند f صفر است اگر من پاکش کنم مدل حل می‌کند ولی F_1 را صفر نمی‌دهد. اتفاقاً من دارم به اجبار می‌گم باید F_1 صفر باشد.
وقتی جزء محدودیت است نمی‌توانید حذف کنید. اینها را نمی‌شمارند محدودیت ضمنی می‌گویند. محدودیت‌هایی که فقط علامت را می‌گه توی تعداد شماره نمی‌شوند به اینها محدودیت‌های کارکردی می‌گویند.
Integer (عدد صحیح) بودنش را هم باید بنویسیم. یعنی مدل باید عدد صحیح باشد خطی، عدد صحیح است که با چی حل می شود ؟ ببینیم کسی میگه *simplex* ، گفتید ؟ می‌گم عدد صحیح ، *simplex* چرا ؟ باید با شاخه و کران حل شود. خطی است ولی عدد صحیح است پس چرا در جلسه قبل گفتیم که با *simplex* حل شود چون آن مسئله یک ویژگی خاص داشت و همان ویژگی اش باعث شد که با *simplex* حل کنیم و این بود که گوشه‌هاش عدد صحیح بود یعنی استثنا بود در واقع هر مسئله خطی، عدد صحیح را که با *Simplex* حل نمی‌کنیم. آن یک استثنا بود. البته تنها نیست مسائل شبیه آن استثنا است که ساختارشان طوری است که گوشه‌هایشان عدد صحیح است به اصطلاح *LP Relaxation* اش عدد صحیح است ولی این اینطوری نیست. اینها نمی‌توانید با *simplex*حل کنید با شاخه و کران یا صفحات برش باید حل کنید.
در ادامه ، آقای ون‌هاک *(Vanhoucke)* برای مسئله *WETPSP* روشی را پیشنهاد داده است.
پس ما مسئله را مدل کردیم و اگر بخواهیم براساس اطلاعات قبلی حل کنید با روش شاخه و کران و یا صفحات برشی باید حل کنید که خیلی وحشتناک است چون مثال‌هایی که تا الان داشته‌اید و با شاخه و کران حل کردید ۲ تا ۳ تا

متغیر داشت و ۲ تا ۳ تا محدودیت داشت. اما در این مثال که ۵۰ تا فعالیت است و دارای ۱۵۰ متغیر و بیش از ۱۴۰ و ۱۵۰ تا هم محدودیت دارد حل کردن این مسئله با روش فوق راحت نمی‌باشد. آقای ون هاک روشی کاراتری را نسبت به شاخه و کران و یا صفحات برشی ارائه دادند که دارای دو گام است :

گام یک : به اصطلاح خودش یک *due date tree* می‌سازد که تنها یکبار انجام می‌شود.
گام دو : دوباره یک *recursive search* (روش جستجوی بازگشتی) دارد.
در گام اول این روش به ترتیب شروع می‌کند از فعالیت ۱ ، ۲ ، ۳ تا n همه فعالیت‌ها را سعی می‌کند کجا بچیند تا تمام بشوند ؟ سعی میکند فعالیت‌ها را طوری برنامه ریزی کند که همزمان با هم یعنی $f_i = h_i$ ، *due date* بشود تا جریمه نشود. اما اگر به خاطر پیش نیازی‌ها نتوانست این کار را انجام بدهد چی کار می‌کند ؟ دیرتر
مثلاً ، فرض کنید محاسبات *forward* را انجام دادم و برای فعالیت زودترین زمان پایان $F_i =$ ۱۷ است محاسبات *forward* را انجام دادم *earliest finish time* اش ۱۷ ام است آن وقت *due date* این روز ۱۲ ام است. $h_i =$ ۱۲ است. در قرارداد پروژه نوشته شده است این پروژه باید ۱۲ ام تمام شود ولی طبق پیش‌نیازی محاسبات رفت را انجام می‌دهید زودترین زمان پایانش ۱۷ ام است.
نتیجه: نمی‌توانید این را ۱۲ شروع کنید هر چند که توی قرارداد نوشته شده است اصلاً غیر ممکن است که بخواهید ۱۲ ام تمام کنید به گفتن نیست پیش‌نیازی‌هاش اجازه این را نمی‌دهد که ۱۲ ام تمام کنید پس ناچار هستید ۱۷ ام تمام کنید یعنی دیرتر. پس گام یک سعی میکند در h_i بچیند. ولی مثل این مثال غیر ممکن بود. ناچار می‌شود دیرتر بچیند در ۱۷ ام بچیند. پس آخر گام یک همه فعالیت‌ها یا *on-time*اند در h_i تمام شدند یا دیرتر تمام شدند. زودکرد نداریم.
حالا گام دو دنبال چی می‌گردد که اینها را بیاورد عقب‌تر بر خلاف روش *NPV* است که فعالیت‌ها را به تأخیر می‌انداخت الان به دنبال این است که اینها را چی کار کند ؟ به اصطلاح خودش گفته *backward shift* (یعنی به سمت صفر بکشاند) نه اینکه به سمت جلو هل بدهد. چرا به سمت جلو هل دادنش فایده‌ای ندارد ؟ مگر نگفتید همه‌اشی یا *on-time* اند یا دیرکرد دارند .

حالا اگر یک فعالیت که *on-time* است یا دیرکرد دارد به تأخیر بندازید چه حسنی دارد ؟ اصلاً حسنی دارد؟ آنی که تأخیر دارد که به تأخیر انداختنش ، تأخیرش را بدتر می‌کند و آنی که *on-time* است و بازم به تأخیر می‌افتد پس هیچ حسنی قطعاً ندارد. پس *forward shift* نداریم ولی *backward shift* چه حسنی دارد ؟ چرا ممکن است حسن داشته باشد ؟ به هر حال آنی که دیرکرد داره را دیرکردش را کم می‌کند اما یک مشکلی وجود دارد آنی که دیرکرد ندارد چرا اصلاً دیرکرد دارد . چرا از اول *on-time* تمام نشد؟ چون این فعالیت پشت سرش یک پیش‌نیاز داشت که این پیش نیاز نداشته . این مشکی‌ها *on-time* است این آبی دیر کرد دارد من اگر بخواهم این را بیارم عقب‌تر ، دیرکرد آبی را کم می‌کنم مطمئناً ولی اینی که *on-time* بود (مشکی) دارم برایش زودکرد ایجاد می‌کنم. خوب اگر هزینه دیرکرد این ۱۰۰ دلار باشد و هزینه زودکرد این ۷۰ دلار باشد برآیندش می‌ارزد من وقتی یک روز این را می‌آورم عقب ۱۰۰ دلار جریمه کم می‌دهم هر چند که ۷۰ دلار دوباره جریمه می‌شود. در نتیجه ۳۰ دلار منفعت است . گام دوم

دنبال پیدا کردن این مجموعه به اصطلاح خودش همان اسم قبلی را دوباره گذاشته *SA*هایی را مجموعه فعالیت‌هایی پیدا کنید جابجایی ***backward*** شان مقرون به صرفه باشد.



در مثال بالا :

عرض کنم که فعالیت ۵ ، ***on-time*** است و فعالیت ۴ و ۶ دیرکرد دارند. من بررسی کردم برآیند سه تا فعالیت که با *SA* نشان داده‌ام اگر جابجا شود یعنی عقب‌تر بیاید به نفع است . فرض کنید حالا چرا برآیندش به نفع من است به خاطر اینکه فعالیت‌هایی که دیرکرد بابت فعالیت ۴ ، ۱۰۰ دلار و بابت فعالیت ۶ ، ۹۰ دلار جریمه می‌دهم و بابت فعالیت ۵ که ***On-time*** است و الان جریمه نمی‌دهم اگر زود تمام شود ۸۰ دلار جریمه خواهم داد . اشکال نداره من حاضرم ۹۰ دلار از اینجا کم بشود، ۸۰ دلار اینجا اضافه شود درکل اش توجیح دارد که جابجا شود حالا اینجا باشید. حالا سوال میزان جابجایی است ؟ چند روز باید عقب تر برود ؟ توجیح دارد چند روز ؟ دیرکرد ۴، ۲ روز می‌باشد و دیرکرد ۶، ۳ روز است و ۵ هم ***on-time*** است و دیرکرد ندارد. الان ۴ و ۵و ۶ بیاید عقب چند روز جابه‌جا شوند باهم ؟ دو یا سه روز ؟ دو روز باید جابجا شوند. ببینید اصلاً اینها را برای چی جابجا می‌کنید چون دیرکرد دارند و من می‌خواهم دیرکردشان را کم کنم اما خوب اگر شما از یه حدی بیشتر عقب بیایید آن وقت از این طرف می‌افتید پایین. متوجه هستید. دیر کرد داشته است آوردید عقب و ***on-time*** شده خوب دیگه عقب‌تر نیاید دیگه ***on-time*** شده کافی است. و عقب تر بیایید زودکرد می‌شود.

پس چقدر باید به عقب بیاد؟ به اندازه دیرکرد. به اندازه کدام دیرکرد ؟ آن دیرکردی که مینیمم است. *W* را زیرش یک فلش بزنید همین ۲ در این مثال است. *W* یعنی مینیمم دیرکرد آنهایی که *SA* هستند و دیرکرد دارند. پس *W* مینیمم دیرکرد فعالیت‌های مجموعه *SA* که دارای دیرکرد هستند. البته یک مثال را کامل حل خواهیم کرد.

$$w = \min_{\substack{y \in SA \\ f_y > h_y}} \{ f_y - h_y \}$$

۲

جدای بحث دیرکردها موضوع دیگری هم هست که مانع عقب‌نشینی زیاد می‌شود آن هم پیش نیازهایشان است. ۵ تا کجا می‌تواند عقب بیاید تا جایی که ۲ و ۳ اجازه بدهند. و چهار چقدر عقب می‌تواند بیاید ؟ تا جایی که ۲ اجازه بدهد و مماس شود.

حالا فرض کنید فاصله ۲ و ۴ مثلاً دو روز است و فاصله ۲ و ۵ یک روز است و فاصله ۳ و ۵ هم ۷ روز است با توجه به فاصله اینها مینیمم فاصله بین ۲ و ۵ یک است نتیجه یک روز بیشتر از نظر پیش‌نیازی نمی‌توانند عقب بیایند. این یک همان V است . V مینیمم فاصله فعالیت‌های مجموعه SA با پیش‌نیازهایشان.

$$v_{k^*l^*} = \min_{\substack{(k,l)\in E \\ k\notin SA \\ l\in SA}} \{f_l - d_l - f_k\}$$

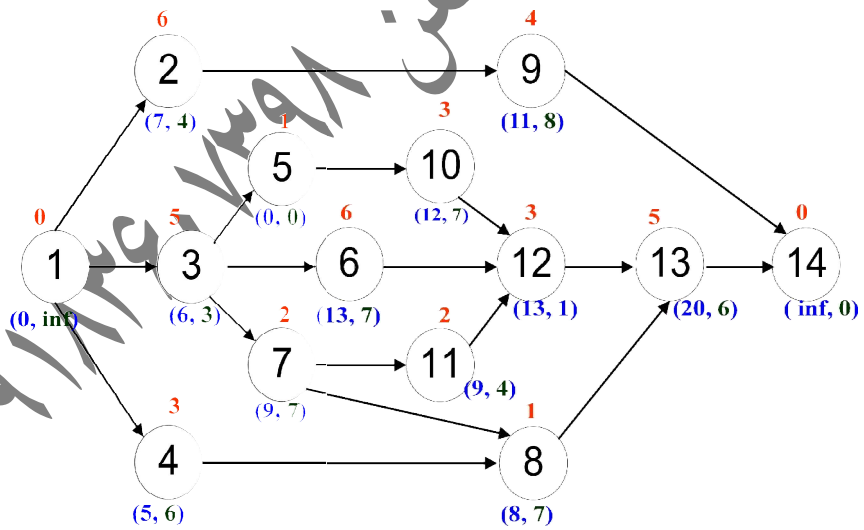
۱

خلاصه کدام مبنا است V یا W ؟ باز هم مینیمم شان یعنی خود Wمینیمم یک عبارتی است و خود V هم مینیمم عبارتی است ولی آخر سر بین V و W مینیمم تعیین کننده جابجایی است.

نهایتاً $min\{V,W\}$ مشخص کننده میزان $backward\ shift$ مجموعه SA است.

خوب حالا که جابجا کردید چه می‌شود آن $tree$ به روز می‌شود. دوباره. $Recursive$ است . من یکبار انجام دادم و حالا این باید بارها و بارها تا زمانی که جواب داره بهتر می‌شود تکرار شود.

مثال :



یک مثالی است که قرار صفر تا ۱۰۰ اش را حل کنیم . در این مثال که ۱۴ فعالیت دارد و فعالیت یک و ۱۴ مجازی هستند اعدادی که بالای فعالیت‌ها یا قرمز نوشته شده است $duration$ است و اعدادی که آبی نوشته شده است $due\ date$همان h_i است و آن یکی عدد که با رنگ سبز نوشته شده است $unite\ penalty\ cost$ است. یعنی جریمه یعنی کدام ؟ ما دو تا جریمه داشتیم یکی e_i و دیگری t_i است. اینجا برای جلوگیری از شلوغ شدن فرض

کرده‌ام که e_i با t_i برابر است یعنی چه زود چه دیر جریمه‌اش روزی اینقدر است. به اصطلاح *weighted* دیگر نیست. یه روز زودتر یا یه روز دیرکرد فرقی نمی‌کند. برای فعالیت ۱۱، ۴ دلار است یه روز زودتر تمام کنید ۴ دلار و یه روز دیرتر تمام کنید بازم ۴ دلار است. فرقی نمی‌کند. اما درست تر آن بود که برای e_i یک عدد و برای t_i هم یک عدد می‌نوشتم ولی در اینجا فرض بر این است که $e_i = t_i$ است. در حل تأثیری در شدت یا سخت راه حل ندارد.

گام یک : سعی کنید فعالیت‌ها را *On-time* بچیند و اگر چاره‌ای نبود دیرتر هم اشکال ندارد.

۱ ← $F_1 = 0$ چون مجازی است.

۲ ← $F_2 = 7$ ، ۷ از کجا آمد ؟ چون $h_i = 7$ است و پیش‌نیازش یک و مجازی است عملاً ندارد می‌تواند روز یکم شروع شود و ۶ روز هم طول می‌کشد ۷ ام تمام می‌شود. پس پایان می‌نویسیم، شروع یکم است و پایان هفتم، ۷ ام *due date*

۳ ← $F_3 = 6$ اینم یک شروع می‌شود ۵ روز طول می‌کشد ۶ ام تمام می‌شود.

۴ ← $F_4 = 5$ دوم شروع می‌شود (۲=۳-۵) سه روز بعد ، ۵ ام تمام می‌شود.

۵ ← $F_5 = 7$ خودش گفته کی باید تمام شود در قرارداد نوشته شده که امروز یعنی صفرام ولی این همان حالت است دیگر گفته ولی شما که نمی‌توانید آن را صفرم تمام کنید علت هم خیلی واضح است که پیش نیاز که ۳ است ۶ ام تمام شده است فرض کنید بلافاصله هم حالا این را انجام بدهیم که خودش هم یک روز کار دارد می‌شود ۷ ام . من این را قرمز می‌نویسم به معنی این که دیرکرد دارد *on-time* نشد.

۶ ← $F_6 = 13$ ببینید اینطوری حساب کنید گفته ۱۳ ام ، ۶-۱۳ کنید و ببینید با توجه به پیش‌نیازی مشکلی دارد یا نه ؟ ۷=۶-۱۳ ، ۳ کی تمام شده است ؟ ۶ ام . خوب باشه ۶ ام تمام شده من ۷ ام شروع می‌کنم تضاد ندارد پس این معنی اش این است که این فعالیت را می‌شود ۱۳ ام و به موقع تکمیل کرد.

۷ ← $F_7 = 9$ ، ۷=۲-۹ ، پیش‌نیازی آن ۶ ام تمام شده است

پس عملاً در ستون اول فقط یک فعالیت ۵ به موقع نشده است بقیه همه *on-time* هستند.

۸ ← $F_8 = 10$ دو تا پیش نیاز دارد باید هر دو را چک کنید. ۷=۱-۸ ، ۷ نمی‌شه چون فعالیت ۷، ۹ ام تمام شده است پس نمی‌توان این را ۷ ام شروع کرد. ۹ ام یک روز هم خود این کار دارد. ۱۰ ام خوب این ۱۰ را یادتان باشد. ۸ ام مطمئناً نیست، ۱۰ ام است. این را چک کنید با این مشکل ندارد ولی خوب به هر حال اون یکی ماکسیمم مبنا است. پس فعالیت ۸ را باید بپذیرم که دیرکرد کرد دارد و ۱۰ ام.

۹ ← $F_9 = 11$: مشکل ندارد چون ۷=۴-۱۱ با پیش نیازی تضاد ندارد

۱۰ ← $F_{10} = 12$: ۹=۳-۱۲ ، ۹ ام این ۷ ام تمام شده است ۱۰ هم مشکلی ندارد.

۱۱ ← $F_{11} = 11$: ۷=۲-۹ نمی‌شه

(۱۲) ⟵ $F_{۱۲} = ۱۶$: ۱۳-۳=۱۰ ولی سه تا پیش‌نیازی که دارد را دقت کنید . ۶ ، ۱۳ ام . ما می‌خواهیم ۱۰ ام شروع کنیم پیش نیازش تازه ۱۳ ام تمام شده است پس نمی‌شود. این یکی (۱۰) ، ۱۲ ام تمام شده است و ۱۱، ۹ ام تمام شده است. ماکسیمم مبنا است پس ۱۳ ام ، سه روز هم که طول می‌کشد ۱۶ ، پس این فعالیت دیرکرد دارد.

{۱۳,۱۲,۹} ⟶max=۱۳ ⟶۳+۳=۱۶

(۱۳) ⟵ $F_{۱۳} = ۲۱$ خوب ۱۶ و ۵ می‌شود ۲۱ پس ۲۰ نمی‌شود در نتیجه ۲۱ ام می‌شود

(۱۴) ⟵ $F_{۱۴} = ۲۱$: آخری هم که مجازی است همان ۲۱ است.

$F_۱ = ۰$ $F_۲ = ۷$ $F_۳ = ۶$ $F_۴ = ۵$

$\mathbf{F_5 = 7}$ $F_۶ = ۱۳$ $\mathbf{F_7 = 9}$ $\mathbf{F_8 = 10}$

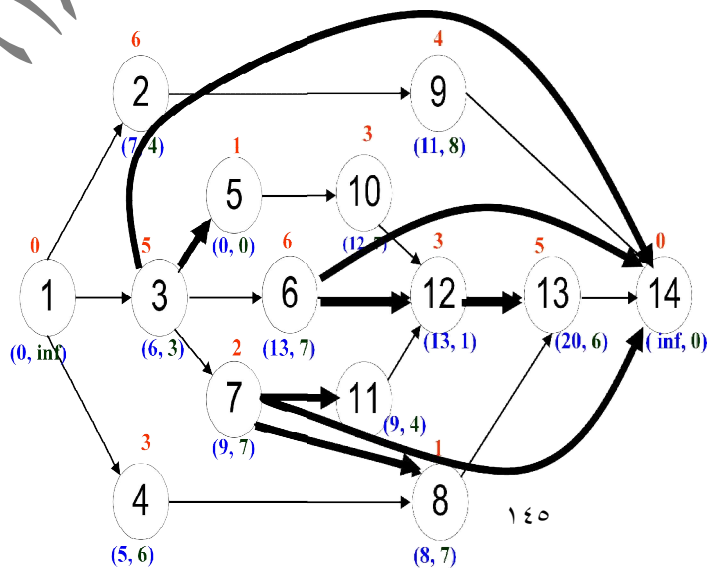
$F_۹ = ۱۱$ $F_{۱۰} = ۱۲$ $\mathbf{F_{11} = 11}$ $\mathbf{F_{12} = 16}$

$\mathbf{F_{13} = 21}$ $F_{۱۴} = ۲۱$

گام یک هنوز به اتمام نرسیده است تنها محاسباتش انجام شده است رسم tree مانده است.

کدام بردارها عضو tree می‌شوند. آنهایی که دیرکرد دارند با پیش نیازی که باعث این دیرکرد شده است چون آنها مماس‌اند مثلاً فعالیت ۱۲ را ببینید . فعالیت ۱۲ چرا ۱۶ ام شروع کردیم براش دیرکرد زدیم به خاطر اینکه فعالیت ۱۲ به خاطر فعالیت ۶ عقب افتاده است چون ۶ در ۱۳ ام تمام شده است یعنی ۶ اگر باشد بعد ۱۲ این می‌خواست ۱۳ ام تمام شود ولی فعالیت ۶ پشت سرش بود این را جابجا کرد پس اینها عضو tree هستند پس این بردارها عضو tree می‌شوند.

کدام ؟ الان گفتم آنهایی که دیرکرد داردبا آن پیش‌نیازی که باعث این دیر کرد شده است. این بردار ها عضو tree هستند. (۳,۵) , (۶,۱۲) , (۱۲,۱۳) , (۷,۱۱) , (۷,۸) اما اگر دقت کنید این tree چه اشکالی دارد. تعریف tree این بود که باید همبند باشد ولی در اینجا همبند نیست start اینها را وصل کنید به چند ؟ به ۱۴ که به اصطلاح همبند شود یعنی ۱۴ نقطه اتصال اینها است. الان گام یک به طور کل تمام شد. البته این بردارهایی که پر رنگ هستند فقط عضو tree هستند.



گام دوم : سه تا پارامتر *SA ,CA , ETC* دارد البته *SA , CA* دیگه نوشتن دوباره نمی‌خواهد چون عیناًهمان است .
SA: مجموعه فعالیت‌های در حال بررسی بود.
CA : مجموعه فعالیت‌های بررسی شده است.
ETC : برآیند مالی مجموعه *SA* است . *ETC* همان حروف اختصاری عبارت *Earliest-Tardiness Costs* است. این مجموعه که جابجا می‌شوند برآیند زودکرد و دیرکردش چقدر هزینه دارد که اسمش را *ETC* گذاشته است.
با این تعریف گام ۲ را با اصلاح *recursion* ها شروع می‌کنیم.
از کجا شروع می‌شود در *NPV* از یک بود ولی الان *tree* را ببینید اصلاً یک عضو *tree* نیست باید از آخر شروع کنیم یعنی برعکس قبل است یعنی ریشه *tree*آخرش است یعنی ۱۴ . از ۱۴ باید شروع کنیم.

Recursion (١٤)

SA={١۴}, CA={١۴}. As $f_{١۴}<h_{١۴}$, set the earliness-tardiness costs ETC=٠

برآیند مالی اش صفر است چون ۱۴ مجازی است خط بالا خط تشریفاتی بود.
بعد از ۱۴ کدام است ؟ پیش نیاز ۱۴ که شمارش نزدیک تر به ۱۴ باشد . نگویید فقط ۱۳. باید عضو *tree* باشد. ۳ و ۶ و ۷ ولی شماره‌ای که به ۱۴ نزدیک تر است ۷ است پس بعد از ۱۴ ، ۷ می‌باشد. در نتیجه

Recursion (٧):*predecessor node ٧*

SA={٧}, CA={٧,١۴}. As $f_{٧}=h_{٧}$, set ETC=٧

ETC چند است ؟ یعنی ۷ را جابجا کنید چقدر به نفع شما است ؟ پاسخ ۷ این که با مشکی نوشتم یعنی چه ؟ *on-time* است. (توخالی هستند) و توپرها دیرکرد دارند. وقتی که *on-time* است اگر بیارید عقب چه مشکلی پیش می‌آید ۷ واحد در البته حالا این اسم‌ها شماره فعالیت است این جریمه . ۷ یعنی این که ۷ واحد جریمه می‌شوید . ۷+ به ضرر است پس جابجایی انجام نمی‌دهیم.
بعدی چند است ؟ ۸
وقتی وارد این شاخه شدید باید خود شاخه را انجام بدهید و به شاخه دیگر نروید ۷ توجیه ندارد ولی ادامه شاخه از بین ۸ و ۱۱ کدام ؟ آن شماره‌ای که نزدیکتر به ۷ است ۸ می‌باشد. پس ۸ را انجام می‌دهیم.

Recursion (٨):*predecessor node ٨*

SA={٨}, CA={٧,٨,١۴}. As $f_{٨}>h_{٨}$, set ETC=-٧

ETC'=-٧<٠: SA={٧,٨}, ETC=-٧+٧=٠

ETC هشت چند می‌شود؟

۸ جزء آنهایی که دیرکرد دارد (توپر هست) اگر جابجا کنیم به نفع من است چقدر ؟ ۷ واحد . این ۷ به نفع تان است در اینجا و در قبلی ۷ واحد به ضرر بود برآیند این دو تا صفر می‌شود شما الان می‌خواهید ۸ را جابجا کنید به نفع شما است ولی پشت سرش ۷ هم هست. اگر جابجا کنید ۰=۷-۷+ خنثی می شود صفر می‌گردد.

بعدی ۱۱ ، دیرکرد دارد پس ۴ واحد به نفع تان است الان شاخه تمام شده است برآیند کل شاخه را بگویید.

Recursion (۱۱):*successor node* ۱۱

SA={۱۱}, CA={۷,۸,۱۱,۱۴}. As $f_{۱۱}>h_{۱۱}$*, set ETC=-۴*

ETC'=-۴<۰: SA={۷,۸,۱۱}, ETC=-۴+۰=-۴

۴-=۴-۷-۷+ پس توجیه شدم که این شاخه را جابه‌جا کنم یعنی برآیندش شاخه‌ای است کل شاخه یا توجیه دارد یا ندارد اگر توجیه دارد یعنی جمع اینها منفی است باید انجام بشود وگرنه انجام نمی شود. که اینجا هم منفی است باید جابجا شود و در غیر اینصورت هیچی حالا چند روز ؟ من تا اینجا را انجام دادم.

۱۴ که هیچی ، ۷ را انجام دادم شد ۷ ، ۸ را انجام دادم شد ۷- و جمعاً صفر بعد ۱۱ را انجام دادم شد ۴- که با صفر کلاً شد ۴- تا اینجا.

حالا چند روز ؟ برای چند روز این را باید به من بگویید. ***W*** چند است ؟ دو فرمول داشتیم . ***W*** چی بود ؟ ***W*** مینیمم دیرکرد فعالیت‌هایی که در ***SA*** که دیرکرد دارند. الان ***SA***ها کدام اند ؟ ۱۱ و ۸ و ۷ ⟵ ***SA*** هستند.

کدام یک از این ***SA*** ها دیر کرد دارند.

۷ ← ***On-time*** است ***W=۲***

۱۱ و ۸ ← دیرکرد دارند. ***V=۱***

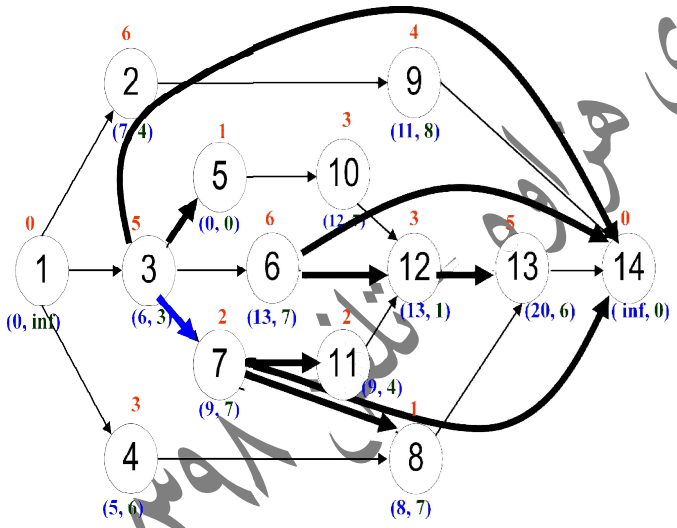
چند روز دیرکرد دارند ؟ ۸ را ببینید؟ ۸ ، ۱۰ تمام شده که باید ۸ام تمام می‌شده ، ۲ روز دیرکرد دارد . ۱۱ هم به جای اینکه ۹ ام تمام شود ۱۱ام تمام شده است اونم ۲ روز دیرکرد. ۲ روز و ۲ روز مینیمم ۲ می‌باشد. مینیمم دیرکردشان ۲ است. حالا ***A*** را بگویید. پیش‌نیازهای اینها کدام‌ها هستند و چقدر اجازه عقب نشینی می دهند. پیش‌نیازهای ۱۱، ۸، ۷ ببینید همین جای روی شکل. پیش نیاز ۷ ، ۳ می‌باشد و پیش نیاز ۸ ، ۴ می‌باشد . همین پیش‌نیازهای مستقیم . ۷ و ۳ بین‌شان چقدر فاصله است. ۳، ۶ ام تمام شده است، ۷ کی شروع شده است چون فاصله فعالیت اینطوری است که پایان تا شروع را باید ببینیم ۷-۲=۹ خوب ۶ ام تا ۷ ام یک روز. پس روی این بردار یادتان باشد عدد یک روی فاصله‌شان بنویسیم. بین ۴ و ۸ چقدر فاصله است. ۴ ، ۵ ام تمام شده است و ۸ هم ۱۰ ام تمام شده است. (۱-) ، ۹=۱-۱۰ و ۹ ام شروع شده است ۵ ام تا ۹ ام می‌شود چهار روز (۴=۵-۹) بین یک و چهار مینیمم یک است.

دوباره مینیمم بین اینها یک است. به طور خلاصه یک روز باید جابجا شوند می‌آیند عقب کدام‌ها ؟ ۷ و ۸ و۱۱ یک روز عقب می‌آیند. ۷ ایی که تا الان ***on-time*** بود زودکرد دارد که من با آبی نوشتم، تو خالی‌ها ***on-time*** هستند و توپرها دیرکرد هستند. و آنهایی که معمولی هستند زودکرد دارد ***on-time*** بود ولی الان دیگر نیست. ۸ یک روز می‌آید عقب می‌شود ۹، ۱۱ هم همین‌طور می‌شود ۱۰.

اینها یک روز عقب آمدند ولی من با توپر نوشتم ؟ چرا ؟ چون دو روز دیرکرد داشت یک روزش را حل کردید هنوز یک روز دیگر دیرکرد را حل نکردید پس هنوز ***on-time*** نشده است. پس خلاصه از دیرکرد ۸ و ۱۱ یک روز کم کردم البته به بهای یه روز زودکرد بابت ۷ ولی خوب مهم نیست. برآیندش مهم است که برآیند ۴ واحد هزینه را کم کردم.

در *tree* چه اتفاقی می‌افتد ؟ اولاً به شما بگم وقتی این شاخه عقب می‌آید این شاخه از ۱۴ کنده می‌شود این شاخه الان در ادامه این اتفاق برایش افتاد این قسمت از *tree* کنده شد.

اگر *V<W* باشد (اینجا دقت کنید که در این مثال هم *V* از *W* کمتر است) بردار K_L به *tree* ΔT اضافه کنید. K_L همان برداری است که با پیش‌نیازی‌هایش فاصله‌اش مینیمم شد. یعنی این K_I است. چرا این نیست؟ گفتم آن که فاصله اش کمتر است این را به *tree* اضافه کنید اگر *node K* یعنی همان ۳ در این مثال *starting node* یک *Arc* در ΔT نیست پس *Arc (K,n)* را هم اضافه کنید.
این که حذف شده بود این هم اضافه شد اما در این مثال اگر دقت کنید ما قبل از اینکه این را اضافه کنید خود ۳ قبلاً به ۱۴ وصل بود ولی اگر نبود باید این را چی کار می‌کردید؟ باید اضافه می‌کردید اینجا آن *if* کار نکرد گفته بود اگر نیست اضافه کن. در صورتی که وجود داشت و عملاً نیازی نبود.
گام دو یکبار انجام شد.



الان تکرار بعد، *reset* دوباره از اول پس یکبار دیگر از اول .

Recursion (١٤)

SA={١۴}, CA={١۴}. As $f_{١۴}<h_{١۴}$, set the earliness-tardiness costs ETC=٠

بعدی را شما بگید ٣ یا ۶ . معلومه ۶ . *ETC* اش چند است ؟ ٧ به موقع است با تو خالی نوشتم پس اگر عقب‌تر بیاید ٧ واحد به ضررتان است *(ETC=+٧)* یعنی بهتر است جابجا نشود یعنی *On-time* است عقب تر بیاد الکی زودکرد ایجاد می‌شود.

Recursion (٦):*predecessor node ۶*

SA={۶}, CA={۶,١۴}. As $f_{۶}=h_{۶}$, set ETC=٧

Recursion (١٢): *successor node ١٢*

SA={۱۲}, CA={۶,۱۲,۱۴}. As $f_{۱۲}>h_{۱۲}$, set ETC=−۱

بعدی‌اش ۱۲ است ۱۲ را مایل به جابجایی هستم چون دیرکرد دارد چقدر به نفع‌تان است ۱− آن یکی۷+ بود من حاضر نیستم بابت یک واحد صرفه‌جویی درهزینه‌ها ۷ واحد بیشتر هزینه کنم پس برآیندش۶+ می‌شود و فایده‌ای ندارد.

Recursion (۱۳):*successor node ۱۳*

SA={۱۳}, CA={۶,۱۲,۱۳,۱۴}. As $f_{۱۳}>h_{۱۳}$, set ETC=−۶

ETC'=−۶<۰: SA={۱۲,۱۳}, ETC=−۶+(−۱)=−۷

ETC'=−۷<۰: SA={۶,۱۲,۱۳}, ETC=−۷+۷=۰

ETC'=۰: SA={۶,۱۲,۱۳,۱۴}, ETC=۰+۰=۰

بعدی ۱۳ ، دیرکرد دارد پس جابجایی‌اش ۶ واحد به نفع تان است ولی خلاصه این شاخه ۷−=۱−۶− با ۷+ خنثی است پس هیچی در واقع این شاخه هم فایده‌ای ندارد برخلاف دفعه قبل که این شاخه را جابجا کرده بودم این بار برای این شاخه اتفاقی نمی‌افتد.

اینجا جمله‌ای گفته شده است گفته این شاخه‌ای را که تا آخر بررسی کردید توجیه نداشت را کنار بنویسید ***inactive*** (غیرفعال) چرا ؟ برای اینکه بعداً دوباره اشتباهاً سراغ این شاخه مجدد نیایید. این شاخه بررسی شده است. شاخه‌ای که بررسی کردید تا آخر توجیه نداشت ***ETC*** اش توجیه نداشت میگه اسم این شاخه را ***inactive*** بنویس این فقط محض یادآوری است که بعداً دوباره اشباه نکنی بیای این شاخه را الکی حساب و کتاب کنی یکبار حساب کتاب شده است.

خوب بعدی را چی کار کنم ؟ الان کجا برم ؟ ۳

Recursion (۳):*predecessor node ۳*

SA={۳}, CA={۳,۶,۱۲,۱۳,۱۴}. As $f_{۳}=h_{۳}$, set ETC=۳

الان تکرار دوم هستیم یکبار جابه‌جایی انجام دادیم تمام شد تکرار بعد. اینجا که ***inactive*** می‌شود ***reset***نکنید ادامه بدهید تا وقتی که شما جابجا نشدید هنوزدر این گوشه‌اید در این نقطه هستید. پس جابه‌جایی اتفاق نیافتاده است. این که رفتم به شاخه دیگر به معنی اینکه دوباره ***reset*** کنیم نیست ادامه همان است فوقش این شاخه‌اش ***inactive*** شده است توجیه نداشت شاخه بعدی ولی نه تکرار بعدی شاخه بعدی یک شاخه دیگر را چک کنید یعنی وقتی ***reset*** می‌کنیم دوباره از اول که جابجا کنید دوباره باید از اول از صفر یعنی از ۱۴ دوباره از اول انجام بدهید.

۱۴ ⟵ ۶ ⟵ ۱۲ ⟵ ۱۳ ⟵ ***inactive*** ⟵۳ ، ۳ به موقع است جابجا کنیم، ۳ واحد به ضرر است.

بعدی ۵:

Recursion (۵): *successor node ۵*

SA={۵}, CA={۳,۵,۶,۱۲,۱۳,۱۴}. As $f_{۵}>h_{۵}$, set ETC=۰

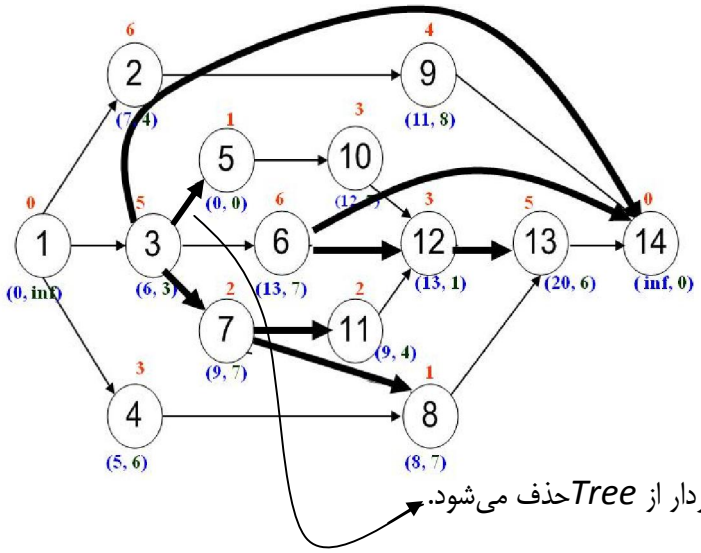
ETC'=۰: delete arc (۳,۵) from due date tree

|SA'|=۱, so no connection is made between node ۵ and node ۱۴

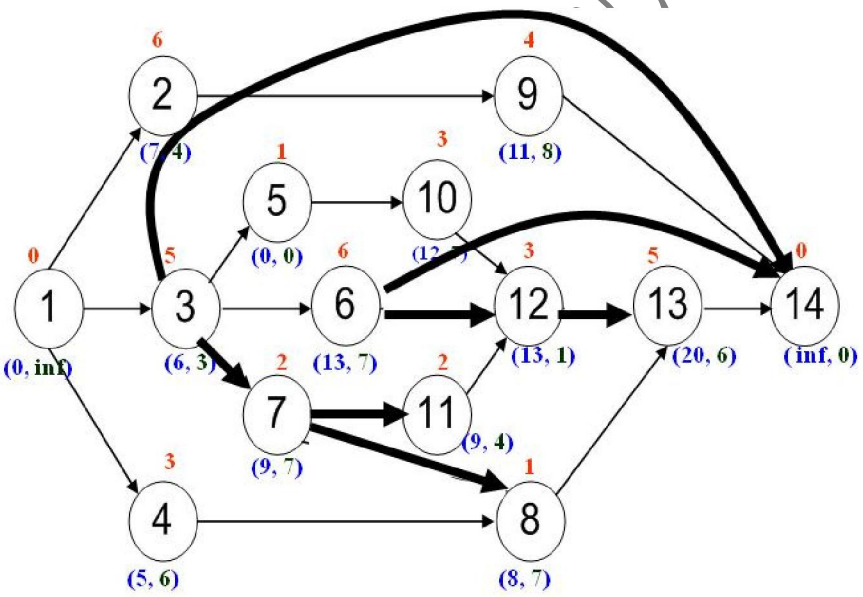
۵ دیرکرد دارد جابجا کنید اینجا چون جریمه‌اش صفر است هیچ تأثیری ندارد.

نکته : این آخر شاخه است ادامه ندارد. اگر آخر شاخه فعالیت *ETC* آن مثبت است این را از *tree* پاک کنید. به طور کل اگر فعالیت انتهای شاخه دارای *ETC* نامنفی باشد این بردار از *tree* حذف می‌شود.

پس این را حذف کردم.



اگر فعالیت انتهای شاخه دارای *ETC* نامنفی باشند این بردار از *Tree*حذف می‌شود.



بعدی ۷:

Recursion (۷):*successor node ۷*

SA={۷}, CA={۳,۵,۶,۷,۱۲,۱۳,۱۴}. As $f_۷<h_۷$*, set ETC=۷*

زودکرد دارد اگر جابجا کنید این بار ۷ واحد به ضررتان است اونم که ۳ واحد به ضررتان بود تا اینجا ۱۰ واحد می‌شود پس جابجا نمی‌کنیم.

بعدی ۸:

Recursion (۸):*successor node ۸*

$SA=\{8\}$, $CA=\{3,5,6,7,8,12,13,14\}$. As $f_8>h_8$, set $ETC=-7$

$ETC'=-7<0$: $SA=\{7,8\}$, $ETC=7+(-7)=0$

۱۱ و ۸ ، شماره نزدیک ۸ است ۸ آن بالا دیرکرد دارد و جابجا کنید ۷ واحد به نفع‌تان است ولی ۷ واحد این ۱۰ واحد بود برآیند فایده ندارد.

بعدی ۱۱، دیرکرد دارد ۴ واحد به نفع‌تان است به نظر توجیه دارد ۷ و ۴ می‌شود ۱۱ واحد منفی به نفع ما است و ۱۰ واحد به ضرر ما ولی برآیند ۱- می‌شود. که به نفع ما است نتیجه ؟ جابجا می‌کنیم. کدام‌ها ؟ این شاخه را کلاً جابجا می‌کنیم $\{3,7,8,11\}$ چند روز ؟ باید W و V را باید بگویید.

Recursion (۱۱): *successor node* ۱۱

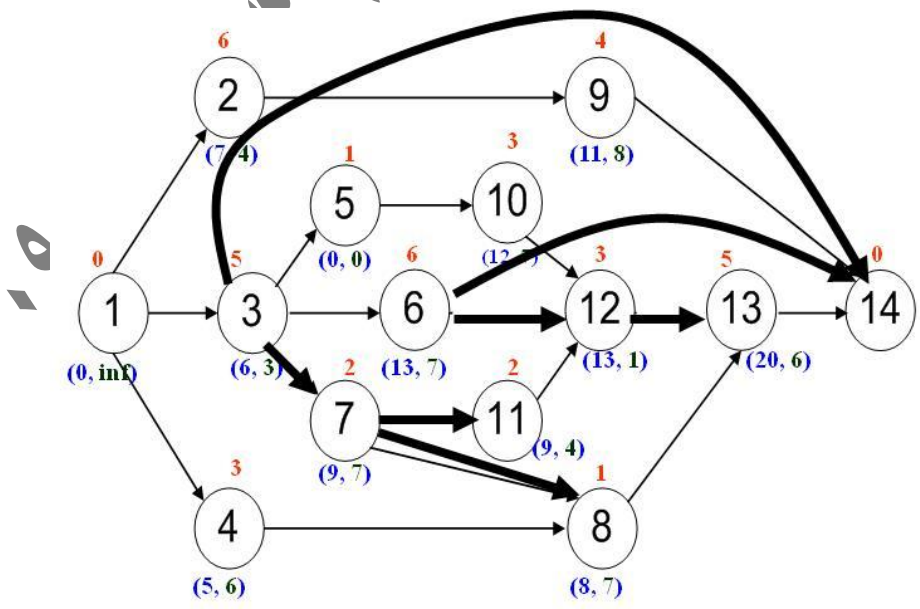
$SA=\{11\}$, $CA=\{3,5,6,7,8,11,12,13,14\}$. As $f_{11}>h_{11}$, set $ETC=-4$

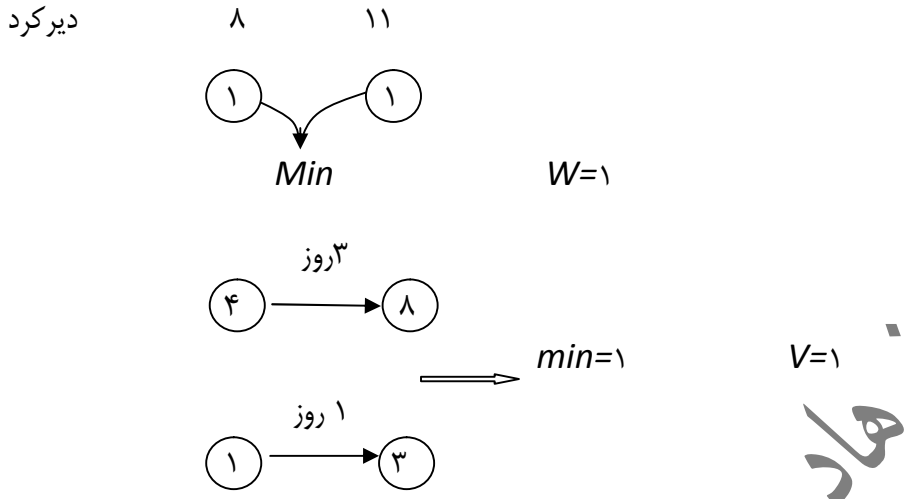
$ETC'=-4<0$: $SA=\{7,8,11\}$, $ETC=0+(-4)=-4$

$ETC'=-4<0$: $SA=\{3,7,8,11\}$, $ETC=-4+3=-1$

W این ۴ تا کدام‌ها دیرکرد دارند؟ این که *on-time* است و این یکی که زودکرد دارد هیچی کدامها دیرکرد دارند؟ این یک روز و اینم یک روز پس مینیمم یک روز حالا چقدر پیش نیازها به شما اجازه می‌دهد که عقب‌تر بیایند.

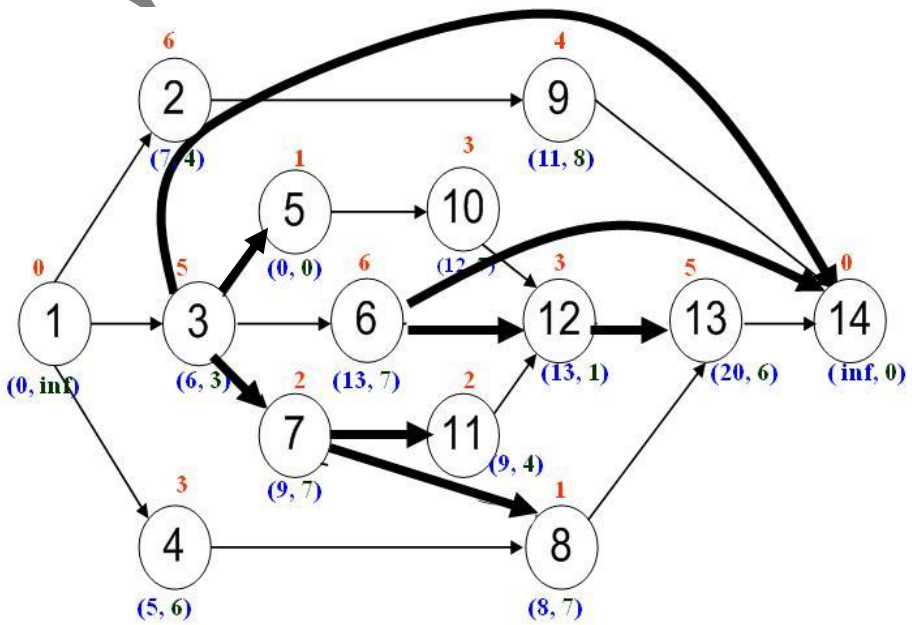
اگر اینها جابه‌جا شوند ۳ با یک درگیر می‌شود ۸ با ۴ چند روز بین آنها گپ است این که یادتان نرفته است ۴ روز بود نه ، یک روز آمدیم ۳ روز دیگه مانده این فاصله ۳ روز، این را حساب کنید چون من نمی‌دانم ۱، صفرم و ۳ ، ۶ ام تمام و ۵- ، یکم . صفرم تا یکم ، یک روز است بین ۱ و ۳ مینیمم یک می‌باشد بین ۱ و ۱ مینیمم یک است خلاصه یک روز. کدام‌ها ۳، ۷، ۸، ۱۱ باید یک روز کم شود. ۳ که *on-time* بود می‌شود زودکرد یعنی الان اومد عقب یه روز زود کرد شد ۵ اینم یه روز زودکرد داشت الان شد ۲ روز زودکرد.





اما در عوض ۸ و ۱۱ *on-time* شدند یعنی من اینها را معمولی می‌نویسم. چون اینها الان دیگه دقیقاً *due date* شان تمام نشده ولی خوب به هر حال فرآیند اینها تا اینجا توجیه داشت.

در *tree* چه اتفاقی می‌افتد، اول بگید که کدام بردار قطع می‌شود ببینید وقتی شاخه کنده می‌شود می‌آوردیش کنارش اتصالش با ۱۴ قطع می‌شود پس الان ۳ و ۱۴ را قطع شده بدانید. بعد آن اسلاید نصفه بود که نصفش را خواندم . چرا نصفش را خواندید به خاطر اینکه در آن مثال $V < W$ بود تا اینجا جواب می‌داد ولی الان چه طور شده ؟ *V=W* و گفته اگر $V \geq W$ باشد که مشمول این بند می‌شود و *SA* شامل بیش از یک فعالیت باشد و ما می‌دانیم که اینجا *SA* اینجا ۴ تا فعالیت بود، آنگاه بردار *(i,n)* را دوباره اضافه کنید یعنی *(۳,۱۴)*که حذف کرده بودیم باید دوباره اضافه کنیم که به اصطلاح با انجام این کار تضمین می‌شود ΔT، *Tree* ما جدا نشود و همبند بماند. من این بردار *(۳,۱۴)* را قطع کردم ولی دوباره باید اضافه کنم پس *tree* جدید شد.



الان reset چون جابجایی انجام دادیم برای سومین بار

Recursion (١٤)

SA={١۴}, CA={١۴}. As $f_{١۴}<h_{١۴}$, set the earliness-tardiness costs ETC=٠. Arc (۶,١۴) is inactive

بعدی ۶ یا ۳، آن inactive کجا بود، این را دیگر نباید بیایم وقتی reset می‌کنیم این شاخه دیگر به کل می‌گذاریم کنار (۶,١۴) این بررسی شده قبلاً توجیه نداشته است الان توی این فاصله جابجایی مگر برای (۶,١۴) اتفاقی افتاده ، نه ، دوباره حساب کنیم فقط وقت صرف می‌شود inactive ها دیگر چک نمی‌کنیم، پس فقط یک شاخه می‌ماند که باید دوباره چک شود در simplex یک متغیر می‌رود بیرون پایه دیگه داخل نمی‌آید. چرا می‌تواند دوباره باز هم بیاید Local دیگه شما ممکن در مقطعی فعالیتی حذف کردین و بعداً دوباره اضافه بشود.

پس این دوباره چک شود، inactive فقط ادامه ندارد. فعالیت ۷ را در دو مرحله کم کردیم کم کم تغییرات را ایجاد کردیم، فعالیت ۷و ۸ و ١١ ، ٢ بار خط زدیم، یعنی در ٢ تکرار یعنی این نیست که یک فعالیت یک بار جابجا شد دیگر کلاً تمام است قطعاً دیگر جابجا نشود ممکن باز هم جابجا بشود یا نشود.

Recursion (٣):predecessor node ٣

SA={٣}, CA={٣,١۴}. As $f_{٣}<h_{٣}$, set ETC=٣

٣ زود کرد دارد یعنی ٣ واحد به ضرر است .

Recursion (٧): successor node ٧

SA={٧}, CA={٣,٧,١۴}. As $f_{٧}<h_{٧}$, set ETC=٧

٧ زودکرد دارد یعنی جابه‌جایی ٧ واحد ضرر دارد با آن ٣ می‌شود ١٠ واحد.

Recursion (٨):successor node ٨

SA={٨}, CA={٣,٧,٨,١۴}. As $f_{٨}=h_{٨}$, set ETC=٧

ETC'=٧>٠: delete arc (٧,٨) from due date tree

|SA'|=١, so no connection is made between node ٨ and node ١۴

٨ این بار on-time ، جابه‌جایی آن دیگر توجیه ندارد چون ٧ واحد ضرر می‌شود. این بردار را باید پاک کرد، چون انتهای شاخه فعالیت توجیه ندارد این بردار پاک می‌شود.

Recursion (١١):successor node ١١

SA={١١}, CA={٣,٧,٨,١١,١۴}. As $f_{١١}=h_{١١}$, set ETC=۴

ETC'=۴>٠: delete arc (٧,١١) from due date tree

|SA'|=١, so no connection is made between node ١١ and node ١۴

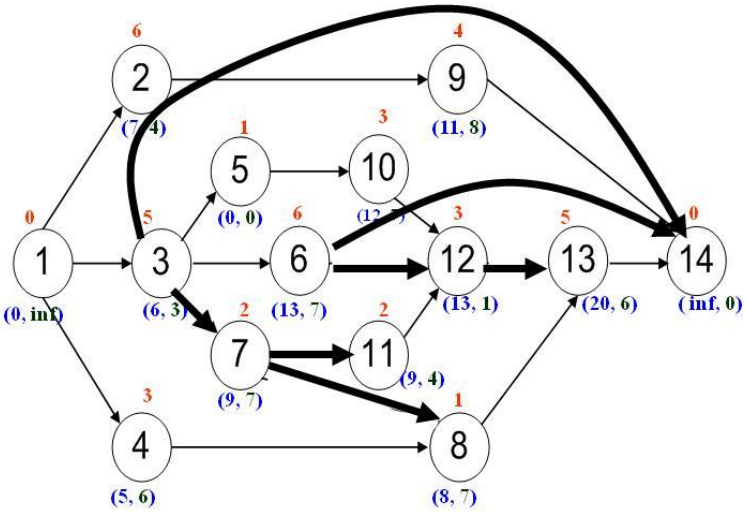
ETC'=٧>٠: delete arc (٣,٧) from due date tree

|SA'|=١, so no connection is made between node ٧ and node ١۴

ETC’=۳≥۰: SA={۳,۱۴}, ETC=۳+۰=۳

Return;

۱۱ ، On-time است، پس توجیه ندارد پس پاک می‌شود، این هم پاک می‌شود، نتیجه اگر این‌ها پاک شوند ، شاخه دیگر تمام می‌شود آن یکی هم inactive بود پس کلاً موضوع تمام شده است.



$F_{۱} = ۰$

$F_{۲} = ۷$

$F_{۳} = ۶$ $F_{۳} = ۵$

$F_{۴} = ۵$

دیرکرد $\mathbf{F_5 = 7}$

$F_{۶} = ۱۳$

$\mathbf{F_7 = 9}$ $\mathbf{F_7 = 8}$ $\mathbf{F_7 = 7}$

$\mathbf{F_8 = 10}$ $\mathbf{F_8 = 9}$ $\mathbf{F_8 = 8}$

$F_{۹} = ۱۱$

$F_{۱۰} = ۱۲$

$F_{۱۱} = ۱۱$ $F_{۱۱} = ۱۰$ $F_{۱۱} = ۹$

$F_{۱۲} = ۱۶$

دیرکرد $F_{۱۳} = ۲۱$

$F_{۱۴} = ۲۱$

جلسه دهم

در پارت‌های قبل محدودیت را برای منابع در نظر نگرفتیم این هر چند باعث ساده شدن بسیار مسئله شد ولی خیلی غیرواقع‌بینانه است که ما در واقع منابع پروژه را نامحدود فرض می‌کنیم در هفته های باقیمانده در واقع می‌خواهیم موضوع را با واردکردن این فرض، با وجود محدودیت منبع هم مدل سازی را صحبت کنیم و هم نحوه حل آن . پس عنوان طبق سرفصلی که قبلاً داشتیم محدودیت زمانبندی پروژه با محدودیت منابع است که به صورت اختصاری مسئله RCPSP می‌گویند. اگر RCPSP را search کنید شاید صدها مقاله را با همین کلید واژه پیدا کنید. مسئله RCPSP پس عنوان یک کلمه تغییر کرد اونم اینکه محدودیت منابع هم اضافه شد یعنی عنوانش خیلی ساده تغییر کرد ولی ماهیت مسئله چه تغییری کرده است ماهیت مسئله تغییرش این بوده است که مسائلی که تا الان داشتیم که به اصطلاح در کلاس P یعنی مسائل آسان بودند ولی به محض وارد شدن محدودیت منبع سختی مسئله بسیار زیاد و در کلاس NP و NP-hard قرار می‌گیرند بنابراین روش‌های حل صرفاً exact ممکن است پاسخگوی کار شما نباشد در کنار روش‌های exact که جواب بهینه را می‌دهند ما روش‌های دیگری را داریم که به اصطلاح روش‌های Heuristic یا حالا ابتکاری، روش‌های تقریبی هر اسمی که می‌خواهید بگویید در روش‌های Heuristic ما دیگر با توجه به محدودیت زمان حل، صرفاً دنبال چی هستیم یک جواب قانع کننده، یک جواب نسبتاً خوب که بتوانیم بگیم

از این که جواب نداشته باشیم این بهتر است یعنی از اینکه جواب نداشته باشیم یا جواب نسبتاً خوب داشته باشید بین بد و بدتر است خلاصه اینکه بهتر از اینکه شما، یک جواب خوب را تا حدی دارید. چرا بهینه را بدست نمی‌آرید؟ عرض کردم مسائلی که NP-hard هستند زمان حل برای مسائل با Scale بالا بسیار بسیار زیاد است و عملاً غیر قابل دستیابی است مسئله RCPSP یک نسخه اولیه دارد یک Version کلاسیک دارد یکسری نسخه‌های دیگه دارد مثلاً نسخه دیگرش را من اینجا آوردم به عنوان RCPSP حالا بگیم تعمیم یافته (generalize RCPSP) . در RCPSP ساده فرض می‌شود پیش‌نیازی از نوع Finish to start ساده است اما اینجا فرض می‌شود پیش‌نیازی ها از نوع GPR اند. از نوع time-lag است.

در حالت کلاسیک فرض می‌شود که فعالیت‌ها بدون انقطاع باید انجام بشوند اما در نسخه‌های دیگه که بهش می‌گویند PRCPSP یا Preemtive RCPSP فرض اینطوری می‌شود فعالیت‌ها می‌توانند منقطع شوند. یعنی فعالیت لزوماً ممکن است یک تیکه انجام نشود. یک مسئله دیگری که به نوعی هم خانواده RCPSP است مسئله‌ای است که در کنترل پروژه با آن آشنا شده‌اید مسئله تسطیح منبع است مسئله دیگر Resource availability cost problem (مسئله دسترس پذیری) هر کدام از اینها داستان خودش را دارد ولی خلاصه همه موضوع در خانواده چی قرار می‌گیرد RCPSP قرار می‌گیرند. ما مثلاً پارت‌های ۹ و ۸ را در دو جلسه گذشته اهداف چی را داشتیم ؟ اهداف مالی بود. یکی ماکسیمم کردن NPV بود یکی مینیمم کردن earliness-tardiness ولی بدون منبع حالا حساب کنید همین موضوع می‌تواند با RCPSP تلفیق شود یعنی مسئله RCPSP با discounted cash flue یا همان NPV یا مسئله با بحثی که در هفته قبل مدل کردیم با earliness-tardiness (مینیمم کردن جریمه‌های زودکرد – دیرکرد) و این در واقع مباحثی است که قرار است در هفته‌های باقیمانده به اصطلاح پوشش داده شود.

مسئله RCPSP از نظرف اقتصادی با آن تعریف α, β, γ ای که قبلاً داشتیم. α اش که چهار نوع α داشتیم :

m,۱/cpm/$C_{\max}$

α_1 که مربوط به ماشین بود را نداریم.

$\alpha_2 = m$: α_2 ⟵ تعداد منبع بود یعنی m نوع منبع داریم

$\alpha_3 = 1$: α_3 ⟵ یک می‌شد یعنی تجدیدپذیر است یعنی منبع نوع‌اش تجدیدپذیر است چون ما هفت نوع منبع داشتیم. تجدیدپذیر، تجدید ناپذیر، doubly constrain ، Special، Comulative حالا خلاصه نماد یک برای تجدیدپذیرها استفاده می‌شود.

α_1 و α_4 که نوشته نشده است یعنی نداریم، تهی است.

$\beta = cpm$ ⟵ یعنی پیش نیازی از نوع Finish to start است. ⟵ $\beta = 5$

γ ⟵ گاما هم که $C_{\max}$ بود یعنی مینیمم کردن زمان پروژه .

پس به صورت اصطلاح نمادین RCPSP را می توان به صورت بالا نشان داد.

جمله آخر در اسلاید : RCPSP مسئله NP-hard از نوع Strong sense است یعنی بدترین‌ها در NP-hard است چون خود NP-hardها سخت‌اند ولی داخل خودشان دوباره یک سخت و سخت‌تر دوباره دارند. RCPSP جزء مسائل Strongly NP-hard اند یعنی جزء مسائل بسیار خشن در بهینه سازی به حساب می‌آیند.

چه طوری مسئله NP-hard اثبات می‌شود در پارت ۲ یا ۳ صحبت شده است هر مسئله ۲ تا نسخه داشته DP,OP نسخه OP برای اینکه NP-hard باشد باید نسخه DP آن NP-complete باشد چه جوری اثبات می‌شود که نسخه DP یک NP-complete است باید یک مسئله‌ای را که قبلاً سختی آن اثبات شده است را ثابت کنید که reduce می‌شود به این مسئله شما، اگر مسئله A ، reduce شود به B معنی اش این است که B از A سخت‌تر است چون A ، N-hard بوده به اصطلاح NP-Complete بوده است پس B هم بدتر از آن است پس آن هم NP-complete است این جوری اثبات می‌کردیم که مسئله NP-complete است آن وقت نسخه DP اش اگه اثبات می‌شد NP-complete است نسخه OP اش ، NP-hard می‌شود حالا مسئله RCPSP اثباتش را فردی به اسم بلازویچ از روی مسئله به نام ۳-partition انجام داد. مسئله ۳-partition الان این مسئله A شما به حساب می‌آید درواقع. جزء مسائلی است که از قبل می‌دانستیم NP-hard است، ایشان ثابت کرده که این مسئله reduce می‌شود به job shop scheduling و آن Reduce می‌شود به مسئله شما یعنی RCPSP بنابراین وقتی این NP-hard بوده است در جهت فلش‌ها هم مسئله سخت‌تر می‌شود اگر این NP-hard بوده پس مسئله RCPSP هم NP-hard به اصطلاح سخت است.

حالا ۳-partition چون اسم‌اش مطرح شد ۲ دقیقه راجع به آن صحبت می‌کنیم که عملاً ربطی به بحث ندارد.

مسائل کلاً NP-hard عموماً مسائلی هستند که ظاهر ساده‌ای دارند و حلشان زمان‌بر است. ۳-partition یک همیچین مسئله‌ای است تعدادی عدد بگویید که فقط این اعداد مضربی از ۳ باید باشد یا باید بگویید ۳ تا ، یا ۶ تا ، یا ۹ تا ، یا ۱۲ تا یا ۱۵ تا مثلاً من ۱۲ تا عدد می‌نویسم که می‌تواند تکراری هم باشد این ۱۲ عدد جمع آنها ۴۴ می‌شود حالا می‌توانست هر چیز دیگری هم باشد. {۳,۴,۱,۱,۷,۵,۳,۵ ,۶ , ۱, ۵, ۳}

۳-partition مسئله‌اش این است که می‌گوید این ۱۲ عدد را به چهار گروه سه تایی تقسیم کنید

{ } { } { } { }

ولی به شرطی اینکه جمع اینها باید با هم برابر باشد. اگر جمع کل این ۱۲ عدد ۴۴ است جمع هر گروه ۳ تایی باید ۱۱ باشد. مثل این است که حالا فردی برای بچه‌هاش کادو خریده باشد چند تا ؟ ۱۲ تا . البته قیمت‌ها با هم فرق می‌کند بعضی‌هاش ۳۰۰۰ هزار تومان ، بعضی‌هاش ۴۰۰۰ تومان بعضی هاش ۱۰۰۰ تومان است والا آخر آخر. حالا می‌خواهد به صور منصفانه بین بچه‌هاش تقسیم کند به تعداد مساوی ولی به ارزش مساوی یعنی هم تعدادشان مساوی باشد هم اینکه نهایتاً ارزش مالی این کادوهایی که می‌دهد یکسان باشد. حالا می‌شد این طوری هم گفت. چطوری می‌شود این را تقسیم کرد؟ ولی حلش به نظر نمی‌یاد که شما الان بتوانید در نیم ساعت حداقل زودتر حل کنید تقسیم‌بندی اش ظاهر مسئله ساده است ولی پاسخ‌اش ممکن است به این راحتی نباشد این ۱۲ تا است حالا فرض کنید بیشتر باشد. چون مسئله NP-hard مسائل سایز کوچک را مشکلی نداریم سایز بزرگ آن مشکل است به جای ۱۲ تا

یک صفر جلوش بگذارید می‌شود ۱۲۰ تا عدد که باید به ۴۰ تا گروه سه تایی تقسیم کنید با تعداد مساوی. خوب بعید است که بشود در عرض کمتر از یک ماه حل کرد به صورت دستی. یعنی بسیار مسئله چالشی است ظاهرش ساده است ولی حلش خیلی کوتاه نخواهد بود اما اگر مسئله Scale اش کوچک بود چرا خیلی سریع در عرض چند دقیقه ، چند ثانیه می‌شد حل کرد.

مثلاًیک مثال دیگر را اینجا آوردم دقت کنید مثال ۱۲ تایی را به ۶ تایی تقلیل داده‌ایم . ۶ تا عدد که جمع‌اش ۲۲ می‌شود $S=\{۳, ۳, ۳, ۴, ۴, ۵\}$ این را به دو تا گروه سه تایی که جمع‌اش ۱۱ می‌شود تقسیم کرده‌ایم. چون مسئله scale اش کوچک است مشکل ندارد ابعاد بالا حلش مشکل است.

$S=\{۳, ۳, ۳, ۴, ۴, ۵\}$ $t = ۲$ $b = ۱۱$

$S_۱=\{۳, ۳, ۵\}$ $S_۲=\{۳, ۴, ۴\}$

RCPSP و روش‌های حل آن

به طور کلی روش‌های حل مسئله به ۳ قسمت تقسیم شده است:

- روش‌های Exact
- روش‌های Heuristic
- روش‌های Metaheuristic

اول راجع به تقسیم‌بندی روش‌های فوق صحبت کنیم.

Exact ها که تکلیفشان مشخص است روش‌هایی هستند که جواب بهینه ، جواب دقیق مسئله را می‌دهند.

روش‌های Heuristic (ابتکاری) و روش‌های Metaheuristic (فرا-ابتکاری) روش‌هایی هستند که تضمینی برای بهینگی نیست فقط تضمین شان این است که زود جواب بدهند و تا حدی هم سعی می‌کنند که جوابشان خوب باشد. این دو روش چه فرقی با هم دارند. چه فرقی بین روش Heuristic و روش Metaheuristic وجود دارد ؟ اصلاً روشی را می‌شناسید که اسمش ابتکاری باشد ؟ در کنترل پروژه روشی به اسم برگس بود که یک روش ابتکاری بود طرح‌ریزی واحدهای صنعتی ، روش کورلپ، آل‌دپ، کرفت، روش جانمایی بود معمولاً آخر طرح‌ریزی صبحت می‌شود آن روش های تضمینی نمی‌دهد قطعاً بهترین جانمایی را می‌دهد. فقط یک روش در کنترل موجودی ۲ روش سیلیور میل، LTC ،LUC روش ها به اصطلاح سفارش دهی پویا این‌ها روش‌های Heuristic اند در تحقیق عملیات روش ووگل، روش راسل در برنامه‌ریزی حمل و نقل اینها روش‌های تقریبی اند. روش‌های ابتکاری

فرا ابتکاری‌ها من چند تا اسم هاشون را گفتم شما که هوش محاسباتی داشتید که مفصل خوانده‌اید و اگر نه شاید به احتمال زیاد در پایان نامه با آن مواجه می‌شوید روش‌های مثل Genetic, algorithms simulated annealing, Taba search اندکلونی electromagnetic شاید تعدادشان به ۳۰ تا برسد.

اما سوال من چیز دیگری بود. چه فرقی بین اینها بود ؟ تشابه‌اش این است که هر دو جواب تقریبی می‌دهند ولی فرق شان در این است که Heuristic ها روش‌های مختص همان مسئله است یعنی روشی که طراحی شده فقط برای همان مسئله . به درد مسئله‌ای دیگری نمی‌خورد. شما روش برگس را کجا استفاده می‌کنید ؟ فقط در تسطیح منبع در

کنترل پروژه استفاده می‌کنید. این روش هیچ جای دیگری قابل استفاده نیست. یا مثلاًروش Craft برای چی استفاده می‌شود ؟ روش جانمایی ، روش کورلپ همان طرح‌ریزی خوب مثلاً شما در درس حمل و نقل روش Craft استفاده می‌کردید. اصلاً ربطی ندارد. آن روش طراحی شده فقط برای جانمایی است اما در روش‌های Meta heuristic روش‌های عمومی هستند مثلاً ژنتیک روشی است برای حل هر مسئله‌ای حالا بستگی به این دارد که شما چطور بتوانید این را کالیبره کنید. بتوانید به اصطلاح. من در کلاس قبلی مثالم را اینطوری زدم که فرض کنید یک زمانی پارچه‌ای را تهیه می‌کنید و می‌دهید خیاط تا متناسب با سایز شما لباسی را آماده کند که کاملاً Fit سایزتان باشد این طراحی شده دقیقاً برای استایل شما ولی یک موقع نه لباسی را آماده می‌خرید که ممکن است خیلی fit سایز شما نباشد یک مقدار باید دست کاری شود تا بتوانید متناسب خودتان آماده‌ش بکنید روش‌های Meta heuristic دقیقاً مثل آن لباس آماده است. که لزوماً ممکن است fit این مسئله نباشد باید شما سعی کنید که تنظیم اش کنید ولی heuristic نه از صفر طراحی شده برای همین مسئله است بنابراین به صورت خاص برای این مسئله فقط قابل استفاده است.

کدام یک بهتر است Heuristic یا Metaheuristic ؟ شاید به خاطر وجود کلمه Meta در اول است، روش فرا ابتکاری است. اسمش درشت‌تر است، اما در واقع اگر یکشون بهتر بود قطعاً آن یکی باید حذف می‌شد. واضح است که جوابی یکی که بهتر باشد پس آن یکی به طور کل باید delete گردد چون آن دیگه نیاز به گفتنش نمی‌باشد. چون آن یکی بهتر است موضوع این است که بستگی به آن خیاط دارد یعنی ممکن است پارچه آماده شده را علیرغم اندازه گرفتن سایز شما ، نتواند درست طراحی کند و بدوزد و برعکس یک لباس آماده ممکن است کاملاً اندازه شما باشد و برعکس یعنی کاملاً به این بستگی دارد که شما چطوری این روش را ساخته‌اید. Heuristic می‌تواند مدعی بشوید که روش شما خاص این مسئله طراحی شده است ولی بسیار فاجعه طراحی شده باشد و جوابی خوبی ازش نگرید و یا اینکه نه خیلی عالی طراحی کرده باشید. اینجا هم همین طور ممکن است روش SA و ژنتیک و غیره را بسیار بد به کار بگیرید در حل مسئله بازم جواب بدی بگیری و ممکن است خیلی خوب تنظیم کنید و جواب عالی بگیرید پس در کل بستگی به این دارد که شما چطور این مسئله را مهندسی‌اش می‌کنید. چطور این برای این مسئله به کار می‌گیرید نحوه به کارگیری است که تعیین کننده است والا کدام بهتر است، وجود ندارد. می‌تواند هر کدام بهتر یا بدتر باشد خود Heuristic ها دو دسته هستند.

دسته اول : constructive Heuristic (سازنده) اند یعنی این روش یک جواب را می‌سازد یک جواب خوب را برای مسئله می‌سازد.

دسته دوم : Improvement Heuristic (بهبود) اند یک جواب اولیه بهش داده می‌شود و روی آن کار می‌کند و آن را بهبود می‌دهد یعنی از صفر خودش جواب نمی سازد.

ایده بهتر کدام است ؟ اولی یا دومی یا هر دو . هر دو اگر باشد خیلی عالی می‌شود یعنی شما دو تا Heuristic پشت سر هم استفاده کنید با Heuristic یک جواب بسازید و آن جوابی که خودش به هر حال با یک منطق خوبی ساخته شده را حالا با یک منطق دیگر بهبود هم بدهید. این معمولاً بهترین ترکیب و حالت است اگر شما بخواهید برای پایان

نامه روشی انتخاب کنید کدام را ترجیح می‌دهید ؟ کدام راحت‌تر است منظورم بین Heurisitc و Meta heuristic است metaheauistic این راحت تر است چون می‌روید یک پایان نامه می‌خوانید دقیقاًکروموزم چه طوری تعریف کرده، عملگر تقاطع را چی کار کرده است جهش چی کار کرده است. خوب در آن مسئله اش استفاده کرده آنجا یاد می‌گیرید و می‌آیید این سمت پیاده‌اش می‌کنید. فوقش این پیاده کردن فقط کپی باشد نتایج‌اش فاجعه بار است چون آن تنظیم تعریف کروموزم یا آن عملگر که آنجا استفاده شده برای آن مسئله خوب بوده و لزوماً ممکن است برای مسئله شما کار نکند و یک جواب خوبی را ندهد. یعنی شما واقعاًباید یاد بگیرید و اینجا که می‌آیید دوباره فکر کنید که فهمیدم آنجا قضیه چی شده ولی در مسئله من باید چی کار کنم پس ممکن است اینجا در واقع نیاز به فکر دارد اگر عیناً کپی کنید نتایج آن قطعاً فایده‌ای ندارد.

ولی بین Heuristic و Meta Heuristic استفاده از اینها راحت‌تر است چون شما کلی براش reference دارید (نمونه دارید) می‌توانید بروید مقاله، پایان نامه بخوانید یاد بگیرید ولی Heuristic باید از صفر بسازی یعنی باید خلاق باشی کلاً راجع به مسئله است با خلاقیت ایده بدهی و بتوانی آن را روی کاغذ بیاوری مثال حل کنی و به هر حال به یکم کار بیشتری نیاز است ولی اگر فرد خلاق باشد قطعاً از Heuristic ها نتیجه بهتری می‌گیرد در کل اگر فرد خلاق باشد از Heauristicها خروجی بهتری می‌گیرد به طور کلی . ممکن است یک مثال بیاورید که اینطور نباشد فرد خیلی خلاقیت هم به خرج داده و آن مسئله‌اش مسئله سختی بوده و برای استفاده از Heuristic خوب نبوده ولی به طور کلی این حرف درست است اما آن چیزی که ما در این کلاس پوشش خواهیم داد کدام یک از این ۳ روش است ما اصلاً قرار نیست از این دو تا صحبت کنیم فقط باید از exact ها صحبت کنیم چون این پایین‌ها پیش‌نیاز می‌خواهد و پیش نیازش هم این است که شما باید هوش محاسباتی را پاس کرده باشید یا حداقل آن را بدانید در واقع دو دلیل وجود دارد که به اینها نمی‌پردازیم اولی همان پیش‌نیازها است و دومی زمان می‌باشد چون آنقدر زمان نداریم که وارد این مباحث شویم. باید برویم سراغ روش‌های exact چون بحث مستقلی است و نیاز به پیش زمینه خاصی نداریم می‌توانیم از صفر تا ۱۰۰ اش را در اینجا صحبت کنیم. خوب یک سوال اساسی پیش می‌آید با توجه به اینکه مسئله NP-hard است و مسائل با Scale واقعی را اصلاً نمی‌شود با exact حل کرد آن وقت توسعه exact ها چه فایده‌ای دارد ؟ روش Exact را می‌سازیم ولی بعد خودمان می‌دانیم این قرار نیست خیلی پاسخگو باشد برای ابعاد بالا. جواب مسئله کوچک را می‌دهد مثلاً این پروژه را که ۳۰ تا فعالیت دارد فقط می‌توانیم حل کنیم با exact اگر ۳۰ تا ۳۰۰ تا بشود مطلقاً نمی‌توانیم با exact حل کنیم یعنی پروژه‌ای که ۳۰۰ تا فعالیت داره قطعاً با exactحل نمی‌شود با منبع البته.

Heuristic ها و Metaheuristic ها مدعی خواهید شد که جواب‌های خوب می‌دهند این خوب را از کجا می‌خواهید بیاورید من یک روش Hearistic توسعه دادم و تابع هدف من هم make span بوده است حل کردم $C_{max} = ۹۶$ است من چه طور مدعی شود که جواب خوب است . جواب پروژه شده ۹۶ روز ولی خوب بودنش وقتی مشخص است که شما جواب exact اش را داشته باشید یعنی شما نمی‌توانید این روش Heuristic ها و MetaHeuristic ها هر چقدر هم عالی باشد نمی‌توانید ازش دفاع کنی به نظر من جواب خوبی می‌دهد قطعاً

پاسخگو نیست یعنی توجیه ندارد باید با چی مقایسه شود ؟ خلاصه روش‌های exact به چه دردی می‌خورند ؟ برای اثبات اعتبارسنجی به اصطلاح روش‌های MetaHeuristic و Heuristic چه طوری این اعتبارسنجی انجام می‌شود ؟ یکی، دو تا ، سه تا ، چهار تا ، پنج‌تا حداقل ۴۰ تا مسئله حل کنید این ۴۰ از کجا آمد ؟ لیسانس آمارمهندسی. قضینه حد مرکزی باید حداقل ۳۵ تا ۴۰ تا داده داشته باشید که فرض کنید جامعه‌ات توزیع نرمال دارد بعد آزمون فرض انجام دهیم اگر جامعه‌ات محرز است نرمال است که نمی‌خواهد با ۱۰ تا کافی است با ۵ تا هم کافی است ولی چون معمولاً محرز نیست نرمال بودن باید داده‌های زیادی داشته باشید پس حداقل ۴۰ تا مسئله داشته باشید با چی حل بشه هم دقیق‌اش حل شود و هم جواب تقریبی حالا تقریبی یا MetaHeuristic و Heuristic یا جوابی از این دو تا روش‌ها بدست می‌آید.

مسئله شماره یک را حل شد $C_{max} = ۹۶$ این exact است بهترین جواب است آن یکی چند می‌تواند باشد می‌تواند ۹۳ باشد ؟ تابع هدف مگر C_{max} مینیمم سازی است وقتی جواب دقیق‌اش ۹۶ است بهینه‌اش ۹۶ است آن که نمی‌تواند بهتر از این باشد آن یا همین است یا بدتر از این است بدتر از این یعنی بیشتر از این است این در بهترین حالت ۹۶ می‌دهد والا ممکن است ۹۹ یا بیشتر ۱۰۳ مثلاً بشود. پس جواب تقریبی از exact بهتر نیست اگر بهتر شده یه جا اشتباه حساب شده است. exact را ظاهراً اشتباه حل کرده‌اید.

مسله شماره دو را حل کردیم شد ۷۴ دقیق و جواب تقریبی‌اش شده ۷۶ است کمی بدتر است خیلی بد نیست نزدیک است.

مسئله شماره سه که حل شد جواب دقیق ۸۱ است و جواب تقریبی ممکن است همان ۸۱ را بدهد یعنی روش‌های Heuristic و MetaHeuristic بعضاً این قدر ممکن است خوب طراحی شوند که جواب دقیق را بدهند ما مدعی نیستیم که جواب دقیق را می‌دهند ولی ممکن است این اتفاق توی مثال هم بیافتد خلاصه تا کجا انجام می‌دهیم این محاسبات را

مسئله شماره ۴۰ را حل کردیم جواب دقیق ۱۳۳ است و ۱۴۹ هم جواب تقریبی است حالا درسته که من ۴۰ تا گفتم ولی عملاً ۴ تا را نوشتم.

Exact Hearistic/Metahearisitc

مساله	دقیق	تقریبی
۱	۹۶	۱۰۳
۲	۷۴	۷۶
۳	۸۱	۸۱
.	.	.
.	.	.
.	.	.
۴۰	۱۳۳	۱۴۹

$\begin{cases} H_۰ : \mu_۱ = \mu_۲ & \text{قبول} \surd \\ H_۱ : \mu_۱ \neq \mu_۲ & \text{رد} \end{cases}$

آزمون t خروجی

بفرمایید روش تقریبی روش خوبی است یا نه ؟ با توجه به این ۴ تا مقایسه‌ای که الان انجام شد اولی ۷ تا اختلاف ، دومی ۲ تا اختلاف، سومی صفر تا و آخری هم ۱۶ تا اختلاف دارد. خلاصه که به نظر من آره ، نظر من اصلاً مهم نیست ما نمی‌توانیم با نظر خودمان اعداد را مقایسه کنیم و بگوییم به نظر من این الان خوب است یا بد راه کار چیست ؟ راه کار استفاده از آمار است مقایسه میانگین جواب‌های جامعه اول یعنی جواب‌های دقیق با میانگین جواب‌های تقریبی. یک آزمونی داشتیم که احتمالاً اسمش یادتان مانده است آزمون t زوجی خروجی‌اش بعد از حساب کردن چی می‌شود. دو تا حالت بیشتر نیست یا میگه H_0 ، رد یا میگه قبول. اگر H_0 را قبول کنیم یعنی چی را قبول کرده‌ایم در واقع $\mu_1 = \mu_2$ را پذیرفته‌ایم یعنی چی ؟ روش تقریبی شما تا حدی مساوی همان دقیق به شما جواب داده است. الان دیگه به نظر من نیست الان به نظر آمار است طبق محاسبات آماری نتایج آماری نشان می‌دهد که روش ما دارد خوب کار می‌کند از کجا این را آوردیم جواب‌های این روش را با جواب‌هایی که از روش قبلی داشتیم با ۴۰ تا مسئله به صورت آماری تست کردیم، اما اگر این H_0رد شد یعنی کار شما، پایان نامه شما ، مقاله شما قابل دفاع نیست . یعنی خوب خیاطی‌اش نکردید. حالا ممکن است آن روش روش خوب بوده باشد ولی شما خوب استفاده نکردید خلاصه‌اش برای این مسئله شما جواب‌هایی که گرفتید قابل استفاده نیست خوب حالا اگر قرار باشد پروژه‌ای واقعی داشته باشید با ۳۰۰ تا فعالیت این روش که تأیید نشده را به کار بگیرید خوب جوابی که می‌دهد قطعاً جواب مناسبی نخواهد بود. پس مبنای مقایسه بر مبنای اعتبارسنجی روش‌ها تکنیک آمار است.

پس قرار شد به سراغ روش های دقیق برویم که به دو دسته تقسیم‌بندی می‌شوند: Exact Prodedures

- Linear Programming (برنامه ریزی خطی)
- Branch & Bound (شاخه و کران)

روش Linear programming چی کار باید بکنیم مسأله را به صورت ریاضی مدل کنید خطی البته بعد با Lingo ، گمز یا Simplex و Lindo حل کنید اینها نرم افزارهای OR هستند که می‌توانند مدل OR را به صورت بهینه حل کنند پس این راه اولش مدل ریاضی بنویسید و با نرم‌افزارهای OR حل کنید فقط مدلی که می‌نویسید حتماً باید خطی باشد چرا خطی ؟ کسی هست که با Lingo یا گمز کار کرده باشد که جواب بده چرا خطی باشد حالا من مدل نوشتم درست هم است فقط خطی نیست غیرخطی است چی می‌شود که اگر با Lingo یا گمز حلش کنم جواب بهینه Local می‌دهد. تضمین نمی‌کند که بهینه است می‌گه این جواب بهینه Local مسئله است اما اگر مدل شما خطی باشد می‌گوید جواب بهینه global (جواب بهینه مطلق) مسئله است دیگر نسبی نیست.

چون ما اسم رو را چی گذاشتیم بالاش با آبی نوشتم exact ، پس جواب دقیق را می‌خواهم جواب مطلق را می‌خواهم جواب نسبی را نمی‌خواهم چون اسم روش را نوشتید روش exact پس باید مدل خطی بنویسید که محرز شود که جوابی که می‌گیری جواب بهینه است نه جواب تقریبی اگر مدل غیرخطی باشد بنویسی آن وقت جواب دوباره مثل اینها می‌دهد جواب تقریبی می‌دهد. پس خطی بودن به خاطر این است.

روش دیگر که تا حدی با اسمش آشنا هستید روش شاخه و کران است.

برویم سراغ روش exact اول یعنی برنامه ریزی ریاضی مسئله و حلش با نرم‌افزار . برنامه‌ریزی یعنی متغیرهایش ، تابع هدف و محدودیت‌ها را بنویسید یک مدل روی اسلاید الان هست فقط چند خط اولش کپی است کاملاً به جزء محدودیت آخری بقیه‌اش کپی از هفته‌های گذشته است. مینیمم کنید f_n همان $C_{\max}$ (finish time فعالیت n ام ، فعالیت n ام آخرین فعالیت یعنی پروژه) شما فعالیت n ام را مینیمم کنید یعنی پروژه را مینیمم کرده‌اید.

$\min f_n$

این را قبلاً نوشته‌اید این هم معرف حضور شما هست محدودیت همیشگی پیش‌نیازی

$s.t.$

$f_i \le f_j - d_j \quad \text{for all} \quad (i,j) \in A$

$f_1 = 0$ ۰ = f_1 هم پروژه در زمان صفر شروع شود

اما محدودیتی که تا الان نداشتیم و برای اولین بار می‌بینیمش

$\sum_{i \in S_t} r_{ik} \le a_k \quad \text{for} \quad k = 1,...,m \quad \text{and} \quad t = 1,...,f_n$

این محدودیت است اینو باید معرفی کنم چی هست

این عبارت سمت راست

a_K ⟵ میزان موجود از منبع تجدید پذیر K ام

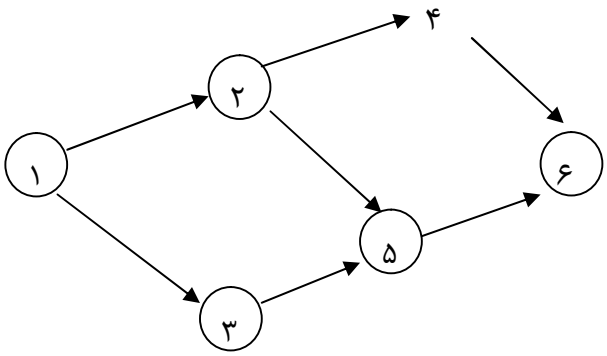
r_{iK} ⟵ میزان نیاز فعالیت i به منبع نوع K

S_t ⟵ مجموعه فعالیت‌های در حال انجام در زمان t

این a_Kکه تا الان نوشته نمی‌شد چند بوده است فرض بر این بود که بی‌نهایت است ولی الان بی‌نهایت نیست a_K تا ازش داریم منبع شماره K، a_K تا ازش داریم بی نهایت نیست چون تا الان بی‌نهایت بود وقتی بی‌نهایت باشد این بی اثر است می‌توانید پاکش کنید اصلاً مهم نیست که شما اگر بی نهایت منبع دارید پس خیالت راحت است موقع برنامه‌ریزی نگران آن نیستی ولی الان دیگر اینطوری نیست.

خوب آن وقت $\sum r_{iK} \le a_K$ است چرا باید اینطوری باشد ؟ می‌گه در مجموع کل استفاده شما از این منبع باید کمتر از آنی باشد که دارید مثلاًدر این دانشکده این کلاس‌ها به عنوان منبع تجدیدپذیر است مثلاً کلاً ۴۵ تا کلاس داریم به فرض می‌گم نمی‌دانم چند تا کلاس داره استفاده می‌شود مجموع استفاده شما از این کلاس حالا هر درسی که می‌خواهد i می‌شود اسم درس‌ها ، فعالیت ها اینجا درس‌ها هستند. زمانبندی پروژه خودش یک فعالیت است آنجا و هر جا یه درسی داره تشکیل می شود. مجموع استفاده کلاس‌ها باید کمتر از ۴۵ تا باشد این تعریف این است فقط کدام درس‌ها یا کدام فعالیت‌ها باید جمع زده شود. امروز ۵ ام آذر، کدام فعالیت‌ها باید جمع شود آنهایی که امروز در حال، مثلاً یک درسی داریم درس تئوری صف ولی امروز نیست اصلاً درسش چهارشنبه است چه ربطی به امروز دارد من دارم راجع به محدودیت امروز صحبت می‌کنم محدودیت درس صف را باید روز چهارشنبه بیارم فعالیتی که در آن روز در حال انجام نیست توی این Sumation بازی داده نمی‌شود پس اگر دقت کنید زیرSumation چی نوشته شده است ؟ $i \epsilon S_t$ است. S_t بیانگر مجموعه فعالیت‌هایی که در زمان t در حال انجام اند. پس S_t مجموعه فعالیت‌های در حال انجام در زمان t است. این می‌شود مدل مسئله‌ای که فقط خط آخرش البته فرق داشت. فقط خط آخر چند بار باید نوشته شود ؟ این محدودیت چند بار باید نوشته شود ؟ جلوش گفته است برای هر منبعی و هر روزی مثلاً برای کلاس در هر روزی برای حالا منابع مختلف در هر روزی باید نوشته شود پس یک خط نوشتیم ولی یک محدودیت

نیست یک مجموعه‌ای از محدودیت‌ها است که خودش به ازای هر منبعی که m تا منبع داریم و به ازای هر روزی باید نوشته شود.



مثال : اگر یک شبکه‌ای مثل این وجود داشته باشد که همه اطلاعاتش را داشته باشید پیش نیازی‌هاش و فعالیت ۱ و ۶ مجازی هستند و فقط ۴ تا فعالیت واقعی وجود دارد، duration ها ، منابع ، i_K و r_{ik} همه اینها در این مسئله باشد و خواسته شود که مدل ریاضی‌اش را با Lingo حل کنید با فرض اینکه همه برنامه Lingo یا گمز را می‌دانند اما کسی مطلقاً نیست که این مسئله را حل کرده باشد نه اینکه نتواسته‌اید بلکه این مدل قابل حل کردن با گمز یا Lingo نمی‌باشد به این مدل مدل ، فرمول بندی Consputional (مفهومی) می‌گویند. مدل مفهومی مدلی است که درست است مفهومش ولی قابل استفاده نیست یعنی قابل پیاده‌سازی نیست. چرا قابل پیاده‌سازی نیست. یعنی کدام موضوع باعث می‌شود که این مدل قابل نوشتن نباشد ؟ این زیگما را روی چی باید ببندی ؟ روی S_t ، S_t مجموعه فعالیت‌ها که در روز t ام یعنی امروز ۵ ام در حال انجام‌اند این پروژه کدام فعالیت‌هایش روز ۵ ام در حال انجام است. اینو ندارم نمی‌دانم مسئله را که هنوز حل نکردم ما هنوز زمانبندی نکردیم که کدام‌ها در حال انجام است پس چون S_t معلوم نیست این مدل قابل نوشتن نیست.

مفهوم (گفتار) درست است. ولی موقعی که می‌خواهی روی کاغذ بیاوری نمی‌توانی آن را بنویسی الان این زیگما را باز کنید و برای این بنویسید. واضح است که نمی‌توانید بنویسید چون نمی‌دانید S_t کدام‌ها هستند. الان S_t ها کدام‌ها روز ۵ ام در حال انجام هستند ۲، ۳ ، ۲ با ۴،۳، ۴ با ۳ ،۵ ، ۵ با ۴ نمی‌دانید این‌ها را ممکن است هر کدام از این‌ها اتفاق بیافتد من نمی‌دانم که کدام در حال انجام است آن روز. چون پروژه هنوز زمانبندی نشده است چون ما تازه می‌خواهیم زمانبندی اش کنیم پس این مدل را مدل مفهومی می‌گویند.

اما مدل‌هایی که قابل نوشتن باشد را چند تا را معرفی می‌کنیم تعدادشان زیاد است که سه نمونه آن را در اسلاید موجود است پس مدل‌هایی که الان می‌خواهم بگم دیگر مدل مفهومی نیست مدل‌های ریاضی‌اند که دقیقاً قابل پیاده‌سازی خواهند بود. می‌توانید برای یک مثال بنویسید و هفته بعد حل کنید و بیاورید. اینها دیگر قابل حل هستند.

مدل اول که حالا فردی به نام پریستکر (Pritsker) اولین بار ارائه داده است چون اولین بار بوده است لازم بود که توضیح آن حتماً گفته شود.

در این مدل بازم متغییر، تابع هدف و محدودیت باید بنویسیم. متغیرها را چی تعریف کرده است ؟ در اسلاید قبلی چی متغیر بوده است ؟ f_i ها متغیر بودند. اما در این مدل جدید f_i ندارد به جای آن گفته شده x_{it} متغیر است.

x_{it} ⟵ متغیر صفر یا یک است اگر فعالیت i در زمان t تمام بشود یک و در غیر اینصورت صفر است. پس متغیری است که اگر فعالیت i در زمان t تمام بشود می‌شود یک والا می‌شود صفر.

فرض کنید یک فعالیتی duration آن ۳ روز است duration جزء داده‌های مسئله است یک فعالیت ۴ روز طول می‌کشد با محاسبات رفت و برگشت حساب کردیم زودترین زمان تمام شدن این فعالیت ۳ ام آذر است و دیرترین زمان تمام شدن همین فعالیت ۹ ام آذر است پس این فعالیت کی تمام می شود ؟ در این بازه تمام می‌شود ولی کدام یک نمی‌دانم. کدامش متغیر است فعالیت i یا روز ۳ ام ، یا ۴ ام ، یا ۵ ام ، یا ۶ ام ، یا ۷ ام ، یا ۸ ام ، یا ۹ ام تمام می‌شود. ولی کدام یک از حالات بالا اتفاق می افتد هنوز معلوم نیست یعنی برای این فعالیت فقط به تنهایی باید چند تا متغیر تعریف کنید به تعداد ۷ تا، از زودترین زمانی که می‌تواند تمام شود تا دیرترین زمانش ولی کدام یک از اینها اتفاق می‌افتد باید مسئله حل شود و مدل تصمیمم بگیرد که بهینه‌اش به نفع مدل (تابع هدف) کدام است که اتفاق بیافتد.
پس ۷ تا متغیر باید تعریف کنیم برای یک فعالیت بین EFT_i و LFT_i که در این دو مورد به صورت فرض نوشته شده است. این مثال است حالا EFT یا LFT هر عددی که عوض شد و اینها را دوباره تغییر می‌دهیم. اگر احیاناً این $x_{i5} = ۱$ یک بشود یعنی چه ؟ یعنی فعالیت i کی تمام شده است ؟ ۵ ام . اگر ۵ ام تمام شده است duration ، ۳ روز است اگر x_{i5} بشود یک یعنی فعالیت i روز ۵ ام تمام شده است یعنی کی شروع شده است duration ، ۳ روز است یعنی ۳ روز قبل، یعنی ۲ ام شروع شده است و ۵ ام تمام شده است ولی خوب این را شانسی گفتم نمی‌دانم کدام یک می‌شود یا نه.

$EFT_i = ۳$ $d_i = ۳$

i

$LFT_i = ۹$

f_i ⟵ $x_{i۳}$ $x_{i٤}$ $x_{i٥}$ $x_{i٦}$ $x_{i۷}$ $x_{i۸}$ $x_{i۹}$ (۱ روی $x_{i٥}$)

ببینید بین f_i (finish time i)، و این x هایی که الان تعریف کردیم به سادگی می‌توان این رابطه را پذیرفت این f همان finish time است. در این مدل آقای پریستکر می‌گوید که شما x_{it} را اینطوری تعریف کنید می‌توانید از روی x ها می‌توانید f را به همین راحتی حساب کنید من این را باز می‌کنم می‌نویسم اینجا f_i در این مثال $\sum t.x_{it}$ می‌شود حال شما عدد بگویید.

$$f_i = ۳x_{i۳} + ٤x_{i٤} + ٥x_{i٥} + ٦x_{i٦} + ۷x_{i۷} + ۸x_{i۸} + ۹x_{i۹} = ٥ = f_i$$

(۰ ۰ ۱ ۰ ۰ ۰ ۰)

حالا اگر احیاناً $x_{i۵}$ بشود یک چه اتفاقی می‌افتد ؟ بقیه صفر می شود $x_{i۰}$ اگر بشود یک میشود ۱ × ۵ در نتیجه $f_i = ۵$ می‌شود finish time می‌شود ۵ یعنی از روی x_{it} ها می‌شود به راحتی طبق این رابطه f_i را حساب کرد ما یادمان نره مجهول اصلی ما f_i هستند اینجا x_{it} نوشته خوب اشکال نداره می‌گه شما x_{it} را حساب کردید نگران نباش می‌توانی از روی x_{it}ها ، f_i ها را طبق این رابطه محاسبه کنید. حالا برگردیم به تابع هدف ، تابع هدف در مدل قبلی چی بود؟ $\min f_n$

حالا الان چی باید بگم، مینیمم بازم همان f_n ولی f نگید به جای f باید برحسب x_{it} صحبت کنید برحسب x_{it} فرمول داریم به جای f_n باید بگویید زیگمای این عبارت فوقش به جای n، i است ($f_i \longleftarrow f_n$) تابع هدف $\sum_{t=EFT_i}^{LFT_i} t.x_{it}$ است این همان تابع هدف شما است این زیگما همان f_n است فقط f_n ننوشتم برحسب x_{it} باید بنویسیم .

این محدودیت چه چیزی می‌گوید.

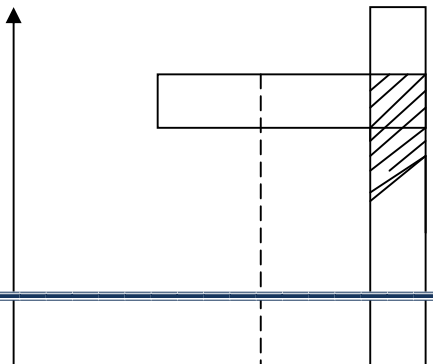
$\sum x_{it} = ۱$ می‌شود چرا باید یک بشود اینجا شما گفتید بین این‌ها یکیش باید یک شود چون منطق است فعالیت باید یکجا تمام شود نمی‌شود که در دو جا تمام شود اگر بخواهد دو تا یک شود به طور مثال بی معنی است تضاد است . شما اینجا می‌گویید فعالیت i ، ۵ ام تمام شود دوباره اینجا می‌گویید فعالیت i ، ۸ ام تمام شود ولی این داره مجبور می‌کند که فقط یکی‌اش یک شود یعنی یک جا باید تمام شود این نیاز به گفتن ندارد قابل حدس است دو تا اسلاید عقب‌تر $f_i \le f_j - d_j$ ، پیش‌نیازی . فقط به جای f_i و f_j رابطه‌ای نوشتیم برحسب x نوشتیم

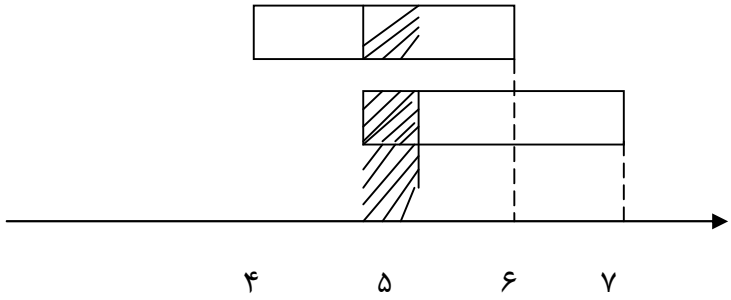
$$\sum_{t=EFT_i}^{LFT_i} t.x_{it} \le \sum_{t=EFT_j}^{LFT_j} t.x_{jt} - d_j \quad \text{for all } (i,j) \in A$$

پس این از این. اما اصل قضیه این است که S_t را چطور از شرش باید خلاص کنیم چون S_t نگران کننده بود دیگه مشکل داشتید و ما نتوستیم مدل قبلی را بنویسیم حالا اینجا را بنویسیم این محدودیت چه جوری این منبع را برقرار می‌کند بدون نوشتن S_t

$$\sum_{i=1}^{n} \sum_{q=\max\{t,EFT_i\}}^{\min\{t+d_i-1,LFT_i\}} r_{ik}.x_{iq} \le a_k \quad \text{for } k=1,...,m \text{ and } t=1,...,T$$

توی این محدودیت که شما دفعه قبل یک زیگما روی S_t نوشته بودید x وجود نداشت الان S_t را پاک کرده است S_t اینجا نداریم. r را در x ضرب کرده است اگر فعالیت i در روز q (q که خودش یک محدوده دارد) در حال انجام نباشد چی می‌شود فعالیت i که امروز ۵ ام است در حال انجام نیست اگر در حال انجام نیست این می شود صفر، این r در صفر ضرب می‌شود خوب هیچی دیگه به حساب نمی‌آید اما اگر در حال انجام باشد این می‌شود یک . این r در یک ضرب می‌شود به حساب می‌آید. حالا این را از کجا می‌فهمد در حال انجام است یا نه ؟ چه طور مقدار می‌گیره که در حال انجام است یا نیست.





وزن ۵ یا ۶ و ۷

Min (۵+۳-۱,۹) ←۷

$\sum$

Max (۵,۳) ۵

این موضوع را با یک مثال من می‌گم که دیگه برای هر مثال دیگه شما همین طوری تفسیرش کنید. من روز ۵ ام را اینجا جدا کنم پس این روز ۵ ام پروژه است من می‌گم ۵ ام آذر. فعالیت i که خودش ۳ روز است تحت چه شرایطی امروز در حال انجام خواهد بود. ضمن اینکه این فعالیت سه روز طول می‌کشد یکی اش این است که این فعالیت امروز پایانش باشد. فعالیتی که امروز آخرین روز انجام اش است پس در حال انجام بوده است یعنی روز ۵ ام تمام شود دیگه چی ؟ یه حالت هم این است که فردا تمام شود فعالیت ای که ۳ روز است اگر ۶ ام تمام بشود یعنی کی شروع شده است. ۴ ام دیگه ، ۴ ام، ۵ ام پس ۵ ام هم توش هست. ۴ ام ، ۵ ام ، ۶ ام. یعنی فعالیت باید از ۴ ام شروع شود که ۶ ام تمام شود خلاصه ۵ ام آن مهم است که در حال انجام است. دیگه چه حالتی ؟ ممکن است امروز اصلاً شروع‌اش باشد و کی تمام شود ۷ ام ، پس پایان ۵ ام، پایان ۶ ام و پایان ۷ ام. این ۳ اتفاق باعث می‌شود این که فعالیت امروز در حال انجام باشد. از چندم شد ؟ شروع را نگویید از پایان بگویید. چرا پایان را باید بگوییم؟ چرا شروع را نمی‌گویم از پایان می‌گویم ؟ چون متغیر خود را پایان تعریف کرده‌اید اگر شروع تعریف کرده بودید هم درست بود آن وقت به طور کل با شروع بود اما چون متغیر پایان تعریف کرده‌اید یعنی finishیک فعالیت را متغیر گرفته‌اید فقط باید با پایانش بحث کنید. پس یک فعالیت کی روز ۵ ام در حال انجام است.

سه حالت : یا روز ۵ ام ، یا روز ۶ ام ، یا روز ۷ ام پایان یابد پس این زیگما از چند تا چند باید x ها را بشمارد ؟ همین الان گفتم: ۵، ۶، ۷ فوقش این فرمول چطور کار می‌کند؟ این شروع زیگما را از چند می‌گیرد. $\max\{t, EFT\}$. که t در این مثال امروز است ۵ ام است. EFT این بالا نوشته شده است ۳ است. بین ۵ ام و ۳ ام ماکسیمم کدام است ؟ ۵ ام پس این شروع ۵ ام . حالا بالا. ۷=۱-۸=۳+۵ (d ۳) آن وقت بین ۷ ام و LFT که ۹ ام است بین ۷ و ۹ مینیمم ۷ ام می شود. پس چی شد ؟ ۵ تا ۷ یعنی همانطوری که تصویری پیش می‌رویم این فرمول داره اینطوری می‌شمارد میگه بین t و EFT از ماکسیمم تا بین t+d-۱ و LFT مینیمم . در این بازه x ها را جمع بزن

چرا باید جمع بزنیم چون توی این بازه، توی این x ها فعالیت شما در حال انجام است چون در حال انجام است پس این منبع اش باید به حساب بیاد یعنی شمرده شود این یک بشه که شمرده شود اگر نه پس در حال انجام نیست. پس در حال انجام نیست شما نباید سهمش را بشمارید. البته این برای یک فعالیت است یک زیگما دیگه هم می‌خواهد که

چی‌ها را بشماری این پشت ؟ همه فعالیت‌ها را جمع بزنیم و من فقط فعالیت این کلاس را گفتم کلاس‌های دیگه چی ؟ فعالیت‌های دیگر چی ؟ باید همه را جمع بزنید و جمع اینها باید از موجودی، از سقف منبع کمتر باشد. S_t چی شد؟ دیگه با این تعریف x ، ما به S_t را نیاز نداریم این دیگه الان قابل نوشتن است. شما الان در واقع اگر این نرم‌افزار را بدانید می توانید این را بنویسید و حلش کنید. این مدل تمام شد.

چند تا متغیر دارد ؟ این مدل چند تا متغیر تصمیم دارد ؟ یک گروه x_{it} . ولی x_{it} یکی نیست. i,t اندیس هستند. مثلاً شما فقط برای فعالیت i چند تا نوشتید ۷ تا فقط برای فعالیت i نوشتیم. تعداد اگر بخواهید بگید باید اندیس‌ها را دقت کنید i چند تا است ؟ i تا n است، t هم تا dead line که من T نوشتم اسمش را که می‌شود nT تا متغیر دارد. البته حداکثر نوشته. چرا حداکثر ؟ چون شما از لحظه صفر تا آخر پروژه که متغیر تعریف نمی‌کنید فقط بین EF,LF اش تعریف می‌کنید. کل بازه پروژه را متغیر تعریف نمی‌کنید فقط بین EF,LF بنابراین حداکثرش nT است چند تا محدودیت دارد. نوشتم البته فقط بگویید این اعداد از کجا آمده است ؟ این از کجا آمد ؟ دقیقاًاین تعداد محدودیت دارد. این اولیش چند تا است ؟ n را از کجا آوردید ؟ جلوش گفته برای ۱ تا n اگر بخواهید ببینید یک محدودیت چند تا است، جلوش را بخوانید و ببینید اندیس آن تا چند رفته است. محدودیت i تا n . پس n این است این چند تا است. این به ازاء هر بردار است بنابراین به تعداد بردارها، بردارهای پیش‌نیازی شبکه و این چند تا است؟ این برای هر منبعی که د تا است و برای هر روز ، T تا است. مثلاً اگر شما ۱۰ نوع منبع دارید و پروژه ۹۰ روز است باید ۹۰۰ تا بنویسید (۹۰×۱۰) پس این تعداد به طور خلاصه محدودیت داریم. این مدل پرستیگر به عنوان اولین مدل در RCPSP بود.

مدل دوم :

مدل دوم را فردی به اسم کاپلان (Kaplan) در سال ۱۹۸۸ معرفی کرد. قبل از شروع، چرا باید شخص دیگری برای مسئله‌ای که یکبار مدل شده است دوباره مدل ارائه دهد ؟ خوب چه کاری است که مسئله مدل شده و تمام شده است را دوباره مدل کند ؟ دوباره مدل نوشتن یعنی چی ؟ بهبود یعنی چی ؟ خوب آن مدل قبلی دقیق (exact) بود ولی بهبودش به این معنی نیست که جوابش را بهتر کند به معنی این است که با محدودیت‌های کمتر و با متغیرهای کمتری مسئله را بنویسید که مسئله راحت‌تر حل شود. وقتی که در نوشتن مدل‌های بیشتر رقابت می‌کنیم. هدف ما این است که مسئله را کوچکتر بنویسیم. شاخص‌های یک مسئله (مدل خوب) چه چیزهایی هست؟ دوتاش را گفتم : محدودیت کم و متغیر کم. خطی هم که حتماً باید باشد. ماتریس ضرایب محدودیت‌ها خلوت باشد تونوک باشد یعنی چی ؟ مثلاً این یک محدودیت است دیگه

$$۳x_{i۳} + ۴x_{i۴} + ۵x_{i۵} + ۶x_{i۶} + ۷x_{i۷} + ۸x_{i۸} + ۹x_{i۹} \leq ۱۵$$

$$\begin{cases} x_{i۳} + x_{i۶} \leq ۳ \\ x_{i۴} + x_{i۹} \geq ۳ \end{cases}$$

موقع شمردن شما چند تا محدودیت را می‌شمارید. یکی . اینها را چند تا می‌شمارید ؟ تعدادی این دو تا است آن یکی است. کم است به ظاهر آن خوب است چون کم است . کم است ولی شلوغ است اگر دقت کنید ضرایب ۳ و ۴، ضرایب زیادی دارد درست است که یکی است ولی ماتریس‌اش ضرایب زیادی دارد این درست است که دو تا است ولی مثلاً شما اولی را دقت کنید از ۷ تا متغیر چند تا شون اینجا هستند ؟ دو تا پس پنج تای دیگر کجا است ؟ ۵ تا دیگه

ضریبشان صفر است خوب صفر حالا چه حسنی دارد ؟ حالا صفر باشد یا نباشد بحث محاسبات است دیگر . صفر یک عامل خوب در محاسبات است اگر جمع بشود که خنثی است که تأثیر ندارد و اگر ضرب بشود که عالی است آن را صفر می‌کند و محاسبات را کم می‌کند. این دیگه بحث محاسباتی است صفر هر چقدر در جمع و ضرب صفر درگیر باشد بهتر است می‌گویند ماتریس تونوک‌تر است پس تعدادی بیشتر است ولی از نظر تونوک بودن این بهتر است آن شلوغ است. یه اصطلاح دیگه هم می‌گن که چگالی‌اش پایین‌تر است چگالی این پایین آن چگالی‌اش زیاد است چگالی یعنی همان شلوغ بودن ضرایب است.

بین این دو مدل کدام بهتر است ؟ محدودیت کم، متغیر کم ، ماتریس ضرایب با چگالی کم یا تونوک‌تر هم می‌توانستم بگویم. در مدل کاپلان متغیر را چی تعریف کرده است ؟

در قبلی این بود که اگر فعالیت i در روز t تمام شود یک و در غیر اینصورت صفر است.

$$x_{it} = \begin{cases} 1 & \text{if activity } i \text{ finishes at time instant } t \\ 0 & \text{otherwise} \end{cases}$$

اما در این مدل گفته شده است یک فعالیت در زمان t در حال انجام باشد یک پس تمام شدن آن متغیر نیست و در غیر اینصورت صفر است.

$$x_{it} = \begin{cases} 1 & \text{if activity } i \text{ is in progress in period } t \\ 0 & \text{otherwise} \end{cases}$$

از همین اول یک مشکل کوچک پیش می‌آید : تکلیف فعالیت‌های مجازی چی می‌شود ؟ چون مجازی ها مشکلشان این بود که اصلاً duration آنها صفر است یعنی هیچ وقت در حال انجام نیست آن وقت با این تعریف تضاد پیش می‌آید. آنها نمی‌توانند در این تعریف باشند. شما الان برای فعالیت یک مجازی است و duration آن صفر است اصلاً نمی‌توانید x تعریف کنید چون هیچ وقت در حال انجام نیست. کاپلان این مشکل را با یک ترفند حل کرده است. گفته فعالیت‌هایی که مجازی هستند می‌دانیم که duration آنها صفر است ولی فرض کنید که duration آن یک است که این مشکل در حال حاضر حل شود و در آخر این یک را دوباره بردار و پروژه را یک روز عقب بیار که درست می‌شود. پس با یک ترفند ساده حل کرد. پس در نتیجه آنهایی که مجازی هستند فعالیت اول و آخر که مجازی هستند علیرغم اینکه می‌دانید duration صفر است می‌گه فعلاً فرض کنید duration یک است که با تعریف بالا x تضاد نداشته باشد حل که شد دوباره برگردان سرجاش . برگردان سرجاش فرقش این است که پروژه شما چی می شود ؟ یک روز عقب‌تر می‌آید. چرا یک روز عقب‌تر ؟ چون فعالیت یک را یک روز پر کردی دیگه درحالی که اصلاً نیست. این روز برمی‌داری خالی می‌شود یک روز می‌آوریم عقب‌تر پس این نگرانی ندارد با این تعریف x جدید تابع هدف و محدودیت‌ها را نوشته البته ما به اندازه قبلی وارد detail اش نمی‌شویم من بیشتر تفاوت‌ها می‌گم.

محدودیت اول را دقت کنید .

$$\sum_{t=EST_i+1}^{LFT_i} x_{it} = d_i$$

x_{it} را چی نوشته است ؟ $\sum_{t=EST_i+1}^{LFT_i} x_{it} = d_i$ دفعه قبل نوشته شده بود $x_{it} = 1$ اما الان نوشته شده مساوی d_i است. چرا باید مساوی d_i باشد؟ در قبلی صحبت از تمام شدن بود. خوب باید یکبار تمام بشه ولی در الان صبحت از در حال انجام شدن است این فعالیت باید چند روز در حال انجام باشد ؟ ۳ روز . چرا ؟ چون duration آن ۳ روز است پس این تفاوت اول که محدودیتش جمع x_{it} ها باید مساوی d_i باشد نه یک. دیرکرد نداریم . C_{max} هدف است. earliness-tardiness ندارد. این version ساده RCPSP است که هدف اش C_{max} است.

مثلاً در این مثال از این ۷ تا x باید چند تاش یک باشد ؟ ۳ تا . کدام ها از این ۳ تا ؟

$x_{i۳} = ۰$ $x_{i۴} = ۱$ $x_{i۵} = ۱$
$x_{i۶} = ۰$ $x_{i۷} = ۱$ $x_{i۸} = ۰$
$x_{i۹} = ۰$

این یک‌ها درست است که باید ۳ تا باشند ولی باید پشت سر هم باشند وگرنه فعالیت بریده میشود. محدودیت زیر برای این است از بریدگی فعالیت جلوگیری می‌کند. این محدودیت تضمین می کند که فعالیت پیوسته یک باشد نه اینکه بریده بریده باشد. من مثال بالا را بریده نوشتم ولی این محدودیت جلوی این بریدگی را می‌گیرد.

$$d_i .(x_{it} - x_{i,t+1}) - \sum_{q=\max\{EST_i+1,\,t-d_i+1\}}^{t-1} x_{iq} \le 1 \quad \text{for } i=1,...,n \text{ and } t = EST_i+1,...,LFT_i-1$$

این دو تا محدودیت زیر را با هم بنویسید هر دو با هم پیش‌نیازی. وقتی تعریف متغیر را عوض کرده، پیش‌نیازی را مجبور شده در ۲ خط بنویسید یعنی ۲ تا محدودیت با هم پیش‌نیازی را کنترل می‌کند پس اینطوری نیست که ما به ازاء هر موضوع یک محدودیت داشته باشیم بعضاً ممکن است به خاطر نوع تعریف متغیرتان مثل این مورد مجبور باشید در ۲ تا ۳ خط بنویسید.

$$x_{jt} \le x_{i,t-d_i} \qquad \text{forall}(i,j)\in A \text{ and } t = EST_j+1,...,\min(LFT_i, EST_j+d_i)$$

$$x_{jt} \le \sum_{q=EST_i+d_i}^{t-d_i} x_{iq} \qquad \text{forall}(i,j)\in A \text{ and } t = EST_j+d_i+1,...,LFT_i$$

پیش نیازی

آخری هم که قابل حدس است چی را تعریف می‌کند. از وجود ik معلوم است منبع را. اگر در حال انجام باشد می شود یک و r به حساب می‌آید و در غیر این صورت صفر است و r به حساب نمی‌آید. (منبع)

$$\sum_{\substack{i=1 \\ t\in\{EST_i+1,LFT_i\}}}^{n} r_{ik}x_{it} \le a_k \qquad \text{for } k=1,...,m \text{ and } t=1,...,T$$

$x_{it} \in \{0,1\}$ for $i=1,...,n$ and $t = EST_i+1,...,LFT_i$

این مدل بهتر است یا قبلی. آن سه تا شاخص که یادتان نرفته است. آن شاخص‌ها چی بودند ؟ متغیر کم، محدودیت کم ، ماتریس ضرایب با چگالی کم.

اول متغیرها را بگویید. متغیرهای مدل کابلان کمتر از قبلی است یا خیر ؟ چرا بیشتر ؟ ایشون یه چیزی را می‌گه، گفته‌اش درست است . می‌گه اینجا x هایی بیشتری یک می‌شوند . آره درست می‌گه اگر حل کنید اینجا x های بیشتری یک می‌شوند در مدل قبلی چند تا x یک می‌شود برای هر فعالیتی یکی x یک می‌شود ولی الان برای هر فعالیت به اندازه duration یک می‌شود. ولی سوال من این نبود که کدام بیشتر یک می‌شوند؟ گفتم چند تا قبل از اینکه حل کنید بگویید. قبل از اینکه حل کنید متغیرها کمترند یا بیشتر ؟ در قبلی برای هر فعالیتی در چه بازه‌ای x تعریف کردید ؟ (۳ تا ۹ در این مثال) بین EFT , LFT چرا بین EFT , LFT ؟ چون finish time بود. زودترین زمان پایان تا دیرترین زمان پایان. ولی الان چی ؟ الان متغیر شما پایان که نیست در حال انجام بودن است. پس باید بین چی تعریف کنید؟ بین EST تا LFT تعریف کنید. اگر در این زیگما دقت کنید. زیگما را از چند بسته است از EST شروع کرده است یعنی ما اینجا در این مثال که شما می‌بینید که ما ۷ تا متغیر تعریف کردیم این برای مدل قبلی بود که ۷ تا بود ولی الان اگر بخواهید بنویسید باید ۱۰ تا تعریف کنید به خاطر اینکه این مسئله EST اش مساوی صفر است و از روز EST+۱ ، یعنی ۱، ۲، ، ۳ تا ۱۰ باید بنویسید متغیر بیشتری می‌خواهد پس متغیرهایش بیشتر است .

$$EST_i = ۰ \quad , \quad LFT_i = ۹$$

محدودیت‌هایش چی ؟ البته جلوش هست. این چند تا است. این n ، T تا است. nT و $n + nT + |A|T + mT$ اینم که ظاهراً بیشتر است یعنی هم متغیرهاش بیشتر است و هم محدودیت‌هاش. چه چیزی باقی ماند. چگالی‌اش هم که نمی‌تونیم با بحث‌های کلاسیک که ما الان بلدیم نمی‌توانیم اندازه‌گیری کنیم.

سوال اگر تمرین به شما بدهم که برای هفته بعد انجام بدهید و با فرض اینکه Lingo یا گمز هم می‌دانید اثبات کنید که بین این دو مدل کدام بهتر است ببینم چی کار می‌کنید بگید که چی کار می‌کنید برای هفته بعد. اگر قرار شد این را انجام بدهید مسیر حل را بگید (در جمله) دو تا مدل داریم: مدل کاپلان و مدل پریستیگر کدام مدل بهتر است ؟ جوابتان را با یک عدد شروع کنید. ۴۰ تا مسئله را باید با مدل پریستیگر حل کنید که همان ۴۰ تا با این مدل کاپلان هم حل کنید در دو ستون CPU time ها را بنویسید، آزمون آماری انجام بدهید و بگویید که کدامش بهتر است یعنی حرف از مقایسه می شود تنها راه آمار است باید آماری بحث کنید.

اگر من یکی، دو تا ، ۳ تا مثال را حل کنم خوب مثلاً با این حل کردم متوسط توی این ۱۳/۳۰ ثانیه حل شده است با اون یکی مدل هم حل کردم متوسط ۱۲/۸ ثانیه حل شده است. الان ۲ تا روش به نظر شما برابرند ؟! ۰/۷ ثانیه خیلی مهم نیست. عملاً می‌توان گفت که اینها با هم فرقی ندارد ولی خب این نظر من یا نظر شما مبنای این بحث نیست ما

در آمار با ۴۰ تا داده با توزیع نرمال علمی اینها را مقایسه می‌کنیم دیگه نظر دخالت ندارد. به صورت آماری تست می‌کنیم ثابت می‌کنیم که این در واقع برابر است یا نیست.

مدل سوم که البته چند ثانیه من فقط متغیرش را بگم چون دیگه وقت را نمی‌خواهم درگیر بکنم.

مدل سوم که آقای کلین (Kelin) که در سال ۲۰۰۰ پیشنهاد کرده است. اگر فعالیت i در زمان t در حال انجام باشد یا قبل تر در حال انجام بوده باشد یک و در غیر اینصورت صفر است.

$$x_{it} = \begin{cases} 1 & \text{if activity } i \text{ is in progress in period } t \text{ or was so before} \\ 0 & \text{otherwise} \end{cases}$$

خوب به هر حال این تعریف با دو تا تعریف قبلی فرق دارد پس نتیجه محدودیت‌ها ، تابع هدف هم فرق می‌کند این هم دوباره ثابت کرده این تعداد متغیر و محدودیت است.

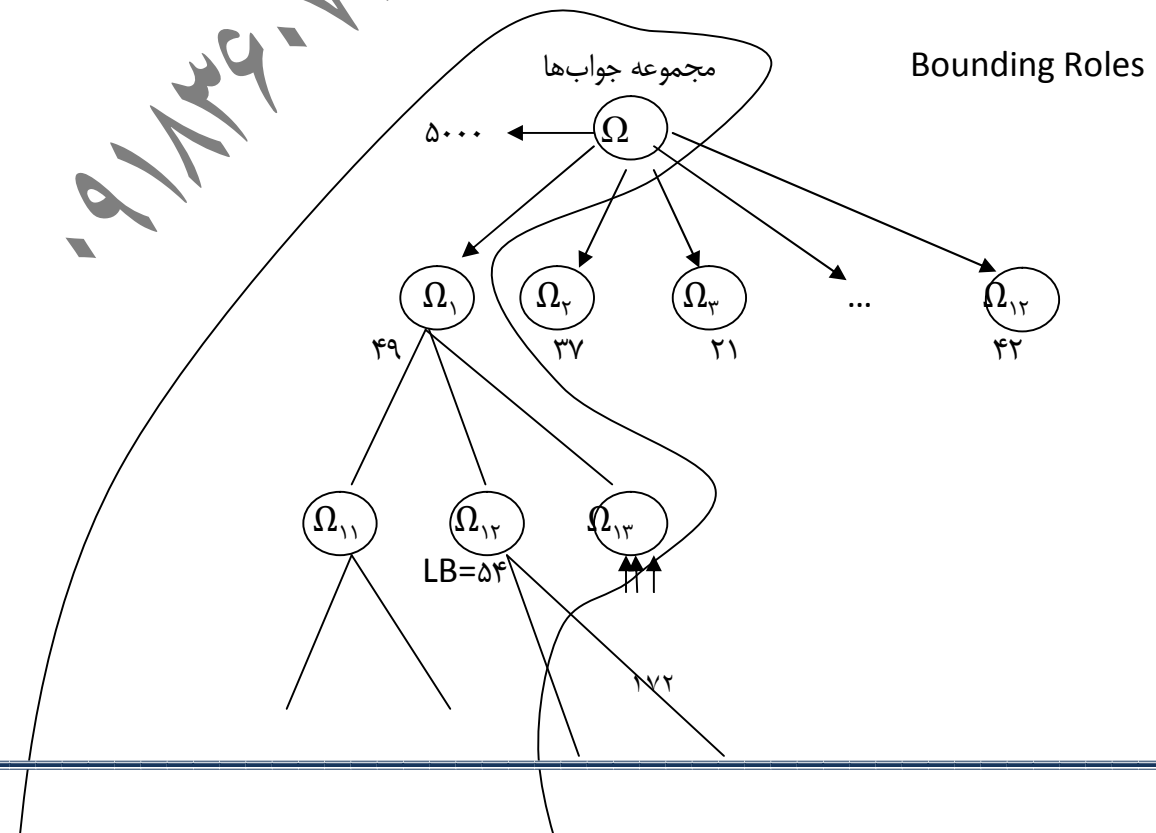
افراد دیگه‌ای هم هستند که من فقط اسم‌هاشون را آوردم. آلوارزه در سال ۱۳۹۳ و مینگازی در سال ۱۳۹۸ دو تا مدل پیشنهاد دادند کتاب هست ولی دیگه مثال نیاوردم. به جز اینها سال‌های اخیر کلی مقاله‌ها می‌توانید پیدا کنید مدل‌های RCPSP هست ولی هر کدام سرشون این بوده که به هر حال یه جوری بتوانند این مدل را سریعتر جواب برسانند.

این راه اول که به اصطلاح برنامه‌ریزی ریاضی است.

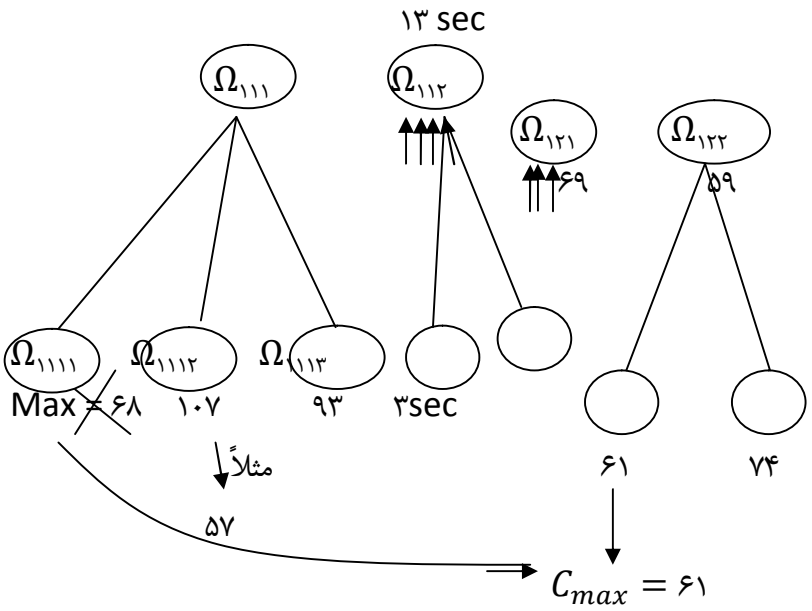
راه دوم بازم تو exact ها روش شاخه و کران است. شاخه و کران را در تحقیق ۲ خواندید چی کار می‌کنید ؟ یعنی منطقش را بگید تو روش شاخه و کران چه اتفاقی می‌افتد ؟ یک مسئله عدد صحیح را حل کردید باهاش دیگه ، دقیقاً چی کاری کردید مثلاً می‌گفتید این روش، روش شاخه و کران است. اول یک سوال بپرسم روش شاخه و کران برای چه مسئله‌ای است ؟ پاسخ: برای هر مسئله گسسته‌ای بدون استثناء هر مسئله گسسته‌ای را می‌شود با شاخه و کران حل کرد. حتی اگر غیر خطی باشد یعنی خطی بودن اصلاً شرط نیست، خوب ایده این روش از دو تا کلمه است.

Branching و Bounding حالا انشعاب زدن و حد گذاشتن (محدود کردن) ؛ کران گذاری

تفکیک فضای جواب به چند قسمت

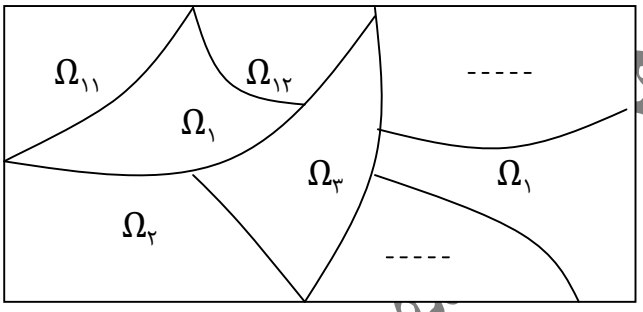


جواب‌ها همه بالای ۹۴ حد پایین LB=۹۴



fathom گم نشود

بهتر است تکرار نباشد



Full Enumaration

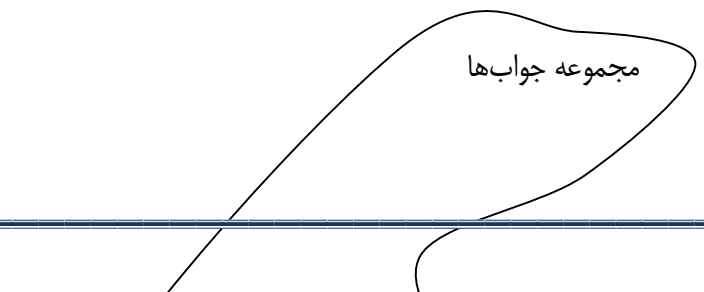
اول branching : در branching چه اتفاقی می‌افتد. در branching اگر ما مجموعه جواب‌ها را کلاً با Ω نشان بدهیم مثلاً فرض کنید یک مسئله‌ای ۵۰ هزار تا جواب دارد. تعداد است ۵۰ هزارتا دارم عددی می‌گم می‌شمارم ۵۰ هزار تا جواب دارد. گسسته است مسئله در branching ما به اصطلاح بهینه در این ۵۰ هزار دنبال چی هستیم؟ یک جواب که به اصطلاح بهینه است برای اینکه بتوانیم مسئله را handle کنیم فضای جواب را تجزیه می‌کنیم، تقسیم می‌کنیم یعنی چی ؟ یعنی اگر فضای جواب Ω باشد من این را تقسیم می‌کنم به Ω_1، Ω_2، Ω_3، Ω_4 تا Ω_K حالا کلی به این K می‌گویند شاخه‌زنی پس branching یک جمله کوتاه بیشتر نیست تفکیک فضای جواب به چند قسمت. این می‌شود شاخه‌زنی. حالا چند قسمت ؟ به دو تا ، سه تا ، ده تا این دیگر به مسئله بستگی دارد یکی ممکن است به دو قسمت بشکنید و جای دیگر به ۱۰ قسمت بشکنید بستگی به مسئله دارد منم اینجا نوشتم K یعنی کلی نوشتم چون نمی دانم آن K چند است خوب فقط در شاخه‌زنی یک نکته بسیار ریزی باید دقت کنید. در این شاخه زنی هیچ جوابی باید گم نشودیعنی چی ؟ یعنی اگر Ω_1 تا Ω_K به اصطلاح دوباره یک کاسه کنید باید چی بشود باید همان Ω بشود پس وقتی تجزیه می‌کنید و بعد دوباره اینها را مونتاژ (سرهم) می‌کنید باید همان اصلی شود هیچ چیز از بین

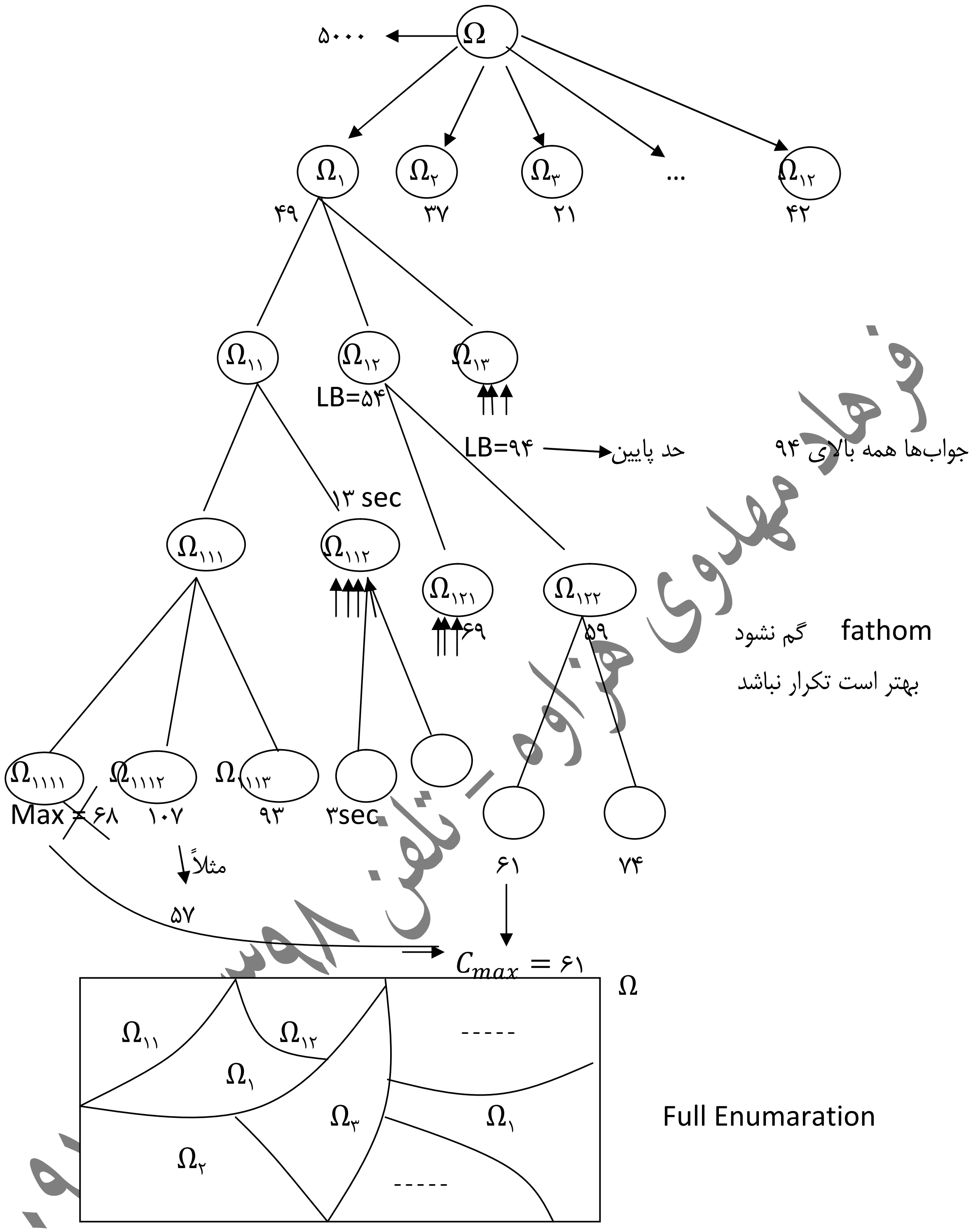
نرود تکراری می‌تواند باشد ؟ یعنی یک جواب هم اینجا و هم آنجا می‌تواند باشد. بهتر است که نباشد ولی اگر شد اشکالی ندارد اگر یک جوابی را در دو جا داری اشکال ندارد یعنی غلط نیست فقط دوباره‌کاری می‌شود یعنی یک جواب را دوبار چک می‌کنید وقتت گرفته می‌شود ولی اشکال ندارد چون جواب را از دست نخواهی داد چون جواب تکراری شمردن باعث از بین رفتن جواب نمی‌شود یک جواب را دوبار شمردید یکم بیشتر وقتت تلف شده موقع حساب و کتاب ولی اگر از بین برود آن جوابی که از بین رفته ممکن است بهینه باشد پس آن وقت شما دیگه بهینگی را از دست داده‌اید.

بعد چی کار می‌کنیم. بعد این بسته را که اسمش Ω_1 است یا یکی از بسته‌ها رو مثلاً Ω_1 رادوباره چی کار می‌کنیم تجزیه می‌کنیم به Ω_{11}، Ω_{12}، Ω_{13} من اینجا به سه قسمت کردم ولی بازم ممکن است بیشتر از این یا کمتر تقسیم‌بندی شود ولی بازم آن شرط هست. هیچ جوابی گم نشود خوب دوباره تا کی؟ نمی‌دانم ممکن است ۱۰تا ، ۲۰ تا ، ۱۰۰ تا لازم باشد سطح شاخه زنی انجام شود بیاید پایین‌تر ، آخر سر به کجا می‌رسید ؟ این بسته‌ها را وقتی کوچک و کوچک تر می‌کنید آخر سر چه اتفاقی می‌افتد؟ این را دوباره بازش کنم Ω_1 و دوباره Ω_{11} را بازکنم و در آخر بسته‌های بدست می‌آید که توش فقط یک دانه است دیگه بیشتر از این قابل شکستن نیست این جا می‌گم شاخه چی شده است ؟ به عمق رسیده (fathoming) این شاخه دیگر آخرش است به اصطلاح به عمق رسیده است دیگه بیشتر از این قابل تفکیک نیست. به زبان بهتر یعنی در این شاخه الان چند تا جواب ته این شاخه وجود دارد ؟ یکی، اگر دو تا بود بازم می‌شد تفکیک کرد یکی‌اش بیاد این سمت و دیگری بیاد آن سمت. ولی وقتی یکی است دیگر قابل تفکیک نیست حالا این یکی‌ها که هستند C_{max} مثلاً تابع هدف باشد. C_{max} این چقدر است ؟ اینجا نوشتم ۶۸.

من اولین جوابم را پیدا کردم چند است؟ ۶۸ خوب فرض کنید روی این میز یک بایگانی داریم یک آرشیو داریم این جواب را اینجا ذخیره می‌کنم به عنوان اولین و بهترین جوابی که تا این لحظه پیدا کردم. بعدی ۱۰۷ است . آن را بردارم و ۱۰۷ را جای آن بذارم ؟ نه چون ۱۰۷ از آن بدتر است اگر بهتر بود اگه مثلاً این ۵۷ بود به فرض آن وقت این بایگانی را به روز می‌کردم آن را پاک می‌کردم این بهتره را جایگزین می‌کردم ولی چون ۱۰۷ است به درد نمی‌خورد . ۹۳ ام که هیچی به هر حال توی این شاخه‌ها که الان بررسی کردم بهترین جوابی که تا این لحظه پیدا کردم این اعداد را از کجا آوردم و نوشتم فرض است دیگه من که مثال حل نکردم من دارم فلسفه شاخه و کران را می‌گم الان مثال حل نمی‌کنیم می‌گم فلسفه شاخه زنی اینه که مسئله را (جواب‌ها را) به مجموعه‌های کوچکتر و کوچکتر و کوچکتر می‌شکند تا برسد به بسته‌های یک عدد جوابی است این یک عدد جواب ، C_{max} اش را می‌نویسید هر کدام که بهتر از همه است این باید بایگانی کند و قبول کنیم . بقیه شاخه‌ها به همین صورت. یک چند ثانیه فکر کنید واضح است اگر همه شاخه‌ها را تا ته باز کنید چند تا از این بسته‌ها این پایین خواهید داشت از این بسته‌های یک عدد به اندازه آن بالایی یعنی اگر مجموعه اصلی شما ۵۰۰۰۰ تا جواب داشته است شما الان خرد و خرد و خرد کردید الان ۵۰۰۰۰ تا بسته یکی‌ایی دارید به این کار شکستن، branching می‌گویند. این همان چیزی بود که شما نوشتید.

تفکیک فضای جواب به چند قسمت





و به این نحو حل مسئله Full Enumaration (شمارش کامل) می‌گویند. یعنی شما عملاً ۵۰۰۰۰ هزار تا جواب را تک تک بررسی و چک کردید. این شاخه و کران به حساب نمی‌آید این فقط شاخه‌اش بود کران در آن مطرح نبود پس من تا اینجا استراتژی شاخه‌زنی را گفتم. اگر روش شاخه و کران ، کران را بردارید و فقط شاخه‌زنی بماند باعث این می‌شود همه شاخه‌ها تا آخر باز می شوند یک درخت بسیار گسترده‌ای می‌شود و اسمش هم می‌شود شمارش کامل اما کلمه Bounding چه زمانی مطرح می‌شود ؟ کران گذاری در این بسته که به عنوان فرض در نظر گرفتم $\Omega_{۱۳}$ چند

تا جواب داشته نمی دانم شاید ۷ یا ۸ هزار تا جواب در آن باشد. بسته است هنوز باز نشده البته یکی، دو بار باز شده ولی بازم خیلی جواب در آن است در این بسته با یک حساب کتابی من به نتیجه رسید که lower bound(حد پایین) این بسته ۹۴ است هفته قبل از شما میان ترم گرفتم نمره‌ها را اعلام نکردم فقط بهتون می‌گم که مینیمم نمره ۱۱ بوده است. این یعنی چی ؟ ۱۱ ، lower bound است یعنی همه شما متوجه شدید که نمره‌هاتون هم ۱۱ و به بالا است چون گفتم که کمترین نمره کلاس ۱۱ است الان دقیقاً هیچ نمره‌ای مشخص نیست مطلقاً ولی اطلاعتش تا حدی هست که قطعاً بالای ۱۱ است ولی دقیقاً چند است معلوم نیست پس lower bound کلی داره می گه در این مجموعه جواب‌ها همه شان بالای ۹۴ است حالا یکم فکر کنید با توجه به اینکه ۶۸ ایی که قبلاً بایگانی کردیم خوب آن بسته را من دیگه اصلاً بازش نمی‌کنم چون من دنبال جواب بهتر از ۶۸ می‌گردم چون ۶۸ را همین الان آماده و نقد دارم دنبال جواب بهتر از ۶۸ می‌گردم وقتی مطلع می‌شود که آن شاخه lower bound اش ۹۴ است این شاخه بسته می‌شود محدود می‌شود یا کران گذاری می‌شود من ۳ تا فلش زیرش گذاشتم که این شاخه دیگه بسته شده است و دیگر ادامه نخواهد داد. خوب این بستن شاخه خوب است یا بد ؟ چه خوبی دارد؟ زمان. خوبی‌اش این است که در این ۷ ، ۸ هزار تا جواب را که می‌خواستید تک تک بررسی کنید الان به طور یک جا ، به طور کلی و یکجا حذف کردید و این کار را خیلی راحت تر کرد خوب شاخه‌های بعدی همین طور این را ببندیم یا نه ؟ این ۵۴ را ؟ نه . چرا نمی‌توانیم ببندم چون Lower bound اش ۵۴ است و امید بخش است. سوال : پس ۶۸ را از آرشیو پاک می شود و به جای آن ۵۴ می‌نشیند ؟ درست است ؟ خیر ۵۴، Lower bound است C_{max} که ۵۴ نیست . من گفتم همه نمره‌ها بالای ۱۱ است ولی نگفتم که نمره شما ۱۱ است یا هیچ فردی را نمره‌اش را نگفتم. گفتم نمرات بالای ۱۱ است

خلاصه یک نمره‌ای بوده که ۱۱بوده است. من یه چیز به شما بگم از الان با اطمینان مطلق بگم نمرات همه شما بالای ۲ است ولی منظورم این نیست که قطعاً یکی از نمرات شما ۲ است درست است ؟ قطعاً نمرات شما بالای ۲ یا بالای ۵ است. من زیر ۵ نمره نمی‌دهم مثلاً پس lower bound لزوماً یکی از جواب‌ها حتماً آن مقدار را دارد و حتی به فرض هم داشته باشد ممکن است جوابی باشد که موجه نباشد بنابراین این ۵۴ من را فعلاً خوش بین نگه می‌دارد که این شاخه را نبندم حالا ممکن است یک سطح پایین تر بیام مثلاً وقتی یک شاخه می‌زنم $\Omega_{۱۲۱}$ ، $\Omega_{۱۲۲}$ و lower bound حساب می‌کنم این ۵۴ ممکن است بشود ۶۹ این هم بشود ۷۷

من می‌گم نمراتون همه بالای ۲ است ولی هیچ وقت به این معنی نیست که حتماً یکی‌اش ۲ است آنجا گفتم همه جواب‌ها بالای ۵۴ است همین ولی نگفتم که حتماً یکی‌اش ۵۴ است حالا جلوتر اومدم به نتیجه رسیدم که همه نمراتون بالای ۶۹ است اونم همه بالای ۷۷ است.

خوب ۶۹ و ۷۷ هر دو چی می شوند هر دو مغلوب ۶۸ هستند این شاخه‌ها دیگه بسته می‌شود. این کار تا کی دنبال می‌شود ؟ همه شاخه‌ها. گوش کنید فقط یک موضوع را بگم . بعد بگم کی بسته می‌شود حالا اگه احیاناً این را من یکی را باز نگه دارم این ۷۷ را اجازه بفرمایید که ۵۹ نگه دارم بعد شاخه می‌زنیم بعدی می‌شه ۶۱ و بعدی میشه ۷۴ خوب چی میشه اینجا نه این تمام شد fathom شد این شاخه‌ها بسته شد به عمق رسید اینی که قبلاً در بایگانی به اسم ۶۸ داشتیم حذف می‌شود به جاش الان C_{max}، ۶۱ است یعنی از این لحظه به بعد دیگه ۶۸ ملاک نیست ۶۱

مبنا است یعنی هر جوابی که از ۶۱ بدتر باشد من دیگر ادامه نخواهم داد وقتی به اصطلاح این بایگانی را به روز می‌کنید با مقدار به روز شده‌اش باید بقیه راه را بروی برای شاخه‌ها این داستان پیش می‌رود تا همه شاخه‌ها یا به عمق برسند یا بسته شوند. به چه دلیل بسته می‌شوند؟ به سه دلیل شما مجاز به بستن شاخه هستید. من یکی‌اش را گفتم.

- دلیل اول : بستن شاخه‌هایی که lower bound هستند.
- دلیل دوم : تکراری بودن است. اگر یک شاخه‌ای تکراریه ، چون گفتم ممکن است جواب تکراری هم بشود توی اینها شاخه‌های که تکراری اند دیگه نیاز به بررسی ندارد.

در روش شاخه و کران به سه دلیل می‌توان شاخه‌ها را بست.

- (LB)- Lower Bound
- تکراری
- ناموجه بودن (تهی بودن)

اگر شاخه‌ای Lower bound از بهترین جوابی که در آرشیو دارید بدتر بود ادامه ندهید.

اگر جوابی (شاخه‌ای) تکراری بود ادامه ندهید.

اگر جوابی (شاخه‌ای) داشتید که تهی بود یعنی ناموجه بود ادامه ندهید.

حالا اینها در هر مسئله‌ای نحوه حساب، کتابش با هم فرق می‌کند. مسئله کی تمام می‌شود که همه شاخه‌ها یا بسته شوند یا به عمق برسند آن وقت آخر از همه بهترین جواب کدام است؟ آخرین جوابی که در آرشیو دارید بهینه است. زمانی که همه شاخه‌ها به عمق برسد یا به یکی از دلایل فوق بسته شود آخرین جواب ذخیره شده بهینه است.

من نمی‌گم $C_{\max}$ چرا چون این موضوع اصلاً به $C_{\max}$ ربطی ندارد دارم عمومی صحبت می‌کنم.

پس آخرین جواب ذخیره شده بهینه است. خوب یک سوال تکراری تو شاخه زنی به چه موضوعی باید توجه کنید ؟ هیچ جوابی گم نشود (از دست نرود) . در کران گذاری باید به چه چیزی توجه کنید ؟ بستن شاخه باید با دلیل باشد یعنی الکی نیست به نظر من نیست باید محرز شود که این جواب تکراری است یا محرز شود که ناموجه است یا محرز شود که lower bound اش اینه یعنی اصلاً نظرسنجی وجود ندارد یعنی قطعاً پشتوانه منطقی یا ریاضی داشته باشد که یکی از اینها باید محرز شود ۳ تاش لازم نیست. یکی از دلایل کافی است این شاخه را ببندم یا نه ؟

۱- Lower bound =۳۷ جوابی که من دارم ۶۸ خوب با دلیل اول (۶۱) قانع نشدم که ببندم چون lower bound اش نمی‌خونه

۲- تکراری است یا نه ؟ با یک مکانیزمی چک می‌کنم تکراری هم نیست

۳- ناموجه است یا نه ؟ ناموجه هم نیست پس چک می‌کنم

ولی وقتی هیچ کدام از اینها اتفاق نیافتد باید شاخه را ادامه بدهید اما اگر یکی از اینها اتفاق می‌افتاد برای بستن شاخه کافی بود.

سوال : کدام شاخه باید اول بررسی شود ؟ آیا فرقی می‌کند که این شاخه بررسی شود بعد این یکی شاخه و بعد دیگری ؟

باید همه بررسی شود ولی معمولاً دو استراتژی برای اول بررسی شدن شاخه وجود دارد:

- Depth first ⟵ روی شکل جدا شده است با خط
- Best first ⟵ (۲۱)

Depth first این است که الان من اینجا رسم کردم البته این Depth first نیست مگر اینکه من اینها را پاک کنم تا Depth first بشود. Depth first به این صورت است که شاخه‌ای که شروع می‌شود مثلاً به ترتیب این اولیه تا کجا می‌رود ؟ تا به عمق برسد. یعنی یک شاخه که شروع کرد ولش نمی‌کند تا به عمق برسد یا بسته شود یعنی تکلیف شاخه مشخص بشود از این شاخه به آن شاخه پریدن جابه‌جا شدن ندارد. شاخه‌ای که شروع شد تا به عمق یا به یکی از سه دلیل بسته شدن پیش می‌رود الان وقتی این بسته شد بعد سراغ کدام شاخه می‌رود سراغ آخرین شاخه‌ای که سرگردان بوده است بعد این بعد این بعدش این یعنی وقتی که شاخه به عمق رسید بعدش سراغ آخرین شاخه‌ای که رها کرده می‌رود نه اولی یعنی در واقع Lifo، Lifo یادتان هست ؟ یعنی آخرین شاخه‌ای که سرگردان گذاشتید اول بررسی می‌شود درست شد یا به زبان بسیار ساده تر از سمت چپ شروع کنید شاخه‌ها را پر کنید تا بیاد به سمت انتهای راست یا از این سمت شروع کنید به آن سمت فرق نمی‌کند. از این سر شروع کنید ولی به ترتیب همه را چک کنید. به این Depth first می‌گویند.

Best first : اسمش روش هست. بهترین را اول، یعنی چی ؟ بهترین کدام است الان ؟ بهترین یعنی بهترین Lower bound منظورش است. ۴۹ ، ۳۷ ، ۲۱ باید این رو باز می‌کرد من این را Depth first نوشتم برای اینکه best first پیش برود باید اول این شاخه را باز می‌کرد یعنی شاخه‌ای که امید بخش تر است. اوضاعش بهتر است یعنی بیشتر به نظر می‌آید که جواب بهینه در آن باشد. آن را اول بررسی می‌کنیم ولی خروجی آخرش یکی است یعنی خلاصه شما یک جواب بهینه یکسان برای مسئله خواهید داشت چون به هر حال همه شاخه‌ها خلاصه بررسی بشوند راجع ترتیبش حالا کدام اول ، کدام دوم دیگر این دو استراتژی را معمولاً می‌گن. کدام یک بهتر است ؟ اگر بهتری بود قطعاً آن یکی حذف می‌شد پس در بعضی از مسائل Depth first بهتر است و در بعضی از مسائل دیگر best first بهتر است چطور می‌توانید چک کنید که برای یک مسئله کدام بهتر است ؟ ۴۰ تا مسئله را برای آن مثال واحد با هر دو باید حل کنید تا ببینند کدام زودتر به جواب می‌رسد یعنی اینها کاملاً باز آماری است.

تعداد این قوانینی را که گفتم نوشتید یعنی قواعد بستن شاخه را Bounding Role می‌گویند.

- Lower bound
- تکراری
- Infeasibility

این قواعد هر چه قدر زیاد باشند خوب است یا کم باشند خوب است ؟ یعنی من ۵ تا bounding role داشته باشم بهتر از سه تا است ؟ ۵ تا مگه می‌شود داشته باشم مگه سه تا ننوشتم آنها سه تا نیست سه گروه است یعنی چی ؟ یعنی خود lower bound ممکن است در مسئله‌ای ۱۰ تا lower bound داشته باشد ۱۰ مدل LB، تو تکرار

می‌توانید ۱۰ مدل تکرار داشته باشید آن infeasibility را می‌توانید به ده مدل بسنجید پس آن هر کدام یک مجموعه است نه اینکه شما حداکثر الان فکر کنید حداکثر ۳ تا bounding role وجود دارد نه می‌تونه بیشتر باشه مثلاً من می‌گم ۵ تا چطوری می‌گم ۵ تا ۲ تا قاعده lower bound ایی دارم ۲ تا قاعده تکرار دارم یک عدد قاعده ناموجه دارم پس شد ۵ تا ، ۵تا بهتر است یا ۳ تا ؟ گفتید ۵ تا . چک کنیم با هم چرا لزوماً۵ تا بهتر نیست. این شاخه را می‌خواهم بررسی کنم که ببندم یا نه. قاعده یک Lower bound اش را می‌خواهم چک کنم که هیچی بسته نمی‌شود ، دو هم بررسی کردم بسته نمی‌شود. با سه هم چک کردم بسته نمی‌شود با ۴ و ۵ هم بسته نمی‌شود. پس بسته نشد ادامه می‌دهم ولی وقتی که تلف شد چی ؟ زمان مگه ملاک نیست اینجا الان. یعنی lower bound های زیاد لزوماً حسن نیست. جمله‌اش را باید چی درست کنیم. Lower bound هایی موثر زیاد. یعنی باید موثر باشد lower bound هایی که باعث بسته شدن شاخه نشود فقط وقت را تلف می‌کند شما دارید چک می‌کنی ولی شاخه را نمی‌بندید ولی وقت را تلف کرده است پس یک جمله دیگه تعداد bounding role ها (تعداد قواعد) بستن شاخه‌ها که موثر باشند هر چه بیشتر باشد بهتر است.

حالا موثر را باید از کجا تشخیص بدهم موثر است یا نه ؟ مثلاً می‌گم یک bounding role این شاخه را اینجا بسته است حالا اگه نمی‌بست چی می‌شد. اگه نمی‌بست من یک ثانیه وقت بیشتری می‌گذاشتم اینم باز می‌کردم یعنی وقتی شاخه‌ها را پایین ببنده در سطوح پایین دیگه خیلی فایده ندارد باید بتواند از ریشه بزند اینها رو یعنی هر چقدر بتواند بالاتر اینها را cutکند حسن به حساب می‌آید شما وقتی که دیگر خیلی دیر متوجه شدی خوب پس کلی دیگه محاسباتش را انجام دادی دیگه بنابراین خیلی دیگه توفیری ندارد. بستن با نبستن‌اش خیلی فرق نمی‌کند حتی ممکن است نبستنش حسن باشد این که پردازش ۱۳ ثانیه طول کشید ۱۳ ثانیه حساب کتاب کرده که فهمیدم این را باید ببندیم در حالی که اگر آن bounding role نبود در ۳ ثانیه می‌توانستم این را باز کنم و حساب کنم نتیجه شما اصلاً ادامه می‌دادید بهتر بود با ۳ ثانیه بررسی می‌کردید بهتر از این بود که با ۱۳ ثانیه به نتیجه بستن این شاخه برسید.

سوال آخر : از کجا متوجه می‌شویم که یک bounding role موثر است یا نه ؟ با عدد ۴۰ شروع کنید ۵ تا bounding role دارم مسئله را حل کردم که هر ۵ تا این موثر است و حالا یک ذهنیتی دارم که آیا یک bounding role دیگری اضافه کنم یا نه یک bounding role حساب کردم که به هر حال حساب کتابش درست است فقط موثر بوندش را می‌خواهم سوال بپرسم ؟ پاسخ ۴۰ تا مسئله را اول با همان ۵ تا bounding role حل می‌کنیم و cpu time ها شون را بنویسیم حالا همان ۴۰ تا را با (حالا اینم اضافه کنید) با ۶ تا bounding role بعد ببینید از نظر آماری از نظر زمان این ۶ امی باعث کمتر شدن زمان شد یا نه ؟ یعنی چی ؟ چی رو باید آزمون کنید این μ_1 باشد این μ_2 آیا $\mu_2 \leq \mu_1$ است یا $\mu_2 > \mu_1$ است ؟ یعنی منطقاً bounding role موثر یعنی باید بیاد و زمان را کم کند ببینید آیا واقعاً کم کرده است ؟ آیا μ_2 یعنی زمان اونی که با ۶ تا هست کمتر از اونی است که با ۵ تا بود ؟ اگر این $H_.$ تأیید شد تمام. اثبات کردید ۶ bounding role موثر است . بنابراین آن را هم اضافه کنید.

	μ_1	μ_2	
۵	۵	۶	

$$\begin{cases} H_.: \mu_2 \leq \mu_1 \end{cases}$$

$$H_1: \mu_2 > \mu_1$$

برای یک مسئله‌ای می‌شود ۱۰ ها تا role نوشت ولی موثرهایش مهم است این خیلی مهم است موثرهایش مهم هستند. تعدادشون خیلی منبا نیست راجع به شاخه و کران فقط از چی صحبت کردیم کلی صحبت کردیم من وارد مسئله نشدم. یک تمرین بهتون می‌دهم برای هفته بعد با شاخه و کران حل کنید بیارید تمرین ربطی به زمانبندی پروژه ندارد. در تحقیق یک یک مسئله‌ای داشتیم به اسم تخصیص ۵ تا ماشین داریم ۵ تا کار می‌خواهیم کارها را تخصیص بدهیم به ماشین‌ها. یک کار به هر ماشین هزینه انجام کار با هر ماشین مشخص است. به روش شاخه و کران تخصیص بهینه را تعیین کنید. با روش مجارستانی که در تحقیق یک خواندید نه با روش شاخه و کران.

می‌خواهید ۵ کار مختلف را به ۵ ماشین طوری تخصیص دهیم که هزینه کل انجام کارها مینیمم شود با فرض اینکه تخصیص یک به یک است یعنی یک کار به یک ماشین به روش شاخه و کران تخصیص بهینه را تعیین کنید.

خوب کار A را که اگر به ماشین یک بدهید ۳۷ واحد هزینه دارد به ماشین ۲ ، ۴۳ واحد و الی آخر. این داده‌های مسئله است این مسئله چند تا جواب دارد. یعنی Ω اش چند تا جواب دارد؟ Ω اش چند تایی است؟ احتمالات لیسانس ۵ تا کار به ۵ تا ماشین اصل شمارش $۵! = ۱ \times ۲ \times ۳ \times ۴ \times ۵ = ۱۲۰$، این مسئله ۱۲۰ تا جواب دارد کلاً ما در این ۱۲۰ تا دنبال آن یکی هستیم که به اصطلاح بهینه است پس $\Omega = ۱۲۰$ است ۱۲۰ تا را باید به چند قسمت بشکنیم این شاخه اول را بشکنید چه طوری باید به ۵ قسمت تقسیم کنیم شاخه اول را باید برای کار A بشکنید کار A را به کدام می‌شه داد کار A را به یک به دو به سه ، چهار ، پنج مطمئن‌اید که این همه را شمرده جوابی گم نشد باید تضمیمن بشود که جوابی از بین نرفته است.

کار \ ماشین	۱	۲	۳	۴	۵
A	۳۷	۴۳	۵۷	۳۹	۴۸
B	۳۹	۴۷	۴۸	۶۱	۵۷
C	۴۳	۴۸	۶۲	۳۸	۵۳
D	۴۹	۵۱	۵۳	۵۶	۵۹
E	۶۰	۴۸	۴۷	۳۹	۶۰

پارت ۱۰ را تا یه جایی صبحت کردیم که البته بیشتر مفاهیم کلی از شاخه و کران مطرح شد که اصلاً موضوع در رابطه با چی هست اگر در چند ثانیه خلاصه کنیم در واقع branch and bound یک روش شمارش است ولی نه شمارش کامل، شمارش به اصطلاح ضمنی یعنی برخی از جواب‌ها به اصطلاح به صورت گروهی بدون اینکه چک یا بررسی شوند کلاً حذف می‌شوند . که به سه دلیل زیر می‌باشد :

- تکراری بودن شاخه

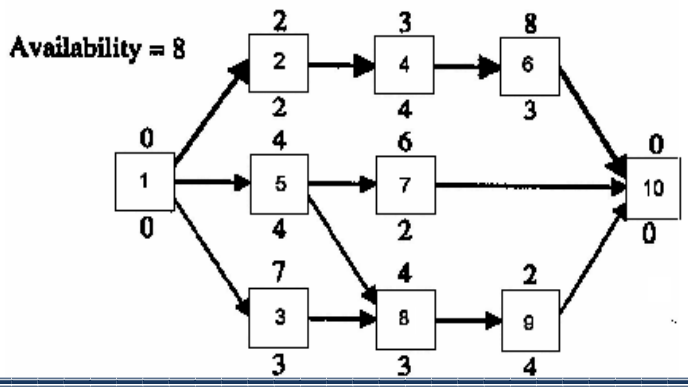
- ناموجه بودن شاخه
- به دلیل lower bound که به آن جواب مغلوب هم می‌گویند.

جوابی که به دلیل Lower bound مغلوب است یعنی lower bound اش طوری است از بهترین جوابی که شما دارید بدتر است پس مغلوب به حساب می‌آید و dominate بهش می‌گویند. Dominate شده آن جواب در واقع حذف می‌شود.

ولی این بحث که صحبت شد کلی بود اما به صورت خاص، از مسئله شاخه و کران بخواهیم حل کنیم RCPSP را می‌شود به چندین روش شاخه و کران حل کرد. ببینید این نیست که شما برای حل مسئله فقط یک بار از شاخه و کران استفاده کنید چون نحوه شاخه زنی و نحوه bounding گذاری (کران گذاری) می‌تواند متنوع باشد اینها می‌تواند باعث شود برای یک مثال بتوانید ۱۰ تا ، ۱۵ تا روش شاخه و کران توسعه بدهید ما برای RCPSP واقعاً بیش از ۱۰، ۱۵ تا روش شاخه و کران داریم در مقاله‌ها. حالا در این کلاس ۴ تا از این روش‌ها گفته می‌شود پس مسئله RCPSP که مسئله ما بوده آن هفته محدودیت منابع در زمانبندی پروژه را می‌خواهیم به صورت exact با روش شاخه و کران حل کنیم.

روش Precedence-tree :

این روش مفهومش مفهوم شاخه و کران است و آقای پترسون و اسپرشو این روش را ارائه کردند. شما در این روش بیشتر به چی باید دقت کنید ؟ چه جوری شاخه‌زنی انجام شده است ؟ جواب‌ها را به چه صورت دارد می‌شمارد؟ و چه جوری شاخه‌ها را می‌بندد یعنی کلاً تو شاخه کردن دو تا موضوع را باید متوجه بشوید که شاخه‌زنی چه طور انجام شده است و شاخه‌ها چه طور بسته شده است . در هر کدام یک نکته آن هفته توشتیم در شاخه زنی به چی باید توجه کنید؟ هیچ جوابی گم نشود. در bounding هم باید به چی توجه می‌کردید ؟ دلیل محرز داشته باشید که این شاخه تکراری است باید محرز شود که این شاخه infeasible است یا محرز شود که این شاخه lower bound اش این است و بسته شود والا الکی نمی‌توانید شاخه را الکی ببندید پس در شاخه زنی و کران‌گذاری هر کدام یک نکته بسیار ظریف را باید دقت کنید. حالا این روش را که داریم شروع می‌کنیم براساس یک اصطلاحی است به اسم Feasible-sequence یعنی چی feasible sequence ؟ یعنی توالی شدنی ، توالی موجه. می‌گه هر جوابی را می‌شود به صورت یک توالی موجه، به اصطلاح به صورت یک کد، عین شما که هر کدام یک کد دانشجویی دارید من از روی لیست اسم شما را بخوانم یا کد دانشجویی‌تان را فرقی نمی‌کند. یعنی یک رابطه یک به یک بین اینها وجود دارد. حالا اینجا هم می‌گه هر زمانبندی را می‌شه با یک feasible sequence ای نشان داد. یعنی چی feasible sequence ؟ ببینید خیلی ساده است اصلاً فرمولی ندارد.



در این مثال ساده دقت کنید، این مثال ۱۰ تا فعالیت دارد که اعدادی که بالای فعالیت نوشته شده است duration می‌باشد و اعدادی که پایین نوشته شده است r (نیاز منبع) است یعنی الان مثلاً این چی داره می‌گه ؟ می‌گه فعالیت ۸، ۴ روز طول می‌کشد روزی هم ۳ واحد منبع می‌خواهد مثلاً ۳ تا کارگر می‌خواهد. فعالیت ۹ ، دو روز طول می‌کشد و روزی ۴ تا منبع می‌خواهد و الا آخر

فعالیت اول و آخر مجازی هستند هم duration و هم نیاز به منبع‌شان صفر است. کلاً چند تا منبع داریم ؟ ۸ تا اینم اینجا گفته پارامتر a_k را گفته. پس یک مثال ساده را انتخاب کردم که روش را از روی مثال صحبت کنیم. یک نوع منبع داریم از آن ۸ تا داریم ، شبکه پروژه، duration و نیاز به منبع مشخص است. حالا یک feasible-sequence تعریفش چیست؟ از یک تا ۱۰را به ترتیب بگویید ولی دقت کنید که پیش‌نیاز را قبل از پس نیاز بگویید به این می‌گویند یک feasible-sequence . یعنی همه فعالیت‌ها را به ترتیب بشمارید. بچینید کنار هم ولی با رعایت پیش‌نیازی مثلاً یک feasible-sequence الان بگید.

۱ ۲ ۵ ۳ ۴ ۷ ۸ ۹ ۶ ۱۰

این feasible بود به خاطر اینکه من در این ترتیبی که گفتم همواره پیش نیاز را قبل از پس نیاز گفتم اما اگر مثلاً بگویید ۴ ۱ غلط است چون ۲ را هنوز نگفته‌اید نمی‌توانید ۴ را بگویید ۳ هنوز در گفتار من نیامده بود که ۴ را گفتم این غلط است این feasible-sequence به حساب نمی‌آید.

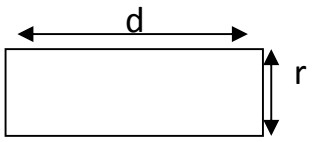
خوب این یکی چی ؟ ۸ ۵ ۲ ۱ ، اینم غلط است. چون باید ابتدا ۳ گفته شود بعد ۸. یعنی اگر شما پیش نیاز را بگویید اون دیگه feasibleبه حساب نمی‌آید. پس اینجا می‌گه که هر جوابی را می‌شود با چنین توالی بیان کرد که feasible sequence گفته می شود. پس به جای اینکه جواب‌ها را بشمارید می‌توانید بگویید feasible-sequence ها را بشمارید. چرا ؟ چون هر جوابی هرfeasible sequence ای با هم فرق نمی‌کنند این که شما را به اسم صدا کنم یا کد دانشجویی‌تان را بگم زیاد با هم فرقی نمی‌کند. آن کد دانشجویی خوبی‌اش فقط به این است که به صورت عددی است. بحثش راحت‌تر است اینجا هم یک توالی را صحبت کنیم در واقع کار ساده‌تری را خواهیم داشت.

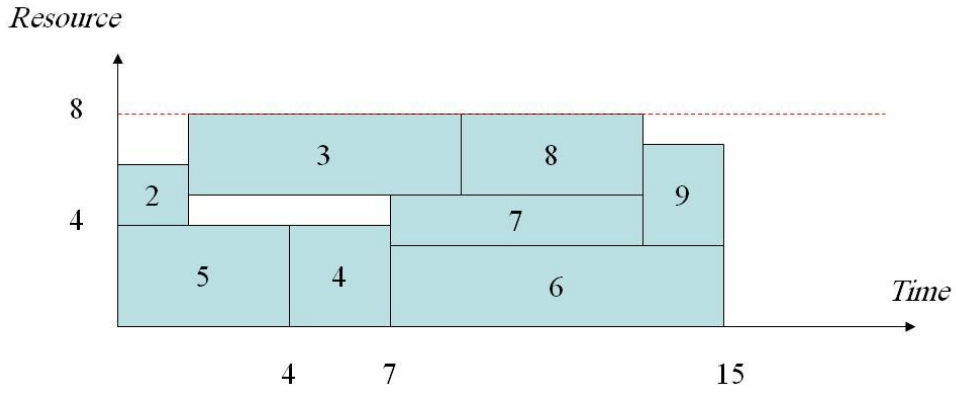
مثلاً این الان یک feasible sequence است یا نه ؟

۱۰ ۹ ۸ ۷ ۶ ۴ ۳ ۵ ۲ ۱ ، آیا این feasible sequence است یا خیر ؟

بله این یک feasible sequence است. اما این تنها feasible sequence نیست این یک feasible sequence است شما می‌توانستید به صورت دیگه‌ای هم بگویید؟ حالا این feasible sequenc را چه طوری می‌شود تبدیل به زمانبندی کرد را ببینید این که شما یک جواب را به صورت کد می‌گویید بهش کد کردن می‌گویند

حالا عکسش را چه می‌گویند ؟ decodeکردن . یعنی شما به اصطلاحی می‌توانید رمز گشایی کنید حالا این را چه طور می‌توان تبدیل کرد به یک زمانبندی ؟ decode کردن اینطوری است که شما یک گانت خالی رسم کنید محور افقی زمان و محور عمودی منبع می‌باشد. که من سقفش را روی ۸ بستم. از این خط چین بالاتر نباید بیایید . چون مسئله گفته ۸ تا بیشتر منبع نداریم این جزء داده مسئله است خوب حالا کدام فعالیت اول انجام شود ؟ طبق توالی کدام یک اول است ؟ یک. ۱ را بچینید اینجا البته یک مجازی است و چیدنش زیاد فرقی نمی‌کند بعدی ؟ ۲ را اینجا بچینید . فقط دقت کنید که هر فعالیت شما باید یک مستطیل یا مربع که طولش durationو عرضش resource است چرا طول durationو عرض resource است ؟ چون محور افقی زمان و محور عمودی منبع است.





حالا اگر شما جای محورها را برعکس نوشتید خوب این را هم باید برعکس بگویید. حالا چون من محور افقی را زمان در نظر گرفتم طول مستطیل *duration* است حالا طبق توالی فعالیت ۲ چند روز است و چقدر منبع می‌خواهد؟ فعالیت ۲، دو روز است و دو تا منبع می‌خواهد پس یک مربع ۲×۲ می‌شود. این مربع ۲×۲ را کجای گانت قرار بدهیم؟ در *ESS* در زودترین زمان . چرا زودترین زمان ؟ یک جمله‌ای را ده بار قبلاً نوشتیم چون تابع هدف *make span* و منظم است و اهداف منظم بهترین جواب *ESS* است پس طبق توالی فعالیت در *ESS* یعنی زودترین زمان (اولین جایی که هم پیش‌نیازی به اصطلاح راضی شود و هم منبع موجود باشد) پس ۲ را چیدیم.

بعدی ۵ می‌باشد که ۴ روز طول می‌کشد و به ۴ تا منبع نیاز دارد یک مربع ۴×۴

سوال : چرا ۲ پایین ۵ کشیده نشده است ؟ مهم نیست می‌توان آن را پایین هم کشید بالا و پایین کشیدن مهم نیست تقدم و تأخر مهم است. فرقی نمی‌کند.

بعدی ۳ می‌باشد. ۳ را چرا از اینجا شروع کردم مگه نباید صفرام شروع شود چون ۳ پیش نیاز ندارد درست است که ۳ پیش نیاز ندارد ولی منبع نبود اگر منبع بود قطعاً از صفر شروع می‌کردم یعنی من ۳ را چون پیش نیازش یک است (یعنی مجازی است) یعنی هیچی دیگه. عملاً پیش نیاز ندارد باید همین الان از صفر شروع کنم ولی خوب باید منبع باشد ولی چون منبع نداشتم از اولین جایی که منبع است شروع می‌کنم.

بعدی طبق توالی ۴ است ببینید اینجا ۳ روز و ۴ تا منبع دارد پیش‌نیازش هم ۲ است زودترین زمان، من اگر جا بود ۴ را بعد از ۲ انجام می‌دادم ولی خوب منبع نیست مجبور شدم با چند روز دیرتر شروع کنم. این منبع هست اینجا این چی هست ؟ منبع هست ولی کافی نیست منبع یک عدد است این فعالیت ۴ تا منبع می‌خواهد منبع کافی نیست.

بعدی طبق توالی ۶ است. ۶ خیلی طولانی است به خاطر اینکه ۶، ۸ روز است و ۳ تا منبع می خواهد.

بعدی طبق توالی ۷ است ۶ روز است و ۲ تا منبع می‌خواهد اینجا جا بود این ۲ تا می‌خواست من این را این داخل جا دادم چون دو تا منبع اینجا هست.

بعدی ۸ است، بعدی ۹ می‌باشد و فعالیت ۱۰ هم که مجازی است. به این کار ***decode*** کردن می‌گویند پس این کد این ***decode*** اش زمانبندی‌اش. حالا حساب کن اگر شما آن عوض کنید چی می‌شود یک توالی دیگر بنویسید. خوب مشخص است عین همین باید با توالی جدید اینها را بچیند این روش منطقش این است که هر جواب معادلش یک ***code***وجود دارد. این کد را حالا کد نمی‌گویند می‌گویند ***feasible sequence*** (توالی موجه) می‌گویند. ولی به همین سادگی می‌شود یک توالی موجه را تبدیل کرد به ***decode*** به این نحوه ***decode*** کردن می‌گویند ***SSGS (Serial Schedule generation Scheme)*** یعنی ***decode*** کردن ردیفی یا ***Serial*** ای یعنی به ترتیب یعنی طبق توالی تک تک می‌آوریم اولین جایی که می‌توانی آن را بچینی که به این ***SSGS*** می‌گویند. فقط یک نکته‌ای که الان وجود دارد از این لحظه به بعد به جایی اینکه بگویم جواب‌ها را بررسی کنید نمی‌گویم جواب‌ها می‌گویم ***Feasible sequence*** ها را بررسی کنید. اشکال که ندارد نه ، چون هر جواب همون .... الان شما این زمانبندی را چک کنید یا آن را (توالی) چک کنید فرقی نمی‌کند این همان است عین داستان دانشجو و کد دانشجویی می‌باشد.

فقط یک نکته ظریف اینجا است بین دانشجو و شماره دانشجو و یا کد ملی رابطه چی وجود دارد ؟ رابطه یک به یک ، منحصر به فرد وجود دارد یعنی این کد دانشجویی انحصاراً برای شما است یا این کد ملی انحصاراً برای شما است . اینجا چی ؟ آیا این به اصطلاح این کد و این زمانبندی رابطه‌شان یک به یک است ؟ اینجا را خوب دقت کنید من یک کد دیگه بنویسیم. این یک کد دیگری است که بالایی که نوشته شده است فرق می‌کند.

۱ ۲ ۵ ۳ ۴ **۸** **٦** **۷** ۹ ۱۰

پس این سه تایی را که فرق می‌کند که در اسلاید هم با یک رنگ دیگر است . خوب حال این را هم ***decode*** می‌کنیم.

یک که مجازی است هیچی . ۲، ۵، ۳، ۴ هم که عین قبلی است

بعدی ۸ است . چرا ۸ آنجا رفت اینجا که جا هست چرا رفته آن بالا. چون طبق شکل ۸ پیش‌نیازی اش ۳ است اولین جایی که منبع باشد نیست باید منبع باشد و پیش‌نیازی‌اش هم حل شود اینجا به خاطر اینکه پیش‌نیازی‌اش ۳ است رفته آنجا اگر ۳ پیش نیازی اش نبود ۸ را من اینجا می‌چیدم که منبع داریم. ولی علیرغم داشتن منبع پیش‌نیازی باید رعایت شود. بعدی ۶ است و بعدی ۷، ۹ ، و در آخر ۱۰ است.

این جواب همان است دیگر. نتیجه : منحصر به فرد نیست. ۲ تا کروموزن (کلمه‌ها را عوض می کنم) کروموزم هم می‌توانید بگویید (۲ تا رشته یا ۲ تا کروموزم یا ۲ تا کد یا ۲ تا توالی) متفاوت است ولی وقتی که ***decode*** کردید یک جواب به دست آمد.

خوبی یکی دیگر را هم ببینید این با قبلی چه فرقی دارد ؟ این بار به جای ۲ و ۵ نوشتم ۵ و ۲

(۱۰ ۹ ۸ ۷ ۶ ۴ ۳ ۲ ۵ ۱) حال ***decode*** را انجام دهید.

یک که هیچی، بعدی ۵ است بعدی ۲ را چیدم نتیجه بازم همان شد نتیجه باز هم همان شد.

۳ تا کد متفاوت و یک جواب. نتیجه : جمله‌ام را اصلاح می‌می‌کنم نمی‌گم تمام ***feasible sequence*** ها را بررسی کنید بلکه چی باید بگم تمام ***feasible sequence*** هایی را بررسی کنید که جواب‌هایی متفاوتی بدهند مثلاً این ۳ تا یک جواب دارند پس یکی هستند، یک نتیجه تکراری هستند. یکی از این ۳ تا را که بررسی کردید می‌توانید دو تای دیگری را بررسی نکنید. بررسی نکردن دو مورد دیگر جز لم‌هایی است که دیگه چرا بررسی نکنیم. چون تکراری است محرز باید بشود که تکراری است که بررسی نکنیم. از این سه تا محرز است که تکراری است چون من هر سه تا را ***decode*** کردم پس اگر من در یک شاخه‌ای مثلاً این جواب اول را بررسی کردم شاخه‌های دیگر که با این جواب یا این جواب مواجه شدم دیگر آن شاخه را ادامه نخواهم داد. چون تکراری است پس این در واقع مکانیزم بستن شاخه‌ها برای تکراری است از این ۳ تا جواب فقط یکی از آنها باید بررسی شود.

حذف ~~۱ ۵ ۲ ۳ ۴ ۶ ۷ ۸ ۹ ۱۰~~

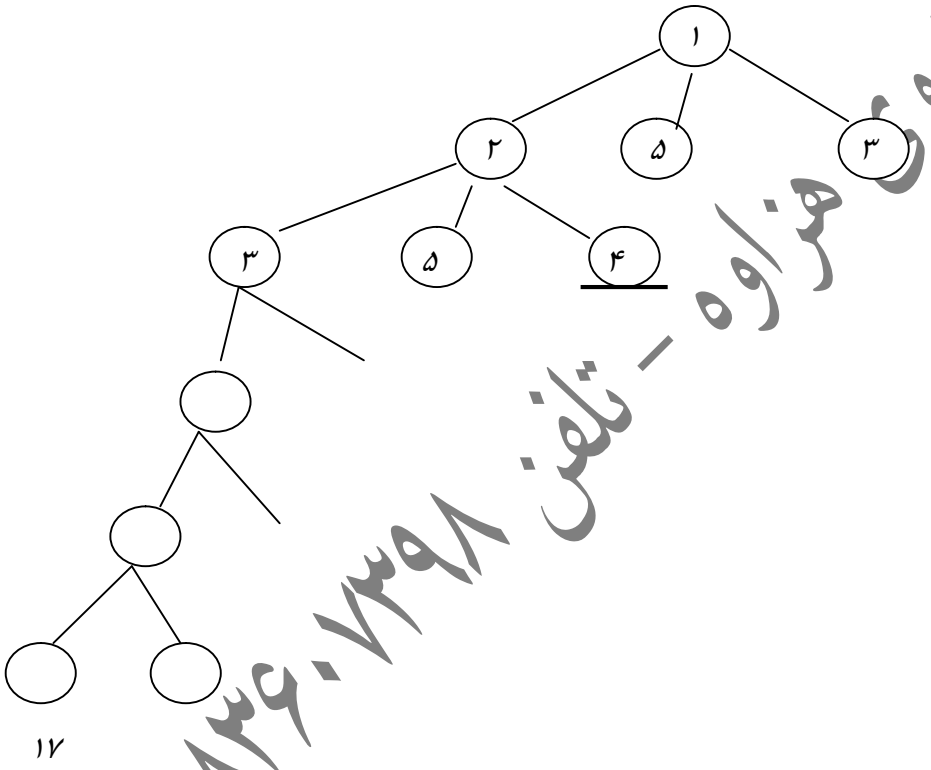
۱ ۲ ۵ ۳ ۴ ۶ ۷ ۸ ۹ ۱۰

حذف ~~۱ ۲ ۵ ۳ ۴ ۸ ۶ ۷ ۹ ۱۰~~

یک توالی دیگر بگویید. اصلاً به طور کل در اینجا ، چند تا جواب که موجه باشد وجود دارد. حالا باید یک خورده هوشمنداش کنیم جلوتر که برویم خواهیم دید که یک مقدار این‌ها را قانونمندش کرد فرمولی‌اش کرد. الان شما از کجا فهمیدید که اینها تکراری هستند. ***Decode*** کردیم یعنی باز کردیم شکل را رسم کردیم و از روی شکل داریم می‌گوییم. ولی قرار نیست که از روی شکل گفته شود.

قرار است منطقی‌اش کنیم فرمول محورش کنیم. اینها الان جلوتر فرمول‌هاش را می‌سازیم نه فرمول ریاضی منظور فرمول مفهومی است. شما لطفاً یک توالی دیگر را بگویید . غیر از این ۳ تا بگویید. آیا می شود با یک شروع نکنید ؟ خیر نمی شود. قطعاً همه توالی‌ها با یک شروع می‌شود بعد از یک قطعاً کدام می‌آید یا ۲ را خواهید گفت یا ۵ یا ۳ را. امکان ندارد بعد از یک غیر از این ۳ تا چیز دیگری بگویید به خاطر اینکه اگر بگویید دیگر موجه نمی‌شود. من دارم فضا را آماده می کنم برای شاخه‌زنی یعنی شاخه‌زنی اینجا خیلی کار ساده‌ای است میگه تمام این توالی‌ها با یک شروع می‌شوند حالا تمام جواب‌هایی که شما دارید و با یک شروع شدند بعدی شان یا با ۲ یا با ۵ یا با ۳ است حالا فرض کنید شما این شاخه را به طور مثال ادامه می‌دهید. بعد ۲ کدام است ؟ ۱ و ۲ را پاک کنید یا با ۳ یا با ۵ با ۴ است و به همین صورت تا آخر (این شاخه‌زنی) را ادامه پیدا می‌کند. یعنی تمام جواب‌ها ، گروه اول که نوشتم یک آن تمام جواب‌های شما هستند همه با یک شروع شدند ولی بعد از این جواب‌ها سه قسمت می‌شوند که بعد از یک ۳ آمده است. یک فکر کنید یک سری آنهایی هستند که بعد از یک ، ۲ آمده است یا یک سری آنهایی هستند که بعد از یک ، ۵

آمده است و یک سری آنهایی هستند که بعد از یک، ۳ آمده است، فکر کنید جوابی را احتمال دارد گم کرده باشید یعنی جوابی هست که از بین رفته باشد. نه دیگه. چون بقیه جواب‌هایی که من به اصطلاح بررسی‌شان نکردم آنها موجه نیستند بعد از یک ، ۴ بیاید چی ؟ این موجه نیست و من به خاطر همین ننوشتم چون اگر موجه هستند همه را دارم می‌نویسم. حالا اگر شما شاخه ۱ و ۲ را پیش بروید بعد از ۱ و ۲ چی می‌تواند بیاید ؟ نوشتم ۳ ، ۵ ، ۴ جز این هم می‌تواند باشد یکم فکر کنید ؟ نه . پس توی این نحوه تقسیم‌بندی هیچ جوابی از بین نمی‌رود. تکرار نه ، ممکن است تکراری باشد این سه تا مگه تکراری نیستند الان این ۳ تا تکراری هستند. در حالی که یکی‌شان تو کدام شاخه است؟ ببینید یک‌شون در این شاخه است اما دو تای دیگه در این شاخه‌اند. یعنی یکی از جواب‌های این شاخه با دو تا جواب‌های شاخه دیگر تکراری اند که من باید جلوتر کشفش کنم که آن را ببندیم. پس ممکن است تکراری باشد پس مدعی نیستم که این جواب تکراری نیستند ممکن است تکراری هم باشد ولی مطمئناً جوابی از بین نرفته این نحوه شاخه‌زنی در این مسئله است.



۱۷

خوب این شاخه زنی را که انجام دادید کدام شاخه‌ها بسته می‌شوند ؟ سه دلیل آن هفته نوشتید. تکراری ناموجه اینجا نداریم چرا ناموجه اینجا نداریم ؟ چون اصلاً ناموجه‌ها را بازی ندادیم که بخواهیم ببندیم مثلاً چرا ۱ و ۶ را موقع شاخه زنی ننوشتید. چون ناموجه بود اصلاً ناموجه را از ریشه قطع کردیم اصلاً نیاوردیم که بخواهیم ببندیمشون بنابراین اینجا از آن سه تا دلیل ۲ تا وجود دارد. تکراری و ***lower bound*** (مغلوب‌ها). مکانیزم شاخه زنی این روش اینطوری است که اصلاً ناموجه ای وجود ندارد که شما بخواهید ببندیش . اصلاً آنها را در حساب کتاب نیاوردید خوب این باعث می‌شود که کار یکم راحت‌تر شود. خوب حالا کدام‌ها تکراری هستند که باید بسته بشوند؟ یا کدام به اصطلاح مغلوب هستند که باید بسته شوند.

از سه تا دلیل چند تاش را داریم ؟ دو تا ، ناموجه‌اش را نداریم . تکراری و ***lower bound*** ولی آن هفته گفتم که تکراری و ***lower bound*** و ناموجه هر کدام یک مجموعه است یعنی شما ممکن است برای تکراری بودن ۱۰ تا دلیل بنویسید و برای ناموجه بودن ۷ دلیل بنویسید. این روش ، روش آقای پیترسون برای تکراری‌ها دو تا دلیل و برای lower bound ها هم دو تا دلیل در واقع اثبات می‌کند و با ۴ تا دلیل شاخه‌ها را می‌بندد. پس الان می‌خواهم ۴ تا دلیل بگم برای بستن شاخه‌ها ، دو تا تکراری بودن‌ها را حذف می‌کند ، دو تا هم lower bound ایی را. حالا اینجا بخواهم به ترتیبی همین برم اول ***lower bound*** ها را خواهم گفت.

کی یک شاخه به دلیل lower bound بسته می‌شود.

پاراگراف اول ، اول تکراری‌ها گفته می‌شود . آن می‌تواند یک کار خوبی باشد که سعی کنیم بشماریم فقط ***feasible sequence*** هایی که منجر به جواب هایی متفاوت می‌شوند یعنی لازم نیست هر سه تا ***feasible sequence*** ها را چک کنید یکی‌اش کافی است چون ۳ تا جواب یکسان می‌دهند.

پاراگراف دوم : اولین دلیل از آن چهار دلیل که باعث بسته شدن شاخه می‌باشد است که فقط جزء تکراری‌ها هستند دو تای اول جزء تکراری‌ها است و دو تای بعدی lower bound است . ما می‌توانیم حذف کنیم همه ***feasible sequenc*** هایی که منجر بشوند به یک ***ESS*** که یک فعالیت در توالی شروع شده است زودتر از فعالیت قبلی در توالی . این توالی را مطمئن باشید که تکراری است باید پاکش کنید. چرا باید پاکش کنم ؟ می‌گه توی فعالیت، توی توالی شما ۸ زودتر آمده است یا ۶ ؟ خوب می‌بینید ۸ ولی وقتی که ***decode*** می‌کنید چه اتفاقی می‌افتد ؟ ۸ ایی که در توالی زودتر از ۶ است ولی موقع چیده شدن برعکس شده است یعنی ۸ ایی که زودتر بوده دیرتر شروع شده و ۶ ایی که بعداً اومده زودتر به همین سادگی می‌گه که شما این توالی را دیگه بررسی نکنید چون تکراری است هر توالی که فعالیت زودتر بیاد ولی دیرتر شروع شود تکراری است با چی تکراری است اگر شما الان جای ۸ و ۶ را عوض کنید چی می‌شود ؟ توالی باز درست است ولی خروجی‌اش باز دوباره همین است و آن حذف نخواهد شد ولی این یکی که به اصطلاح مشابه‌اش که همین است دارین حذف می‌کنید. اگر در یک توالی فعالیت ***i*** قبل از فعالیت ***j*** آمده باشد ولی در زمانبندی ***ESS***، ***j*** قبل از ***i*** شروع شود این توالی تکراری است و این شاخه بسته می‌شود. یعنی شاخه‌ای که اینطوری شده است را دیگه ادامه نمی‌دهم. این دلیل اول بود.

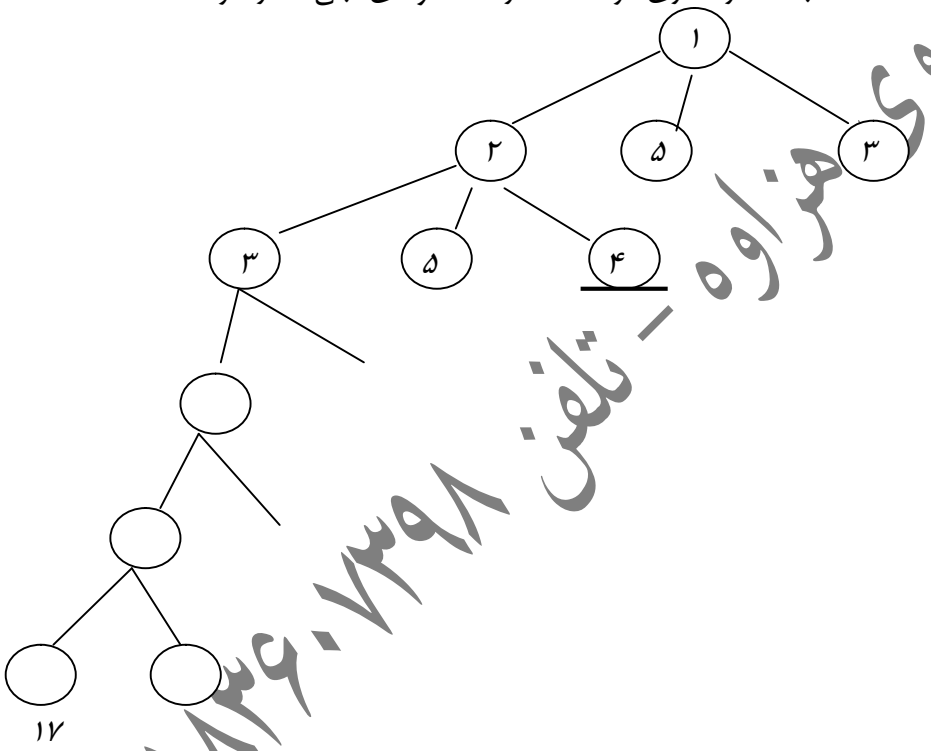
دلیل دوم : این چرا بسته می‌شود ؟ اینم دیدیم دیگه به عنوان مثال دیدم که تکراری بوده است ولی خوب می‌خواهیم به اصطلاح این را فرمولی‌اش بکنیم قانون مندش کنیم. چرا این تکراری است ؟ ***(۱۰ ۹ ۸ ۷ ۶ ۴ ۳ ۲ ۵ ۱)*** تا آخر چیدم چرا این تکراری است اگر شما جای ۲ و ۵ را عوض می‌کردید، می‌شد ۵ و ۲ ولی وقتی ***decode*** کنیم باز همان می‌شود. دیدیم قبلاً که باز همین بدست می‌اید میگه اگر دو تا فعالیت ***i,j*** داشته باشید که با هم شروع بشوند خوب شما بگید ۱***,(i,j)*** یا بگید ۱***,(j,i)*** فرقی نمی‌کند خروجی چون با هم شروع می شوند نتیجه خوب وقتی این دو تا یک جواب می‌دهند یکی‌اش تکراری است حالا کدام تکراری است ***(i,j)*** تکراری است یا ***(j,i)*** . فرقی نمی‌کند اگر ***(i,j)*** را نگه داشته‌اید پس اون یکی را پاک کن و اگر ***(j,i)*** را نگه داشته‌اید آن یکی را پاک کن. در هر صورت یکی‌اش تکراری است حالا باید با هم توافق بکنیم یا بگیم ۲ و ۵ تکراری است یا بگیم ۵ و ۲ تکراری است حالا این متن اینطوری نوشته گفته که شماره به سمت زیاد شدن است اشکال ندارد و آنی که شمارش کم شده است (شمارش کاهشی بوده است) آن را

حذف کن. ولی اگر عکسش هم می‌گفت بازم اشکال نداشت تکراری کپی هم اند. حالا شما هر کدام را می‌خواهی بذار کنار این یا آن فرقی نمی‌کند مهم این است که یکی‌اش خلاصه بماند.

دلیل دوم : اگر در یک زمانبندی یا توالی $j<i$ باشد ، i و j همزمان شروع شوند و در توالی i قبل از j ظاهر شود این توالی تکراری است و شاخه مربوطه بسته می‌شود.

پس سه تا دلیل نوشتیم نه می‌دونم دومی خودش سه تا نکته داشت در دومی سه تا اگر نوشتید. اگر $i>j$ باشد همزمان هم شروع بشوند و i هم قبل از j بیاد شما دقت کنید که هر سه این اگرها باید باشد که شما شاخه را ببندید یعنی همین $i>j$ شاخه را نبندید یا همین که با هم مساوی شدند نبندید باید سه تا دلیل هر سه تا با هم اتفاق بیافتد توی متنی که نوشتید سه تا اگر نوشتید $i>j$ باشد و i,j با هم شروع بشوند و i هم قبل از j بیاید یعنی به این سه تا دلیل اگر همزمان بود آن شاخه بسته می‌شود خوب خلاصه شماره‌های ۱ و ۲ شاخه‌هایی که تکراری هستند را می‌بندد.

اما دو تا بعدی که در این اسلاید یکی‌اش را گفته پس در کل می‌شود شماره ۳ ولی چرا شماره یک زده است چون راجع به *lower bound* ها است جدا شماره‌گذاری کرده است در ادامه دو تای قبلی شماره نزده است.



این دیگه شاخه‌هایی را نمی‌بندد که تکراری هستند این شاخه‌هایی را می‌بندد که *lower bound* شان (از بهترین جوابی که در آرشیو داریم بیشتر است) من یک شاخه‌ای را تا آخر ادامه دادم مثلاً یک جوابی پیدا کردم که *lower bound* اش شده است به فرض ۱۷ یک شاخه را تا (یادتان می‌آید یک کلمه گفتم *fathom* شدن یا به عمق رسیدن) تا عمق ادامه دادم و شده ۱۷ این جواب چی می‌شود ؟ بایگانی می‌شود به عنوان بهترین جواب زمانبندی که من تا این لحظه پیدا کردم این یادتان باشد این البته عدد ۱۷ را همین‌طوری گفتم . انجام ندادم به فرض حالا اگر توی یک شاخه دیگری *lower bound* ایی پیدا کنید که بالای ۱۷ و یا ۱۷ باشد دیگه ادامه نمی‌دهیم حالا آن *lower bound* چطور حساب می‌شود.

دو تا روش برای محاسبه ***lower bound*** اینجا داریم :

مثلاً فرض کنید در این شاخه می‌خواهم ***lower bound*** حساب کنم چه جوری ***lower bound*** در این شاخه حساب می‌شود بعد من جمله‌اش را می‌گم می‌نویسید اضافه می‌کنیم چی را اضافه می‌کنیم ؟ باقیمانده طول مسیر بحرانی ***currently schedule activity*** یعنی فعالیتی که الان دارد زمانبندی می‌شود یعنی کدام ؟ اینجا یعنی ۴ . شما هر بار یک فعالیت داری اضافه می‌کنی به توالی دیگر آن فعالیت می‌شود ***currently schedule activity*** پس باقیمانده طول مسیر بحرانی فعالیتی که در حال حاضر دارد زمانبندی می‌شود را اضافه کنید به ***start time***اش اگر این مقدار یعنی جمع این دو تا (مسیر بحرانی با این حساب کنیم) تخطی کند یعنی بیشتر باشد یا مساوی باشد با ***currently best solution*** (بهترین جوابی که الان دارید) ما می‌توانیم این شاخه را ببندیم.

خوب من یک مثال دیگه بگم بعد جمله‌اش را بنویسید. کدام دارد اضافه می‌شود دقت کنید. ۴ دارد اضافه می‌شود. این یادتان باشد مسیر بحرانی ۴ چند روز است ؟ از روی شبکه حساب کنید یعنی چی مسیر بحرانی ۴ ؟ یعنی از ۴ به بعد چند روز دیگر این پروژه کار دارد ؟ خود ۴ را هم بگویید. ۴ و به بعد ۴ ، سه روز بعدی ۸ روز پس به طور کل می‌شود ۱۱ روز . مسیر بحرانی باقیمانده یعنی چی ؟ یعنی از آن فعالیت تا آخر پروژه طولانی‌ترین مسیر طولش چقدر است به این می‌گویند ***remaining critical path*** یا مسیر بحرانی باقیمانده طولش در این مثال ۱۱ است . این ۱۱ را با چند جمع کنم ؟ با ***start time*** اش ، ۴ کی دارد شروع می‌شود ۱۵=۴+۱۱ . ۱۵ از بهترین جوابی که شما دارید بیشتر است ؟ اگر بیشتر است یا مساوی است ادامه نده ولی اگر نه که باید ادامه دهید. این جا الان شاخه بسته نمی‌شود. پس من می‌خواستم نحوه محاسبه‌اش را بگم این فعالیت که داره چیده می‌شود ***remaining critical path*** اش چقدر طولانی است مقدارش ؟ آن به اضافه ***start time*** این درست شد. حالا این را بگویید که آیا این بسته می‌شود یا نه ؟ خوب ۸ کی داره شروع می شود ؟ ۹ ام . خودش هم چند روز است . ۸ رو پیدا کنید. ۶ روز . ۶ و ۹ می‌شود ۱۵ بازم در مقایسه با ۱۷ که اشکال نداره ادامه می‌دهیم.

۱۱ به اضافه ***start time*** اش قبلی ۴ بود. البته این مثال را کامل باید حل کنیم من فعلاً دارم لم‌هاش را می‌گم پس طول باقیمانده مسیر بحرانی فعالیتی که هم اکنون به توالی اضافه می‌شود را به زمان شروع آن اضافه می‌کنیم اگر مقدار حاصل از بهترین جواب به دست آمده بیشتر یا برابر باشد این شاخه مغلوب است و بسته می‌شود. پس این تکراری را نمی‌بست جواب‌هایی را می‌بست که نا امیدکننده بود. ممکن است تکراری هم نباشد ولی جوابش جواب خوبی نبود.

و دلیل آخر شماره ۴ و البته تو اسلاید شماره ۲ هم طولانی است و هم یکم سخت‌تر از ۳ تای قبلی است این دقت کنید این را بگم روش شاخه و کران تمام شده است. خوب دقت بفرمایید اول من با مثال توضیحی بگم بعد متنش را می‌گویم بنویسید. فرض کنید توالی زیر که روی شکل می‌گویم تا موجه باشد *(۶ ۳ ۴ ۲ ۱)* تا اینجا موجه است چک کنید کدام فعالیت شمارش کمتر است ولی هنوز در این توالی ظاهر نشده است ؟ آره ۵ . *۲ ۱)* *۶ ۳ ۴* ظاهر شده ولی ۵ هنوز در توالی ظاهر نشده است شمارش کمتر است ولی هنوز نیامده است خوب

حالا یکم فکر کنید اگر ۵ الان بخواهد اضافه بشود آیا می‌توان همزمان با ۶ شروع بشود یا زودتر از ۶ شروع بشود یا نه ؟ دو تا سوال پرسیدم اول همزمان آیا ۵ می‌تواند با ۶ همزمان شروع بشود اگر بخواهم الان به توالی اضافه‌اش بکنم آیا ۵ می‌تواند الان به توالی اضافه بشود موجه است اگر اضافه بشود و همزمان با ۶ شروع بشود مشکلی ندارد اصلاً به منبع کار نداشته باشید دلیل‌های ۱ و ۲ را فقط نوشتید دلیل ۲ را یکبار بخوانید.

اگر در یک توالی یا زمانبندی $i>j$ باشد و i,j همزمان شروع بشوند در توالی i قبل از j ظاهر شود این توالی تکراری است و شاخه مربوطه بسته می‌شود.

خوب نتیجه : پس نمی‌تواند با هم شروع بشوند قرار ما این شد که وقتی یک فعالیتی شمارش کوچکتر و دیرتر می‌آید نمی‌تواند با این همزمان شروع شود اگر همزمان شروع بشود تکراری می‌شود این که یادتان نرفته است بین (۲ ۵ ۱) و (۵ ۲ ۱) قرار شد شما (۲ ۵ ۱) را بازی ندهید و ببندید و بگویید که این تکراری است چرا تکراریه ؟ چون i یعنی ۵ قبل از ۲ یعنی j آمده است شمارش هم بزرگتر است با هم، هم شروع شده‌اند و اگر هم با هم شروع بشوند تکراریه نتیجه اینجا هم همین است دیگر ۶ که قبل از ۵ آمده است خوب شماره ۶ هم که از ۵ بیشتر است اگر بخواهند با هم شروع بشوند پس با هم نمی‌توانند شروع بشوند. حالا زودتر چی ؟ زودتر می‌توانند شروع بشوند ؟ ۵ بعد از ۶ بیاید ولی زودتر شروع بشود؟ آیا فعالیت ۵ می‌تواند بعد از ۶ بیاد ولی زودتر شروع بشود ؟ این اسلاید یادتان رفت ؟ آیا ۶ می‌تواند بعد ۸ بیاد ولی زودتر شروع بشود اینم تکراری است یعنی فعالیتی که شمارش کوچکتر است بعد از یک فعالیتی می‌آید شمارش کوچکتر است ولی دیر می‌آید در توالی. این نمی‌تواند با فعالیت قبلی نه همزمان و نه زودتر شروع بشود. هر دو تکراری است پس مجبور است کی شروع بشود همزمان که نه زودتر هم که نه. تنها می‌ماند دیرتر. باید دیرتر شروع بشود چند روز دیرتر ؟ حداقل یک روز باید دیرتر شروع بشود حالا ***lower bound*** را چطوری می‌شود حساب کرد الان کدام داره اضافه می‌شود ؟ باقیمانده مسیر بحرانی ۵ چند روز است ؟ ۱۰ روز اینم ۱۰ ماکسیمم می‌شود ۱۰ باقیمانده مسیر بحرانی ۵، ۱۰ روز است. این ۱۰ رابه اضافه ***start time*** ، ۶ می‌کنیم حالا نمی‌دانم چندام است مثلاً فرض کنید ۷ ام است به اضافه عدد یک این عدد یک به خاطر چیست ؟ یک روز حداقل باید بعد از آن شروع شود. ۱۸=۱+۷+۱۰ اگر این عدد از بهترین جوابی که دارید بیشتر است دیگر ادامه نمی‌دهید.

پس جمله‌اش را با مثال صحبت کردیم اینجا متنش خیلی سخت است یکبار هم از روی متن بخوانیم بعد جمله‌اش را بگم بنویسید. هر فعالیت زمانبندی نشده با شماره کوچکتر یعنی کدام ؟ (۵) . در توالی نیامده و شمارش هم کوچکتر است باید داشته باشد که یک زمان شروعی که حداقل یک واحد زمان (حداقل یک روز) بیشتر از فعالیتی که زمانبندی شده و شمارش هم بالاتر است یعنی (۶) . چرا باید حداقل یک روز بیشتر باشد ؟ چون اگر بخواهد همزمان یا زودتر باشد تکراری می‌شود پس یک روز باید دیرتر شروع شود بنابراین یک ***lower bound*** برای این اینطوری می‌تواند حساب شود که چی کار کنید ؟ ***start time*** فعالیتی را که الان زمانبندی کردید را به اضافه ۱ کنید به اضافه باقیمانده مسیر بحرانی ۵ کنید اگر این نتیجه از بهترین جوابی که دارید بیشتر یا مساوی شد آن می‌تواند ***eliminate*** یا حذف شود.

اگر فعالیتی مانند i هنوز در توالی نیامده است در حالی که فعالیت j بزرگتر از i در توالی آمده است در این صورت i حداقل باید یک روز بعد از j شروع شود (قبل یا همزمان تکراری است) لذا می‌توان زمان شروع j به اضافه یک به

اضافه باقیمانده مسیر بحرانی *i* را به عنوان یک *lower bound Lb* در نظر گرفت که اگر *Lb* محاسبه شده از بهترین جواب موجود تا این لحظه بیشتر یا برابر باشد این شاخه بسته می‌شود.

۱ ۲ ۴ ۳ ۶ ۵

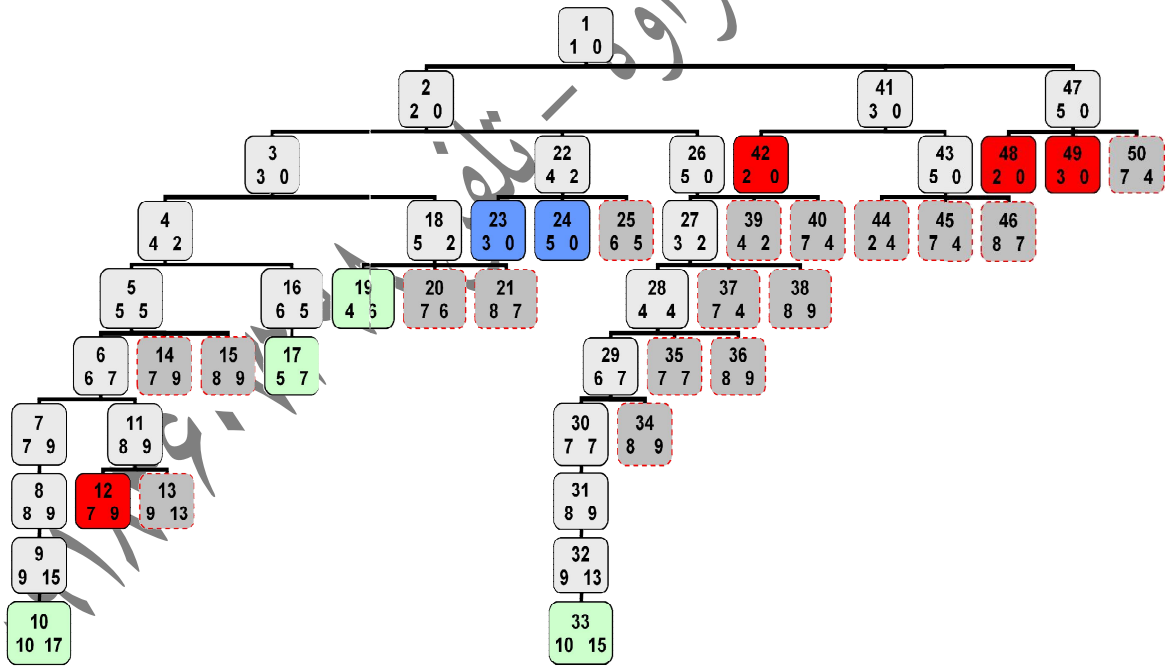
۱۰+۷+۱=۱۸

۱۸ بیشتر از ۱۷ است پس ادامه نمی‌دهیم.

مثال : به این چی می‌گویند ؟

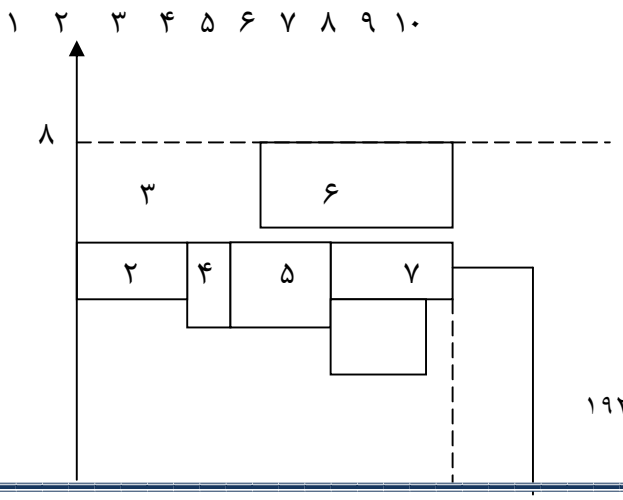
Branch and bound tree (درخت شاخه و کران)

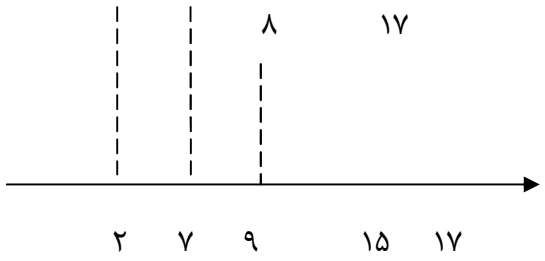
یعنی به اصطلاح ساختار این که شاخه می‌زنیم و می‌بندیم خروجی آخرش را *branch and bound tree* (درخت شاخه و کران) می‌گویند. به هر یک از این مربع‌ها یک گره *(node)* می‌گویند. در هر گره یک سری عدد نوشته شده است این عددها چی هستند ؟ قطعاً باید وقتی عددها زیاد هستند شما یک راهنما کنارش ببینید که کنارش گفته است این اعداد چه معنی دارند عدد بالایی شماره گره *(node number)* است در پایین دو عدد وجود دارد عدد سمت چپ شماره فعالیت *(activity number)* است و عدد سمت راست زمان شروع *(start time)* است مثلاً بخواهیم یکی‌اش را بگوییم. در گره شماره ۳۲ فعالیت ۹ روز ۱۳ ام شروع شده است ۹ به توالی اضافه شده و روز ۱۳ ام هم شروع شده است و الی آخر



Tree به آن می گویند ریشه *tree* یا *root* . ریشه *tree* همیشه گره شماره یک می‌باشد. در گره شماره یک کدام فعالیت به توالی اضافه می‌شود با هم صحبت کردیم همه *sequence* ها با فعالیت یک شروع شود فعالیت یک مجازی است پس اینجا در توالی اضافه می‌شود بعدش هم کجا شروع می‌شود و یک مجازی است دیگر *ESS*=۰ است این دقیقاً همین چیزی بود که من اینجا شروع کردم خوبش *detail* اش را ننوشته بودم کجا شروع می شود صفر، *ESS* بعد از یک کدام می‌آید یا ۲ یا ۵ یا ۳، بقیه موجه نیستند یا باید ۲ بیاید یا ۵ یا ۳. پس شما می‌بینید که به سه شاخه ۲، ۳، ۵ تقسیم شد. به هر حال هر کدام که اضافه بشوند باید کی شروع بشوند ؟ صفرم. سه هم صفرم ، پنج هم

صفرم. کدام شاخه اول بررسی شده است ؟ آن هفته ۲ تا استراتژی داشتیم ***depth first*** و ***best first*** . این روش ***depth first*** است یعنی همین اولی را باید تا آخر دنبال کنید تا به عمق برسد یعنی شاخه که اول باز می‌شود باید تا آخر ادامه بدهم پس اینجا الان چی کار می‌کند ؟ بعد از یک دو را می‌چیند بعد از ۲ می‌تواند ۳، ۴ ، ۵ بیاید. خوب به ترتیب، چون باید به ترتیب پیش برود اول کدام را انجام می‌دهد ؟ (۳). خوب ۳ را دوباره در زمان صفر قرار می‌دهد.
من یک شاخه را که به عمق رفتم تا آخر انجام دادم پس اینجا ۸، یک را که چیدم هیچی، صفر است بعد از ۱ ، ۲ آمد . ۲ کی شروع شد ؟ صفرم کی تمام شد ؟ ۲ ام . بعد از ۲، الان ۳ را اضافه می‌کنم ، ۳، ۷ روز است و ۳ تا منبع می‌خواهد پس فعالیت ۳، با ۳ تا منبع ۷ روز. روز صفرام شروع می‌شود و ۷ ام تمام می شود پس یک که هیچی بعد ۲ و بعد ۳ را چیدم. بعد از ۳ می‌تواند ۴ یا ۵ طبق پیش‌نیازی بیاید. چون من به صورت ***depth first*** حل می کنم اول کدام را چیدم ؟ ۴ اما اگر در شکل دقت کنید ۴ پیش نیازش ۲ است پس ۴ باید بعد از ۲ شروع شود ۲ کی تمام شده است ؟ ۲ ام. ۴ باید بعد از ۲ تمام بشود یعنی ۲ ام . چند روز است ؟ ۳ روز اگر ۳ روز است که می‌شود تا ۵ ام. ۴ تا منبع دارد. پس ۴ از ۲ ام تا ۵ ام با ۴ تا منبع دارد بعد از ۴ ؟ الان ***(۴ ۳ ۲ ۱)*** را چیدم شما وقتی آن را چیدید روی شکل به طور فرضی آنها را خط بزنید اینجا چیده شده اند باید از روی بقیه بگویید که کدام می‌تواند اضافه شود با رعایت پیش‌نیازی چون شکل را دیگه ندارم نمی‌خواهم هر بار جابه‌جا شود ***(۴ ۳ ۲ ۱)*** را چیدم الان کدام می‌تواند اضافه بشوند ***(۶,۵)*** . ***(۷,۸)*** را نباید بگویید چون هنوز ۵ را نگفته‌اید نباید ۷ و ۸ را بگویید. فعلاً ۵ و ۶ پس دو شاخه می‌شود که اول ۵ را انجام می دهم که خود ۵ هم ۴ روز است و ۴ تا منبع دارد بین ۵ و ۶ که ۵ داره اضافه می‌شود ۵ کی باید شروع شود منبع کجا داریم بعد از ۴. ۵ ام تا ۹ام فعالیت ۵ گذاشته می‌شود. خوب حالا بعد از ۵، ۵ که انجام شد دیگه الان ۷ و ۸ هم آزاد شدند پس ۶، بعد ۷، بعد ۸ که ظاهراً هم اول ۶ باید انجام بدهیم چون ***depth first*** ۶، ۷ ام شروع می‌شود چند روز است ۶ ؟ ۸ روز یعنی میشه از ۷ ام تا ۱۵ ام خوب بعد از ۶ هم ۷ و هم ۸ . حالا من ۷ رو زودتر انجام دادم . ۷ چند روز است ؟ ۶ روز. چند تا منبع می‌خواهد ؟۲ تا اینم از ۷ . بعد از ۷ ، ۸ می‌باشد یعنی فقط یک شاخه داریم آن چند روز است ؟ ۴ روز میشه تا ۱۳ ام . چند تا منبع می‌خواهد ؟ ۳ تا و بعد از ۸ ، ۹ می‌باشد ۹ چند تا منبع می‌خواهد؟ ۴ تا اینجا ۲ تا داریم پس مجبور ۱۵ شروع بشود بعدی ۱۰ می‌باشد و چون مجازی است در واقع بعد از پایانش می‌شود ۱۷. این ۱۷ چی می‌شود ؟





۱۷ ⟵ *best solution* و مبنای مقایسه *Lower bound*

این *best solution* ایی است که ما تا این لحظه داریم این میشه همان مبنای مقایسه *lower bound* ها این را به اصطلاح فقط ذخیره می‌کنیم در ادامه دنبال جوابی بهتر از ۱۷ هستیم پس این شاخه‌ای که ما به عمق رسیدیم یک زمانبندی به این شکلی را می‌توانید براش مستقل بشید. حالا بقیه شاخه‌ها ، شاخه بعدی باید کدام شکل باشد. ببینم می‌توانید بگویید الان این بسته شد بعدی کدام است ؟ آن هفته گفتم آخرین شاخه‌ای که رها کردید کدام شاخه بوده است ؟ همین جا است . ۱۰ و ۱۱ این را باز کرده و ۱۳ و ۱۲ بعد این را باز کرد ۱۶ و ۱۵ و ۱۴ یعنی از این ور شاخه‌ها باید ببندید . تا به انتها برسیم آخرین شاخه‌ای که بررسی شده اینجا شماره خود ۵۰ است. اینجا ۵۰ تا گره را بررسی کرده است. این شاخه‌ها دیگه کجا به عمق رسیده است؟ یعنی بعد از این ۱۷ دفعه ، دفعه بعد شاخه‌ها دیگه به عمق نرسیدند همه بسته شدند فقط شاخه ۳۳ بسته نشده است و به عمق رسیده است که چند شده است ؟ آن ۱۷ را از آرشیو پاک کنید این توالی از آن بهتر است پس این می‌آید به جای آن می‌نشیند از این به بعد بهتر از ۱۵ را می‌خواهم پیدا کنم که دیگه ظاهراً هیچ وقت همچین جوابی پیدا نشده است پس خلاصه این در واقع *first visible schedule* بود ولی در گره ۳۳ *optimal schedule* به دست آمد یعنی کدام توالی بهینه است توالی‌اش را بگویید.

(۱۰ ۹ ۸ ۷ ۶ ۴ ۳ ۵ ۲ ۱*)* این توالی بهینه است . بقیه شاخه‌ها چرا بسته شدند به یکی از ۴ دلیل.

دو تا تکراری بود، دو تا *Lower bound* . اینهایی که با رنگ قرمز و آبی مشخص‌اند اینها تکراری هستند آنی که با آبی مشخص است دلیل شماره یکی که نوشتید؟ چی بود ؟ اگر فعالیت *i* در توالی قبل از *j* بیاید ولی در زمانبندی دیرتر شروع بشود تکراری است در توالی زودتر بیاد ولی در زمانبندی دیرتر شروع شود تکراری است خوب حالا اینجا بگویید گره‌های ۲۳ و ۲۴ چه طوری بسته شدند از همین جا بگویید به شکل نیازی نیست از روی این بگویید. (گره ۲۳ و ۲۴ آبی هستند)

۴ زودتر آمده است کی شروع شده است ؟ ۲ ام اما ۳ یا ۵ ایی که بعداً اضافه شدند کی شروع شدند ؟ صفرام خوب این تکراری است دیگر فعالیت ۳ بعد از ۴ آمده است ولی وقتی می‌خواهد شروع شود زودتر شروع شده است این توالی تکراری است این سمت هم ۵ اتفاق افتاده است. ۵ بعد از ۴ آمده است ولی این ۲ ام است و این صفرام پس زودتر شروع شده است در توالی دیرتر آمده است یعنی به نوبت باید شروع بشوند. فعالیتی که بعداً اضافه شده باید بعداً شروع بشود نه زودتر زودتر اگر بشود تکراری است این لم یک بود.

اینهایی که با قرمز بسته شدند با لم ۲ بسته شدند. تکراری ۲. درست می‌گم یا نه ؟ آره. چی بود لم ۲ ؟ اگر *i>j* باشد و همزمان باشد و در توالی هم *i* قبل از *j* شروع شود. یعنی *i* شمارش بیشتر از *j* باشد در توالی هم زودتر آمده است و با هم شروع شده است این ۳ تا *if* اگر با هم باشند آن شاخه بسته می‌شود مثلاً در این جا گره ۱۲ را ببینید *i* یعنی ۸ و *j*

یعنی ۷ است. *i* از *j* بزرگتر است *i* و *j* با هم شروع شدند. یعنی ۹ ام. در حالی که *i* زودتر از *j* آمد. ۸ قبل از ۷ آمد شمارش هم بزرگتر است ولی با هم شروع شدند این توالی تکراری است گره ۴۲ هم همین اتفاق افتاده است. ۳، صفرم ، ۲ ، صفرم پس ۳ شمارش بزرگتر است و زودتر آمده است ولی با هم شروع شدند با ۲ و ۵ هر دو صفرم. اشتباه است چون ۵ از ۲ بزرگتر است زودتر آمده است ولی با هم شروع شده است پس اینها خیلی راحت قابل بحث است. (گره‌های ۱۲، ۴۲ و ۴۸ و ۴۹ قرمز هستند) دوتایی که با حالا به اصطلاح رنگ فسفریی مشخص هستند گره های ۱۷ و ۱۹ . (۳۳ و ۱۰ نیستند چون به عمق رسیدند بسته نشدند). گره‌های ۱۷ و ۱۹ به دلیل سوم بسته شدند. در واقع ***lower bound*** اول در کل دلیل سوم. چی بود دلیل سوم ؟ باقیمانده مسیر بحرانی فعالیتی که الان اضافه شد را به زمان شروعش اضافه کنید اگر این از ***lowr bound*** ایی که دارید بیشتر یا برابر باشد ادامه ندهید. اگر خوب توجه کنید در شماره ۱۷ کدام دارد اضافه می‌شود ؟ ۵ . اگر باقیمانده مسیر بحرانی ۵ را یادتان بیاد ۱۰ روز می‌شود ۱۰ را با چی جمع کنید ۱۷=۷+۱۰ (آن یک روز اضافه برای دلیل آخری بود.) ۱۷ . گفتیم برابر یا بیشتر باشد ادامه ندهیم. این برابر است دیگر ادامه ندهیم. گره ۱۹، ۴ روز است و ۶ ام بسته شده است باقیمانده مسیر بحرانی ۴ را بگید ۱۷=۶+۱۱ بازم همین طور این شاخه بسته شد.(۳۳ و ۱۰ چون به عمق رسیدند اینها بسته نشده‌اند) (گره‌های ۱۷ و ۱۹ فسفری هستند).
اما بقیه که به رنگ طوسی مشخص است. ۱،۲،۳،۴،۵،۶،۷،۸،۹،۱۰،۱۱،۱۲،۱۳،۱۴،۱۵،۱۶،۱۷،– ۱۷ بار بسته شده است. دلیل آخر یعنی بهترین دلیل به نظر کدام است ؟ دلیل آخر. که ۱۷ بار باعث بستن شاخه‌ها شده است دلیل آخر چی بود ؟ فعالیتی که شمارش کمتر است ولی در توالی هنوز نیامده است آن اگر الان به توالی اضافه شود از فعالیتی که قبلاً شمارش بیشتر بوده و آمده نمی‌تواند هم زمان یا زودتر شروع بشود حداقل یک روز دیرتر باید شروع شود پس مسیر بحرانی به اضافه ۱ و به اضافه ***start time*** آن می‌تواند یک ***Lower bound*** خوب باشد این ۱۷ بار ***run*** شده است ولی من ناچار هستم تنها به یک مورد اکتفا کنم.
مثلاً گره ۳۸ را می‌گم چرا گره ۳۸ ؟ فرقی نمی‌کند. چرا گره ۳۸ بسته شده است؟ این تا الان کدام‌ها را نوشتید در این شاخه *(۸ ۳ ۵ ۲ ۱)* کدام فعالیت شمارش کمتر است ولی هنوز در توالی نیست. مثلاً ۴ هم ۴ و هم ۶ و هم ۷ را می‌توانید بگویید. من ۴ را می گویم آن وقت ۴ مسیر بحرانی‌اش باقیمانده چند روز است ؟ ۱۱ روز. ۱۱ را با یک جمع کنید که می شود ۱۲حال ۱۲ را با ***start time*** ، ۸ جمع کنید. ۲۱=۹+۱۲ خوب بهترین جوابی که دارید با این ۲۱ مقایسه کنید. ۲۱ از ۱۵ بیشتر است . حالا چرا با ۱۵ مقایسه کردید چرا با ۱۷ مقایسه نکردید. چون از گره ۳۳ به بعد با ۱۷ کار ندارید مبنا ۱۵ می‌باشد قبل از ۳۳ ، مقایسه با ۱۷ بود ولی بعد از ۳۳ ، ۱۷ به روز شده و ۱۵ شده باید با ۱۵ سنجیده شود به همین سادگی بسته شد. حال یکی را خودتان بگویید گره ۵۰ چرا بسته شده است؟ چک کنید و بگویید چرا گره ۵۰ بسته شده است.

۱ ۲ ۵ ۳ ۸ ﷺ

۱۱++۹=۲۱>۱۵

همین کار که من انجام دادم توالی‌اش را بنویسید و ببینید کدام شمارش کمتر است هنوز نیامده، مسیر بحرانی‌اش چقدر است ؟ گره ۵۰. *(۷ ۵ ۱)* آمدند کدام‌ها نیامدند هم ۲ ، هم ۳، هم ۴ و هم ۶ یکی از اینها کار کنه کافی است ۲ چقدر است مسیر بحرانی ۱۳ است. ۱۴=۱+۱۳ و ۱۸=۴+۱۴ ، ۱۸ هم بالای ۱۵ است حال فرض کنید بالای ۱۵

نمی‌شد مثلاً می شد ۱۴ یعنی بسته نمی‌شد یعنی با ۲ چک کردیم بسته نشد چی می‌شود بسته نمی‌شود خوب با بعدی چک کنید یعنی ممکن است با یکی از اینها همه کار نکند اشکال نداره با بعدی اگر با بعدی هم کار نمی‌کرد با بعدی اگر با هیچ کدام کار نکرد پس دیگر اصلاً نباید ببندی اما با یکی‌اش که بسته شد کافی است.

الان با ۲ بسته شد دیگه اصلاً بقیه مهم نیست هر چند با بقیه هم بسته می شود مثلاًبا ۴ چک کنید اگر ۴ ، ۱۱ بود باقیمانده مسیر بحرانی ۱۲=۱+۱۱ و ۱۶=۴+۱۲ . بازم ۱۶ بالای ۱۵ است بازم بسته شد.

۱۳+۱+۴=۱۸>۱۵

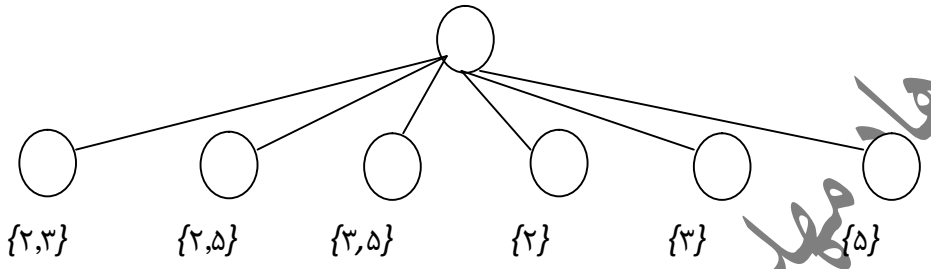
این مسئله چند تا جواب داشت ؟ دقیقاً نمی‌دانم. ولی چند تا را عملاً چک کردید؟ واقعاً چند تا جواب را چک کردید؟ دو تا و فقط دو شاخه را کامل دیدید دو تا جواب مطلق بررسی کردید بقیه را وسط‌های جواب منصرف شدید ادامه بدهید. یعنی تمام شاخه‌هایی که بسته می‌شوند معنی‌اش این است که وسط‌های جواب دیگر منصرف شدید که ادامه بدهید به این شمارش ضمنی می‌گویند. حالا نمی‌دانم شاید ۴۰ ، ۵۰ تا حداقل جواب دارد و یا شاید هم بیشتر جواب واقعی دارد یعنی اگر تا آخر باز می‌کردید کلی جواب هست ولی خیلی از اینها اصلاً ادامه ندارد وسط جواب منصرف شدم که ادامه بدهم فوقش این است که منصرف شدن دلیل محرز داشت یعنی مطمئن شدم تکراری است یا مطمئن شدم جوابش بالای ۱۵ است بالای ۱۷ است و بستم والا اگر شما غیر از این عمل کنید دیگر آن اصطلاح بهینگی را نمی‌توانید به کار ببرید یعنی کلمه بهینه را دقت کنید که آن کلمه بهینه را که استفاده می‌کنی یعنی همه جواب‌ها را باید با دلیل قانع‌کننده‌ای کنار گذاشته باشی پس عملاً ما دو تا جواب داریم.

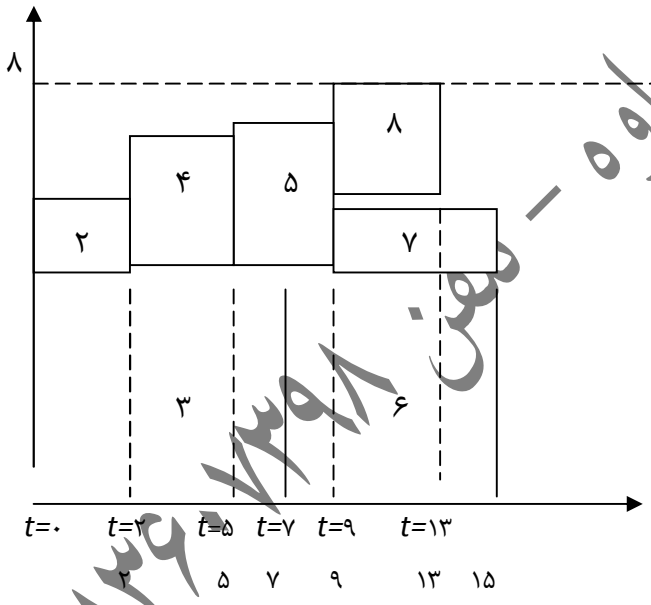
روش دوم مسئله همان است همان ***RCPSP*** است روش اسمش همان است (شاخه و کران) یعنی دوباره منطقش شاخه و کران است فوقش یک مکانیزم دیگری برای شاخه زنی استفاده شده و یک مکانیزم دیگری برای کران گذاری استفاده شده است در روش اول ***Feasible sequence*** و این داستان‌ها بود ولی در این روش ***feasible sequence*** نداریم.

روش *Extension alternative*

این روش توسط آقای استینسون ارائه شده است. مجموعه فعالیت‌هایی را بگویید که قابل زمانبندی هستند در یک زمان مشخص شامل فعالیت‌هایی که وجود ندارد یک نوع منبعی که اینها نتوانند انجام بشود در واقع هیچ محدودیت منبعی به اصطلاح نصب نشود شروع هم لحظه صفر است . $t=۰$. در همین مثالی که داریم که بگویید که کدام فعالیت‌ها در لحظه صفر می‌توانند انجام بشوند، شروع بشوند ولی منبع و پیش‌نیازی نقض نشوند یک را نگویید چون مجازی است کدام‌ها می‌توانند انجام بشوند . *{۲,۳,۵}* ولی چیزهایی دیگه هم باید بگویید. *{۲,۳}* با هم می توانند *{۲}*

به تنهایی و {۳}به تنهایی درست است {۲,۳} نیز با هم درست هستند {۳,۵} درست اند. {۲,۵} هم درست‌اند. {۲,۳,۵} چی ؟ ۲ و ۵ و ۳ از لحاظ پیش‌نیازی مشکل ندارد با هم می‌توانند انجام بشوند ولی منبعی که می‌خواهد نیست یعنی از لحاظ منبع مشکل دارد {۳,۸} از لحاظ پیش نیازی مشکل دارد پس ما کلاً به سمت اینها نرید. چون اصلاً پیش‌نیازی‌هاش انجام نشده است.فقط این آمد این ور. یک که هیچی فقط اینها ترکیب های مختلفش تکی می‌توانند انجام بشوند دوتایی و دو به دو هم می‌توانند انجام بشوند ولی سه تایی نه. پس الان می‌خواهم شاخه بزنم باید به چند قسمت این *tree* را شاخه بزنم ؟ به ۶ قسمت. این ۶ تا {۵} ,{۳} ,{۲},{۳,۵} ,{۲,۵} ,{۲,۳}



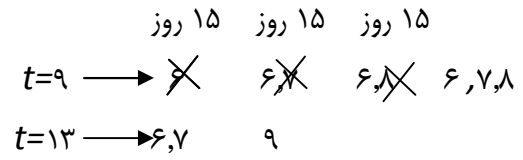


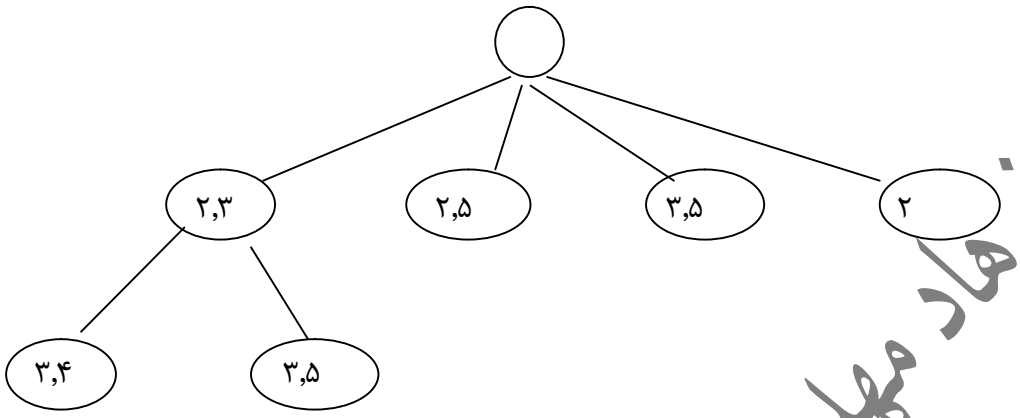
$t=۲$ ⟶ ~~۲~~ ۳,۴ ۳,۵

مغلوب شد. {۳} و {۴,۳} هر دو ۷ روز می‌شود.

$t=۵$ ⟶ ۳ ۳,۵ ۳,۶

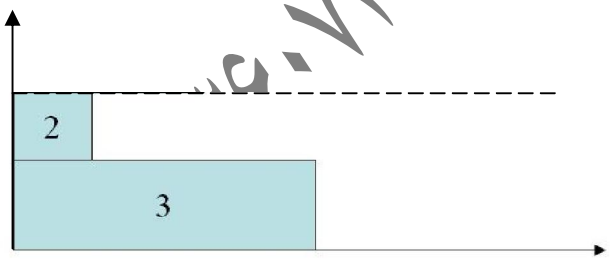
$t=۷$ ⟶ ۵ ۵,۶

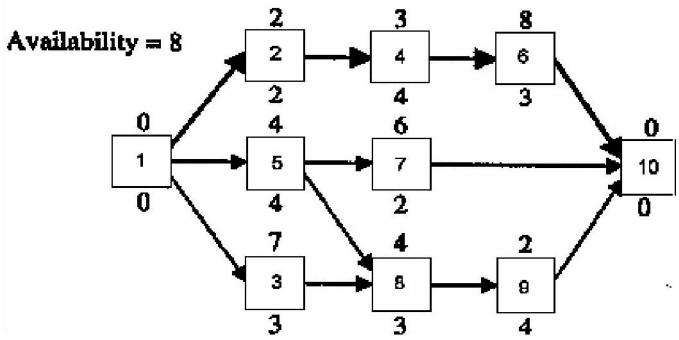




این الان کدام است مشخص است دیگر این {۲,۳} است اگر {۲,۳} را بچینید. این پروژه تا اینجا چند روز انجام می‌شد ؟ ۷ روزه. اگر {۲,۳} را با هم بچینید این پروژه تا اینجا ۷ روز می‌شود. این چیه آن وقت ؟ این کدام شاخه است؟ {۳} چند روزه ؟ ۷ روزه . خوب منطق چی می‌گوید؟ منطق می‌گه اگر {۲,۳} با هم ۷ روزه می‌توانند انجام بشود و {۳}هم به تنهایی ۷ روزه است خیلی واضح است که شما بهتر است در طی ۷ روز دو تا فعالیت انجام بشود تا یک فعالیت . نتیجه ۳ را حذف کردم. آیا حذف کردن به این معنی است که موجه نبود؟ نه موجه است مغلوب بود. شاخه‌ای که مغلوب است یعنی جوابی است که واضح است هیچ وقت بهینگی در آن نیست

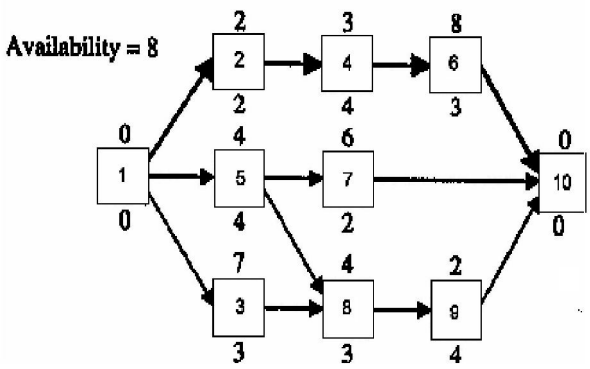
{۲, ۳}, {۲,۵}, {۳,۵}, {۲}, {۳}, {}





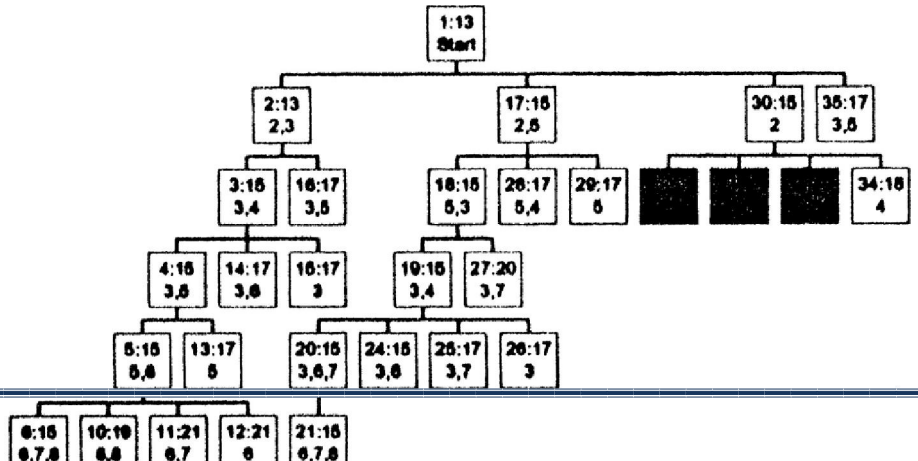
؟ ۴ روز و خود {۵} هم به تنهایی انجام شده است بازم ۴ روز. این یعنی چی ؟ {۵} , {۳} انفرادی‌هاشون مغلوب است، مغلوب چی ؟ {۳} مغلوب {۲,۳}شده است {۵} مغلوب {۲,۵} شده است. حالا در مورد {۲} چی ؟ مغلوب کدام می‌شود ؟ چرا مغلوب نمی‌شود. اگر {۲} به تنهایی انجام شود ۲ روز است. بین {۲,۵} {۲,۳} کدامشان ۲ روزه می‌شوند ؟ هیچ کدام پس ۲ مغلوب نمی‌شود. پس خلاصه اگر بخواهید فرمولش کنید کدام مجموعه‌ها مغلوب می‌شوند؟

{۲, ۳}, {۲,۵}, {۳,۵}, {۲}, {۳}, {۵}

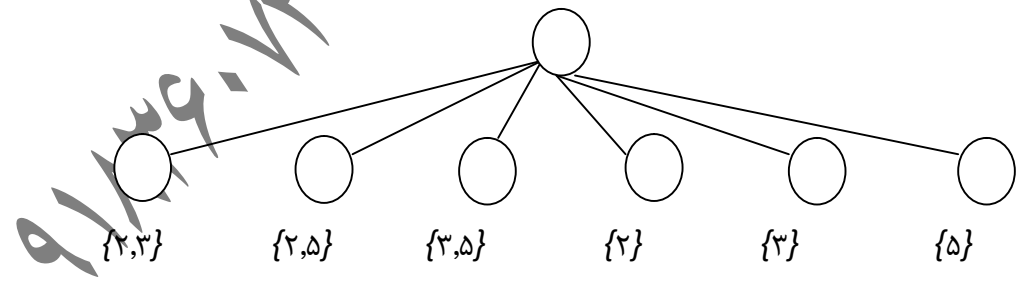


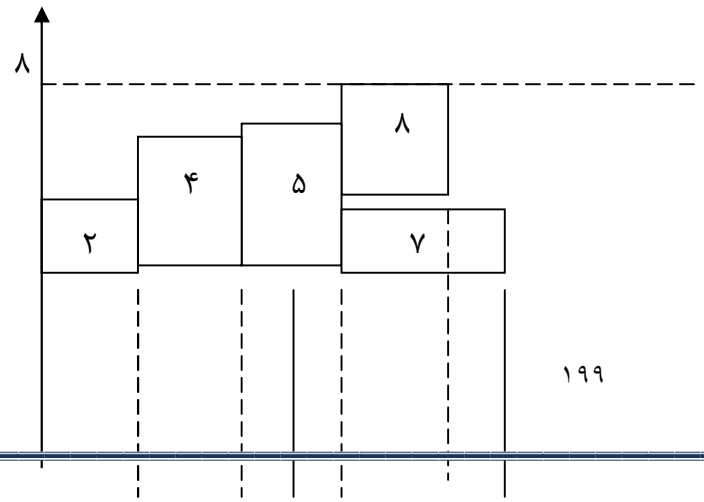
اسم اینها را مجموعه می‌گذارید. اگر یک مجموعه‌ای A زیر مجموعه B باشد.$(A \subset B)$ ولی وقتی زمانبندی می‌کنید زمان آخر *make span* اش برابر شود A مغلوب می‌شود (آن زیرمجموعه مغلوب می‌شود). اگر $A \subset B$ باشد و از نظر زمانبندی زمان اتمام A با B برابر باشد در این صورت مجموعه A مغلوب است و بررسی نمی‌شود آن شاخه را حذف می‌کنیم. با این کار شما عملاً تعداد زیادی از جواب را تصمیم گرفتید از اول بررسی نکنید. سوال این کار اشکال ندارد ؟ خیر چون منطقی است که این جواب‌ها جواب هستند ولی مغلوب هستند یعنی هیچ حسنی در آن‌ها نیست که بخواهیم آنها را بررسی کنیم قطعاً جواب بهینه در اینها نیست پس اینها دور ریخته می‌شوند پس آنهایی که باقی ماندند چند شاخه است در این مثال ۴ تا باقی می‌مانند. *{۲} {۳,۵} {۲,۵} {۲,۳}* خوب حالا کدام شاخه را اول انجام بدهیم ؟

من این شاخه را شروع کردم آن اول که *root* هست اعداد را ترجمه کنید.

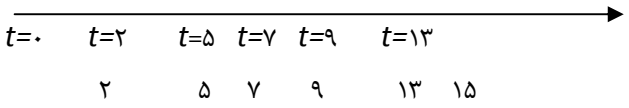


بالا دو تا عدد است عدد سمت چپ شماره گره است عدد سمت راست ***lower bound*** است پایین هم این مجموعه‌ها ***extention alternative*** . شروع یک است که البته مجازی است. ۱، ۱۳= ***lower bound*** اما بعداش کدام‌ها باید بیاید همین ۴ تایی که من نوشتم یعنی کدام ؟ ***{۳,۵} {۲} {۲,۵} {۲,۳}*** ترتیبش با اینی که من نوشتم یکم فرق می‌کند ولی خوب به هر حال همان ۴ تا است. ***lower bound*** ها چند است ؟ ۱۷ ,۱۵ , ۱۵ ,۱۳ حالا چه طوری حساب شده است نگفتم کدام‌ها را اول باز کرده است؟ ۱۳ را. این معنی‌اش این است که چطور عمل کرده است. ***best first*** یعنی آنی که اوضاعش بهتر بوده است ظاهراً اول باز کرده است ولی یادمان نرود که مهم نیست باید همه شاخه‌ها بررسی بشوند حالا فقط آن اصطلاحی که آن هفته گفتم ***best first*** و ***depth first*** . روش قبلی به صورت ***depth first*** بود ولی اینجا از روش ***best first*** استفاده شده است.





۳ ۶



t=۲ ⟶ ~~۳~~ ۳,۴ ۳,۵

مغلوب شد. {۳} و {۴,۳} هر دو ۷ روز می‌شود.

t=۵ ⟶ ۳ ۳,۵ ۳,۶

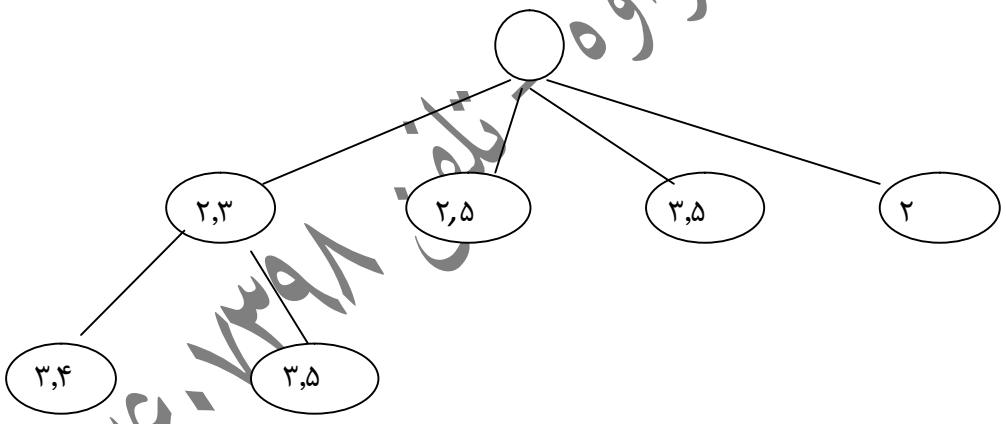
t=۷ ⟶ ۵ ۵,۶

۱۵ روز ۱۵ روز ۱۵ روز

t=۹ ⟶ ~~۶~~ ~~۶,۷~~ ~~۶,۸~~ ۶ ,۷,۸

t=۱۳ ⟶ ۶,۷ ۹

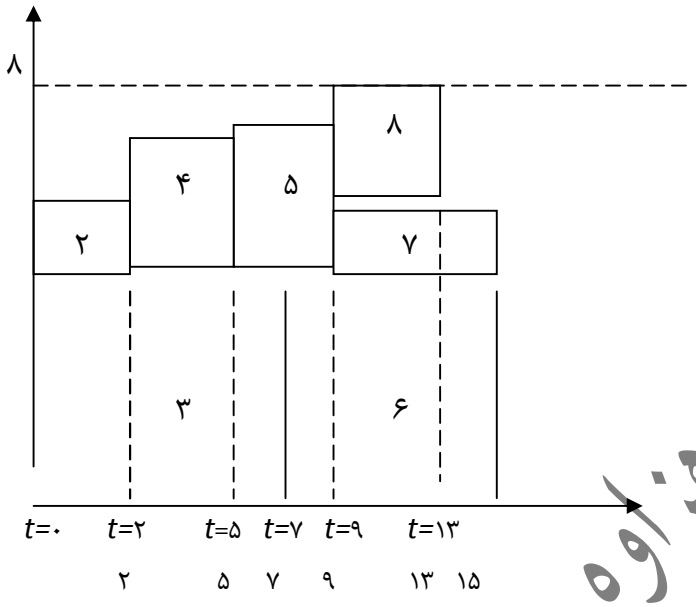
t=۱۵ ⟶ ۹

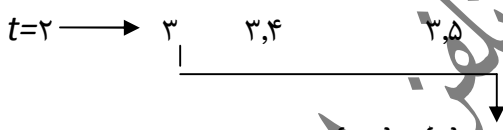


یعنی شاخه‌ای که امیدوار کننده تر است اول بررسی کرده است چون ***Lower bound*** اش ۱۳ است و ۱۳ کمتر است و بهتر است و هدف C_{max} است اول این را بررسی کرده است پس کدام را می‌خواهیم الان بچینیم {۲,۳} . وقتی که ۲ را چیدم ۳ را هم کنارش بچینم پس این الان زمانبندی تا این لحظه است از ۸ تا منبع موجود ۵ تاش استفاده شده است.

t بعدی کجا می‌شود؟ ما الان در t=۰ بودیم. از روی گانت باید بگویید اولین جایی که منبع آزاد می شود کجا است ؟ شما الان شروع کردید این خالی بود گانت کجا منبع داشتید ؟ لحظه صفر ولی الان بین اینهایی که می‌بینید کدام اول یا زودتر تمام می‌شود؟ (۲) . کی تمام می‌شود ؟ ۲ پس t بعدی می‌شود ۲ الان ما در t=۲ هستیم. حالا این سوال را دوباره بگویید در لحظه ۲ کدام فعالیت یا فعالیت‌ها می‌توانند همزمان انجام بشوند از روی گانت، از روی شکل بگویید. فقط یادتان باشد که روی ۱ و ۲ کلاً خط بزنید که اینجا دیگر نیستند چون چیده شدند. کدام‌ها است ؟ {۳,۴,۵} غلط

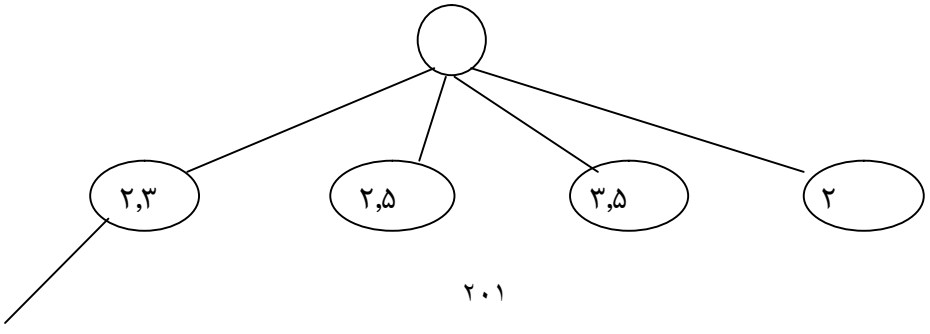
است ، ۱۱ تا منبع می‌خواهد باید جمع منبع زیر ۸ باشد. {۳} {۵} {۴} {۵,۳} {۴,۵} {۴,۳} . مگر ۳ زمانبندی نشده ؟ چرا زمانبندی شده است ولی تمام نشده است. دقت کنید . جواب‌ها غلط بود حتماً باید ۳ را بگویید. چرا ؟ وقتی فعالیتی که وسط کارش است نمی‌شود منصرف بشود و بگویید ۳ نباشد ، ۳ را باید حتماً بگویید پس ۳ حتماً جزء بازی است ؟ حالا در کنار ۳ صحبت کنید. {۳,۵} {۳,۴} {۳} . ۳ که باید باشد حالا به تنهایی باشد یا با ۴ یا با ۵ خلاصه باید ۳ را بگویید. {۳,۴,۵} هم نمی‌شود. {۱,۲} هم نیست. حالا بگویید که از این ۳ تا کدام مغلوب می‌شود ؟

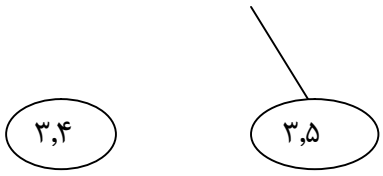




مغلوب شد. {۳} و {۴,۳} هر دو ۷ روز می‌شود.

اصلاً مغلوب می‌شود یا خیر ؟ مثلاً من می‌توانم ۳ را مغلوبش کنم ؟ دفعه قبل شما این‌ها را مغلوب کردید طبق آن $A \subset B$ باشد الان دقت کنید $\{۳,۵\}\{۳,۴\} \subset ۳$ است میشود حذف کرد یا خیر ؟ اگر $A \subset B$ باشد ولی زمان اتمام A,B برابر باشد A حذف می‌شود این جمله‌ای بود که نوشتیم الان $A \subset B$ است $\{۳,۴\} \subset ۳$ است اگر دقت کنید ۳ به تنهایی چند روزه انجام می‌شود ؟ ۷ روزه حالا اگر کنار ۳، ۴ را همین جا شروع کنم چی ؟ ۳ روزه و ۴ تا منبع می‌خواهد یعنی ۲ام تا ۵ ام می‌شود. حالا نظرتان چیست ؟ ۳ مغلوب می‌شود چون ۳ به تنهایی ۷ روز است و {۳,۴} با هم ۷ روز است پس ۳ حذف می‌شود.





t=۷	۵	۵,۶		
	۱۵ روز	۱۵ روز	۱۵ روز	
t=۹ ⟶	~~۶~~	~~۶,۷~~	~~۶,۸~~	۶ ,۷,۸
t=۱۳ ⟶	۶,۷	۹		
t=۱۵ ⟶	۹			

خلاصه باید چند تا شاخه بزنیم ؟ دو تا . این دو تا چی هست ؟ *{۳,۵} {۳,۴}* . ۳ هم البته بود ولی حذف شد. مغلوب شد. حالا اگر دقت کنید در این *tree* چه می‌بینید. *{۳,۵} {۳,۴}* ، ۳ ایی هم بوده که مغلوب شده است به همین دلیلی که الان صحبت کردیم. ولی ۳ باید باشد. یادتان باشد چون ۳ وسط کارش بود باید باشد یعنی فعالیت‌هایی که در لحظه *t* قبلاً شروع شدند ولی هنوز تمام نشدند باید جزء *option* ها باشد. خوب *lower bound*اش را حساب کرده است . *lower bound* چند شده است؟

{۳,۴} ، *lower bound* اش ۱۵ شده است . *{۳,۵}* ، *lower bound* اش، ۱۷ شده است. چون *best first* عمل می‌کند اول کدام را انجام می‌دهد. *{۳,۴}* چیدم خوب *t* بعدی کجا می‌شود ؟ کدام ؟ از روی شکل باید بگویید. اولین جایی که یکی از اینها تمام می‌شود ؟ ۵ ام . فکر کنم دیگه کار الان راحتر باشد . این شکل اینم تا اینجا جواب شما. در لحظه ۵ کدام فعالیت یا فعالیت‌ها می‌توانند در حال انجام باشند. باز باید ۳ را بگویید. ۱ و ۲ و ۴ را کنار بگذارید چون تمام شده است. *{۳,۶} {۳,۵} {۳}*و *{۳,۵,۶}* به دلیل نبود منبع نمی‌شود. شما باید بگویید ۳ را حذف کنم یا نه ؟ نمی‌شود این بار ۳ را حذف کرد ؟ چون ۳ الان به تنهایی می‌بینیدش چند روزه تمام شده ، ۷ روزه ، یعنی ۳ آخرش ۷ ام است دیگه اما در کنار ۳ اگر ۵ یا ۶ را بگذارید دیگه ۷ روزه نمی شد بیشتر می‌شود پس این بار ۳ باید جزء بازی بماند پس اگر دقت کنید ۳ به تنهایی هم جزء *option*ها بماند *{۳,۶}* ، *lower bound*، ۱۵ ، ۱۷، ۱۷ خوب ۱۵ را اول انجام می‌دهم . *{۳,۵}* را می‌خواهم بچینم ۳ که البته هست در کنارش ۵ را هم قرار می‌دهم. ۵، ۴ روز است . ۴ تا منبع هم دارد. پس ما روی *tree* الان توی این نقطه هستیم. من تا اینجا کار را انجام دادم

خوب بفرمایید الان *t* چند می‌شود؟ *t*=۷ است. در *t*=۷ کدام‌ها می‌توانند در حال انجام باشند. این *lower bound* را همین جا بگم بعداً سخت می شود این *lower bound* هایی که می‌بینید ۱۷ و ۱۵ و ۱۳ اینها از کجا آمدند؟ بحث خیلی ساده است آن باقیمانده مسیر بحرانی که یادتان نرفته است وقتی ۳ و ۵ را می‌خواهید بچینید ۳ که بوده ، ۵ اضافه می‌شود باقیمانده مسیر بحرانی ۵ چند روز است در شبکه . ۱۰ روز است و *t* الان چند است ۵ . ۱۵=۵+۱۰ باقیمانده مسیر بحرانی شبکه را حساب کنید به اضافه *t* کنید که این می‌شود *lower bound*.

نحوه محاسبه *lower bound* برای این روش (نحوه محاسبه *LB*)

برای فعالیت‌هایی که هنوز زمانبندی نشدند طول باقیمانده مسیر بحرانی را محاسبه کنید و نتیجه را با t جمع کنید. این *lower bound* به همین سادگی حساب شد. پس الان در این مثال ۱۵ از کجا آمده است ؟ کدام‌ها را حذف کردم. ۱ و ۲ و ۴ را بگذارید کنار و اینها بقیه زمانبندی نشدند. کدام فعالیت مسیر بحرانی‌اش بیشتر است این که ۸ است این که ۶ است این ۱۰ است . این ۱۰ یادتان باشد تا اینجا از همه بیشتر است. اینم ۱۰ است، ۷،۸،۹،۱۰ و ۱۱ سه چیده شده پاک کنید. طولانی‌ترین کدام شد. شماره ۵ بود دیگه ۱۰=۶+۴ می‌شود ۱۰ پس مسیر بحرانی باقیمانده ۱۰ است ۱۰ را با t جمع کنید که t در این جا ۵ بوده است. ۱۵=۵+۱۰ پس آنهایی که زمانبندی شدند. من جمله‌ام را چطور شروع کردم ؟ گفتم برای فعالیت‌هایی که زمانبندی نشده‌اند آنهایی که زمانبندی شده‌اند که هیچی آنهایی که نشدند فقط در مورد آنها دارم صحبت می‌کنم. حالا جلوتر یکبار دیگر این را تمرین می‌کنم.

t الان در شکل ۷ است در t=۷ شما بگویید این مجموعه‌ها چی باید باشند؟ فقط روی ۱،۲،۳،۴ خط بزنید. ۵ را باید بگویید چرا ؟ چون در روز ۷ ام ، ۵ نصفه کار است هنوز ، حتماً ۵ هست. به جزء ۵ فقط ۶ مانده است. پس یکی *{۵}* است و یکی هم *{۵،۶}* . حالا باز سوال تکراری ؟ میشه ۵ را حذف کرد ؟ نه چون ۵ به تنهایی ۹ روز انجام می‌شود ولی ۶ را کنارش بگذارید ۹ روزه نمی شود بیشتر می‌شود پس مغلوب نمی‌شود پس دو تا شاخه می‌خواهد *{۵،۶}, {۵}* اگر {۵} بماند ۱۷ و اگر *{۵،۶}* بیایند ۱۵ پس، مینیمم ۱۵ است در نتیجه *{۵،۶}* را انجام می‌دهیم. ۵ که هست. ۶ که ۸ روز است ۳ تا منبع می‌خواهد که من چیدم بعدی t=۹ است کدام‌ها ، ۶ را باید حتماً بگویید ۶ را در شکل نگاه کنید. ۶ نصفه کار است در کنار ۶، ۷ و ۸ هم هست. یکی به تنهایی *{۶}* بعد *{۶،۸} ,{۶،۷}* در مورد *{۶،۷،۸}*چی ؟ ما تا الان ۳ تایی نداشتیم. *{۶،۷،۸}* اشکال ندارد چون ۳تا و ۲ تا و ۳ تا می‌شود ۸ تا دیگه منبع کافی است این بار این ۳ تا جمع منبع‌اش ۸ است کافی است پس ۴ تا مجموعه می‌شود *{۶،۷،۸} {۶،۸} {۶،۷} {۶}* .

حالا سوال : کدام‌ها مغلوب هستند ؟ $A \subset B$. ۶ را می‌توانید حذف کنید ؟ اگر $A \subset B$ باشد ولی زمانبندی شان با هم یکی باشد می‌شود حذف کرد. الان ۶ زیر مجموعه ۶ و ۷ است اگر خوب دقت کنید من اگر در کنار ۶ ، ۷ را قرار بدهم چه اتفاقی می‌افتد. ۷، ۶ روز است و دو تا منبع می‌خواهد . ۶ روز می‌شود ۹ ام تا ۱۵ ام. ببینید ۶ ، ۱۵ روز بود. ۶ با ۷ هم ۱۵ روزه است پس ۶ به تنهایی لازم نیست.

سوال دیگر : حالا آیا می‌توان خود *{۶،۷}* را حذف کرد. چون خود *{۶،۷}* هم زیر مجموعه *{۶،۷،۸}* است و یا خود *{۶،۸}* هم زیر مجموعه *{۶،۷،۸}* است میشه حذف کرد. باز همان داستان هست دیگه ۸ را اینجا بذارید ببینید چی می‌شود ؟ ۸، ۴ روز است و ۳ تا منبع دارد. ۴ روز می‌شود ۹ ام تا ۱۳ ام. نظرتان چیست ؟ ببینید ۶ به تنهایی انجام بشود یا با ۷ یا با ۸ یا با ۷ و ۸ خلاصه ۱۵ روز است. نتیجه اینها همه حذف می‌شوند یعنی عملاً اینها فقط یک شاخه لازم است آن شاخه را اگر نگه دارید اشتباه نیست ولی داری الکی حساب می‌کنی فقط وقتت گرفته می‌شود آنها بسته می‌شود.

یک اشکال در شکل است این ۳ تا را خط بزنید اینها نباید از هم باز کنی اینجا یک شاخه باید بیاید پایین. همین شاخه فقط بیاد این ۳ تا را خط بزنید اینها اشتباه است. اینها مغلوب ۶،۷،۸ است پس عملاً ۶،۷،۸ را دارید خوب چی می‌شود؟ جواب را ببنید از روی شکل است. خوب t بعدی ۱۳ است در ۱۳ چی باید بگی ؟ کدام‌ها در حال کار هستند هنوزم پس

باید ۶ و ۷ را باید بگویید حالا در کنار ۶ و ۷ دیگه چی می‌تواند بیاید ۹ می تواند بیاید ؟ آره ۹ هم می‌تواند بیاید ولی منبع نداریم پس نمی‌تونه بیاد. پس در t=۱۳ فقط *{۶,۷}* بعدی که می‌شود t=۱۵ اینم ۶ را پاک کنید. ۶ نداره ۹ فقط دارد اینم یک اشکال تایپی دیگر است پس اینجا هم آخر سر ۹ می‌آید و این می‌شود ۱۷ پس ۱۷ را پیدا کردید. ***Lower bound***، ۱۷ است این بهترین جوابی است که تا این لحظه پیدا کردید. به اصطلاح ***first visible schedule*** اش در گره شماره ۹ به دست آمده با $C_{max} = ۱۷$. گانت‌اش کدام است همانی که با هم رسم کردیم حالا شاخه های دیگه را چی کار کنم ؟ آنهایی که بهتر از ۱۷ نمی‌شوند را ادامه نمی دهند. در روش قبلی چند تا ***bounding role*** داشتیم ؟ ۴ تا اما در این جا فقط ۲ داریم فقط به دو دلیل شاخه‌ها بسته می‌شوند یکی ***lower bound*** است.

(۱) Bounding role یا (قاعده کران گذاری ۱) : هر شاخه‌ای که ***LB*** آن از بهترین جواب یافت شده تا این لحظه بیشتر یا برابر باشد حذف می‌شود.

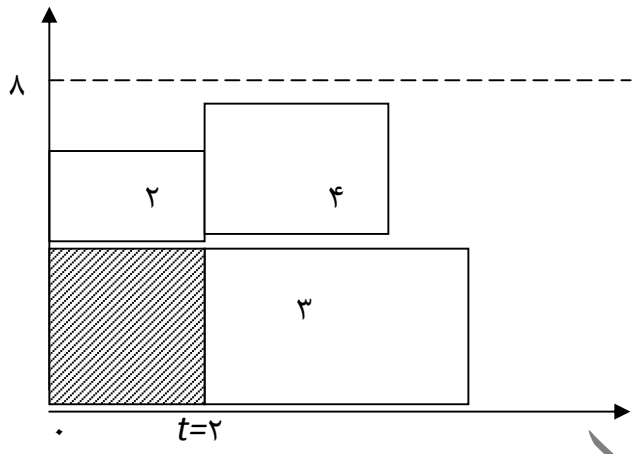
کدام‌ها حذف می شود، این ۱۷ را ذخیره کردم این چند است ؟ ۲۱،۲۱،۱۹، ۱۷،۱۷، ۱۷،۱۷ اینها به درد من نمی‌خورد ولی این چند است ؟۱۵ خوب این را مجبورم باز کنم و ظاهراً تا آخر آمدم آخر سر هم فهمیدم ۱۵ است. نتیجه این پاک می‌شود ۱۵ به جا می‌نشیند حالا از این به بعد توقع ام زیاد است جواب‌هایی که از ۱۵ بدتر هستند را حذف می کنم ۱۵، ۱۷،۱۷،۲۰،۱۷،۱۷، ۱۵ دیگه حذف می شوند هیچ جوابی بهتری ظاهراً وجود ندارد. این می‌شود ***optimal schedule*** البته جواب ۱۵ یکبار حل کرده بودیم البته برای بار دوم است که با یک روش دیگه حل می کنیم پس یک دلیل بستن شاخه ها همین باشد.

(۲) Left shift dominance : در روش قبلی یک همچین چیزی نداشتیم یعنی چی ***Left shif dominance*** ؟ قبل از اینکه بنویسیم رو مثال ببینیم بعد جمله‌اش را راحت‌تر می‌شود نوشت. گره ۳۱ را لطفا ببینید گره ۳۱ به دلیل ***Left shif dominance*** بسته شده است. چرا بسته می‌شود ؟ اصلاً داستانش چی هست ؟ آره تکراری است. ***Left shif dominance*** تکراری را می بندد اما چطور این تکرار اتفاق افتاده است. اول خود گره ۳۱ را تا اینجا بسازیم چطور ساخته شده است از آن بالا بیاید پایین دوباره ، گانتش را دوباره رسم کنم. در لحظه صفر کدام انجام شده است ؟ یک هیچی بعد کدام انجام شده ؟ ۲ به تنهایی. من اینجا ۲ را رسم کنم. t که صفر بود حالا t شده ۲، در روز ۲ ام کدام‌ها چیده شدند؟ ۳ و ۴ خوب اینجا چه اتفاقی افتاده است اینجا که هاشور زدم چی قضیه اش هیچ فعالیتی نیست یعنی منبع چی داریم ؟ منبع آزاد بیکار ***Left shift*** یعنی چی ؟ یعنی هل بدهید به سمت چپ اگر من ۳ را هل بدهم آن ور چی می‌شود؟ اشکال ندارد منبع دارم پیش نیاز نداره منبع هم دارد اگر ۳ را هل بدهم آن ور خیلی خوب است چون باعث می‌شود ۹ روز ، ۷ روز بشود. فقط اگر ۳ برود آن ور جواب شما شبیه چی می شود ؟

آن شکل را تصور کنید که ۳ را ببرید بچسبانید آن ور ، آن این میشه دیگه، کپی این میشه می‌شود تکرار، شما این را قبلاً کدام گره بررسی کردید یادتان می‌اید همین الان باید با هم انجام دادیم یعنی گره ۳۱ و ۳ با هم تکراری می شوند پس دیگر ادامه نمی‌دهم چون یکبار هم انجام داد این کار را. ***Left shif dominance*** یعنی فعالیتی که قبلش

منبع بیکار دارد پیش نیاز هم ندارد عقب‌نشینی می‌کند فوقش عقب نشینی‌اش باعث می‌شود که جواب شما تکراری شود

(۲) *Left shift dominance* : اگر گزینه‌ای وجود داشته باشد که حداقل یک فعالیت آن بتواند بدون نقض منبع یا پیش نیازی به سمت چپ جابه‌جا شود یعنی در گزینه گره بالاتری قرار بگیرد آن شاخه تکراری است و حذف می‌شود.



اگر ۳ بره بچسبد به صفر در واقع چه اتفاقی افتاده است یعنی ۳ رفته کنار ۲ قرار گرفته، آنجا ۲ تنها بود ولی وقتی ۳ می‌رود آنجا قرار می‌گیرد یعنی ۳ و ۲ با هم هستند یعنی چه اتفاقی افتاده ؟ یعنی ۳ از اینجا رفته به گره بالاسری یعنی ۲، یعنی ۳ رفته آنجا کنار ۲ قرار گرفته آنجا دیگه ۲ نیست ۲ و ۳ است اگر ۲ و ۳ هست شما قبلاً ۲ و ۳ را داشتید تکراری بود. یعنی یک فعالیتی در *tree* یک *Level* بالاتر رفته عقب‌نشینی اش دو تا معنی دارد روی گانت یعنی *Left shift* یعنی رفته عقب تر اما در *tree* اگر می‌خواهی صحبت کنی یعنی یک *level* رفته بالاتر.. خوب اگر یک *Level* بالاتر برود تکراری می‌شود.

جلسه یازدهم

۲ تا روش را برای مسئله *RCPSP* صحبت کردیم که هر دو روش، روش‌های *exact* مبتنی بر یا شاخه و کران بود.

در روش اول که *Precedence tree* بود هر مسئله با یک کدی که به اصطلاح می‌گفتیم این *feasible sequence* . کدگذاری می‌شد. *Feasible sequence* چینش فعالیت‌ها بود به ترتیبی که پیش‌نیازی رعایت بشود بعد براساس همین توالی فعالیت‌ها به اصطلاح چیده می‌شد تا نهایتاً یک جواب ساخته شود این روش ۴ تا به اصطلاح *Bounding role* است یعنی ۴ تا لم داشته که کی شاخه‌ها را ببندیم دو تاشون قواعد به اصطلاع تکراری بود شاخه‌هایی را می‌بست که مطمئن بود که این جواب تکراری خواهد داد و دو تا هم شاخه‌هایی می‌بست که *lower bound base* بود به نوعی که شاخه‌هایی که نا امیدکننده بود.

روش دوم روش *extentional ternative* بود که برخلاف قبلی، دیگه به توالی ربطی نداشت یک روش زمان محور بود یعنی جی کار می‌کردیم از روی گانت شروع می‌کردیم از لحظه صفر به چیدن فعالیت‌ها. در لحظه صفر کدام فعالیت‌ها می‌توانند چیده بشوند ؟ همه حالت‌های ممکن را شاخه می‌زدیم و می‌شمردیم . بعد t بعدی. کدام فعالیت‌ها می‌توانند چیده بشوند و الی آخر که نهایتاً تا انتها همه فعالیت‌ها چیده می‌شدند در شاخه‌ها. دو تا *bounding role* داشت برخلاف قبلی که ۴ تا بود. یکشون *lower bound* با یک مکانیزمی حساب می‌شد. شاخه‌ای که *lower bound*اش بالاتر از بهترین جواب یافت شده بود بسته می‌شد یکی هم اصطلاحی به اسم *left shift dominance* شاخه‌ای را می‌بست که فعالیت بتواند *Left shift* داشته باشد بتواند عقب‌تر بیاد به اصطلاح که جواب تکراری بدست می‌آمد.

این خلاصه‌ای بود از دو روش در هفته گذشته. تعداد این روش‌ها در واقع بالای ده تا می‌باشد که در این کلاس قرار بر این است که از ۴ مورد آنها صحبت شود که ۲ روش را در هفته گذشته صحبت کردیم و در پارت ۱۱ به دو روش دیگر پرداخته می‌شود. این ۴ روش چه اشتراکی دارند ؟ همشان شاخه و کران هستند و همه‌شان جواب بهینه می‌دهند. اما چه فرقی دارند ؟ نحوه شاخه‌زنی و نحوه کران گذاری کلاً در روش های شاخه کران برای یک مسئله تفاوت‌هایشان در نحوه شاخه‌زنی و نحوه کران گذاری می‌باشد که حالا هر کدام یک مکانیزم خاص خودش را دارد فقط در این دو مکانیزم یعنی در شاخه زنی و کران‌گذاری دو تا نکته کلیدی بود. در شاخه‌زنی چی بود ؟ باید طوری این تقسیم جواب‌ها صورت بگیرد که به هیچ عنوان جوابی از بین نرود یا گم نشود چون وقتی شما می‌خواهید به اصطلاح تضمین بهینگی را داشته باشید آن وقت اگر یک جوابی را اصلاً بررسی نکرده باشید که دیگه نمی‌توانید بگویید بهینه است چون همان جوابی که بررسی نکردید شاید بهینه بوده و شما آن را اصلاً ندیدید پس اگر کلمه بهینه را استفاده می‌کنید باید این موضوع حتماً دیده شود تمام جواب ها باید چک بشود جوابی از بین نرود. در *Bounding* چی ؟ آنجا هم یک نکته ریز داشت که قبلاً گفته شد. شاخه را که می‌بندید یعنی نمی‌خواهید ادامه بدهید باید محرز بشود که یا جواب تکراری است یا *infeasible* است یا *lower bound* اش بالای *optimal solution* است که تا این لحظه داشته‌اید یعنی باید محرز شود که تکراری است یا محرز شود که *feasible* نیست یا محرز شود که جواب بی‌فایده است *lower bound* آن یعنی نا امیدکننده است یعنی بستن شاخه باید با دلیل باشد. الکی اگر شاخه‌ای را ببندید یعنی تکراری نیست و شما شاخه را ببندید یا *feasible* نیست ولی شاخه را ببندید *lower bound* اش از بهترین جواب

بیشتر نیست ولی ببندید خوب معلومه داری جواب بهینه را ممکن است از بین ببرید پس هر دو این نکات باید رعایت شود

حالا این دو روش شاخه و کران هم باز همین موضوع خواهد بود.

روش سوم که روش حالا عنوانش ***minimal delaying alternatives*** است آقای دمیل مستر و هرولن ***((Demeulemeester and Herroelen (۱۹۹۲))*** که اتفاقاً نویسنده همین کتابی هستند که ما به عنوان ***refrence*** انتخاب کردیم این روش را پیشنهاد کردند.

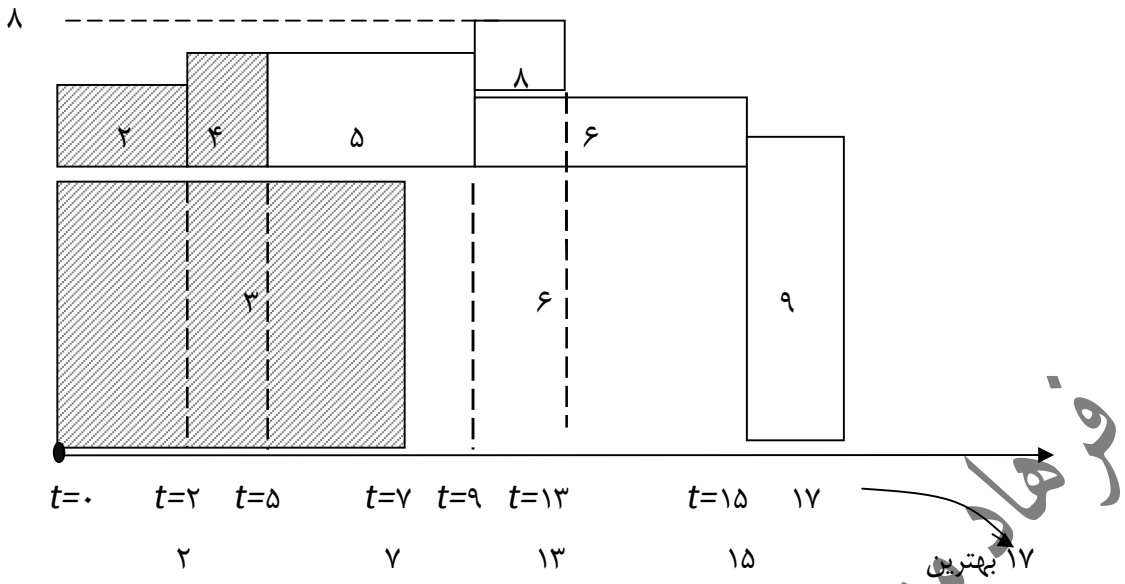
روش ***minimal delaying alternative*** (گزینه‌های تأخیر)

در این روش ***minimal delaying alternative*** فرد زمانبندی می‌کند فعالیت‌ها را تا یک ***Resource conflict*** (یعنی تضاد منبع ، تناقض منبع) ایجاد شود منظور از تضاد منبع چیست ؟ آره یعنی مثلاً شما برای انجام فعالیت‌ها ۱۰ تا منبع نیاز دارید ما کلاً ۷ تا منبع داریم یعنی منبع نداریم به اصطلاح. یعنی تضاد وجود دارد بین آن چیزی که شما می‌خواهید منبعی می خواهید و آن منبعی که ما داریم به این می‌گویند ***resource conflict*** می‌گویند. پس از لحظه صفر شروع می‌کند به چیدن فعالیت‌ها تا به یک ***resource conflict*** برسد خوب بعد فقط در چنین موقعی دوباره فرد (منظور شما هستید، کسی که مسئله را حل می‌کند) چه کار می‌کند ؟ در نظر می‌گیرد چندین ***Alternatives*** (راهکار) برای حل این تضاد برای حل کردن این مشکل منبع یعنی وقتی که تضاد منبع وجود ندارد یعنی شما کارتان را انجام می‌دهید فعالیت‌ها را به ترتیب می‌چینید ولی اگر تضاد پیش آمد باید این را حل کنید. چه طوری باید حل کنید ؟

برای حل ***resource conflict*** کاملاً طبیعی است که باید یک یا چند فعالیت را به تأخیر بندازید که منبع کافی آزاد شود برای انجام بقیه فعالیت‌ها. این خیلی ساده است. شما امروز ۲۶ ام آذر می‌خواهید ۵ تا فعالیت را همزمان انجام دهید و اما این ۵ تا فعالیت همزمان ۳۰ تا منبع می‌خواهد شما ۳۰ تا منبع ندارید خوب خیلی واضح است که راهکارش این است که یک یا دو فعالیت را بعداً انجام دهید که منبع برای بقیه باشد حالا یک یا دو تا یا سه تا ؟ بستگی دارد با چند منبع ، با چند تا تأخیر منبع آزاد خواهد شد. این یک یا چند تا بستگی به صورت مسئله دارد.

برای یافتن جواب زمانبندی بهینه تمام ***delaying alternatives*** ها ، تمام گزینه‌های تأخیر را باید بررسی شود. چرا باید همه آنها بررسی شود ؟ همان اول گفته شد شما دنبال جواب بهینگی هستید باید تمام حالت‌های ممکن بررسی شود اگر کلمه تمام را نیاورید دیگر آن کلمه بهینگی را آن وقت نمی‌توانید بگویید همه حالت‌های ممکن باید چک شوند که شما مطمئن باشید که جواب بهینه را پیدا کرده‌اید.

به عنوان مثال حالا برای اینکه مقایسه انجام بدهیم ما همان مسئله‌ای را برای حل کردن انتخاب کردیم که در جلسه قبل با دو روش قبلی حل شده است. مسئله را کامل می‌شناسید ۸ تا منبع دارد با ۱۰ فعالیت، بالایی ***duration*** و پایینی هم نیازمندی منبع را به شما داده است می‌خواهیم با روش ***minimal delaying alternatives*** حل کنیم.



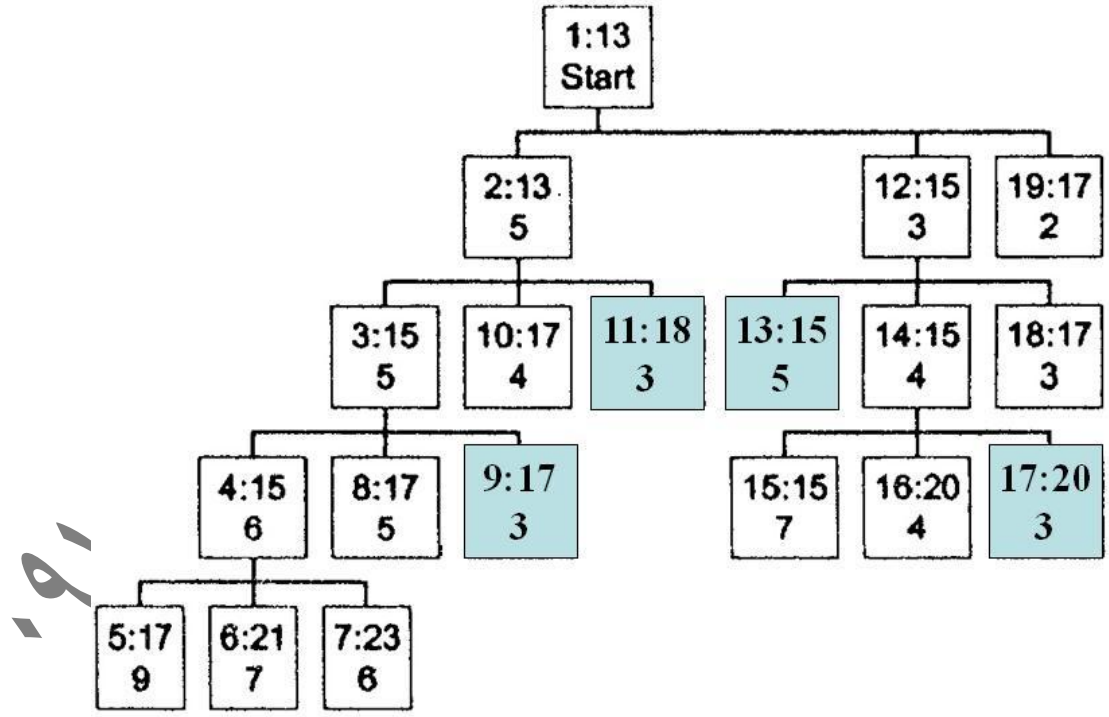
خوب یک گانت خالی به طور مشخص، محور افقی زمان است و محور عمودی منبع است. ۸ تا منبع داریم که در شکل مشخص است در این مثال با این روش اولین نقطه‌ای که قطعاً باید فعالیت را بچیند لحظه صفر پروژه است ***(t=۰)*** در لحظه صفر با توجه به پیش‌نیازی کدام‌ها می‌توانند انجام شوند البته یک را نگویید چون مجازی است پس هیچی به جز یک. کدام‌ها می‌توانند در لحظه صفر انجام شوند ؟ خوب واضح است ۲ و ۵ و ۳ خوب اینجا بچیند ولی مشکل چیست؟ از همین اول که شروع کردیم به چی برخوردیم؟ به یک ***resource conflict*** ایی که اشاره شد به خاطر اینکه ۲ و ۵ و ۳ ، نه تا منبع نیاز داریم ما ۸ تا منبع داریم . راهکار چیست ؟ تأخیر یکی از فعالیت‌ها .

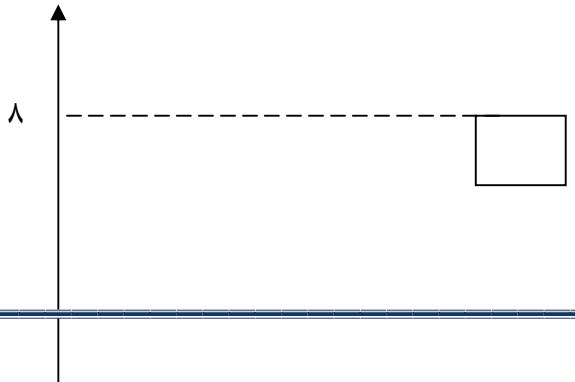
- ۲ باید به تأخیر بیافتد تا مشکل ۳ و ۵ حل شود
- ۳ باید به تأخیر بیافتد تا مشکل ۲ و ۵ حل شود
- ۵ باید به تأخیر بیافتد تا مشکل ۲ و ۳ حل شود
- ۲ و ۳ به تأخیر بیافتد تا مشکل ۵ حل گردد.
- ۲ و ۵ به تأخیر بیافتد تا مشکل ۳ حل کردد
- ۳ و ۵ به تأخیر بیافتد تا مشکل ۲ حل گردد.

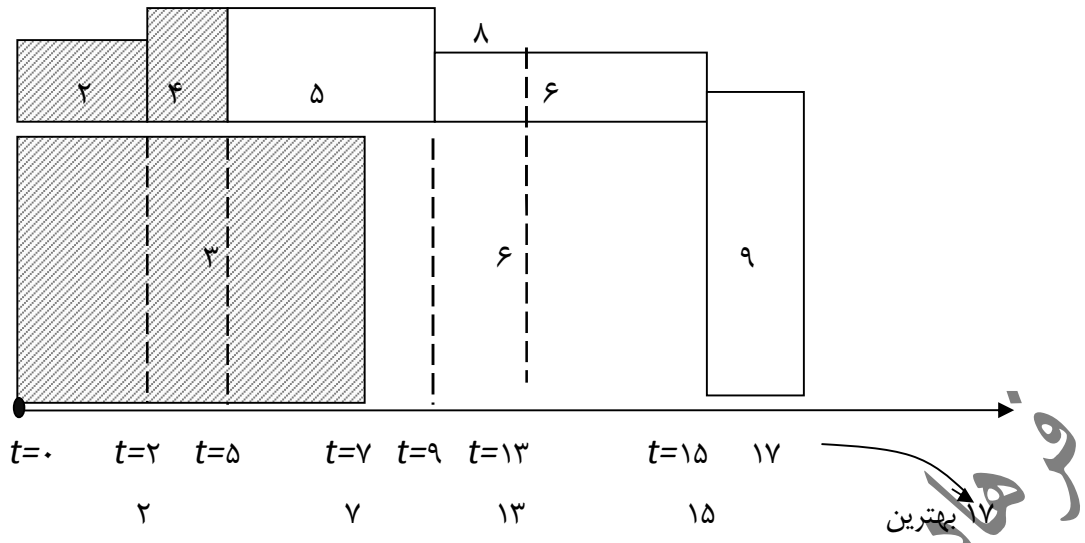
پس ۶ حالت می تواند اتفاق بیافتد. اگر شاخه بزنی باید با ۶ قسمت شاخه بزنی که تضمین بشود که همه حالت‌های ممکن داره چک می‌شود اما اگر چند ثانیه فکر کنید، سوال من این است ؟ اگر با به تأخیر افتادن ۲ مشکل حل می‌شود دیگر چه نیازی است که ۲ و ۳ را به تأخیر بندازیم وقتی فقط با به تأخیر انداختن ۲ می‌شود ***conflict*** را حل کرد ۳ چرا به تأخیر بیافتد ؟ بعضی از شاخه‌ها را جلوتر می‌بندیم به یکی از آن ۳ دلیل . ولی بعضی از شاخه‌ها را از ریشه می‌بریم یعنی شاخه اصلاً نمی‌سازیم نه اینکه جواب نیست چرا این شاخه هم یکسری جواب دارد ولی شاخه‌هایی هستند که از همان اول مشخص است که این شاخه منجر به بهینگی نیست این شاخه را مغلوب می‌گوییم.

Dominate شده یا مغلوب. در واقع ***alternative*** ، ***{۲,۳}*** یک ***alternatives*** است خلاصه یعنی یک راه برای حل مسئله است که شما ۲ و ۳ را به تأخیر بندازید که منبع آن آزاد شود که ۵ انجام گردد. ولی یک جواب به

اصطلاح مغلوب است چون شما فقط با به تأخیر انداختن ۲ می‌توانید مشکل را حل کنید. دیگه ۳ را وقتی به تأخیر می‌اندازید هیچ حسنی ندارد. جواب بهتری را به شما نخواهد داد. نتیجه آن کلمه اولی که در اسم این روش است کلمه چیست ؟ *minimal* است که دقیقاً می‌خواهد به این اشاره کند می‌گه با کمترین تأخیر یعنی فقط گزینه‌ها باید چک شوند که با کمترین تعداد که به تأخیر می‌افتد مشکل حل شود پس خلاصه در این ۶ تا حالت فقط به ۳ حالت انفرادی بررسی می‌شود دوتایی ها را نمی‌خواهد اما اگر بخواهیم قاعده مند کنیم بخواهیم فرمول بنویسیم چی باید بگوییم ؟ کدام *alternative* ها حذف می‌شوند اگر بگویید دوتایی ها حذف می‌شوند غلط است چون در این مثال دوتایی ها بودند در کل یک قانون کلی بخواهید بگویید این ۳ تا را می‌خواهم حذف کنم. با کمترین تعداد. منظور شما الان این است بین گزینه *{۳}* و *{۲,۵}* کدام حذف می‌شود ؟ یعنی بین *{۳}* و *{۲,۵}* ، *{۲,۵}* حذف می‌شود آره باید ربط بینشان باشد. بین *{۳}* و *{۳,۵}* آره ۳ ولی بین *{۳}* و *{۲,۵}* اینها همدیگر را حذف نمی‌کنند در واقع اینطور ی بگم اگر A زیرمجموعه B باشد $(A \subset B)$ ، B لازم نیست بررسی شود. ۲ زیرمجموعه کدام است ؟ *{۲,۳}* پس دیگه *{۲,۳}* لازم نیست. ۳ زیر مجموعه *{۳,۵}* است پس *{۳,۵}* لازم نیست ، ۵ زیر مجموعه *{۲,۵}* است و *{۲,۵}* لازم نیست پس همین جا یک خط بنویسید. اگر *alternative*، A زیر مجموعه B باشد فقط کافی است *alternative*، A بررسی شود در واقع *alternative*، B مغلوب است.





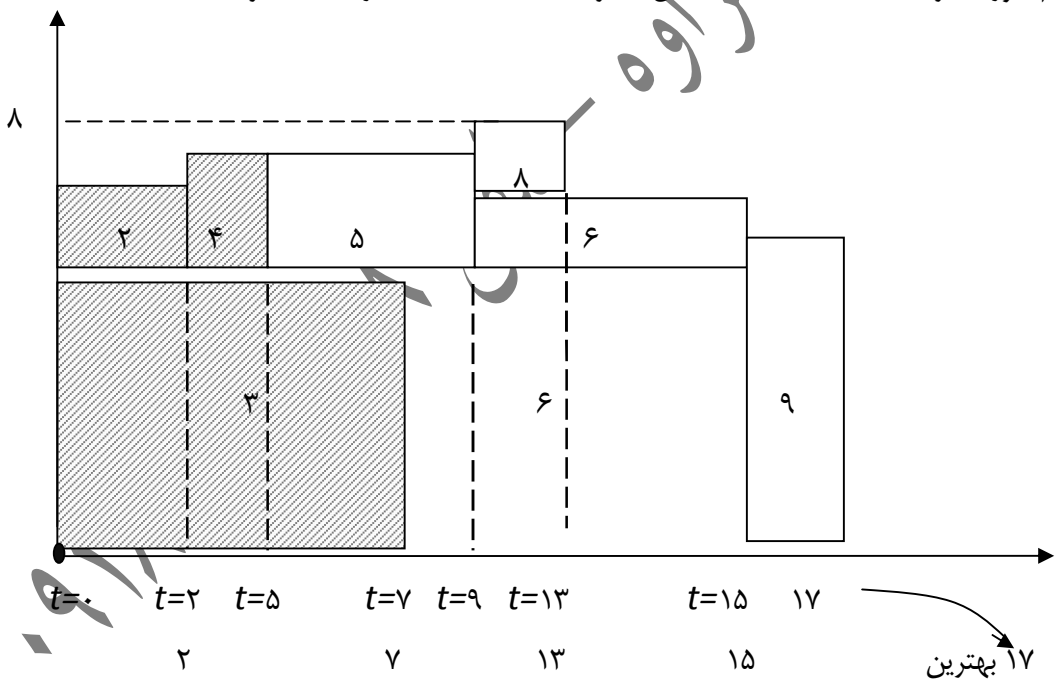


پس به این ترتیب ما اینجا فقط ۳ تا شاخه خواهیم زد. پس مسئله که شروع می‌شود به ۳ قسمت شاخه می‌زنیم. شاخه اول. اول بگم این گره‌ها چی هستند. اعدادی که پایین نوشته شدند همان ***delaying alternatives*** ها است. اعداد بالا سمت چپ شماره گره است و سمت راست ***lower bound*** است. الان اینجا ۲ و ۳ و ۵ به سه شاخه می‌شکنیم ***lower bound*** اش اگر ۲ به تأخیر بیافتد می‌شود ۱۷ و اگر ۳ به تأخیر بیافتد می شود ۱۵ و اگر ۵ به تأخیر بیافتد می‌شود ۱۳. این روش به صورت ***best first*** شاخه را پیش می برد یعنی گره‌ای را اول انجام می دهد که ***lower bound*** اش بهتر است الان اینجا کدام اول انجام می‌شود ؟ گره شماره ۲ که ***lower bound*** اش ۱۳ است یعنی ۵ باید به تأخیر بیفتد، ۵ به تأخیر بیافتد یعنی کدام‌ها انجام بشود ***{۳ , ۲}*** یادتان نرود مجموعه اصلی ***{۵ , ۳ , ۲}*** بوده است. ۵ به تأخیر بیافتد پس من اینجا فعالیت ۳ و فعالیت ۲ را در گانت می‌چینم ولی ۵ را نه چون ۵ قرار شد که فعلاً انجام نشود این در واقع لحظه صفر و این شاخه بود. آن یکی شاخه‌ها را بعداً صحبت می‌کنیم. الان شما اینطوری ببینید که ***{۱,۲}***، الان ***t*** بعدی کجا می‌شود الان در لحظه صفر بودیم ، ***t*** بعدی یعنی لایه بعد کدام ***t*** باید بریم. اولین جایی که منبع آزاد می شود ببینید الان منبع را اشغال کردیم ولی اولین جایی که منبع آزاد می‌شود کجاست فعالیت ۲ تمام شود روز دوم. خوب حالا از روی گانت ***{۱,۲}*** را بذارید کنار با ***{۱,۲}***کاری نداشته باشید. کدام فعالیت یا فعالیت‌ها می‌توانند انجام شوند ؟

{۴} {۵} {۳} . ۳ را هم باید بگویید چون هنوز تمام نشده است فعالیت‌هایی که هنوز تمام نشده‌اند جزء ***alternatives*** ها باید بگویید. پس ***{۴} {۵} {۳}*** خوب می‌توانیم زمانبندی کنیم یا نه ؟ چون ***{۴} {۵} {۳}*** ، ۴تا و ۴تا و ۳تا با هم می‌شود ۱۱ تا یعنی به اصطلاح ***resource conflict*** اینجا وجود دارد. دوباره باید همان داستان تکرار شود. به چند قسمت شاخه بزنیم یا باید ۴ به تأخیر بیافتد یا ۵ یا ۳ . دوتایی‌هاش را که نمی‌خواهد چون قرار شد فقط ***minimal*** ها فقط چک شود پس در ادامه این مثال اگر دقت کنید ما اینجا هستیم شاخه می‌زنیم به ۳ قسمت یا ۳ به تأخیر بیفتد یا ۴ یا ۵ اگر ۳ به تأخیر بیافتد***Lower bound*** اش ، ۱۸ است. بعدی ۱۷ و بعدی ۱۵ است و ما قرار شد ***best first*** عمل کنیم پس این را انجام می‌دهیم یعنی ۵ را به تأخیر می‌اندازیم آن دوتای دیگر یعنی ۳ و ۴

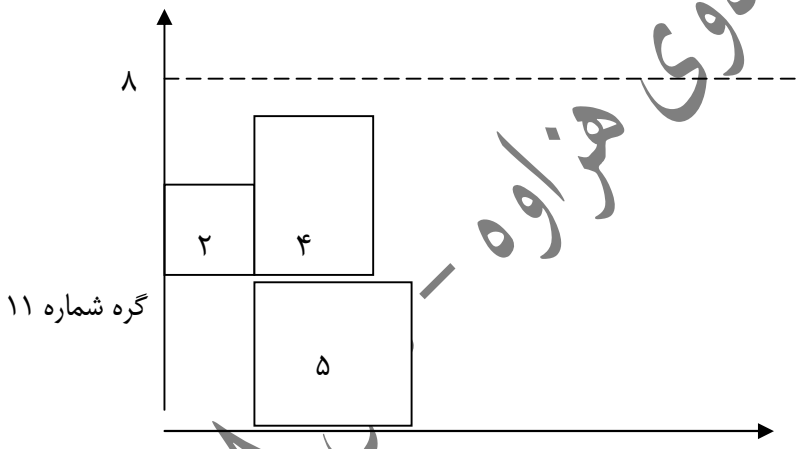
انجام می شوند البته ۳ که چیده شده ۴ را هم الان دارم اضافه می‌کنم. یک سوال اینجا پیش می‌آید؟ سوال ایشون این است ؟ ۳ که دیگه نباید به تأخیر بیافتد؟ فعالیت‌ها قطع نمی‌شود اگر قرار شد یعنی شما این شاخه را پیش برید ۳ را باید از اول جابه جاش کنید. ببینید ما الان پروژه را انجام نمی دهیم برنامه ریزی می‌کنیم اینطوری برداشت نکنید که ما الان وسط انجامش هستیم الان در حال برنامه‌ریزی هستیم پروژه قرار است ۳ ماه دیگه شروع شود.

ما برنامه را می‌توانیم دست کاری کنیم اینجا تصمیم گرفتید که از ۳صفر شروع کنید به تأخیر بندازیم و بگوییم از ۲ ام شروع شود پس ما الان داریم برنامه‌ریزی می‌کنیم پس ۳ باید بررسی شود به تأخیر افتادنش اگر بررسی نکنیم یعنی سلیقه‌ای نیست بگیم نه ۳ بررسی نشود چون دیگه آن را چیدم این بدون دلیل که نمی‌توانید بررسی‌اش نکنید باز آن جمله برای سومین بار تکرار می‌شود. همه جواب‌ها باید بررسی شود و حتی به تأخیر انداختن ۳ هم باید بررسی شود اگر بررسی نکنیم باز آن شبهه پیش می‌آید که جواب بهینه را از دست می‌دهد پس باید بررسی شود. سوال دیگری باید بپرسید ؟ ***lower bound*** ها چه طوری محاسبه شد؟ سوال این است؟ این ***lower bound*** ها ۱۷، ۱۵ ، ۱۳. پاسخ مثل جلسه قبل ؟ درست میگه؟ نه هر روشی ببینید جلسه قبل روش اول در آن ***feasible sequence*** داشت ***Lower bound*** از روی ***Sequence*** ها بدست می‌آمد اینجا ما الان ***sequence*** (توالی) نداریم هر روش ***lower bound*** حساب کردنش مختص است یعنی وقتی شما روش شاخه زنی فرق می‌کند قطعاً ***lower bound*** حساب کردن هم طور دیگری است اینجا باید یک مدل دیگر ***Lower bound*** را حساب کرد.



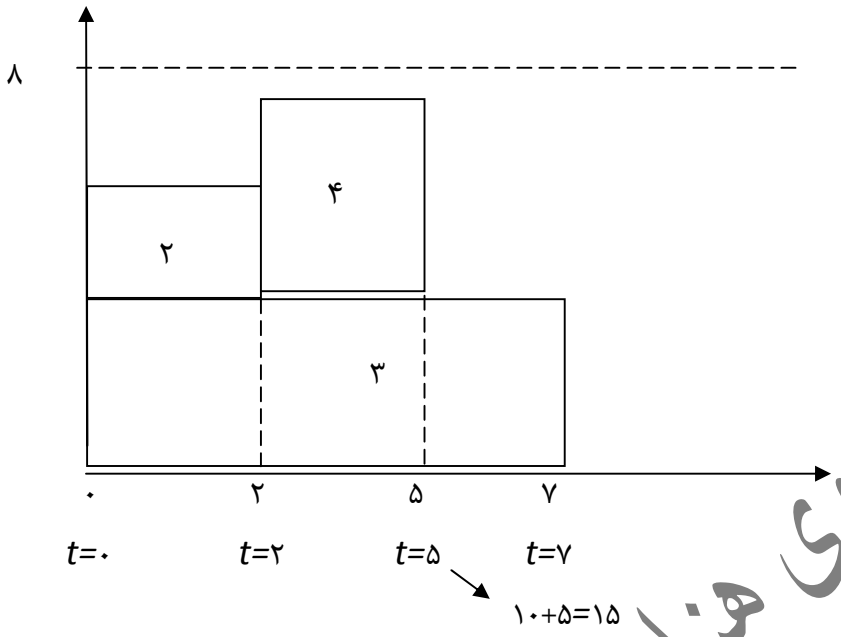
الانم همین را داریم نمی توانیم قطع کنیم. باید ۳ دو باره بررسی شود وقتی که فعالیت شروع شده ما نباید فعالیت را قطع کنیم بنابراین اینجا که شروع شده اگر عقب بندازیم قطع می شود ؟ می‌گویم نه . نه نباید قطع کنیم باید از اول کل مستطیل را کلاً باید جابجا کنید و ببرید جلو. آن تا آنجا زمانبندی بود لی الان تصمیم گرفتیم که ۳ عقب بیافتد اگر مثلاًمن بخواهم این شاخه را رسم کنم این می‌شود من این را رسم کنم من دارم این شاخه را انجام می‌دهم یعنی دفعه

اول شما کدام را چیده بودید . ۵ که به تأخیر افتاده بود ۲ و ۳ را چیده بودید. این ۳ اینم ۲ اینجا هم که ۸ است این که تا اینجا قبلاً دیده بودید. الان من در آن شکل این شاخه را رفتم ***best first*** عمل کردم به اصطلاح این ۱۵ بهتر بود اما اگر شما بخواهید این را انجام بدهید یعنی ۳ به تأخیر بیافتد پس روز ۲ ام چه اتفاقی باید بیفتد. کدام‌ها انجام بشوند ؟ ۳ را بذارید بعداً انجام بدهید پاکش کنید. ۳ که چیده شده بود را پاکش کنید قرار شد بعداً انجام شود. از کجا قرار شد؟ اینجا نوشته ، گفته ۳ بعداً انجام می‌شود پس کدام ها انجام بشوند؟ ۵ و ۴ پس این ۵ و اینم ۴ ، ۳ چی شد؟ ۳ به عنوان ***alternatives*** قرار شد ***delay alternatives*** بعداً انجام شود. خوب آن را که چیده بودیم باشه، چیده بودیم ولی الان منصرف شدیم الان به عنوان ***alternative*** تصمیم گرفتیم بعداً انجام شود. پس فعالیت‌هایی که تمام شدند نه دیگه نه، تمومه. آنی که چیده شده و زمانبندی شده و پایانش اتفاق افتاده دیگه تموم است ولی آنی که وسطهای کار است این به عنوان ***alternatives*** ایی که بعداً انجام بشود باید بررسی شود پس الان من گره شماره ۱۱ را اینجا رسم کردم. ۲ اول انجام شده با ۳ ولی الان تصمیم گرفتیم که ۳ بعداً انجام شود یعنی ۴ و ۵ در این لحظه انجام می شوند این گره ۱۱ می‌شود.



من یکی را با مثال توضیح می دهم بعد جمله‌ها را می‌گویم بنویسید. مثلاً همین گره شمارهٔ ۳ ایی که الان هستیم این ۱۵ از کجا آمده است ؟ شما الان کدام‌ها را داشتید می‌چیدید که تصمیم گرفتید که ۵ را به تأخیر بندازید ؟ ۳ و ۴ و ۵ قرار شد روز ۲ام انجام بشود که ***conflict*** داشت ، تصمیم گرفتیم ۵ به تأخیر بیافتد ۳ و ۴ انجام بشود که من انجام دادم اما ***Lower bound***از کجا ۱۵ شد؟ آنی که به تأخیر افتاده کدام است ؟ ۵ . باقیمانده مسیر بحرانی ۵ را از روی شکل بگویید چقدر می‌شود ؟ ۱۰=۶+۴ و ۱۰=۲+۴+۴ پس ماکسیمم ۱۰. باقیمانده مسیر بحرانی یعنی از ۵ به بعد ماکسیمم مسیر ۱۰ است به اضافه آن دوتایی که به تأخیر نیفتادند یعنی ۳ و ۴ . کدامشان زودتر تمام می شود ؟ ۴ . کی تمام می‌شود ؟ ۵ ام. آن ۱۵=۵+۱۰ که ***lower bound*** می‌شود. درست شد آنی که به تأخیر افتاده مسیر بحرانی باقیمانده‌اش باضافه آنهایی که به تأخیر نیفتادند زودترین زمان تکمیلش. آنی که زودتر تمام شده است ۳ و ۴ اینجا انجام شدند. ولی آنی که زودتر تمام شده یعنی منبع زودتر آزاد می‌شود روز پنجم است فعالیت ۴، روز پنجم است. الان ۵ قرار شد انجام ندهیم، ۵ پس انجام نشده ، اما خود ۵ به بعد چند روز کار مونده حداقل ۱۰ روز از روی باقیمانده مسیر

بحرانی ، ۵ اگر قرار باشد الان انجام دهم کی باید شروع شود اولین جایی که منبع آزاد شود. پنجمف ۱۰ روز کار است خودش از پنجم که شروع کنیم می‌شود ۱۵ روز.



نحوه محاسبه *Lower bound*:

باقیمانده مسیر بحرانی *alternatives* های تأخیری را محاسبه کنید (در مثال فعالیت ۵ بود) و آن عدد را به (حالا من این بقیه‌اش را می‌توانم به دو مدل بگویم هم می‌توانم بگویم به اولین جایی که منبع آزاد می‌شود اضافه کنید یا بگم به اولین جایی که فعالیت تمام می‌شود اضافه کنید فرقی نمی‌کند به زودترین زمان تکمیل فعالیت‌های به تأخیر نیفتاده اضافه کنید (به عبارت دیگر به اولین زمانی که منبع آزاد می‌شود، اضافه کنید) (در مثال ۵ بود).

اگر این *Lower bound* کمتر باشد از *Lower bound* ایی که محاسبه شد در *parent node* (در شاخه و گران به گره بالایی *parent node* و به گره پایینی *child node* گفته می‌شود) باشد *Lower bound* ، *parent node* را استفاده کنید.

اگر *LB* محاسبه شده از(*LB*) *Lower bound* گره بالاسری کمتر باشد ، *LB* گره بالاسری ثبت می‌شود.

در این مثال گره ۲ کدام به تأخیر افتاده است ؟ ۵ . باقیمانده مسیر بحرانی ۵ را حساب کردیم که شده بود ۱۰ . ۱۰ را با چی اضافه کرده ؟ آنهایی که به تأخیر نیفتاده بودند ۲ و ۳ بودند ۲ کی تمام شده ؟ ۲ پس ۱۲=۲+۱۰ =*LB* پس *Lower bound* این شده ۱۲ در حالی که گره *Parent* اش یعنی بالایی ، *Lower bound* اش چند است ؟ ۱۳ . یعنی *Lower bound* ای که شما حساب کردید از *Lower bound* گره بالایی کمتر شده چی کار کرده است؟ اینجا ۱۲ را ننوشته است ۱۳ را نوشته است. چرا ؟ دقیقاً جمله‌ای بود که شما نوشتید. چرا باید این کار را بکنیم ؟ منطق این قضیه را می‌خواهیم صحبت کنیم. چرا اینطوری است ؟ چرا باید بالایی را بنویسیم.

LB وقتی کمتر می شود به درد نمی‌خورد باید آن بالایی نوشته شود. منطقش ؟

فرض کنید دیشب یک مسابقه فوتبال بوده است که شما مشغول دیدن بازی بودید تا جایی که بازی را تماشا کردید ۲ تا گل داشته یک تیمی ۲ به هیچ جلو بوده است. خوب بعد شما خوابتان برده است صبح از دوستان سوال می‌کنید که بازی دیشب چی شد ؟ ایشان در جواب می‌گوید من تا یک گل را دیدم که یک به هیچ جلو بود. شما خودتان تا کجا دیده بودید ؟ ۲ به هیچ. یعنی از نظر شما که تا قبل از اینکه از دوستان سوال کنید این بازی چند تا گل داشته حداقل ۲ تا . شما حداقل ۲ تا گل را دیدید. ولی الان ایشون چی می‌گه حداقل یک گل داشته است. من یک گل را دیدم. نتیجه عملاً صحبت آن فرد هیچ اطلاعاتی را به شما اضافه نکرد. حداقل ۲ تا گل داشته است بنابراین اگر دوست دیگری که بعد از شما خوابش برده باشد بازی را تا ۳ به ۱ دنبال کرده باشد نتیجه ***Lower bound*** ، ۴ است ایشان ۴ تا گل را دیده است ***Lower bound*** بالاتر، عدد بالاتر ، اطلاعاتی را دارد به ما می‌دهد که ما قبلاً نداشتیم الان متوجه شدیم. حداقل ۴ تا گل داشته است ولی عدد کمتر ، ***Lower bound*** کمتر اطلاعاتی را عملاً منتقل نمی‌کند.

اینجا از همان بالا داریم می‌بینیم که پروژه ۱۳ روز طول می‌کشد ولی الان به نتیجه رسیدم که پروژه حداقل ۱۲ روز طول می‌کشد. دقیقاً همین بحث بازی است دیگه. شما اطلاعات ۱۳ روز براتون محرز شد که ***Lower bound*** ، ۱۳ است. حداقل این پروژه ۱۳ روز طول می‌کشد. الان فهمیدید که حداقل ۱۲ روز. یعنی عملاً چیزی اضافه نشده به داده‌های شما به اطلاعات شما . نتیجه همان ۱۳ است . اما دفعه بعد که متوجه می‌شوید پروژه ۱۵ روز طول می‌کشد الان موضع تان قوی‌تر شد. الان اطلاعاتان بیشتر شد به جای ۱۳ روز. ۱۵ . به جای ۱۵. ببینید وقتی بیشتر می‌شود بیشتر نوشته می شود ولی وقتی که کمتر بشه همان قبلی ثبت می‌شود این در واقع منطقی است که اینجا داره رعایت می‌شود خوب ما الان کجا هستیم در گره شمارهٔ ۳ که ۵ به تأخیر افتاده ۴ و ۳ چیده شدند. خوب ***t*** بعدی کجا می‌شود. حداقل یک شاخه را تا عمق پیش برویم اول آنجا از روی شکل بگویید. اول صفر بود بعد ۲ ، الان ***t=۵*** را باید بگویید اولین جایی که منبع آزاد می‌شود. دیگه ۲ و ۴ را کاری نداشته باشید چون ۲ و ۴ تمام شدند، زمانبندی و تکمیل ولی ۳ هنوز تمام نشده پس یک که مجازی بود ۲ و ۴ کنار، بقیه ۶ و ۵ و ۳ به عنوان فعالیت‌هایی هستند که می‌توانند انجام شوند که البته بازم ۳ تا ، ۴ تا و ۳ تا یعنی ۱۰ تا پس ***conflict*** منبع بازم وجود دارد. کدام به تأخیر بیافتد ۳ حالت دارد.

پس یا ۳ یا ۵ یا ۶ باید به تأخیر بیافتد. اومده ***Lower bound*** را طبق مکانیزمی که الان شما نوشتید حساب کرد، ۳ به تأخیر بیافتد ***Lower bound*** ، ۱۷ است ۵ به تأخیر بیافتد ***Lower bound*** ۱۷ و ۶ به تأخیر بیفتد ***Lower bound*** ۱۵ می‌باشد. که بازم مبنا به صورت ***best first*** یعنی ۶ بعداً انجام بشود. ۶ را انجام نمی‌دهم ۳ و ۵ را می‌چینم. ۳ البته که چیده شده است. ۵ را همین جا اضافه می‌کنم. خوب ***t*** بعدی. دیگه الان عملاً حرفی برای گفتن ندارم این‌ها تکرار یک کار ثابتی است تا به نتیجه برسیم. ***t*** بعدی . در واقع ***t=۷*** است. کدام باید انجام شود ؟ ۱ و ۲ و ۳ و ۴ انجام شدند. کدام یک باید انجام شود این را بگذارید کنار. ۶ و ۵ چند تا منبع می‌خواهد. ۷=۴+۳ نتیجه برای اولین بار اتفاق افتاده در این مثال که اصلاً ***conflict*** منبع وجود ندارد. اگر ***conflict*** منبع وجود ندارد خیلی خوب

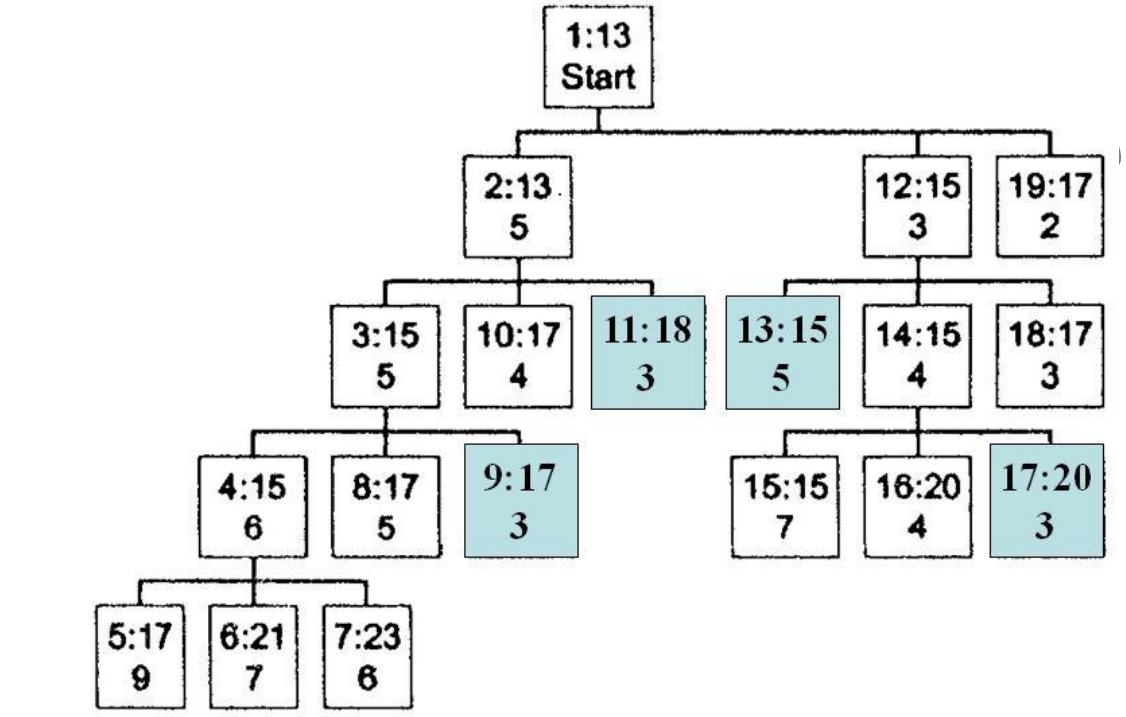
است شما بچینید. دیگه مشکلی وجود ندارد. شاخه‌زنی چی ؟ دیگه شاخه‌زنی نیازی نیست ، شاخه‌زنی فقط وقتی است که *conflict* اتفاق می‌افتد والا اگر منبع وجود دارد دیگر نیازی به تأخیر انداختن فعالیت نیست پس من چی کار کنم در کنار ۵ ، ۶ را هم بچینم هیچ مشکلی پیش نیامد. منبع ۸ تا بود اینها به ۷ تا منبع نیاز داشتند. t بعدی ۹ است بگویید.اینجا را چه کار کنیم؟ هم گانت را نگاه کنید هم گراف را . ۶ و ۷ و ۸ تأخیر نمی‌خواهد. ۸=۳+۲+۳ پس در کنار ۶، ۷ و ۸ را می‌چینیم. اگر دقت کنید الان ۲ سری است که سراغ *tree* نرفتیم. چون *conflict* ایجاد نشده بود و نیازی به تأخیر انداختن فعالیت نبود. t بعدی ۱۳ است. کدام‌ها می‌توانند انجام بشوند؟ ۶ و ۷ و ۹. اینبار دیگه *conflict* اتفاق افتاده است مجبور هستم که شاخه‌زنی را انجام بدهم. . ۶ به تأخیر بیافتد *Lower bound* ۲۳ و ۷ به تأخیر بیفتد *Lower bound* ، ۲۱ و ۹ به تأخیر بیافتد *Lower bound* ، ۱۷ است. مینیمم ۱۷ . پس ۹ بعداً انجام بشود ۶ و ۷ هم که چیده شدند. t آخر ۱۵ است در ۱۵ کدام می‌تواند انجام شود؟ فقط ۹ مانده است. *make span* اش چی شد؟ این شاخه چی شد ؟ *fathoming* یا به عمق‌رسیدن. این شاخه را تا آخر فعالیت‌ها را چیدیم و به اصطلاح به عمق رسید یک جواب بدست آمد که روی برد گانت می‌بینید. مقدار تابع هدف‌اش چند است ؟ ۱۷ . این جواب را به اصطلاح می‌گوییم جواب اولی که پیدا کردیم این را ذخیره می‌کنیم و در بایگانی نگه می‌داریم. به عنوان بهترین جوابی که تا این لحظه پیدا کردیم. ادامه چی ؟ شاخه‌های دیگر چی ؟ من دنبال جواب نیستم. دنبال جواب بهتر از ۱۷ هستم چون که ۱۷ را دارم. بنابراین شاخه‌های که *Lower bound* ، ۱۷ یا بالاتر دارند بسته می‌شوند.

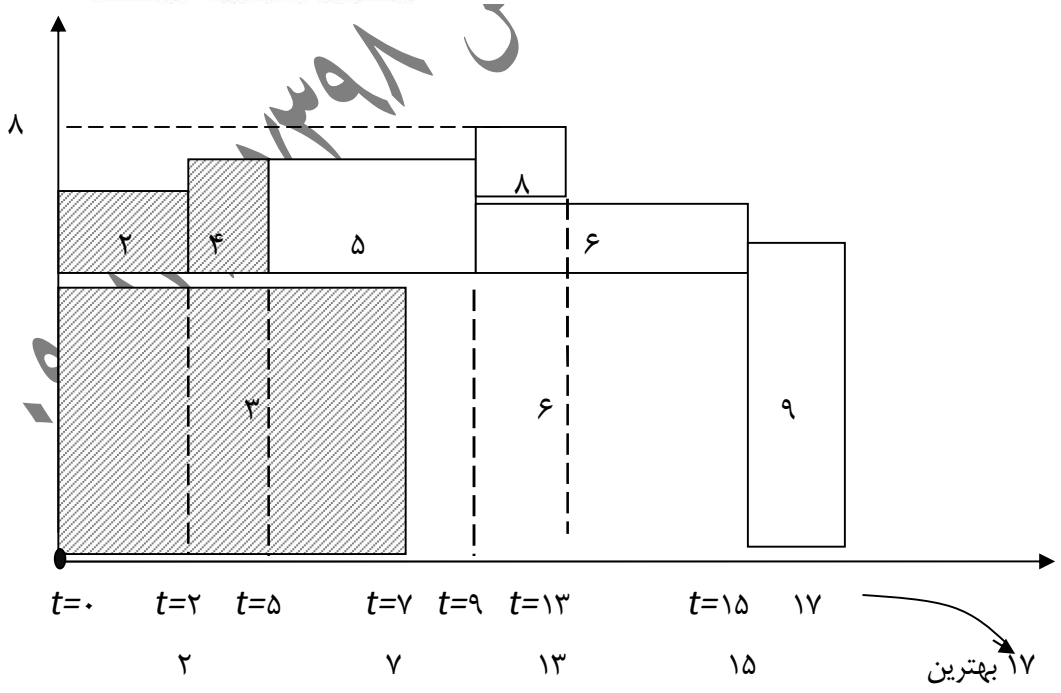
۱ *Bounding role* چرا تا حالا ننوشتیم. چون ما تا الان جوابی فعلاً پیدا نکرده بودیم یعنی شما تا یک جواب نداشته باشید نمی‌توانید با *Lower bound* هیچ شاخه‌ای را ببندی ولی الان که ۱۷ را پیدا کردید دیگه خیالت راحت است یک جواب ۱۷ را دارید. پس آنهایی که بالای ۱۷ هست را می‌توانم ببندم . آن دلیل را می‌خواهم بگویم باید دلیل داشته باشی برای بستن‌اش . باید محرز شود که این جواب، این شاخه بی‌فایده است مثلاً این شاخه را دقت کنید همین شاخه ۱۱ را. *Lower bound* اش چند است ؟ ۱۸ . این فارسی‌اش چند می‌شود. یعنی چی ؟؟ یعنی با ادامه دادن این شاخه جوابی که شما بدست می‌آورید حتماً ۱۸ و بالاتر است *make span*اش. همین دلیل کافی است. من ۱۸ به بالا را نمی‌خواهم چون جواب بهتر را دارم من دنبال جواب بهینه‌ام یعنی جواب مینیمم باشد *make span*اش. ۱۸ بالاتر به درد شما نمی‌خورد.

۱ *Bounding role* : اگر در یک گره *LB* محاسبه شده از $C_{\max}$ (تابع هدف) بهترین جواب یافت شده تا این لحظه بیشتر یا برابر باشد شاخه بسته می‌شود

الان با همین جمله کدام‌ها بسته می‌شوند ؟ از همین جا ، ۲۱ ، ۲۳ و ۱۷ و ۱۷ و ۱۷ و ۱۸ و ۱۷ اینها بسته شدند ولی این چند است؟ ۱۵ . این را نمی‌توانید ببندید و اگر دقت کنید بسته نشده است ادامه داشته است؟ ادامه‌اش چی شده است ؟ ۱۷ و ۱۵ و ۱۵ حالا ۱۷ را می‌توانید ببندید اما این ۱۵ ها را باز نمی‌توانید ببندید . این ۱۵ را باز باز می‌کنید می‌شود ۲۰، ۲۰ ، ۱۵ . این ۲۰ ها بسته می شوند این ۱۵ به عمق رسیده است خوب به عمق رسیده ما دیگه قرار نیست انجامش بدهیم فقط گزارش می‌کنیم این که به عمق رسید دفعه قبلی ۱۷بود این چند شد ؟ ۱۵ . این بهتر است قبلی را پاک می‌کنیم الان دیگه توقع ما زیاد شده الان جواب بهتر از ۱۵ می‌خواهیم که ظاهراً هیچ شاخه‌ای بهتر از

این را نداشته است پس گره ۱۵ ، *optimal solution* را نشان می‌دهد.. که ما آن را انجام ندادیم. می‌توانستیم انجام بدهیم فرقی نمی‌کند یک شاخه تا آخر رفتیم دیگه انجام دادن شاخه‌های دیگر شبیه است و نیازی به تکرار نیست پس گره ۱۵ ، جواب بهینه را به شما داده است. گره ۱۳ چرا بسته شده است ؟ چون گره ۱۳ را شما کی بسته‌اید ؟ دقت کنید ببینید شما ۱۵ را الان پیدا کردید در حالی که ۱۳ را قبلاً بسته‌اید .۱۳ را به دلیل دیگری بسته‌اید.
۲ *Bounding role* ، *left shift dominance* که قبلاً با آن آشنا شده بودید.

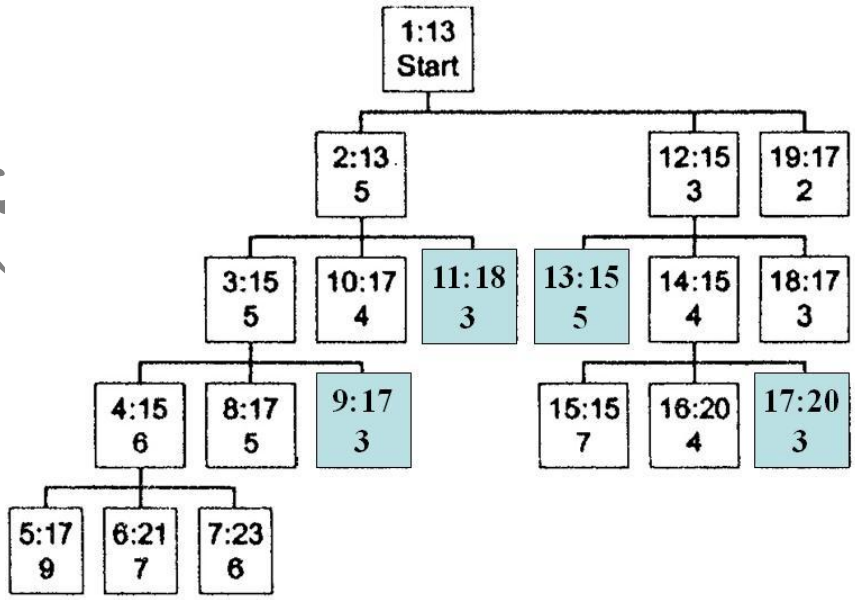




۲ *Bounding role* ، *left shift dominance*

این که هاشور زدم کدام گره است این جواب را از کدام شاخه بدست آوردید. از این شاخه که با هم طی کردیم که این ۳ تایی که الان هاشور زدم کی چیده شده بود در گره شمارهٔ ۳ . این که هاشور زدم گره شماره ۳ است حالا گره شماره ۱۳ را با هم انجام بدهیم در گره ۱۳ از آن بالایی که شروع می‌کنیم ، ۵ را اول به تأخیر انداختیم. اما در این گره اول کدام باید به تأخیر بیافتد ؟ ۳ . یعنی کدام ها انجام بشوند ؟ یعنی ۲ و ۴ و دفعه بعد ۳. این الان گره ۱۳ است. در حالی که گره ۳ کدام است ؟ گره ۳ البته هاشور زدم هست ولی حالا اینجا هم من دوباره آوردمش. اینها یکی هستند این ۴ بالا است و اینجا پایین اش. اصلاً مهم نیست بالا و پایین بودن مهم نیست. روی محور افقی موقعیتش مهم است روی محور عمودی مهم نیست. دقت کنید اگر به گره ۱۳ نگاه کنید فعالیت ۳ به اصطلاح قبلش یا پشت سرش منبع آزاد دارد و پیش نیاز هم ندارد نتیجه : می‌توانید ۳ را شیفت بدهید به سمت چپ اگر این کار را انجام بدهید پاسخ چی می‌شود ؟ گره ۱۳ می‌شود همان گره ۳. نتیجه : آن را قبلاً انجام دادیم یعنی شما اگر توجه کنید در گره ۱۳ ، ***left shift dominance*** اتفاق می‌افتد می‌شود دقیقاً گره ۳ . اگر ادامه بدهید چی می‌شود ؟ همین را دوباره کپی کنید. خوب ما یک بار انجام دادیم برای چی یکبار دیگه عین همان جواب عین همان مسیر را دوباره برویم تکراری است . ***Left shift dominance*** شاخه‌های را که تکراری هستند را می‌بندد و فقط وقتی قابل استفاده است که فعالیتی پشت سرش منبع آزاد داشته باشد یکبار نوشته‌اید ولی چون روش دیگری است ناچار به تکرارش هستیم.

۲ ***Bounding role*** ، ***left shift dominance*** : اگر فعالیتی در یک گره در زمانبندی ایجاد شده بتواند به سمت چپ جابجا شود این زمانبندی تکراری است و شاخه مربوط به آن بسته می‌شود.



خوب این ۳ تا که رنگی شدند قضیه‌اش چیست ؟ اینها را هایلات کردم که بگویم اینها به دو دلیل بسته شدند هم با ۱ ***bounding role*** هم با ۲ ***bounding role*** یعنی ***left shift dominance*** توی گره‌های ۹ و ۱۱و ۱۷ هم هست. فوقش من تو گره ۱۳ بررسی کردم. آیا لازم است هر دو دلیل چک شود ؟ خیر چون با یکی که بسته شود

می‌توانید با آن یکی دیگر چک نکنید. ولی فقط می‌خواستم بگویم که بعضی از شاخه‌ها در اینجا با هر دو دلیل می‌توانستید ببندید تمام.

روش سوم *minimal delaying alternatives* بود که در چند گره مسئله حل شد ؟ ۱۹ تا در حالی که در روش‌های قبلی با چند تا گره حل کردیم ؟ در روش اول با ۵۰ تا گره و در روش دوم با ۳۵ یا ۳۶ گره. تعداد گره‌ها در این روش سوم کمتر است. ولی لزوماً به این معنی نیست که این روش سریعتر است این را قبلاً بحث کردیم چون تعداد گره‌ها اصلاً مبنای زمان نیست شما ممکن است ۱۹ تا گره داشته باشید ولی هر گره متوسط یک دقیقه طول کشیده که بررسی شده است که می‌شود ۱۹ دقیق اما آن روشی که ۵۰ تا گره داشت ممکن است هر کدام در دو ثانیه طول بکشد. خوب ۵۰ تا دو ثانیه کمتر است یا ۱۹ تا یک دقیقه. خوب آن را ترجیح می‌دهیم ۵۰ تا دو ثانیه‌ای کمتر است پس به هیچ عنوان تعداد گره مبنا نیست باید چی کار کنیم ؟ ۴۰ تا مسئله با هر سه روش حل کنید و ببینید متوسط زمان *CPU time* کدام کمتر است؟

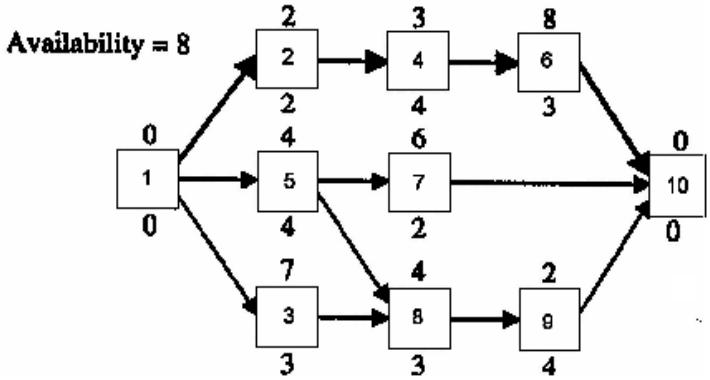
روش آخر روشی به اسم *minimal forbidden sets* (کمترین مجموعه ممنوعه) که افرادی به نام ایگل موند و رادارمارکر پیشنهاد کردند. *Forbidden set* را می‌توانید مجموعه ممنوعه ترجمه کنید. کمترین مجموعه ممنوعه معمولاً بهتر است با خود اصطلاحات کار کنید تا ترجمه‌ها.

قبل از توضیح روش *Minimal forbidden set* دو تا تعریف را باید بگویم.

تعریف چی را باید بگویم؟ خود همین اصطلاحات *forbidden set* یعنی چی اول ؟ بعد *Minimal forbiddent sets* یعنی چی ؟

Forbidden set تعریف می شود به عنوان مجموعه‌ای از فعالیت‌ها که می‌توانند زمانبندی شوند به طور همزمان در طی پروژه (یعنی هیچ پیش‌نیازی بینشان وجود ندارد) و اما اگر *Perform* (انجام) بشوند به طور همزمان، محدودیت منبع را نقض می‌کند. یک تعداد فعالیت که پیش نیاز هم نیستند ولی با هم نمی‌توانند انجام شوند چون منبع نقض می‌شود. به این می‌گویند *forbidden set*

من چند تا درس را اسم‌هاشون را می‌گویم زمانبندی پروژه، تئوری فازی، دوتا *data minig* ، توالی عملیات، تئوری صف. ۵ تا درس این ها پیش نیاز هم نیستند ولی شما اینها را نمی‌توانید با هم داشته باشید درست می‌گم در یک ترم نمی‌توانید ۵ تا درس را داشته باشید. درست است که پیش نیاز هم نیستند ولی به خاطر محدودیت واحدی که دارید نمی‌توانید همه ۵ تا درس را با هم داشته باشید می‌شود *Forbidden set* . با هم نداشتن اش به خاطر پیش‌نیازی نیست به خاطر محدودیت است. محدودیت منبع حالا. در قضیه شما به خاطر محدودیت تعداد واحد. این را می‌گویند *Forbidden set* .



حالا اگر براتون موضوع جا افتاد شما در این مثال بگویید که {۲,۵,۸} آیا *forbidden set* هستند یا خیر ؟ خیر. ۵ پیش نیاز ۸ می‌باشد. پیش نیاز نباید باشد. ببینید دو تا شرط دارد. پیش نیاز نباشند و منبع نقض بشود.

{۶,۷,۸} آیا *forbidden set* هستند یا خیر ؟ بازم خیر. این بار برعکس قبل است پیش نیاز نیستند باشه درست ولی شرط منبع باید نقض شود. نقض نشده است.

{۴,۷,۹} آیا *forbidden set* هستند یا خیر؟ بله این بار *forbidden set* هستند چون اولاً اصلاً پیش نیاز هم نیستند. حتی یکی هم نباید پیش نیاز باشد (به هیچ عنوان) و از طرفی هم ۱۰ تا منبع می‌خواهد که نداریم پس *forbidden set* می‌شوند.

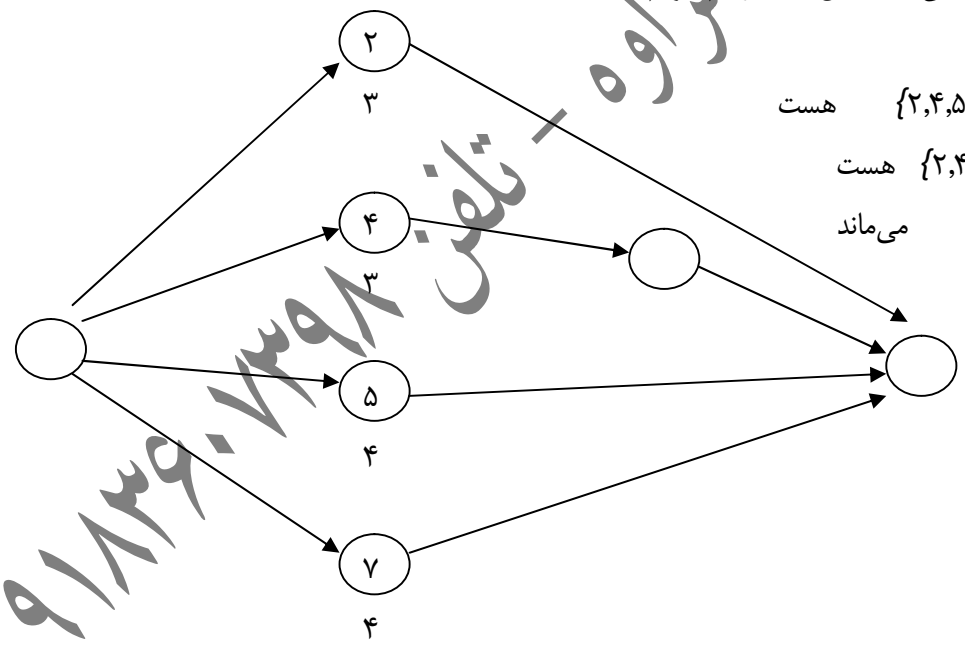
این شبکه چند تا *forbidden set* دارد

{۲,۳,۵}, {۳,۴,۵}, {۳,۴,۷}, {۳,۵,۶}. {۴,۷,۸}, {۴,۷,۹} and {۶,۷,۹}.

این شبکه خاص ۷ تا *forbidden set* دارد.

تعریف بعدی *minimal forbidden sets* می‌باشد . یعنی چی ؟

خوب لطفاً اینجا باشید من یک مثال دیگه رسم کردم



فرض کنید *avalibility*یعنی منبع موجود ۸ تا است این ۴ تا فعالیت را که نوشتم {۲,۴,۵,۷} این *forbidden set* هست یا نه ؟ خیلی واضح است که می‌باشد چون اولاً پیش نیاز هم نیستند ثانیاً ۱۴ تا منبع می‌خواهد که ما کلاً۸ تا منبع داریم. پس این *forbidden set* است. حالا {۲,۴,۵} چی ۷ را پاک کردیم ، علیرغم اینکه یک فعالیت را برداشتیم. ؟ بازم *forbidden set* است علیرغم اینکه یک فعالیت کسر شد. این چنین *forbidden set* را دیگه *minimal* نمی‌گویند . *forbidden set minimal* یک *forbidden set* مثل این را می‌گویند *minimal* که اگر شما یک فعالیت کسر کنید یا بردارین بقیه دیگه *forbidden set* نیست.*minimal forbidden set* ،

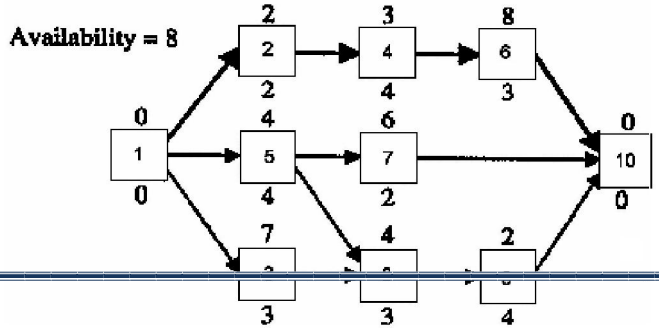
یک *forbidden set* ایی است که با حذف یک فعالیت از آن ، بقیه دیگه *forbidden set* نباشند. پس کلمه *minimal* به خاطر این است حالا چرا یک مثال دیگه زدم این مثال ۷ تا *forbidden set* دارد حالا شانسی است که همه ۷ تا *minimal*هستند. یعنی هیچ *forbidden set* غیر از *minimal* دیگه نداره و من مجبور شدم با یک مثال دیگه موضوع را بگویم. پس الان *{۲,۴,۵,۷}* ، فقط *forbidden set* است کلمه *minimal* براش درست نیست چرا *minimal* نیست به خاطر اینکه با حذف یکی از آنها بازم بقیه *forbidden set* می‌مانند. *Minimal forbidden set* یعنی کمترین تعداد را باید داشته باشد یعنی با حذف یکی از آنها دیگه بقیه‌اش نشد.

Minimal forbidden set عبارت است از *forbidden set* ایی که با حذف یک عضو آن باقیمانده مجموعه *forbidden set* نباشد.

این روش *minimal forbidden set* دنبال چیست ؟ که این مجموعه‌ها را به اصطلاح از هم باز کند ، گره‌ها را از هم باز کند. وقتی سه تا فعالیت *{۷,۴,۳}* با هم نمی‌توانند انجام شوند چه طور می‌توانیم این موضوع را حالی کنم ؟ من نمی‌خواهم شما این ۴ تا درس را همزمان با هم بگیرید چه طوری می‌توانم کاری کنم که شما در انتخاب واحد نتوانید اینها را با هم بگیرید؟ یکی را پیش نیاز آن یکی کنم . درست است مگه نه ؟ اگر من به عنوان مدیر گروه نخواهم که شما این ۴ تا درس را با هم بگیرید می‌توانم یکی را در سیستم پیش نیاز آن یکی کنم که به شما *error* بدهد وقتی انتخاب واحد می‌کنید نتونید اینها را با هم ثبت کنید. دقیقاً همچین مکانیزمی را اینجا استفاده می‌کند. میگه این *forbidden set* را اول شناسایی کنید مثلاً این ۷ تا بودند حالا در هر *Level*از شاخه و کران دنبال این است که یکی از اینها را مشکلش را حل کند چه طوری حل کند یک فعالیت این مجموعه را پیش نیاز یکی دیگه بکنه مثلاً شما برای اینکه مشکل { ۲ و ۳ و ۵ } را حل کنید چند تا راه دارید برای اینکه این *conflict* را از بین ببرید.

- ۲ پیش نیاز ۳
- ۳ پیش نیاز ۲
- ۲ پیش نیاز ۵
- ۵ پیش نیاز ۲
- ۳ پیش نیاز ۵
- ۵ پیش نیاز ۳

یعنی ۶ حالت. البته در این مثال ۶ حالت است چون مجموعه شما ۳ تایی است اگر ۴ تایی یا ۵ تایی بودند دو به دو حالت‌های بیشتری را چک کنید. در این مثال چون مجموعه ۳ تایی هستند هر کدام ۶ تا شاخه می‌خورد. پس این نحوه رفع این *conflict* است یعنی به همین صورت همه شاخه‌ها را باز می‌کند تا آخر سر. خلاصه باید همه این ۷ تا *forbidden set* را حل کند اما نکته جالبی اینجا وجود دارد که اول کدام را انجام بدهیم در این ۷ تا. فرقی می‌کند ؟ مگر قرار نیست همه انجام بشوند فرقی می‌کند که کدام اول انجام بشود؟ نه



ولی این اومده یک کار جانبی را انجام داده است در این روش قبل از اینکه شاخه زنی را انجام بدهد آمده گفته بعضی از اینها با هم دیگه اشتراک دارند. مثلاً اگر ۳ را پیش نیاز ۵ کنید چه اتفاقی می‌افتد کدام‌ها شکسته می‌شوند از آن بالا . این شکسته می‌شود یا نه ؟ شکسته شدن منظور این است که مشکل حل می‌شود اگر ۳ را پیش نیاز ۵ کنید این *forbidden set* حل می‌شود این چی ؟ اینم ۳ و ۵ داره حل می‌شود. این یکی چی ؟ یکم فکر کنید. نه دیگه شما اگر ۳ را پیش نیاز ۵ کنید عملاً به واسطه ۵ ، ۳ را پیش نیاز ۷ هم کردید پس به صورت حالا مستتر است خوب این هم حل می‌شود. ۳ و ۵ پس با یک پیش‌نیازی ۳ با ۵ داری ۴ تا را با هم حل می‌کنید. اما مثلاً اگر شما ۶ را پیش نیاز ۹ کنید آن وقت چه اتفاقی می‌افتد ؟ اگر ۶ را پیش نیاز ۹ کنیم فقط این یکی را حل کردیم. یعنی منظور این است که انتخابش تأثیر دارد بعضی از اینها همپوشانی دارند، همپوشانی زیادی دارند بعضی‌ها همپوشانی کمی دارند.

این اومده برای هر کدام از این *forbidden set* ها به اصطلاح یک کار اولیه انجام داده که بهش می‌گویند پیش پردازش. پیش پردازش یعنی اینکه قبل از اینکه مسئله را حل کنید به اصطلاح حلاجی بکنی یه حساب کتابی بکنی یه تصمیم بهتری بگیری، یه ذره هوشمندش کرده است. اومده برای هر کدام یک امتیازی حساب کرده است بعد *forbidden set* ایی را انتخاب کرده که اول چک کند این امتیاز همپوشانی اش بالاتر است درست شد برای این امتیاز حساب کردن، یک ماتریس‌ایی را به این صورت در نظر بگیرید که البته اینجا انجام شده است ولی در واقع یک ماتریس خالی باید باشد که باید پر شود. سطر و ستون اول فعالیت‌ها را نوشته البته مجازی‌ها را دیگه نیاورده است . ۱ و ۱۰ که مجازی هستند نه فقط ۲ تا ۹ و اونجا هم ۲ تا ۹ فعالیت های اصلی و مجازی هستند. قطر اصلی هیچی نمی‌خواهد در آن بنویسید من خط تیره زدم بقیه خانه‌ها سه تا حالت دارد چی می‌بینید در این جدول یا یک عدد مثل (۲، ۱، ۳، ۲،۵) یا *X* و یا *Lower bound (LB)* می‌بینید. هر کدام از اینجا یک توضیح بدهم کافی است مثلاً این یعنی چی ؟ یعنی ۲ است یا این ۵ یعنی چی ؟ این ۲ به این معنی است اگر شما ۵ را پیش نیاز ۴ کنید ۲ تا از آن ۷ تا *forbidden set* حل می‌شود درست شد پس عددها چه معنی می‌دهند.

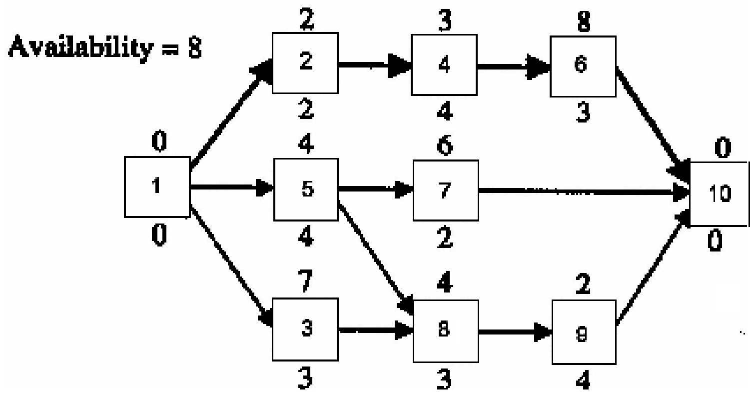
	2	3	4	5	6	7	8	9
2	-	1	X	1	X	0	0	0
3	LB	-	LB	LB	1	1	X	X
4	X	LB	-	5	X	3	2	1
5	LB	LB	2	-	1	X	X	X
6	X	LB	X	LB	-	LB	LB	2
7	LB	LB	LB	X	LB	-	3	2
8	LB	X	LB	X	LB	LB	-	X
9	LB	X	LB	X	LB	LB	X	-

عدد : هر عدد نشان می‌دهد که با اضافه کردن این پیش نیازی چه تعداد از ***forbidden set*** ها حل می‌شود.

X* :** نشان می‌دهد که یک پیش‌نیازی از قبل وجود داشته است. (مثلاً ***(۲,۴) در جدول بالا ***X*** گذاشته شده است که نشان می‌دهد که خود به خود این پیش‌نیازی از قبل هست)

یعنی این هیچ امتیازی وجود ندارد که اگر این را اضافه کنید. چون از قبل بوده است. پس خانه‌هایی که پیش نیاز هستند را شما دوباره اضافه نکنید.

LB: نشان می‌دهد که این پیش نیازی منجر به این می‌شود که طول مسیر بحرانی پروژه از بهترین جواب موجود بیشتر یا برابر شود.



مثلاً خانه ***(۳,۲)***، ***LB*** نوشته شده است. چرا ؟ من برگردم از روی شبکه بگویم. ۳ را پیش نیاز ۲ کنید. مثلاً اگر ۳ را پیش نیاز ۲ کنید مسیر بحرانی شبکه را حساب کنید خیلی راحت می‌توانم بگویم چند روز می‌شود پس فرض کنید ۳ پیش نیاز ۲ است. یک بردار اینجا فرض کنید مسیر بحرانی می‌شود ۷، ۸، ۹+۳=۱۲+۸=۲۰ اینجا روی برد چی داریم یک جواب ۱۷. نتیجه این پیش نیازی اضافه کردنش هیچ امتیازی ندارد چون جوابی خواهد داد که حداقل ۲۰ روز طول می‌کشد و من یک جواب ۱۷ روز را قبلاً روی برد آماده دارم. یعنی شما اگر یک جواب داشته باشید مثلاً ما یک نمونه داریم. من حالا توی این هم نوشتم اگر نداشته باشی چیزی نمی‌توانی بنویسی ***Lower bound*** نداری اگه جواب از قبل نداری دیگه ***lower bound*** نه باید عدد یا ***X*** بنویسی اما اگر یک جواب از قبل داری که حالا اتفاقاً ما این را روی برد داریم. من اینجا ۱۷ را نوشتم به لطف این که اینجا دارم این در واقع پیش‌نیازی‌های مسیر بحرانی بالای ۱۷ روز را می‌دهند را ***LB*** می‌نویسیم. یعنی هیچ امتیازی بهش نمی‌دهم.

پس این خانه‌ها به این ترتیب پر شدند حالا امتیاز بدهید . به چی ؟ به هر کدام از آن ***forbidden set*** ها مثلاً ***forbidden set {۲,۳,۵}*** چند امتیاز کسب می‌کند ؟ ***{۲,۳,۵}*** کدام خانه‌ها هست ؟ ۲ پیش نیاز ۳ یک امتیاز ، ۲ پیش نیاز ۵ هم یک امتیاز با هم می شود دو امتیاز. ۳ پیش نیاز ۲ که هیچی ، ۳ پیش نیاز ۵ هم همین طور. ۵ پیش نیاز ۲ و ۳ هم همینطور پس کلاً چی شد ؟ کلاً دو امتیاز شد. یعنی پیش نیازهای مجموعه ***{۲,۳,۵}*** که دو به دو پیش نیاز هم می‌کنید کلاً ۲ امتیاز شد. پس ***{۲,۳,۵}***یک عدد ۱ بود یک عدد ۱ دیگر، بقیه که ***LB*** و ***X*** هستند هیچ امتیازی ندارد.

{۲,۳,۵}

امتیاز ⟵ ۱+۱+۰+۰+۰+۰=۲

اگر برای بقیه هم همین کار را انجام بدهید متوجه می‌شوید که کدام امتیاز بیشتری دارد ؟ مجموعه {۴,۷,۸} پس بعد از این به اصطلاح امتیازدهی و پیش پردازش این روش به صورت هوشمندانه و نه شانسی {۴,۷,۸} را می‌خواهد زودتر انجام دهد به خاطر اینکه این امتیاز همپوشانی بیشتری دارد.

{۴,۷,۸} به عنوان *forbidden set* ایی که اول بررسی می‌شود خوب چه طور حل کنم این را ؟ گفتم این تکراری است. {۴,۷,۸} را مشکل آن را چه طوری حل کنم؟ یکی را پیش نیاز دیگری کنم. کدام یکی را ؟ همه حالت‌ها را باید بگویید که هیچ جوابی گم نشود من اینجا نوشتم لطفاً روی برد را نگاه کنید.

۴ پیش نیاز ۷

۴ پیش نیاز ۸

۷ پیش نیاز ۸

۸ پیش نیاز ۷

۷ پیش نیاز ۴

۸ پیش نیاز ۴

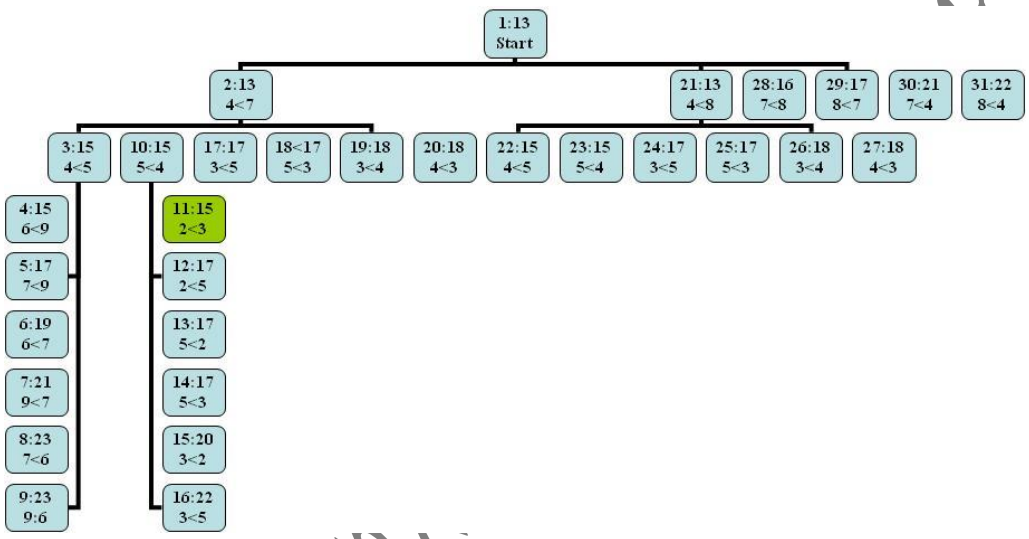
پس بین ۴ و ۷ و ۸ دو به دو پیش نیاز هم کردم *Lower bound*هاش ، ۲۱،۲۲، ۱۷، ۱۶ ، ۱۳ ، ۱۳ به صورت *Best first* . *lower bound* هایی که از همه کمتر است اول انجام می‌شود ۱۳ در این جا ما دو تا ۱۳ داریم که معمولاً در این حالت‌ها تصادفی یک شاخه را انجام می‌دهیم دیگه ولی منطق زیادی توش نیست وقتی دو تا *lower bound* مساوی دارید یکی‌اش را انجام بدهید که من می‌خواهم کدام را انجام بدهم. ۴ را پیش نیاز ۷ می‌کنیم پس الان تو شبکه از این لحظه به بعد تو این شاخه ۴ پیش نیاز ۷ است. درست شد آن وقت که ۴ پیش نیاز ۷ بشود. توی این *forbidden set* های شما کدام‌ها حل می‌شوند. چک کنیم. در {۳,۴,۷} ، {۴,۷} دارد. وقتی ۴ پیش نیاز ۷ بود چرا این حل می‌شود چون تعریف *forbidden set* این بود که باید پیش نیاز هم نباشند ولی الان شدند. الان ۴ پیش نیاز ۷ شد اینها *forbidden set* نیستند از این لحظه . پس این حل شد. اینم همین طور {۴,۷,۸} و {۴,۷,۹} پس از ۷ تایی که داشتم توی این مسیر که الان قرار گرفتیم ۳تا را حل کردیم ۴ تای دیگه مانده است. آن ۴ تای دیگر را می‌شود حدس زد که چی کار باید بکنیم.

آن ۴ تایی که ماندند حالا کدام را انجام بدهم. دوباره امتیاز. این جدول دوباره رسم شده است یعنی این جوری نیست که قبلی را دوباره کپی کرده باشیم باید دوباره انجام بدهید. چون دفعه قبل شما ۷ تا *forbidden set* داشتید ولی این بار ۴ تا *forbidden set* دارید باید دوباره انجام بدهید. خوب من این جدول را دوباره رسم کردم.

	2	3	4	5	6	7	8	9
2	-	1	X	1	X	X	0	0
3	LB	-	LB	LB	1	0	X	X
4	X	LB	-	2	X	X	0	0
5	LB	LB	2	-	1	X	X	X
6	X	LB	X	LB	-	LB	LB	1
7	X	LB	X	X	LB	-	LB	1
8	LB	X	LB	X	LB	LB	-	X
9	LB	X	LB	X	LB	LB	X	-

امتیاز این ۴ تا *forbidden set* را هم حساب کردم که به طور واضح می‌بینید دیگه. کدام امتیاز آورده است ؟ *{۳,۴,۵}* البته معمولاً بین دو تا کدام را انتخاب کنیم بین دو تا یا سه تا ممکن است دیگه نیاز به امتیاز حساب کردن نباشد بین دو تا یکی‌اش را انجام بدهید مهم نیست ولی بین ۱۰ تا یا ۱۵ تا حالا اینجا هم ۷ تا و ۴ تا این کار را انجام بدهید ولی معمولاً وقتی *forbidden set* زیاد است آن اوایل امتیاز حساب کردن می‌خواهد آن آخر که دو تا ، سه تا می‌ماند دیگه می‌تواند شانسی انجام بدهید.

حالا اینجا برای ۴ تا هم انجام دادیم امتیاز گرفتیم. کدام امتیاز آورده است ؟ *{۳,۴,۵}* پس اگر در *tree* دقت کنید، بعد از اینکه ۴ را پیش نیاز ۷ کردید دفعه بعد *{۳,۴,۵}* را می‌خواهد حل کند کدام را پیش نیاز کرده است ؟ ۶ حالت پیش نیاز هم هستند.



۴ پیش نیاز ۵

۵ پیش نیاز ۴

۳ پیش نیاز ۴

۵ پیش نیاز ۳

۳ پیش نیاز ۴

۴ پیش نیاز ۳

Lower bound ها ، ۱۸،۱۸، ۱۷،۱۷، ۱۵

خوب کدام را ادامه بدهم بازم *best first* بین دو تا ۱۵ دیگه فرقی نمی‌کند. این *tree* چرا شکلش این مدلی است ؟ چون در اسلاید جا نمی‌شد ناچاراً این جوری بازش کردم و الا یک ردیف مثل آنها باز شود چون جا نمی‌شود ساختارش را اینطوری رسم کردم پس اینها خیلی مهم نیست. این شاخه باز شده است ظاهراً حالا من دیگه امتیازهاش را نیارودم یا اصلاً نخواستم بیارم به اصطلاح چون دیگه ۳ تا مانده است آن ۳ تا را شانسی انتخاب کردم بین ۳ تا *{۶,۷,۹}* را باز کردم. *Lower bound*ها را ببینید. ۲۳، ۲۱،۲۳، ۱۹، ۱۷، ۱۵ بعد این ۱۵ را باز کردم نکته جالبی که اتفاق افتاده این است که وقتی این ۱۵ را باز کردم یعنی *{۲,۳,۵}* را بررسی کردم ۲ را پیش نیاز ۳ کردم همه ۷ تا *forbidden set* حل شد یعنی این گره به عمق رسید یعنی یک گره کی به عمق می‌رسد . دیگه هیچ*forbidden set* ایی در آن مسیر نماند.

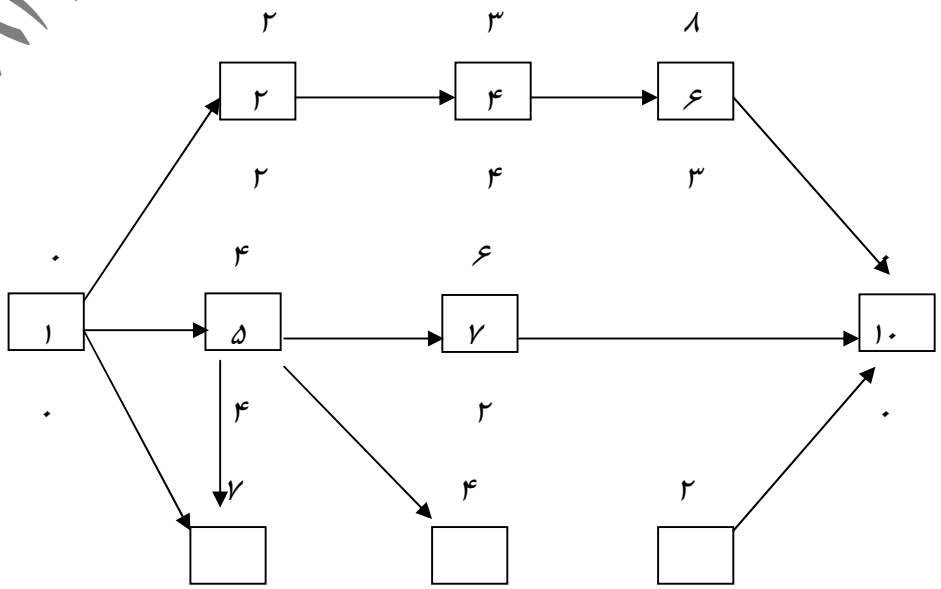
تابع هدفش چند شده است؟ ۱۵ . البته این جواب را *optimal* حساب نکنید من هنوز مطمئن نیستم که بهینه باشد چون هنوز شاخه را چک نکردم، بعداً معلوم می‌شود پس من یک جواب پیدا کردم ۱۵ است. این ۱۵ را چی کار کنیم ؟

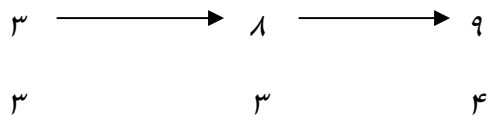
۱۵ را بایگانی می‌کنیم حالا دنبال جوابی هستیم که از ۱۵ بهتر باشد. کدام‌ها بهتر از ۱۵ هستند. ۲۳، ۲۳، ۲۱، ۱۹، ۱۵،۱۷، ۲۲، ۱۷،۲۰، ۱۷، ۱۷ اینها همه بسته شدند چون بالای ۱۵ هستند. آن سمت : ۱۷، ۱۸، ۱۸ اینها هم بسته می‌شوند. اینجا : ۲۲، ۱۷،۲۱، ۱۶ بسته می‌شوند فقط یک ۱۳ داریم که باید بررسی کنید یعنی همه شاخه‌ها به جز یک گره ۱۳ بقیه بسته شدند. ولی*lower bound* ، ۱۳ ظاهراً بهتر از ۱۵ است. باید ادامه بدهم پس من گره ۱۳ وقتی است که ۴ پیش نیاز ۸ شده است. اگر ۴ را پیش نیاز ۸ کنید خوب یک لایه می‌آید پایین تر دوباره مجموعه بعدی و امتیازدهی. *{۴,۳,۵}*را باید چک کنید ۱۸، ۱۸، ۱۷، ۱۷، ۱۵، ۱۵ تمام دیگه هیچ کدام بهتر از ۱۵ نبودند. یا ۱۵ بودند یا بالاتر که فایده نداشت نتیجه الان مطمئن شدم که تنها جوابی که پیدا کرده بودیم بهینه است پس این روش چند تا جواب کاملاً چک کرد ؟ یک جواب و اونم بهینه بود بقیه به چند دلیل بسته شدند ؟

این روش چند تا *bounding role* داشت ؟ این روش تنها یک *bounding role* دارد اونم اگر *Lower bound* باشد. اما شما سوال نپرسیدید که *Lower bound* از کجا می‌آید؟ *Lower bound* اش هم بسیار راحت محاسبه می شود. محاسبات *forward* در کنترل پروژه خواندید. که یکی را هم محاسبه کردید. ۳ را پیش نیاز ۲ کردید طول مسیر بحرانی.

مثلاً این چه طوری ۱۷ شده است؟ ۵ را پیش نیاز ۳ کرده است و *Lower bound* ۱۷ شده است. خوب بالا یادتان نرود که ۴ هم پیش نیاز ۷ است *(۴<۷)* ۵ پیش نیاز ۳ *(۳ <۵)*، *Lower bound* ، ۱۷ شده است. نحوه محاسبه اش باید دو تا بردار به پرینتان اضافه کنید ۴ را پیش نیاز ۷ یعنی از ۴ به ۷ یک بردار و ۵ پیش نیاز ۳ یعنی از ۵ به ۳ یک بردار بعد محاسبات *forward* در کنترل پروژه را انجام دهید. طول مسیر بحرانی. این مسیر *duration* اش چند است ؟ ۱۳=۸+۳+۲ . این مسیر چقدر است ؟ ۱۰=۶+۴ و این مسیر ۱۰=۲+۴+۴ و این مسیر ۱۳=۲+۴+۱۷ اما مسیرهایی که الان اضافه شده است. این یکی مسیر چی ؟ ۱۱=۶+۳+۲ و این مسیر ۱۷=۲+۴+۷+۴و این مسیر ۱۰=۶+۴ و این مسیر ۱۰=۲+۴+۴ نتیجه ۱۷ .

مسیرش : ۱۰ ← ۹ ← ۸ ← ۳ ← ۵ ← ۱، طولش ۱۷ است پس *lower bound* اش ۱۷ شد. یعنی مسیری که طولانی تر است البته من این مسیر را چشمی حساب کردم ولی در پارت ۳ و ۴ محاسبات *forward* به طور مفصل توضیح داده شد.





نحوه محاسبه *LB* : در این روش *LB* از محاسبات *forward* به دست می‌آید.

تنها *Bounding role* : اگر در یک گره *LB* محاسبه شده از $C_{\max}$ بهترین جواب یافت شده تا این لحظه بیشتر یا برابر باشد این شاخه بسته می‌شود.

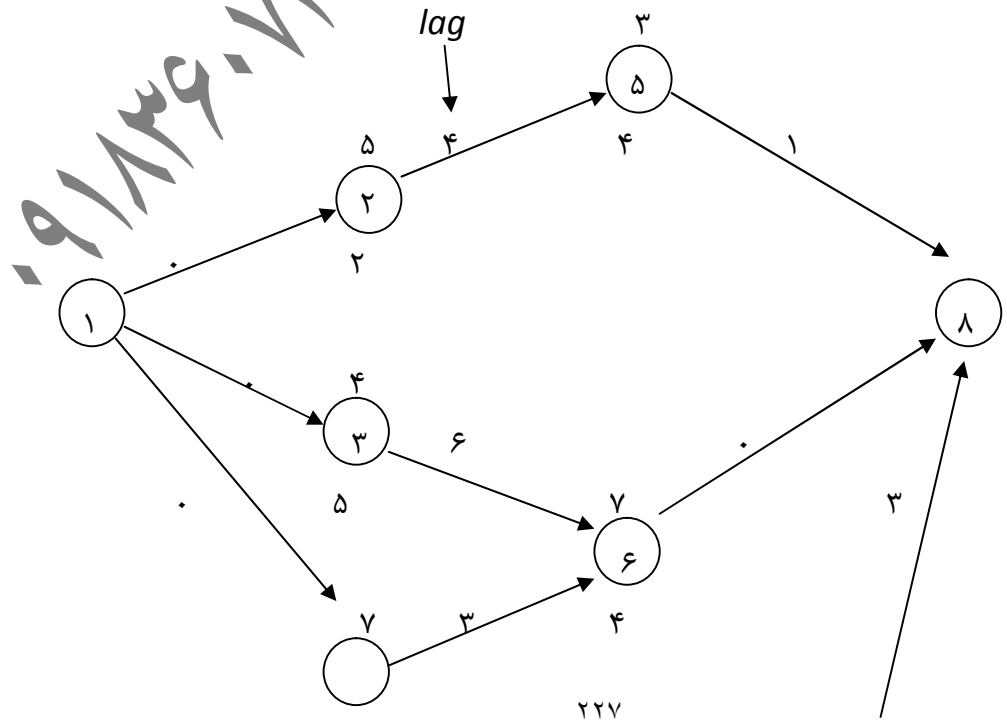
البته دقت کنید *lower bound* برای مسائلی است که نوع مینیمم سازی است اگر مسئله ماکسیمم سازی باشد *Upper bound* باعث بستن شاخه خواهد شد. چون همه مسئله‌هایی که تا اینجا داشتیم فعلاً $C_{\max}$ بوده است همه‌اش از *Lower bound* استفاده کردیم ولی وقتی مسئله ماکسیمم سازی است تابع هدف شما برعکس است دنبال ماکسیمم کردن هستیم بنابراین *upper bound* ها مبنا خواهد شد. این روش نسبت به روش‌های قبلی راحت‌تر بود یا سخت‌تر ؟ درکش راحت‌تر بود. *Bounding role* هم که یکی داشت *lower bound*اش هم از طریق محاسبات *forward* حساب می‌شد که راحت تر است. اما بدترین قسمت کار ، آن *forbidden set* هایی است که آن اول نوشتیم که ۷ تا بود. این مثال چون مثال ساده‌ای بود ما از روی شکل این ۷ تا *forbidden set* را نوشتیم حالا تصور کنید روی این دیوار شبکه‌ای با ۵۰۰ تا فعالیت باشد حالا می‌خواهید *forbidden set* ها را پیدا کنید ؟ اولاً شناسایی همه *forbidden set* ها کار ساده‌ای نیست و اینم می‌دانید که اگر احیاناً یکی را نتوانید پیدا کنید بهینه سازی نیست باید مطمئن باشید که همه را نوشته‌اید از طرفی در شبکه‌هایی که مثلاً۵۰۰ تا فعالیت دارد. *forbidden set* های شما دیگه ۳ تایی نیستند ممکن است یک *forbidden set* ، ۳۰ تایی داشته باشید. آن وقت بخواهید دو به دو پیش نیاز هم کنید. احتمالات یادتان می‌آید. ۳۰ تا فعایت دارم می‌خواهم ۲ را پیش نیاز کنم ؟ کدام ۲ تا ؟ انتخاب ۲ از ۳۰ پس دیگه ۶ تا نمی‌شود ، ۵۰۰ تا یا ۴۰۰ تا می‌شود یعنی هر شاخه‌اش ۶۰۰ قسمت می‌شود در کل این روش روش سختی است برای حل کردن. هر چند که ظاهر و درکش برای این مثال کوچک راحت بود. چون *bounding role* یکی داشت و *lower bound* راحت حساب می‌شد ولی مشکل *forbidden set* ها است. که خودش آخر سر پیشنهاد می‌دهد تا با یک الگوریتم دیگری توسعه بدهی که اول بتواند این *forbidden set* ها را کشف کند و بعد بتوانیم بحث را با این روش ادامه بدهیم.

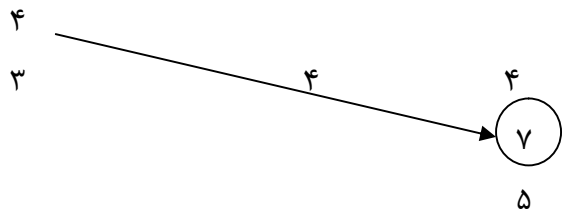
جلسه دوازدهم

این بخش آخر برخلاف پارت‌های قبلی که ما معمولاً یک مسئله یا دو مسئله را به صورت کامل از تعریف ، مدل‌سازی، حل و مثال عددی صفر تا صد را پیش می‌بردیم این جلسه برعکس است قرار است ۸ ا مسئله را ولی نه خیلی عمیق، بلکه به صورت سطحی مطرح کنیم پس تعداد موضوعات مورد بحث در این جلسه زیاد است ولی خیلی عمیق وارد آنها نمی‌شویم. بیشتر به اصطلاح چکیده بحث و مسیر حل را خواهیم دید. به جای اینکه یک مثال را کامل حل کنیم اینم به خاطر محدودیتی که زمانی که داریم و به هر حال ترم را با این پارت ۱۲ می‌بندیم.

در جلسات قبل، خصوصاً دو جلسه قبل برای *RCPSP* ، ۴ تا روش *exact* شاخه و کرانی یاد گرفتیم ولی *RCPSP* نسخه‌های دیگری هم دارد که در این پارت به بررسی آنها می‌پردازیم.

RCPSP-GPR که به اصطلاح *RCPSP* تعمیم یافته می‌گویند یا *Generalize RCPSP* . پاراگراف اول این اسلاید راجع به همین موضوع است. احتمالاً قابل حدس است که موضوع چیست؟ *GPR* قبلاً ، در دو ، سه پارت های قبلی با این اصطلاح سر کار و داشته‌اید. پیش نیازی‌های *GPR* چی بودند؟ آره در حالت کلاسیک وقتی می‌گوییم a پیش نیاز b است و حرف دیگری نمی‌زنیم منظور این است که پیش نیازی از نوع *finish to start* است یعنی وقتی که a تمام شد b می‌تواند شروع شود. امادر *GPR* ها ، a پیش نیاز b است ۸ تا معنی متفاوت می‌تواند داشته باشد که قبلاً صحبت کردیم ولی قرار بود موقع حل کردن همه اینها به حالت استاندارد SS_{min} تبدیل شود. حالا *RCPSP*که پیش نیازهایش از نوع *time lag* باشد را *RCPSP-GPR* می‌گویند. پس پیش‌نیازهایش از نوع *finish to start* نیست.





در مثال بالا ، پروژه‌ای داریم که دارای ۸ تا فعالیت است فعالیت ۱ و ۸ را مجازی فرض کنید یعنی عملاً ما ۶ تا فعالیت داریم. عدد بالای فعالیت نشان دهنده ***duration*** است.

فعالیت ۲، به مدت ۵ روز است

فعالیت ۳، به مدت ۴ روز است

فعالیت ۴، به مدت ۷ روز است

فعالیت ۵، به مدت ۳ روز است

فعالیت ۶، به مدت ۷ روز است

فعالیت ۷، به مدت ۴ روز است

عدد پایین فعالیت نشان دهنده نیازمندی منبع است.

فعالیت ۲، به ۲ منبع نیاز دارد.

فعالیت ۳، به ۵ منبع نیاز دارد.

فعالیت ۴، به ۳ منبع نیاز دارد.

فعالیت ۵، به ۴ منبع نیاز دارد.

فعالیت ۶، به ۴ منبع نیاز دارد.

فعالیت ۷، به ۵ منبع نیاز دارد.

فعالیت ۱ و ۸ مجازی هستند و ***duration*** و منبع آنها صفر می‌باشد.

اما یکسری عدد روی بردار هستند که نشان دهنده ***Lag*** است ***lag***های استاندارد شده این ***lag*** اش صفر است اینم صفر و این یکی هم صفر است و ۴، ۶، ۳، ۴، ۱، ۰، و ۳ . پس اعدادی که روی بردار هستند جدید هست ما قبلاً نداشتیم. چون توی ***RCPSP*** کلاسیک پیش‌نیازها از نوع ***finish to start*** هستند ما ***lag***نداشتیم ولی الان در ***RCPSP-GPR*** می‌خواهیم ***lag*** داشته باشیم. این میشه یک مثال برای این عنوان اول یعنی ***RCPSP-GPR*** حالا می‌خواهیم بگوییم که این مسئله چه طوری حل می‌شود.

این مدل با این ***time lag*** ها چه طور حل می‌شود چون آن هفته بدون ***time lag***ها حل می‌کردیم خوب ۴ تا روش را یاد گرفتیم که اگر به ترتیب بخواهم بگویم.

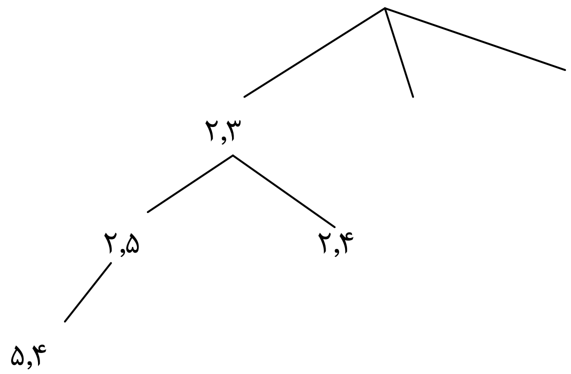
۱- ***Precedence tree*** که کد موجه ، توالی موجه تعریف می‌کردیم.

۲- ***Extentional alternative***

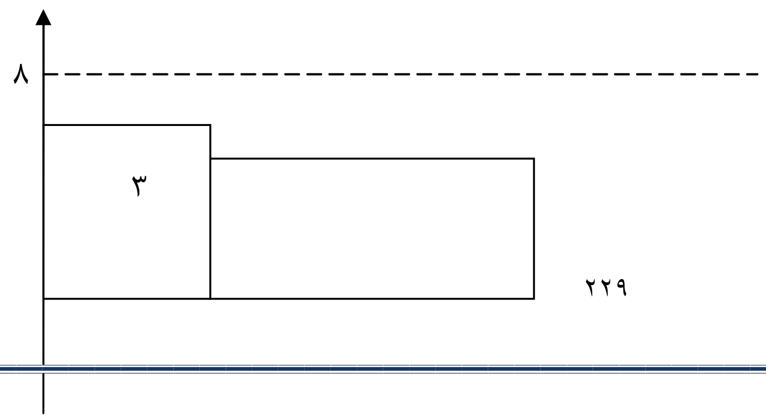
۳- ***Minimal delay alternative***

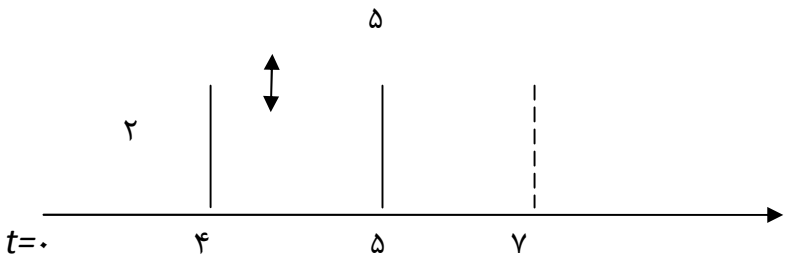
۴- ***Forbidden set***

فکر کنم که *extentional alternative* از همه راحت‌تر بود . یعنی چی ؟ موضوع اش در مورد چی بود ؟ یک گانت خالی رسم می‌کردیم حالا من در این مثال فرض می‌کنم *a=*۸ باشد یعنی چند تا منبع داریم ؟ ۸ تا. یک گانت خالی رسم می‌کردیم و می‌گفتیم اینم ۸ تا منبع. زمان محور بود. از کجا شروع می‌کردیم ؟ از لحظه صفر شروع می‌کردیم به چیدن فعالیت‌ها . در لحظه صفر، در این شبکه کدام فعالیت یا فعالیت‌ها می‌توانند با هم انجام شوند ؟ *{۳,۴} {۲,۴} {۲,۳}* سه تایی با هم که نمی‌توانند چون سه تایی با هم منبع نمی‌شود. پس ۳ تا شاخه باید بزنیم اگر بخواهیم حل کنیم.



حالا فرض کنید مثلاً من *{۲,۳}* را انجام بدهم. خوب اینجا *{۲,۳}* را بچینید. ۲ چند روز است ؟ ۵ روز با دو تا منبع می‌باشد. پس این ۲ ، دو تا منبع ، ۵ روز . ۳ هم که ۴ روز است و ۵ تا منبع. پس من *{۲,۳}* را الان چیدم. *t* بعدی کجا می‌شود ؟ ۴. حالا دوباره از روش شکل صحبت کنید در *t=*۴ کدام فعالیت یا فعالیت‌ها می‌توانند انجام شوند؟ البته ۲ را هم باید بگویید چون ۲ هنوز تمام نشده است . *{۲,۴}* . *{۲,۵}* چی ؟ الان بحث دقیقاً همین جا است *{۲,۵}*هم می‌توانند ؟ ۲ هنوز تمام نشده است. درست است که تمام نشده ولی تمام شدن اینجا مهم است ؟ اینجا تمام شدن مهم نیست باید *{۲,۵}* را هم بگویید. پس *{۲,۵} {۲,۴}* درست است که ۲ تمام نشده است ولی خوب ببنید ما وقتی پیش نیازی‌ها از نوع *finish to start* هستند *finish* هم است پایان بعد شروع ولی الان مهم نیست ۲ تمام بشود یا نه . چی مهم است ؟ مهم این است که آیا از شروع ۲، ۴ روز گذشته یا نگذشته است. آره دیگه من الان در *t=*۴ هستم ۴ روز گذشته است پس می‌تواند شروع شود یعنی می‌خواهم بگویم که کل موضوع ، اگر قرار باشه که ما یک جلسه را جدا برای این مبحث قرار بدهیم شما به شدت حوصله‌تان سر می‌رفت چون باید ۴ تا روش را دوباره می‌گفتم که ۹۰ درصد حرف‌ها تکراری بود فقط ۵ درصد جدید بود. کدام ۵ درصد ؟ که پیش نیازی فرق می‌کند قضیه‌اش. تمام شدن مهم نیست پس هیچی تمام شد بحث ادامه نمی خواهد. یعنی همه ۴ تا روش معتبرند فقط آنجایی که *alternative* ها را بررسی می‌کنید تمام شدن‌ها را مبنا نذارید *lag* ها را مبنا بگذارید. این کل موضوع بود. پس از ۸ آیتمی یک موردش گفته شد.





حل *RCPSP-GPR* مشابه *RCPSP* کلاسیک می‌تواند با ۴ روش شاخه و کران مطرح شده انجام شود تنها با این تفاوت که در بررسی پیش‌نیازی‌ها پایان پیش نیازی مهم نیست بلکه *time lag* بین فعالیت ها مهم است.

فرض کنید می‌خواهم *{۲,۵}* را بچینم ۵ چند روز است ؟ ۳ روز است و ۴ تا منبع می‌خواهد. ۳ روز یعنی تا ۷ ام . حالا *t* بعدی کجا می‌شود ؟ ۵ ام . اینم شاخه‌اش را بگویید دیگه کافی است. درلحظه ۵ کدام فعالیت یا فعالیت‌ها می‌توانند *alternative* باشند. البته اینجا دقت کنید که ۵ را باید بگویید چون ۵ تمام نشده است. ۵ خوب با کدام ؟ *{۵,۴}* . ۸ که نه ؟ چون ۷ و ۶ هنوز شروع نشده‌اند. فقط همین یکی هست یعنی فقط یک شاخه باید بزنید .

نکته : در این لحظه چه اتفاقی افتاده است .هم ۲ دارد انجام می‌شود و هم ۵ (باهم) . در حالی ۲ پیش نیاز ۵ است. ولی خوب در اینجا پیش‌نیازی معنی‌اش فرق دارد. پس آن مثال را که نوشته‌اید من حالا قسمتی از حل را برای اینکه این تیکه را نوشتید توضیح داده باشم را انجام می‌دهم دیگه حل کامل را انجام نمی‌دهم اصلاً قرار نیست که کلاً مسائل را حل کنیم. آره دیگه من دارم این شاخه را ادامه می‌دهم این شاخه *{۲,۵}* را همه شاخه‌ها باید انجام بشوند. ولی شاخه‌ای که من پیش رفتم این شاخه است. شماره شاخه به همان ترتیبی است که شما داری انجام می دهید.

مسئله دوم *PRCPSP*

همان *RCPSP* است که اول آن *P* قرار دارد به اصطلاح *Preemtive RCPSP* یعنی *P* اول *Preemtive* است. *preemption* را فکر کنم چند بار بحث کردیم. یعنی چی ؟ یک فعالیت *preemption* دارد یعنی چی ؟ قبلاً داشتیم نه در مدل‌ها بلکه در صحبت‌ها بود. *Preemption* یعنی بریدگی . قطع شدن فعالیت را *preemption* می‌گویند. اگر بریدگی نداشته باشیم *non-preemptive* است. ما تا الان بریدگی نداشتیم یعنی فعالیت تا شروع می شود تا کی ادامه داشت ؟ تا تمام بشود یعنی فعالیت‌ها یک تیکه انجام می‌شود الان شما همین جا ۵ و ۳ را ببینید ۳ که شروع شد تا تمام شدنش پیش رفت کلاس‌های ما از اول ترم چه طوری بوده است ؟ بدون *preemption* بود ما در این کلاس‌ها هیچ وقت آنتراک نداشتیم کلاس پیوسته بود و وسط درس وقفه‌ای وجود نداشت.

ما دو مدل انقطاع داریم یک انقطاع عمدی است یعنی عمداً برنامه ریزی می‌کنم که این کلاس آنتراک داشته باشد. یک انقطاع هم اتفاقی است. برق رفته و ویدیو پروژکتور هم خاموش شده است و چون درس در این کلاس وابسطه به ویدیو پروژکتور است یک وقفه در وسط کلاس ایجاد می‌شود. مسئول خدمات گفته که نیم ساعت دیگر برق می‌آید نتیجه نیم ساعت دیگه بیاید تا به ادامه درس داده شود. این می‌شود یک *preemption*اجباری، اتفاقی، به خاطر یک حادثه . حالا وقتی می‌گوییم *PRCPSP* فکر می کنید منظور کدام حالت از *preemptive* است ؟ یعنی آیا برنامه‌ریزی شده می‌خواهیم ببریم یا نه اتفاقی بریده خواهد شد. *Preemptive RCPSP* یعنی برنامه‌ریزی شده داریم یعنی شما می‌خواهید ببینید آیا برنامه‌ریزی قبلی بخواهید فعالیت‌ها را ببرید چه طور می‌شود زمانبندی کرد. آن‌ایی

که اتفاقی است را معمولاً *preemption* نمی‌گویند. قطع تصادفی را *intruption* می‌گویند. آن وسط پروژه اتفاق می‌افتد ولی موقع برنامه‌ریزی وقتی ما داریم برنامه را بریده بریده می‌چینیم کلمه *Preemption* به کار می‌بریم و آن یکی را معمولاً *intruption* می‌گویند.

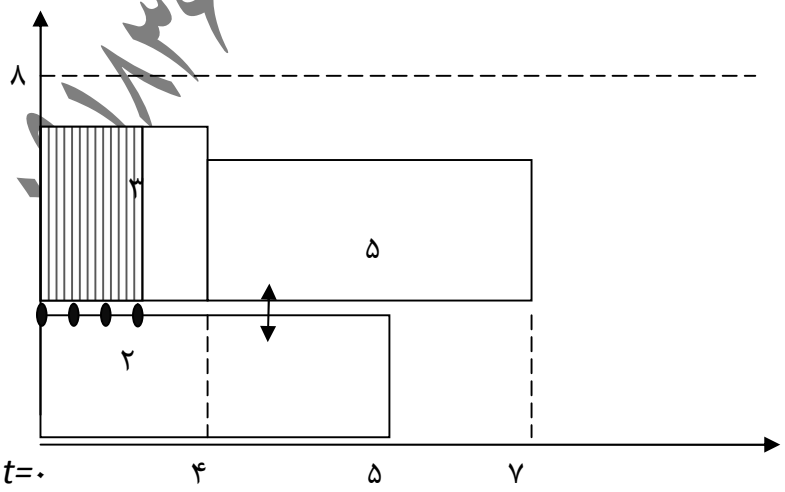
حالا برای این مسئله را حل کنیم آقای اسلاونسکی و گلارز *(Slowinski & weglarz)* برای اولین بار مسئله یعنی مفهوم بریدگی را معرفی کردند و برای یک مدل ریاضی پیشنهاد کردند و یک روش حل را هم ارائه دادند. فقط دقت بفرمایید که من مدلش را خیلی کوتاه در رابطه با آن صحبت کنم. ما در حالت عادی یعنی در حالت *RCPSP*کلاسیک می‌خواستیم مدل بنویسیم متغیرها را چی تعریف می‌کردیم؟ متغیرها را f_i تعریف می‌کردیم. f_i یعنی *finish time* فعالیت *i* یا زمان پایان آن .

الان در این زمانبندی $f_۳$ چند است ؟ ۲ کی تمام شده است ؟ ۵

$f_۳$ چند است ؟ ۳ کی تمام شده است ؟ ۴

$f_۵$ چند است ؟ ۵ کی تمام شده است ؟ ۷

پایان هر فعالیت را متغیر می‌گرفتیم یا شروع را ؟ فرقی نمی‌کرد. معمولاً پایان . هر فعالیت پایانش یک مجهولی است که موقع زمانبندی که باید به دست بیاید یا شروعش که یک مجهول است باید به دست بیاید . اما وقتی مسئله *preemptive* است به جای f_i باید چی تعریف کنید ؟ باید دو اندیس تعریف کنید. $f_{i,j}$. چه طور خوانده می‌شود؟ زمان پایان *j* امین روز فعالیت *i* یعنی برای هر روزش یک *f* باید تعریف کنید. ببنید وقتی شما *non-preemptive* حل کنید فعالیت ۳ اگر به ۴ قسمت تقسیم شود در واقع ۴ تا فعالیت یک روز است اینم چی است ؟ $f_{(۳,۱)}$، این $f_{(۳,۲)}$ است ، $f_{(۳,۳)}$ و این یکی هم $f_{(۳,۴)}$ است این دقیقاً چقدر بینشان فاصله است ؟ دقیقاً یک روز . چرا یک روز ؟ چون اینها پشت سر هم و بدون وقفه باید انجام بشوند اما الان که قرار به اصطلاح اینها بینشان فاصله باشد. یعنی بریده بشوند از هم. یعنی فاصله داشته باشند از هم. دیگه معلوم نیست فاصله‌ها چقدر باشد . دیگه باید این را مدل تعیین کند. یعنی برای هر روزش یک *f* باید تعریف کنید در حالی که ما در دفعه قبل آخرین روز را f_i می‌گرفتیم دیگه بقیه‌اش پشت سر هم بود. بقیه‌اش دیگه واضح بود ولی اینجا باید برای هر روزش یک *f* تعریف کنید.



این تعریف متغیر بود به جای f_i می شود $f_{i,j}$ فقط به صورت توافقی $f_{i,.}$ را چی می‌توانید بگوید. لحظه شروع را $f_{i,.}$ می‌گویند پس $f_{i,j}$ پایان j امین روز i ولی $f_{i,.}$ شروع i را می‌گویند.

خوب وگلارز و اسلاویسکی با این fایی که تعریف کردند این مدل ریاضی را برای مسئله پیشنهاد کردند.

$$\min f_{n,0}$$

subject to

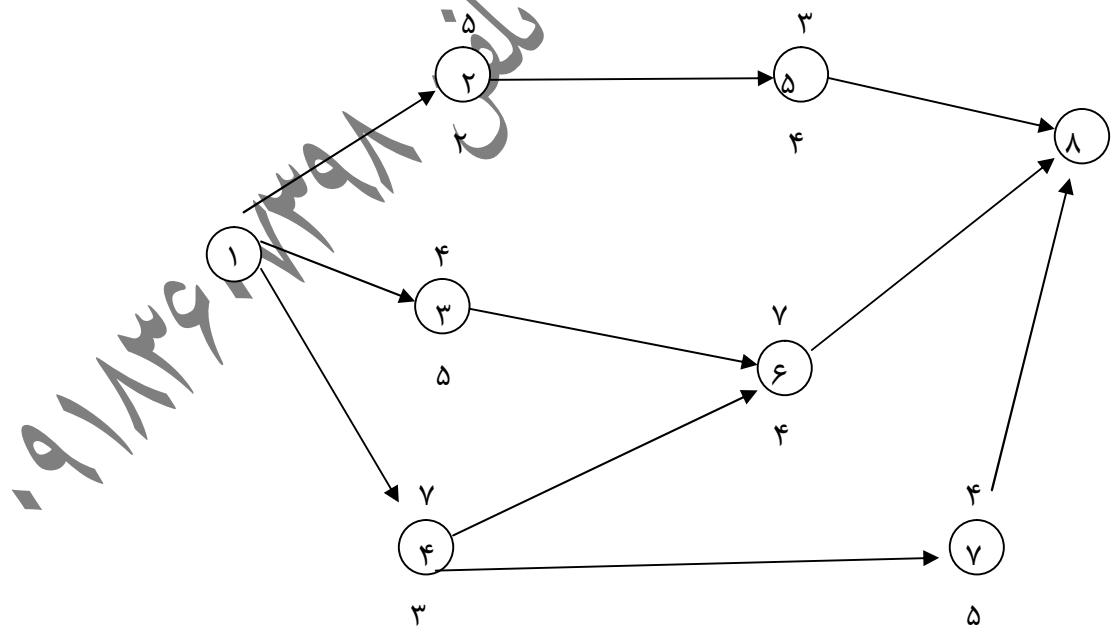
$$f_{i,d_i} \le f_{j,0} \qquad \text{for all } (i,j) \in A$$

$$f_{i,j-1} + 1 \le f_{i,j} \qquad \text{for } i = 1,..,n \text{ and } j = 1,..,d_i$$

$$f_{1,0} = 0$$

$$\sum_{i \in S_t} r_{ik} \le a_k \qquad \text{for } k = 1,..,m \text{ and } t = 1,..,f_{n,0}$$

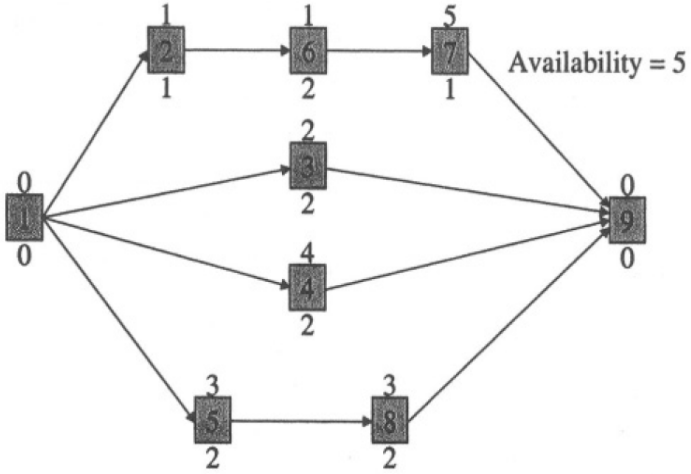
تابع هدفش *make span* است شما قبلاً *make span* را چی نوشتید. مینیمم f_n را الان به جای f_n چی بگویید. $f_{n,.}$ یعنی زمان شروع فعالیت آخر. فعالیت آخر چون مجازی است شروع و پایانش یکی است. این را هر چی بخوانید درست است. $f_{n,.}$ را چه بگویید زمان شروع n یا بگویید پایان n اشکال ندارد. این مجازی‌ها شروع و پایانش یکی هستند. این محدودیت $f_{i,d_i} \le f_{j,.}$ چه می‌گوید ؟ $f_{j,.}$ یعنی چی ؟ زمان شروع j باید بیشتر از پایان d امین روز i باشد. چرا ؟ این شکل را دقت کنید . آیا ۴ پیش نیاز ۷ است ؟ از چه نوع پیش نیازی است ؟ *time lag* ایی نیست که برای مدل قبلی بود . پیش نیازها دوباره همان *finish to start* ساده شد. *time lag* برای مدل یک بود این *GPR* ندارد فقط *RCPSP* است. ۴ پیش نیاز ۷ را می خواهم به صورت ریاضی بنویسیم.



۷ کی شروع می‌شود ؟ $f_{۷,.}$ این کی تمام می‌شود؟ $f_{۴,۷}$ آخرین روزش ۷ . خوب وقتی ۴ پیش نیاز ۷ است یعنی شروع ۷ باید بعد از پایان i باشد بین ۴و ۷ . حالا این دارد می‌گوید که برای همه i,j ها یعنی برای همه بردارها بنویسید. من بین ۴و ۷ نوشتم شما باید بین {۴،۶} و {۳،۶} و {۲،۵} بنویسید باید بین همه پیش‌نیازی‌ها یک همین چیزی نوشته

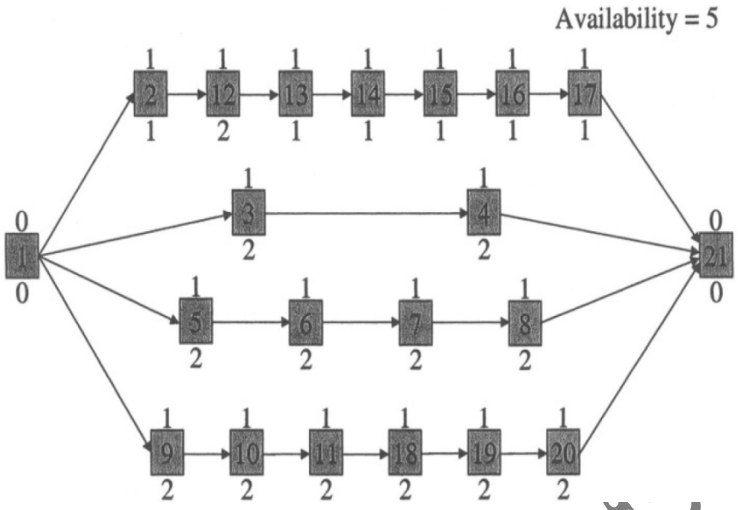
شود. این هم که محدودیت منطقی است و این $\sum_{i\in s_i} r_{ik} \leq a_k$ یک محدودیت منبع است که قبلاً داشتید. این می‌تواند با هر روش تعیین عملیاتی این مدل حل شود.

با جستجو عنوان ***Preemptive RCPSP*** در سایت‌های عملی ، کلی مقاله می‌آید که برای ***Preemptive RCPSP*** روش حل ارائه کرده است اولین روش روشی است که خانم کاپلان ***(Kaplan)*** در سال ۱۹۸۸ پیشنهاد کرد. ولی بعد از آن بیش از ۲۰ تا ۳۰ تا مقاله پیدا کنید که عنوانش این است حل مسئله ***PRCPSP*** . حالا یکی با شاخه کران، دیگری با برنامه‌ریزی پویا و یکی هم با ژنتیک حل کرده است. روش‌های مختلفی برای حل آن وجود دارد. من نمی‌خواهم به توضیح آن روش‌ها بپردازم من از شما می‌خواهم بپرسم که از روی دانسته‌های هفته قبل چه طوری می‌توان این را حل کرد با همان فرمول‌های (۴ تا روشی) که دو هفته قبل یاد گرفتید می‌خواهیم ***PRCPSP*** را حل کنیم. چه طوری ***Preemptive***کنیم. این می‌شود یک ابتکاری که ممکن است تضمین شده بهینه جواب ندهد این مثال را دقت کنید پس من دارم می‌گویم که چه طوری حل می‌شود.
این مثال را دقت کنید در این مثال ۹ تا فعالیت موجود است که ۷ تا از فعالیت‌ها واقعی هستند. ۵ تا منبع داریم و این اطلاعات موجود است.



اگر این مثال را بدون بریدگی حل کنید که بلد هستیم و ۴ تا روش برای حل آن می‌دانیم. جوابش این می‌شود. اما چه طوری می‌توان این مثال را ***preemptive*** حل کرد. برای ***preemptive*** حل کردن این فعالیت‌ها را بشکنید. یعنی چی مثلاً فعالی ۴ که ۲ تا منبع دارد و ۴ روز طول می‌کشد را بشکنید و اینطوری فرض کنید که ۴ تا فعالیت یک روزه است . همین کاری که من اینجا روی ۳ انجام داد. اینها چی ؟ این‌ها به ۳ تا یک روزه، اینها به ۳ تا یک روزه ، اینها به ۲ تا یک روزه و آنهایی که یک روزه هستند را دست نزنید چون خودش یک روزه است آن یکی هم به ۵ تا یک روزه بشکنید.

در آن صورت وقتی این کار را انجام دهید، این همان است این همان شبکه است فقط شکسته شده است. ببینید مثلاً به جای ۴ البته وقتی می‌شکنید در ابتدا شما ۹ تا فعالیت داشتید ولی وقتی می‌شکنید تعدادشان بیشتر می‌شود. که در این مثال ۲۱ فعالیت شده است.



فعالیت‌های ۵،۶،۷،۸ همان فعالیت ۴ است پس الان حل کردید و تمام شد یادتان باشد ۵،۶،۷،۸ با هم یکی هستند یعنی همان ۴ است. این را با همان روش‌های هفته پیش حل کنید انگار نه انگار که *preemptive* است.

این را حل کنیم حالا شماره‌ها بر گردانید سرجاش یعنی چی ؟ مثلاً ۵،۶،۷،۸ را پیدا کنید که همان ۴ است. حالا اگر من شماره‌ها سر جای قبلی بگذاریم دیگه نه ۵،۶،۷،۸ که ۴ است .۴ چه بلایی سرش آمده است. آن انقطاع براش اتفاق افتاده است دیگه کدام بریده شده است ؟ ۳ و دیگه هیچی . در این مثال ۴ و ۳ بریده شدند. و آن یکی‌ها دیگه بریده نشدند. خوب اگر بدون بریدگی حل کردید جوابش ۷ روز بود با بریدگی جواب بازم ۷ روز است. در این مثال فرقی نکرد. یعنی با *preemption* یا بدون *preemption* جواب، *make span* ، ۷ است. حالا سوال این است ؟ به طور کلی وقتی فعالیت اجازه بریدگی دارد جواب چی می‌شود ؟ همان می‌ماند ؟! می‌تواند بهتر هم بشود ؟! یا نه می‌تواند بدتر هم بشود ؟! در این مثال همان جواب قبلی است حالا راجع به بهتر شدن یا بدتر شدن‌اش با دلیل بگویید. یعنی می‌تواند بهتر شود آره بهتر می‌تواند بشود را قانع شدم آیا می‌تواند بدتر شود ؟ مثلاً بدون بریدگی ۷ روز شده با بریدگی ۸ روز . آیا همچنین اتفاقی می‌تواند بیافتد ؟ وقتی یک محدودیت را برمی‌دارید فضای جواب بزرگتر می‌شود جواب بدتر نمی‌شود. وقتی محدودیتی را بردارید فضای جواب بزرگتر می‌شود و جواب بدتر نمی‌شود یا همان می‌شود مثل این مثال و شاید هم بهتر شود حالا اینجا الان شما در تیتر وقتی یک *P* گذاشتید گفتید که بریدگی مجاز است در حالی که گفته بودید که بریدگی مجاز نیست الان وقتی می‌گویند مجاز است یعنی مسئله را چه کار کردی ؟ یک محدودیت‌اش را برداشتی ، چه محدودیتی را ؟ قبلاً بریدگی ممنوع بود ولی الان اجازه دادی بریده بشوند. یعنی محدودیتی را برداشتی پس اضافه کردن *Preemption* یه جوری برداشتن محدودیت است پس ما وقتی محدودیتی را برداری

جواب بدتر نمی‌شود می‌توانست بهتر شود در این مثال، خوب شانسی است فرقی نکرد. اگر یک مثال دیگر حل کنیم شاید بهتر شود بدتر نمی شود.

حل*PRCPSP* از حل یا جواب *RCPSP* کلاسیک بدتر نمی‌شود .

خوب سوال دوم : عرض کنم که چرا در این مثال ۸ یا ۷ یا ۵ بریده نشدند مگر قرار نشد اینها هم بریده شوند ؟ پس چرا بریده نشدند مگر قرار نشد *Preemption* داشته باشد ؟ ۷ بریده نشد. تیکه‌هاش وقتی پشت سر هم هست یعنی بریده نشده است ولی ۴ واضح است که ۳ تیکه شده است ببینید ما می‌گوییم *preemption* مجاز است نمی‌گوییم که باید *preemption* انجام شود. می‌گوییم مجاز است می‌تواند بریده هم بشود ولی نمی‌گوییم باید بریده شود. این مدل خودش موقع حل خودش تشخیص می‌دهد که بریده بشود به نفعاش است یا بریده نشود. ظاهراً ۴ و ۳ به نفعاش بود ولی ۵،۶،۷،۸ به نفع نبوده است. اگر شما بگویید باید بریده بشود چی می‌شود ؟ باید ، اینجا محدودیت اضافه کردن می‌شود. باید را نباید بگویید. باید مجاز کنید عین این است که دانشگاه الزام کرده است باید کلاس های ۳ واحدی را آنتراک بدهید این چی می‌شود ؟ این محدودیت است یعنی من علیرغم میل باطنی‌ام که دوست ندارم کلاس امروز را آنتراک بدهم ولی چون باید است دارم آنتراک می‌دهم که کلی به ضرر کلاس می‌باشد چون ممکن است وسط یک الگوریتم باید کلاس را تطعیل کنیم و در این صورت سر نخ موضوع از دست خواهد رفت. ولی خوب چون باید است انجام دادیم و جواب هم بدتر شده است. چون محدودیت است. ولی یه موقع دانشگاه اینطوری اعلام کرده است کلاس‌های ۳ واحدی می‌توانند آنتراک داشته باشند. ولی الزام نکرده است. گفته می‌تواند داشته باشد. حالا بستگی به روند کلاس تشخیص می‌دهم که آنتراک داشته باشیم یا نه . مثلاً شما خسته شدید موضوع سخت بود می‌گم باشه نیم ساعت توقف. اما اگر ببینم اوضاع خوب است چون همه ترم‌ها اوضاع خوب بوده و نیازی به آنتراک نبوده و موضوع هم آسان بوده است بگویم نه نیازی به آنتراک نمی‌باشد.

پس باید، نبود. *preemptive* اجازه بریدگی را داریم می‌دهم نه اینکه بگوییم باید.

مسئله ۳، البته آشنا هستید مسئله ۳ در بحث امروز ، تنها مسئله‌ای است که از قبل هم کنترل پروژه در لیسانس خواندید فکر کنم یک چیزهایی یادتان هست .

مسئله *Resourc leveling* یا تسطیح منبع :

تسطیح منبع یعنی چی ؟ از اسمش مشخص است تسطیح یعنی چی ؟ منابع را تسطیح کنید یادتان نمی‌آید. تسطیح منبع یعنی از منبع به صورت یکنواخت استفاده کنید. مثلاً این دانشکده ۵۰ تا کلاس دارد یکشنبه به ۱۷ تا کلاس نیاز است و پنج‌شنبه به ۵۳ تا کلاس نیاز است. که ۵۰ تا کلاس داریم و ۳ تای دیگر را مجبور به برگزاری در دانشکده‌های دیگر هستیم. نتیجه : استفاده از این منبع یکنواخت نیست نه به آن یکشنبه‌ها که ۳۳ تا کلاس خالی داری فقط ۱۷ استفاده شده است و نه به پنج‌شنبه‌ها که ۳ تا کلاس هم کم است. به اصطلاح گفته می‌شود که منبع تسطیح شده استفاده نشده است حالا حساب کنید در کنترل پروژه، کارگر ، تجهیزات ، ماشین‌آلات اینها هم به همین صورت است. مثلاً شما ۱۰۰ تا کارگر دارید که در این پروژه استخدام کردید یک روز در سایت پروژه می بینید که از این ۱۰۰ تا کارگر، ۵۰تا ، ۶۰تا ، ۴۰ تا بیکار هستند . چرا ؟ چون فعالیتی در آن روز در حال انجام نیست. بعد یک روز دیگه می‌آیی

و می‌بینی ۱۳۰ نفر فعال هستند ۱۰۰ نفر که بودند ۳۰ نفر هم به صورت *part time* اجاره کردیم یا اضافه‌کار به هر حال نگه داشتیم که کار را بتوانند پیش ببرند.

چون آن روز خیلی کار پروژه شلوغ بوده است خیلی ساده است نه به آن روزی که نصفشان بیکار بودند نه به آن روزی که اضافه کار نگه‌شان داشتید هر دو اینها هزینه‌ای برای یک تیم دارد ؟ چه هزینه‌ای دارد ؟ آنجا بیکاری داری یعنی دستمزد کارگر دادی ولی بیکار است اینجا داری پول اضافه کاری می دهید هر دو اینها پوئن منفی است. این مسئله دنبال این است که همچنین چیزهایی را تسطیح کند. محدودیت‌ها ، کپی همان محدودیت‌های *RCPSP* است فقط یک *dead line* هم براش گذاشته است. *dead line* برای کدام مسائل است ؟ آنهایی که هدفش نامنظم است آنهایی که منظم هستند *dead line* نمی‌خواهد. چون *leveling* نامنظم است *dead line* براش گذاشتیم اما تابع هدف. ما در قسمت قبلی داشتیم *make span*را مینیمم می‌کردیم اما الان *make span* مهم نیست چی مهم است ؟ تسطیح بودن منبع مهم است. یعنی چی را باید مینیمم کنیم ؟ مینمم کنید این عبارت را . آره می‌توانید بگویید هزینه استخدام، اخراج یا می‌توانید بگویید هزینه نوسانات منبع . نوسانات یعنی اضافه و کم شدن منبع . هر دو یکی است

$$\min \sum_{k=1}^{m} \sum_{t=1}^{\delta_n} c_k(u_{kt})$$

subject to

$$f_i \le f_j - d_j \qquad \text{for all } (i,j) \in A$$

$$f_1 = 0$$

$$f_n \le \delta_n$$

$$\sum_{i \in S_t} r_{ik} \le u_{kt} \qquad \text{for } k = 1,..,m \text{ and } t = 1,..,\delta_n$$

این به اصطلاح تسطیح را من در مقاله‌های مختلف ۹ تا پیدا کردم یعنی به ۹ مدل می‌شود تسطیح را سنجید. حالا حوصله شما سر می رود که به ۲ تا ۳ مورد اکتفا می‌کنم. مثلاً یکی‌شون این است این چطوری تسطیح را می سنجد.

$$c_k(u_{kt}) = v_k \cdot u_{kt}^2$$

$$c_k(u_{kt}) = v_k \cdot |u_{kt} - a_k|$$

$$c_k(u_{kt}) = v_k \cdot (u_{kt} - a_k)^2$$

$$c_k(u_{kt}) = v_k \cdot \max(0, u_{kt} - a_k)$$

$$c_k(u_{kt}) = v_k \cdot (\max(0, u_{kt} - a_k))^2$$

$$c_k(u_{kt}) = v_k \cdot |u_{kt} - u_{k,t-1}|$$

$$c_k(u_{kt}) = v_k \cdot (u_{kt} - u_{k,t-1})^2$$

$$c_k(u_{kt}) = v_k \cdot \max(0, u_{kt} - u_{k,t-1})$$

$$c_k(u_{kt}) = v_k \cdot (\max(0, u_{kt} - u_{k,t-1}))^2$$

u_{kt} ⟵ میزان استفاده از منبع k در زمان t

$u_{k(t-1)}$ ⟵ میزان استفاده از منبع K در روز قبلی یعنی $t-1$

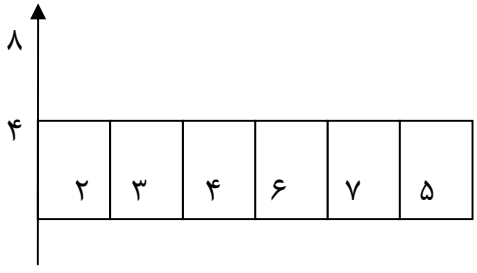
حالا اگر u_{kt} یا $u_{k(t-1)}$ برابر باشد معنی‌اش چی می‌شود ؟ یعنی امروز ۱۷ تا کارگر استفاده کردیم و دیروز هم از ۱۷ تا کارگر استفاده شد یعنی دیروز و امروز نه استخدام داشتیم نه اخراج به اصطلاح میزان استفاده مان ***level*** بوده ، تسطیح بوده است . اختلاف اینها چقدر است ؟ صفر. پس هیچ جریمه‌ای را شما بابت این قضیه استخدام یا اخراج، اضافه‌کاری یا کم‌کاری نمی‌دهید.

اما اگر اینها با هم برابر نباشند چی می‌شود ؟ یا استخدام کردید یا اخراج به هر حال فرقی نمی‌کند قدر مطلق اش یک جریمه‌ای دارد که اسمش v_k حالا چرا قدر مطلق خوب باشد پایین این را آمده چی کار کرده اومده توان ۲ اش را حساب کن البته اینجا فرض کرده که استخدام و اخراج هزینه‌اش با هم برابر است. والا اگر برابر نباشد باید اینها را جدا کند. این عبارتی که الان حساب شد فقط کی را حساب کرد؟ امروز را نسبت به دیروز دیگه پس بقیه روزها چی ؟ اگر دقت کنید ***summation*** دارد برای همه روزها. فقط برای کارگر یا برای همه منابع ؟ یک زیگما دیگری هم دارد که منابع یعنی این عبارتی که در اسلاید بعدی داریم توی کی منبع در یک روز بود حالا برای کل منابع در کل پروژه اینها را جمع می‌کنید اگر این عدد به اصطلاح در کل جمع این بشود صفر یعنی از تمام منابع به صورت کاملاً مساوی در طول پروژه استفاده شده که می‌شود فوق‌العاده عالی. اما اگر غیر از این باشد خوب یعنی شما جریمه اضافه کاری یا کم کاری داری که به هر دنبال این که چی کار کند ؟ این مسئله را تسطیح منبع است.

بازم کلی مقاله در سایت‌های علمی هست که روش حل پیشنهاد دادند برای حل تسطیح منبع شما یک روش به اسم برگس را در لیسانس خواندید . برگس یک روش ابتکاری است چون در این مسئله شما یک روش حل در لیسانس خواندید من دیگه راجع حل‌های این وارد نمی‌شوم نمی‌تونم هم وارد بشوم چون الان هر روش را بخواهم شروع کنم یک جلسه کامل می‌خواهد روش را به طور کامل با مثال توضیح بدهم. چون یک روش حداقل می‌دانید دیگه وارد آن نمی‌شوم.

تابع هدف در آنها چی بود ؟ داشتیم سعی می‌کردیم زمان پروژه را مینیمم کنیم و هدف من این بود که زمان پروژه را مینیمم کنم ولی الان زمان مهم نیست منابع مهم است.

فرض کنید همه فعالیت‌ها ۴ تا منبع می‌خواهد یعنی مسئله راخیلی آسان کردیم. همه فعالیت ها ۴ تا منبع می‌خواهد اگر این را با ***resource leveling*** حل کنید چه جوابی به شما می‌دهد این را چه طوری می‌چینید. من گانتش را همین پایین رسم میکنم.



ولی خوب من ۴ تا منبع از ۸ هم تجاوز نکردم الان دقت کنید که چقدر مسطح چیدم ببینید صاف صاف است.چون همه ۴ تا منبع می خواستند الان این را به اصطلاح مینیمم کنید تابع هدف چی می‌شود ؟ صفر. چون ان مطلقاً در کل پروژه اگر ۸ تا کارگر داشته است نوسانات را بخواهید بسینجید از اول تا آخر پروژه، صاف ۴ تا کارگر در واقع استفاده کرده است. مفهومش این است که *level* کامل. ولی خوب این پروژه چند روز طول کشیده است ؟ ۳۷ روز . ولی خوب اگر همین را الان با هدف *make span* حل کنیم. *Make span* دنبال این نیست که اینها را صاف کند می‌خواهد این پروژه به اتمام برسد پس یک کار مشابه را انجام نمی‌دهند یکی شون زمان پروژه را جمع می‌کند یعنی اینها را جمع می‌کند این ۵ را این بالا قرار می‌دهد و ۷ را اینجا می‌گذارد در کل کاری می کند تا کمی عقب تر بیاید. پس اینها یک کار را انجام نمی‌دهد. چرا اشتراک دارند. اشتراک هر دو در چی هست ؟ سقف منبع در هر دو مسئله محدود است در هر دو مسئله پیش نیازی باید رعایت شود آره اشتراک دارند ولی هدفشان یکی نیست.

Resource availaibility cost problem یا مسئله سرمایه‌گذاری در منابع که به طور اختصار *RACP* نشان می دهند. اسم دیگر این مسئله *Resource investment problem* یا به اختصار *RIP* می‌باشد. *RACP* یا *RIP* هر دو یکی است. خوب این چه فرقی با *RCPSP* دارد از اسمش می‌توان حدس زد. مسئله سرمایه گذاری در منابع . من تفاوت را بگویم و بعد بگویید این سخت‌تر است یا *RCPSP* در این مثال که روی برد است ما چند تا منبع داریم ؟ ۸ تا منبع داریم . این ۸ را مسئله داده است. گفته ۸ تا منبع داریم از شما خواسته که زمانبندی کنید. حالا این *a* را پاک کنید و به جای آن *x* را بگذارید یعنی چی ؟ این *a* را شما تعیین کنید که چند تا منبع داشته باشید بهینه است. این می‌شود مسئله *RACP* . فرقش در این است که تعداد منبع را باید مسئله تعیین کند از قبل معلوم نیست حالا اگر یکبار دیگه اسمش را بخوانید متوجه می‌شوید. مسئله سرمایه‌گذاری در منبع . چه طور می‌شود با کمترین هزینه منبع پروژه را انجام دهیم. حالا فکر کنید *RACP* سخت تر است *RCPSP* ؟ معلوم است *RACP* سخت تر است چون آنجا *a* را داده است خوب خیلی راحت است نصفش را حل کرده است اما الان *a* را شما باید تعیین کنید بعد که تعیین کردید ولی جواب بهتری می‌دهد. آن ۸ از کجا آمده بود مسئله از خودش داده بود. بهینه نبود اما الان شما دارید بهینه تعیین می‌کنید. پس این سخت تر حل می‌شود ولی جواب بهتری می‌دهد.

حالا چه طوری حل می‌شود ؟ البته بگویم چون هدف نامنظم است *dead line* دارد. البته روش‌های زیادی برای حل این وجود دارد که یک روش را براساس اطلاعاتی که می‌دانیم حل خواهیم کرد.

$$\min \sum_{k=1}^{m} c_k(a_k)$$

subject to

$$f_i \le f_j - d_j \qquad \text{for all } (i,j) \in A$$

$$f_1 = 0$$

$$f_n \le \delta_n$$

$$\sum_{i \in S_t} r_{ik} \le a_k \qquad \text{for } k = 1,..,m \text{ and } t = 1,..,\delta_n$$

پس این a را می‌خواهم تعیین کنم که مقدارش را بهینه تعیین کنم آن ۸ دیگه نیست. این a هر چقدر که کمتر باشد بهتر است یا بیشتر باشد ؟ کمتر باشد. چرا ؟ چون هزینه کمتری دارد. مثلاً در این مثال کمتر باشد یعنی چند مثلاً ، یک خوبه ؟ با یک که نمی‌شود در این مثال فقط فعالیت ۶ به تنهایی چند تا منبع دارد ؟ ۵ تا. پس ۵ کمتر نمی‌شود ؟ a از چند شروع می‌شود ؟ ۵ . این ۵ از کجا آمده است ؟ در پروژه ماکسیمم نیازمندی‌مان *Lower bound* مسئله می‌شود. a را با ۵ شروع می‌کنیم. پس a=۵ قرار می دهیم. فقط به شما بگویم که در این مثال *dead line* هم بگیرید ۲۰ یعنی $\delta_n = ۲۰$. با a=۵ چی حل کنیم ؟ *RCPSP* . که چه چیزی به دست بیاید ؟ C_{max} . پس $C_{max} = ۲۹$. آیا جواب موجه است ؟ خیر . *dead line* چند است ؟ ۲۰ . اما این پروژه چقدر شده است ؟ ۲۹ . وقتی C_{max} بیشتر از *dead line* باشد $(C_{max} > \delta_n)$ است یعنی اینکه این جواب به درد شما نمی خورد موجه نیست. خوب حالا چی کار کنیم ؟ باید پول خرج کنید و منبع بیشتری بخرید یعنی a=۵ را a=۶ قرار دهید. دوباره *RCPSP* حل کنید. اینبار $C_{max} = ۲۴$. خوب بهتر شد ولی بازم از δ_n یعنی همان ۲۰ بیشتر است. پس هیچی . دوباره حل کنید. اینبار a=۷ قرار می دهیم بعد محاسبه *RCPSP* را انجام می‌دهیم و $C_{max} = ۱۹$ می‌شود. در این حالت تمام است. همین جواب بهینه است . یک روش حل *RACP* همین است که الان گفته شده است. از مینیمم ترین a که این a مینیمم خودش ماکسیمم نیازمندی است شروع کنید و یکی یکی اضافه کنید تا زمانی که *dead line* راضی بشود تا زمانی که محدودیت *dead line* موجه بشود به محض اینکه موجه شد آن بهینه است یاد کدام روش می‌افتید ؟ روش سیمپلکس ثانویه . از یک جواب فرابهینه ناموجه شروع می‌کرد و گام به گام به سمت موجه شدن جلو می‌رفت. اولین نقطه‌ای که موجه می شد همان جواب بهینه می‌شد.

$\delta_n = ۲۰$

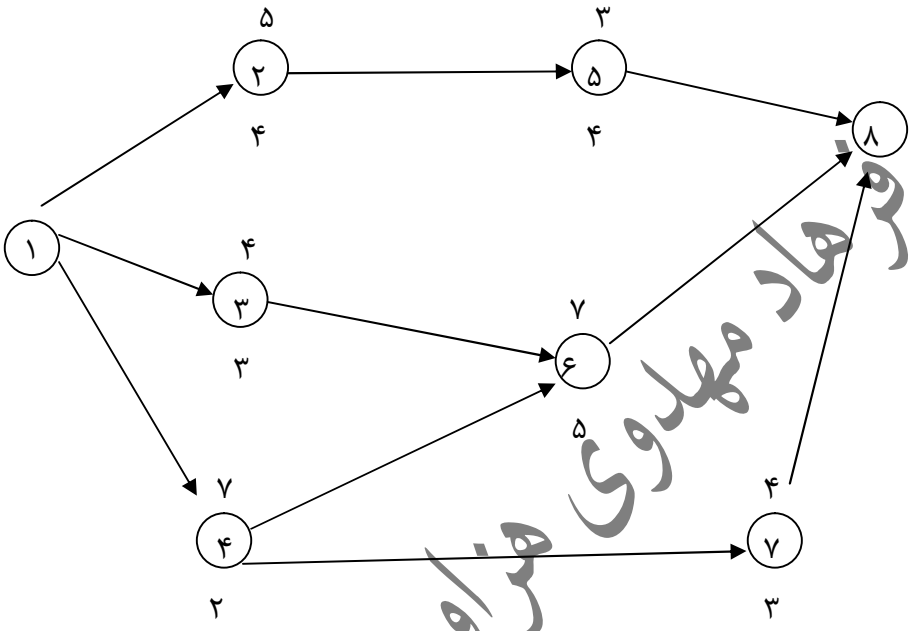
a=۵ $\xrightarrow{RCPSP}$ $C_{max} = ۲۹$

a=۶ $\xrightarrow{RCPSP}$ $C_{max} = ۲۴$

a=۷ $\xrightarrow{RCPSP}$ $C_{max} = ۱۹$ ✓

الان اگر دقت کنید a=۵ خیلی خوب است . عالیه . فرابهینه است ولی موجه نیست یکم نزدیک می‌شویم . ۶ که موجه نشد و بعد ۷ تا زمانی که موجه شد آن جواب بهینه خواهد بود پس ما می‌توانیم با استفاده از اطلاعات هفته گذشته و با استفاده از *RCPSP*، *RACP* را حل کنیم . چون عملاً حق *RACP* حل تعدادی *RCPSP* است. در این مثال حل ۳ تا *RCPSP* بود. ولی ممکن است در یک مسئله‌ای برای حل *RACP* مجبور شوید ۱۰۰ تا *RCPSP* حل کنید. آره *dead line* جزء داده‌های مسئله است اگر *dead line*را به شما ندهد بهینه چند می‌شود ؟ همان ۵ .

اگر مسئله *dead line*را ندهد یعنی آخر پروژه باز باشد خوب بهتر من با کمترین منبع کار را انجام می‌دهم یعنی با ۵ . همان جواب بهینه است ولی وقتی که مسئله *dead line* را می‌دهد مجبور هستم منبع را یکی یکی زیاد کنم تا ببینم که *dead line* چه هنگام راضی می‌شود.



مسئله ۵ : *RCPSP-NPV*

مسئله زمانبندی پروژه با محدودیت منابع و *NPV* هم یادتان است دیگه. که در پارت ۸ یک مسئله حل کردیم برای ماکسیمم کردن *NPV*فوقش نکته این است که آنجا منبع مشکل نداشتیم محدودیت منبع نداشتیم ولی الان محدودیت منبع داریم ببینید ما *RCPSP* را قبلاً صحبت کردیم، *NPV* را هم صحبت کردیم خوب الان چی دارم می‌گویم ؟ یکپارچه ، ببینید یک نوآوری به کل توی تعریف یک مسئله این است که شما دو مقاله جدا بخوانید باید بتوانید اینها را به صورت *merge* شده و یکپارچه مدل و حل کنید و در اینجا این دو موضوع را به صورت جدا آشنا هستید ولی الان می‌خواهیم با هم حل کنیم. *NPV* با چه روشی حل می‌شد ؟ آقای ون هاک یک روش *recursive* ، که ۳ تا گام داشت*current tree , early tree* و بعد یک روش *recursive* بود یک *tree* درست می‌کردید از شاخه کنده می‌شود اضافه می‌شود یک جلسه کامل راجع به آن صحبت کردیم تابع هدفش را هم من کپی کردم.

$$\text{Maximize} \sum_{i=2}^{n-1} c_i e^{-\alpha(s_i+d_i)}$$

subject to

$$s_i + d_i \le s_j \qquad \forall (i,j) \in E$$

$$\sum_{i \in S(t)} r_{ik} \le a_k \qquad k = 1,2,\ldots,m\ ,\ t = 1,2,\ldots,\delta_n$$

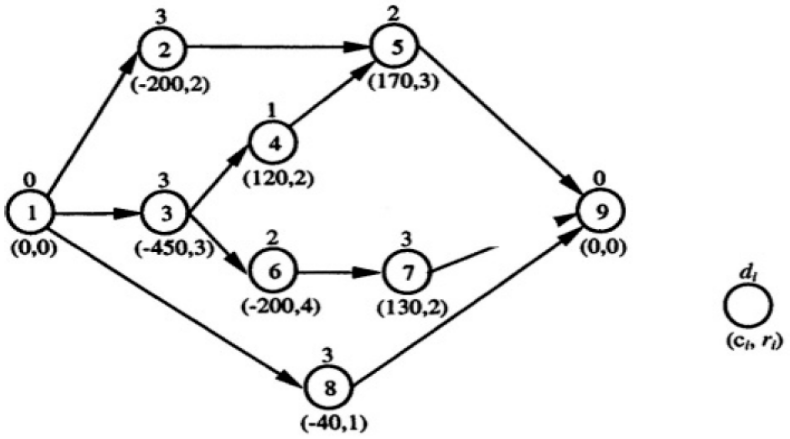
$$s_1 = 0$$

$$s_n \le \delta_n$$

$$s_i \in \mathbf{N} \qquad i = 1,2,\ldots,n$$

تابع هدف از پارت ۸ کپی کردم محدودیت‌ها را هم از *RCPSP* کپی کردم پس مدل‌هایی که *merge* هستند مدل‌سازی خیلی نداریم چون کپی است پس موضوع را به نوعی دارم مونتاژ می‌کنم پس *NPV* را جدا بلد هستید که حل کنید و *RCPSP* ، ۴ روش هفته قبل یاد گرفتید که آخرین روش آن *minimal forbidden set* بود این مسئله می‌توانید با ترکیب *minimal forbidden set* هفته قبل با پارت ۸ حل بشود.

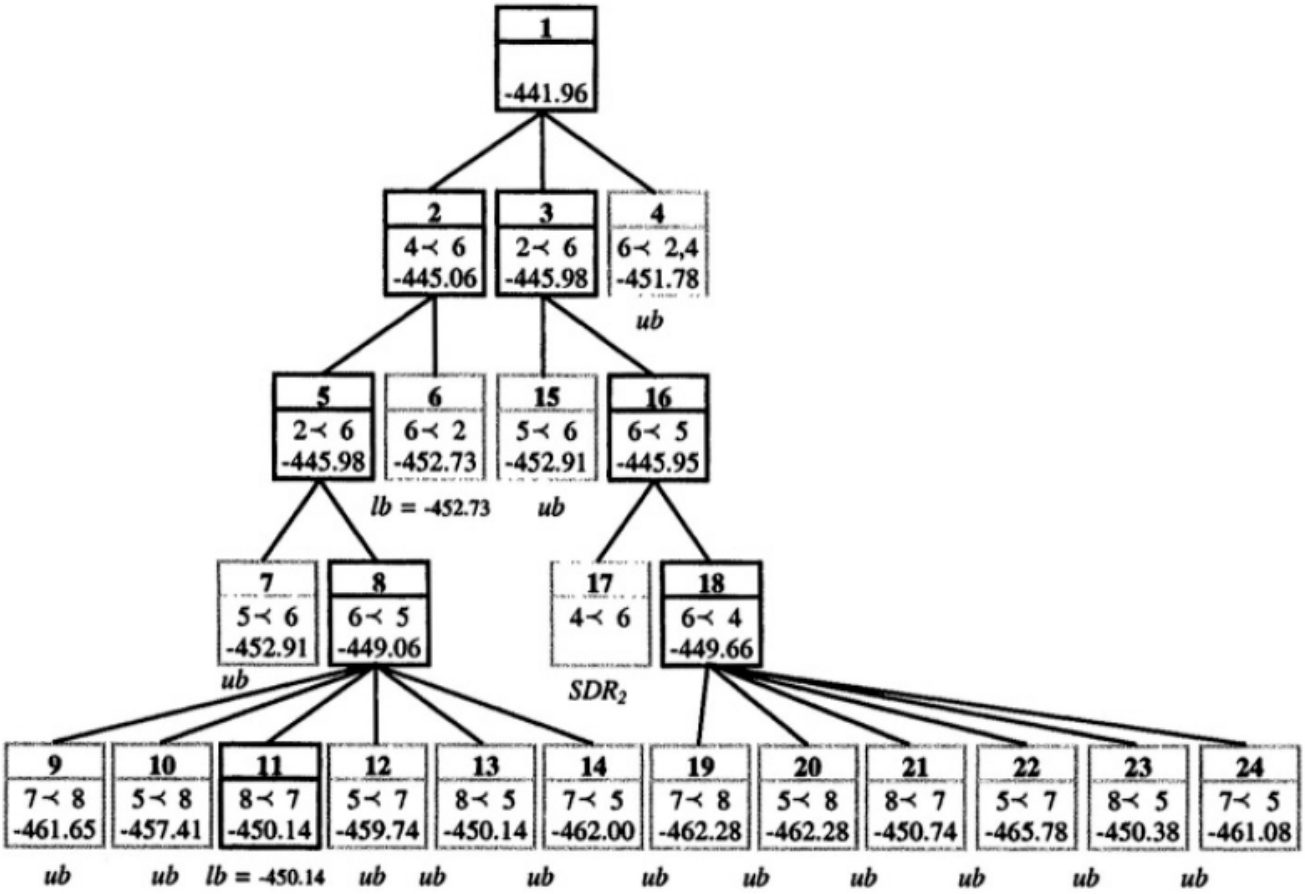
یک مثال بیارم بعد توضیح‌اش. بازم ۹ تا فعالیت داریم که فعالیت ۱ و ۹ مجازی است این اعداد فقط چی هستند ؟



اعدادی که بالای فعالیت هستند *duration* می‌باشند و عدد پایین سمت راست فعالیت نیازمندی منبع را نشان می‌دهد و عدد پایین سمت چپ فعالیت C_i را نشان می‌دهد. C_i همان *cash flue* است یعنی یا هزینه است یا درآمد است آنهایی که مثبت‌اند درآمد می‌باشند و آنهایی که منفی است هزینه می‌باشند. C_i را *cash flue* می‌گفتیم جریان نقدی آن فعالیت . برآیند مالی فعالیت مثلاً فعالیت ۷، ۳ روز طول می‌کشد و ۲ تا منبع می‌خواهد ۱۳۰ واحد پولی درآمد دارد ولی فعالیت ۶، ۲ روز طول می‌کشد ۴ تا منبع می‌خواهد و ۲۰۰ واحد پولی هزینه دارد و الی آخر.

نکته : a=۵ . در این مثال a را ۵ قرار دهید یعنی ما ۵ تا منبع داریم. محدودیت منبع داریم، بفرمایید که ۲ و ۳ *minimal forbidden set* هستند یا نه ؟ *minimal forbidden set* چی بود ؟ فعالیت‌هایی که پیش نیاز هم نیستند و منبع برای انجامشان کافی نیست . آیا ۲ و ۳ *forbidden set* هستند ؟ ۲ و ۳ پیش نیاز هم نیستند ولی منبع ۵=۳+۲ ، ۵ تا منبع دارد کافی است پس یک شرط را دارد و یک شرط را ندارد. پس هیچی *forbidden set* نیستند. اما مثلاً ۴ و ۵ و ۷ چی ؟ بازم نیستند ۴ که پیش نیاز ۵ است نمی‌شود اما مثلاً ۲ و ۴ و ۶ می‌توانید بگویید *forbidden set* هستند ؟ چرا ؟ ۲ و ۴ و ۶ پیش نیاز هم نیستند و ۸ تا منبع می‌خواهد که ۵ تا منبع داریم حالا *minimal forbidden set* برای اینکه مشکل اینها را حل کند چی کار می‌کرد ؟ بعضی‌ها را پیش نیاز اون یکی می‌کرد ؟

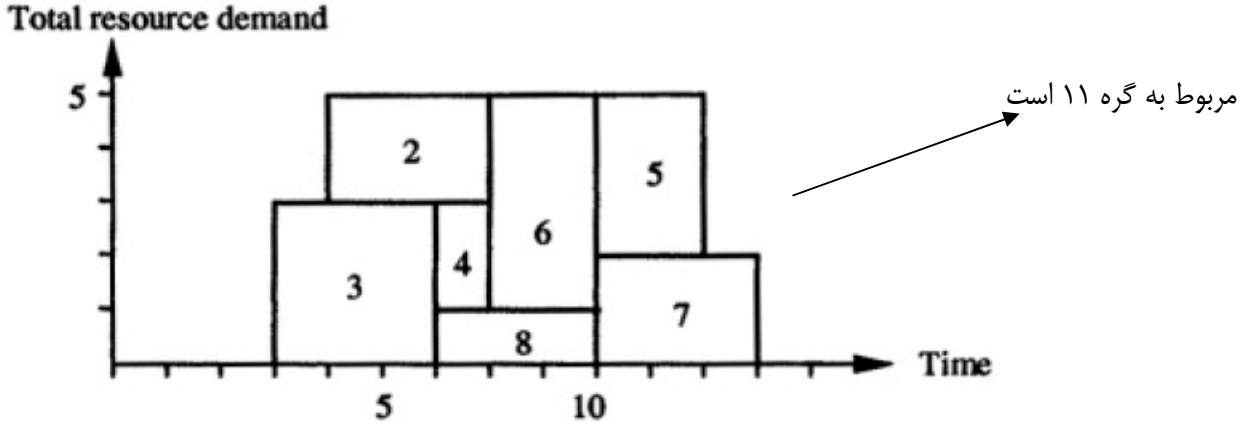
این مثال را من حل کاملش را اینجا آوردم .



این شاخه و کران کلی مسئله است حالا ***forbidden set*** ها را باید امتیاز حساب می‌کرد من دیگه تکرار نکنم الان اگر اینجا خوب دقت کنید اول کدام ***forbidden set*** را حل کرده است *{۲,۴,۶}* چه طوری مشکل اینها را حل کرده است ؟ گفته یا باید ۴ پیش نیاز ۶ بشود یا ۲ پیش نیاز ۶ شود و یا ۶ پیش نیاز ۲ بشود. بعد کدام شاخه را ادامه داده است ؟ ***best first*** آن‌هایی که بهتر است بهتر یعنی اینجا تابع هدف ماکسیمم است یعنی بیشتر ، آنی که چی‌اش بیشتر است ؟ ***lower bound*** نباید بگوید باید بگویید ***Upper bound*** اش بیشتر است. مسائلی که مینیمم هستند ***lower bound*** هست اینجا مسئله ***upper bound*** است پس بین این ۳ تا شاخه کدام را اول انجام دادیم ؟ کدام بیشتر است ؟ این بیشتر است دیگه . ۴ را پیش نیاز ۶ کرده است. بعد بین *{۲,۶}* و *{۶,۲}* بازم ***forbidden set*** . ۲ پیش نیاز ۶ باشد بهتر است یا ۶ پیش نیاز ۲ . ***upper bound*** این بیشتر است. پس این را ادامه بدهید ۲ را پیش‌نیاز ۶ کنید . بعد ۵ پیش نیاز ۶ باشد یا ۶ پیش نیاز ۵ باشد بهتر است؟ این بهتر است. بعد خلاصه شاخه‌ها به عمق رسیده است. این شاخه‌ها که به عمق رسیده است کدام از همه بهتر است ؟ ۴۶۲- و ۴۵۰- و ۴۵۹ و ۴۵۰/۱۴ و ۴۵۷/۴۱ و ... نتیجه : گره ۱۱ به عمق رسیده است . این جواب را ذخیره می‌کنیم به عنوان بهترین جواب تا این لحظه . ۱۳ هم هست ولی اونم عین همین است و بهتر از این که نیست پس بهترین جوابی که تا این

لحظه پیدا کردم جوابش ۴۵۰/۱۴- است ؟ حالا شاخه هایی که کمتر از این هستند دیگه به درد شما نمی خورند کدام ها کمتر از ۴۵۰/۱۴- هستند ؟

از آن بالا چک کنید. مثلا این چرا بسته شده است ؟ چون ۴۵۱ ، ۴۵۱- ، ۴۵۰- بود دیگه این بسته شد. اما این را ببینید این را نتوانسته ببندد چون بهتر است باز کرده است *{۵,۶}* یا *{۶,۵}* اینی که بهتر است *{۶,۵}* است ۶ را پیش نیاز ۵ کرده است. در ادامه، *{۴,۶}* یا *{۶,۴}* این *{۴,۶}* است که نوشته ***SDR*** این را بنویسید ناموجه این شاخه ناموجه بوده است که بسته شده است. آنهایی که ناموجه هستند یا تکراری هستند بسته می‌شوند. خوب در ادامه‌اش این را ببینید بازم بهتر است ادامه داده است ادامه‌اش را ببینید . ۴۶۲ فایده ندارد . ۴۵۰.۷۴ این ۱۴ بهتر بود پس همه بسته شد کدام موند همان گره ۱۱ باقی ماند یعنی چی‌ها پیش نیاز بشوند گره ۱۱ یعنی ۴ پیش نیاز ۶ ، ۲ پیش نیاز ۶ ، ۶ پیش نیاز ۵، ۸ پیش نیاز ۷ اینها را پیش نیاز کرده است با همان پارت ۸ حل کرده است و اینم گانت بهینه‌اش.



این گانت بیهینه مربوط به گره ۱۱ است. یعنی جواب بهینه را رسم کرده است و بقیه را دیگر رسم نکرده است. ۳ روز اول پروژه خالی است درست می‌گویید هیچ منبعی همه بیکار همه ماندند. خوب ***left shift*** بدهیم . نمی‌شه ؟ نه ***left shift*** برای چی بود ؟ برای ***make span*** بود الان ***make span*** نیست.

۳ تا سوال :

۱- آیا این پروژه اقتصادی است ؟ وقتی ***NPV*** منفی است پروژه اقتصادی نیست. پروژه‌ای که اقتصادی نیست بهتر است که انجام نشود اما اگر زوری است باید انجام شود بهتر است کی انجام شود. دیرترین زمان به خاطر این است که این را البته من یادم رفت بگم در این مثال ***dead line***، ۱۳ است. ۱۳ $= \delta_n$ این جزء داده‌های مسئله است. اگر دقت کنید پروژه کی تمام شده است؟ ۱۳

۲- اگر ***dead line*** ، ۱۵بود چه اتفاقی می‌افتاد ؟ عیناً همین فقط ۲ روز دیگه می‌رفت اگر ***dead line*** ، ۱۰۰۰ بود چی ؟ می‌رفت می‌چسبید به ۱۰۰۰ یعنی پروژه‌هایی که اقتصادی نیست در دیرترین زمان ممکن انجام می‌شود از کجا فهمیدید که اقتصادی نیست از آنجایی که ***NPV*** منفی است.

۳- چه طوری این ***upper bound*** ها حساب شدند ؟ از کجا حساب شدند ؟ همان بالای بالا، ۴۴۱/۹۶- از کجا آمده است؟ آره ***RCPSP*** را کنار گذاشته است و رفته از پارت ۸ ، ***NPV*** را حساب کرده است بدون منبع

آن وقت این چه طوری حساب شده است که شد ه ۴۴۵/.. و این چی بازم همین طور فقط با این تفاوت که ۴ را پیش نیاز ۶ کرده باز این را گذاشته کنار رفته از پارت ۸ *upper bound* را حساب کرده است.

برای حل این مسئله از روش شاخه و کران با منطق *minimal forbidden set* استفاده می شود و در هر گره *upper bound* از مسئله *max-npv* محاسبه می‌شود.

مسئله ۶ : *RCPSP –ET*

ET یعنی *earliness – tardiness* ما بازم این را در پارت ۹ داشتیم. مسئله‌ای که مینیمم می‌کرد *earliness – tardiness* را ولی بدون منبع حالا الان با منبع . یعنی بازم یک مسئله به اصطلاح یکپارچه سازی دو تا موضوع است. منبع را جدا بلد هستیم ، *earliness – tardiness* هم جدا و حالا اینجا به صورت یکپارچه. زیاد توضیح نمی‌خواهد فقط مدلش را من اشاره کنم که تابع هدف را از پارت ۹ کپی کرده است.

$$\text{Minimize} \sum_{i=2}^{n-1} (e_i E_i + t_i T_i)$$

subject to

$$f_i \le f_j - d_j \qquad \forall (i,j) \in E$$

$$E_i \ge h_i - f_i \qquad \forall i \in V$$

$$T_i \ge f_i - h_i \qquad \forall i \in V$$

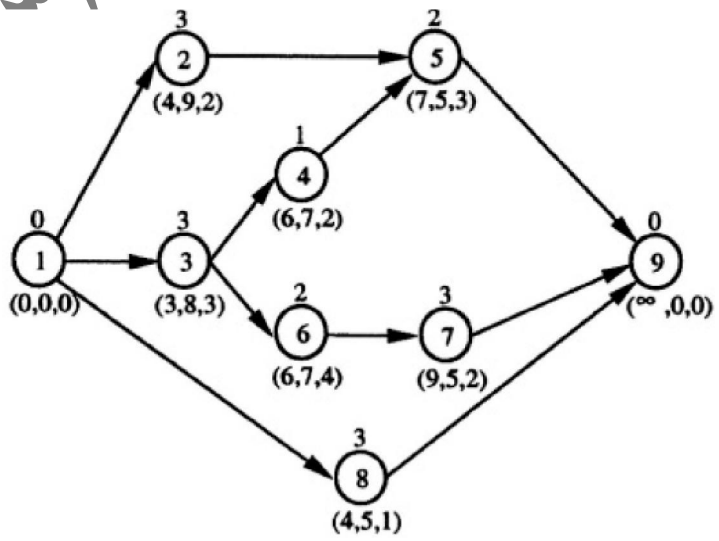
$$f_1 = 0$$

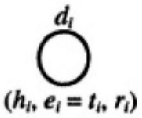
$$\sum_{i \in S(t)} r_{ik} \le a_k \qquad k = 1,2,\ldots,m,\ t = 1,2,\ldots,T$$

$$f_i \ge 0 \qquad \forall i \in V$$

اینها هم محدودیت‌های مربوط همان *RCPSP* و *earliness – tardiness* است و البته محدویت منبع اینجا هم هست.

اینم مثال آن می‌باشد.



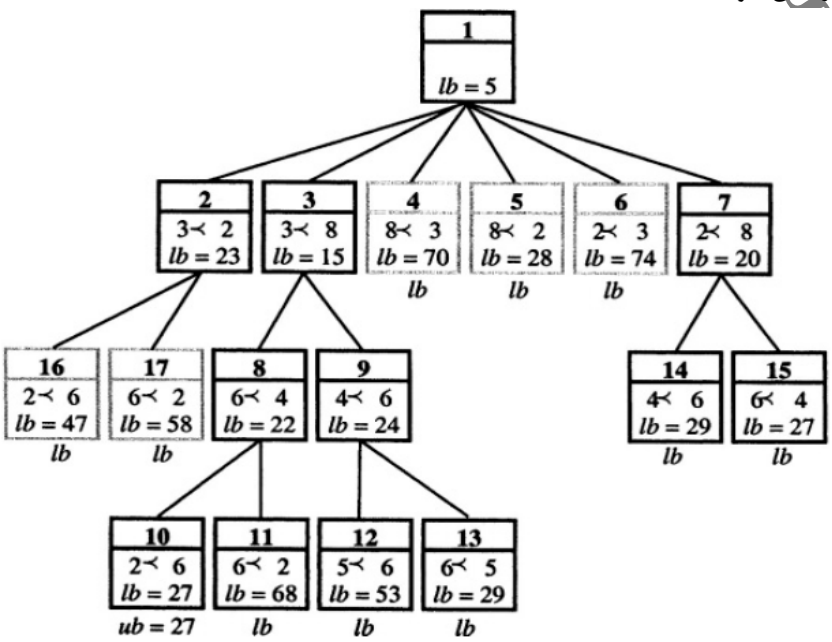


اعداد چی هستند ؟ اینجا راهنما دارد مثلاً فعالیت ۸ ، ۳ روز طول می‌کشد و یک منبع می خواهد عدد اول از سمت چپ h_i است *due date* را در پارت ۹ یادتان می‌آید. *Due date* موعد تحویل است . یعنی بهتر است که این فعالیت ۴ ام تمام شود اگر تمام نشد به ازای هر روز زودکرد یا دیرکرد ، ۵ دلار جریمه می‌شوید عدد وسط در پایین فعالیت جریمه زودکرد و یا دیرکرد است. آره فرض کرده است که جریمه دیرکرد و زودکرد مساوی هستند والا باید دو تا عدد می‌داد فقط تابع هدف دوباره چی شد ؟ مینیمم است . باز *max* بود شد *min*بازم باید *lower bound* حساب کنید . دیگه نباید *upper bound* بگویید. آنهایی که مینیمم هستند *Lower bound* به در می‌خورد و آنهایی که *max* بود *max –NPV*، *upper bound* بود.

خوب بازم ادغام روش *minimal forbidden set* با روش *earliness – tardiness*

این مسئله نیز با روش *minimal forbidden set* قابل حل است و در هر گره مقدار *Lower bound (LB)* از مسئله مینیمم کردن جریمه‌های زودکرد و دیرکرد به دست می‌آید.

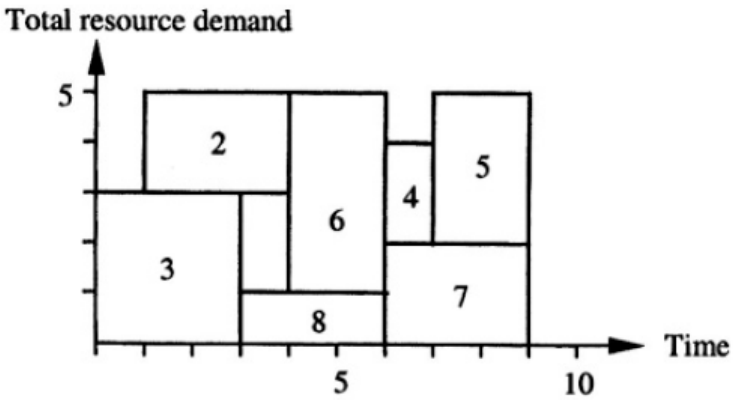
اینم مثالی را که همین الان دیدیم را حل کرده است.



بازم با هم مسیر حل‌اش را با هم طی می‌کنیم از آن بالای بالا شروع کنید از همان شروع *lower boune* که ۵ است را از کجا آورده است ؟ این مسئله *LB=*۵ گفته است از کجا آورده است ؟ نه، ۵ منبع نیست. ۵ جریمه است الان ۵ واحد پولی ، ۵ دلار از کجا آورده است؟ گفته ۵ دلار جریمه زودکرد- دیرکرد دارد هنوز حل نکرده است. گفتم که همین الان نوشتید. *RCPSP* را به طور کل کنار گذاشته و به پارت ۹ برگشته و مسئله را بدون منبع با آن روش‌ایی که در پارت ۹ بود حل کرده است. و جریمه *earliness-tardiness* ، ۵ شده است. این *lower bound* شده است. ولی خوب الان جواب مسئله که ۵ نیست چون بدون منبع حل کردید حالا باید با منبع پاسخ بدهید با منبع را شاخه زده است.

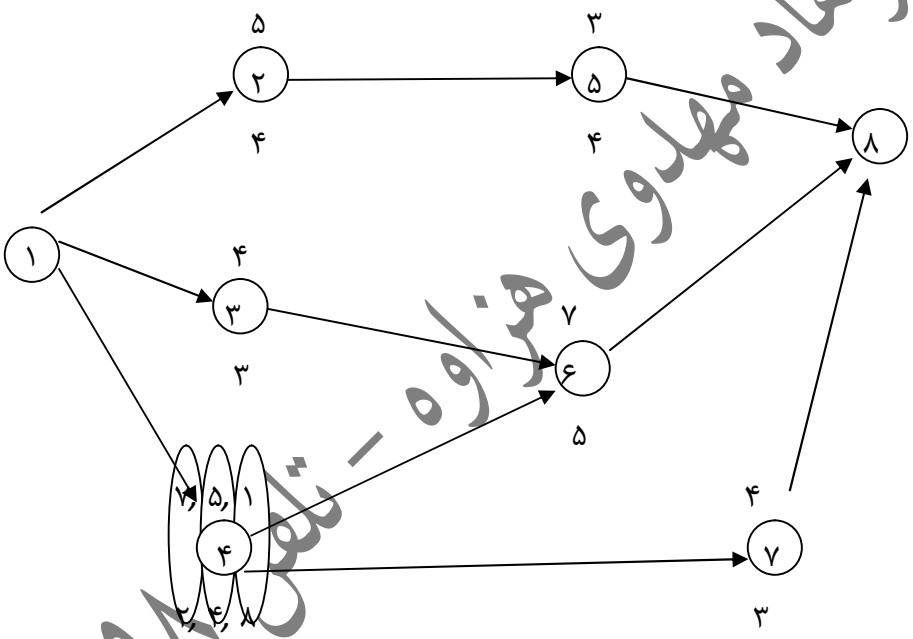
گفته مشکل منبع را حل می‌کنم چه طوری حل کرده است ؟ گفته که مثلاً اینها *Forbidden set* هستند و *forbidden set* ها را با هم دیگه .... مثلاً دقت کنید *{۳,۲,۸}* را پیدا کنید. ۲ و ۳ و ۸ ، *forbidden set* هستند منبع به حد کافی براشون نیست. چه طوری این را حل کرده است؟ گفته یکی باید پیش نیاز یکی دیگه بشود مثلاً گفته ۲ پیش نیاز ۸ شود یا ۸ پیش نیاز ۲ شود تا الا آخر بعد *lower bound* ها را چه طوری حساب کرده است؟ باز با همان *ET* ولی مثلاً اینجا این ۷۴ را با همان پارت ۹ بدست آورده است که فقط فرقش با آن بالایی این است که یک پیش نیازی ۲ به ۳ را هم اضافه کرده است یعنی در این شبکه‌ای که شما می‌بینید یک بردار از ۲ به ۳ وصل کرده است یعنی ۲ را پیش‌نیاز ۳ کرده است بعد رفته به پارت ۹. پس با این مکانیزم *Lower bound* را همه را حساب کرده است فوقش هر بار با یک پیش نیازی متفاوت. همه حالت‌ها. آن وقت کدام بهتر است ؟ ۲۰، ۷۴، ۲۸، ۷۰، ۱۵، ۲۳ که ۱۵ از همه بهتر است چون *best first* است اونی که بهتر است . پس اول این شاخه را پیش رفته است یعنی دفعه بعد اومده بین *{۴,۶}* ، *{۶,۴}* بازم *Lower bound* ۲۲ بهتر است. دفعه بعد بین *{۲,۶}* یا *{۶,۲}* ، ۶۸ ، ۲۷ که ۲۷ بهتر است که این به عمق رسیده است. پس این کنارش بنویسید اولین جواب موجه، این به عمق رسید یعنی جوابی پیدا کرد به اسم ۲۷ .

این را نخوانید این *upper bound* را نخوانید این را فعلاً ننوشته هنوز، بعداً قرار بنویسید. یک *Lower bound* ۲۷، به دست آوردید این دیگه یک دلیلی می شود برای بستن شاخه‌ها کدام شاخه‌ها بسته می‌شوند آنهایی که *Lower bound*اش، ۲۷ یا بالاتر است مثلاً ۶۸، ۵۳، ۲۹، ۷۰، ۲۸، ۷۰ اینها بسته شد. اما این ۲۰ یا این ۲۳ را نمی‌توانید ببندید مثلاً این ۲۰ را باید ادامه بدهید ادامه داده شده ، ۲۹ ، ۲۷ . جفتشان بسته شدند. این ۲۳ را باید ادامه بدهید که شده ۴۸، ۴۷، ۵۸ خوب بازم بسته شد. تمام همه را بستیم نتیجه : جواب بهینه. الان اینجا چی نوشته ، *lower bound* =۲۷ و *Upper bound*=۲۷ است یعنی چی ؟ یعنی جواب ۲۷ است. چه زمانی اینها رون نوشته ؟ وقتی که مطمئن شد که همه شاخه‌ها بسته شدند آن جوابی که بدست می‌آید بهینه هم *upper bound* است و هم *Lower bound* . من گانت بهینه این گره را هم آوردم . آن جواب بهینه این مسئله است کدام‌ها پیش نیاز شدند ؟ ۳ پیش نیاز ۸، ۶ پیش نیاز ۴ و ۲ پیش نیاز ۶ شده است. بقیه چی ؟ چرا اینها ؟ خوب بهینه‌اش را گفتم بقیه دیگه بهینه نبودند و بسته شدند آنهایی که به دلیل *lower bound* بسته شدند گزارش نکردم فقط گانتی را که بهینه بود را آوردم. و الا آنها هم بررسی شدند.



مسئله ۷ : *MRCPSP*

کلمه *M* در *MRCPSP* به معنی *Multi-mode* می‌باشد پس *multi mode MRCPSP* *Mode* یعنی روش انجام . ما در حالت کلاسیک وقتی می گوییم *RCPSP* منظور این است که هر فعالیت چند تا روش انجام دارد؟ یکی آنها هم *single mode* می‌گوییم. ولی نمی‌نویسیم می‌گوییم *RCPSP* نمی‌گوییم *single mode RCPSP* وقتی چیزی نمی‌گوییم منظور این است که *mode* اش یکی است . مثلاً فعالیت ۴ را دقت کنید این یک *mode* چه طور انجام می‌شود. ۷ روز با ۲ تا منبع . یک راه دیگه انجام این کار این است که این را تو ۵ روز هم انجام بدهیم ولی با ۴ تا منبع . یک راه دیگه اش این است که در یک روز انجام بدهید ولی با ۸ تا منبع و هر کدام از اینها یک *mode* می‌گویند.



Mode اول ، *mode* دوم یعنی هر ترکیبی از منبع و زمان را یک *mode* می‌گویند. بعضاً هم ترکیبی از زمان و هزینه را می‌گویند. نمی‌گویند منبع، می‌گویند هزینه یعنی چی ؟ فارسی بگم.

Mode اول : ۷ روز با ۲ میلیون تومان هزینه نمی‌گه ۲ تا منبع ، پولش را می‌گوید هزینه را می‌گوید.

Mode دوم : ۵ روز با ۴ میلیون تومان

Mode سوم : ۱ روز با ۸ میلیون تومان می‌توان انجام داد.

این مسئله را *time cost* یا *time resource trade off* می‌گویند. *Trade off* یعنی موازنه . موزانه زمان و هزینه . چرا موازنه می‌گویند. اگر این *mode* ها دقت کنید کدام بهتر است ؟ چرا می‌گویید *{۱,۸}* ؟ چون یک روزه است ؟ ولی خوب کلی منبع می‌خواهد یا کلی هزینه می‌خواهد.

این یکی هم که خیلی هزینه‌اش کم است ولی طول می‌کشد. یعنی می‌خواهی زمان را کم کنی منبع زیاد می‌خواهد هزینه زیاد می خواهد، برعکس می‌خواهی منبع را کم کنی به زمان بیشتری نیاز خواهی داشت. *Trade off* یعنی ما دنبال این هستیم که این را یک طوری *balance* کنیم که نه خیلی طول بکشد نه خیلی هزینه بخواهد.

این مسئله سخت‌تر است یا *RCPSP* البته خیلی واضح است که این سخت تر می‌باشد. چرا ؟ چون دو مسئله است باید *mode assign* انجام بدهید تخصیص *mode* یا *mode selection* بعد تازه حالا می‌شود یک *RCPSP*. این در واقع یک گام بیشتر دارد باید اول *mode* تخصیص بدهید که کدام *mode* انتخاب بشود بهتر است بعدش که حالا *mode* ها انتخاب شد تازه یک*RCPSP* می‌شود. هر فعالیت باید یک *mode* براش انتخاب بشود کدام *mode* ؟ آنی که بهینه است. از کجا بدانیم که بهینه است ؟ باید مسئله را حل کنیم.

من این مسئله را ، چند تا فعالیت داریم ؟ ۷ تا فعالیت واقعی داریم. فرض کنید اینها این جوری هستند . اینها همه ۳ تا *mode* دارند پس این ۳ تا *mode*. اینها را من دیگه ننویسم شلوغ می‌شود ولی شما فرض کنید که همه فعالیت‌ها در این مثال ۳ تا *mode* دارند. قسمت *Mode assign* چند حالت مختلف دارد. یعنی به چند حالت به اینها می‌توانید *mode* تخصیص بدهید؟ هر فعالیتی ۳ تا حالت دارد هر فعالیت ۳ تا *mode* دارد. در کل اگر بخواهم به فعالیت‌های پروژه *mode* تخصیص بدهم به چند حالت مختلف می‌توانم انجام بدهم ؟ درس احتمالات است. $۳^{۷} = ۲۱۸۷$ یعنی قسمت *mode assign* اش بیش از ۲۰۰۰ حالت دارد برای هر کدام چی باید حل کنیم ؟ یک *RCPSP* که چی بشه ؟ که ببینید کدام *mode assign* بهتر است. ۲۱۸۷ حالت مختلف می‌شود *mode* تخصیص داد به این مسئله بسیار ساده چون خیلی کوچک است همش ۷ تا فعالیت دارد که فعالیتی نیست. الان اگه شما از اینجا تا خانه بروید ۱۰ تا فعالیت دارد یک پروژه معمولاً در حد چند صد تا ، هزار تا فعالیت دارد نه در حد تعداد انگشت.

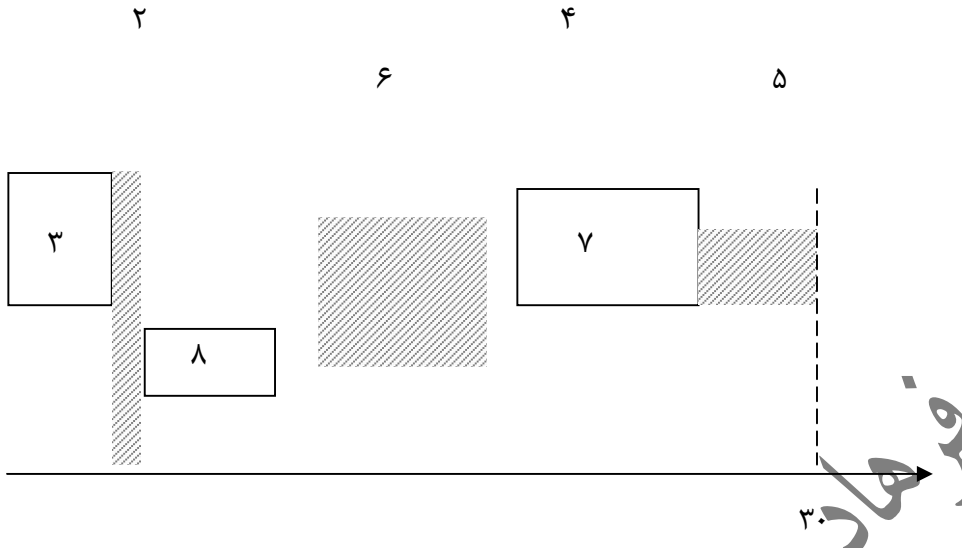
پس من می‌خواهم بگویم که مسئله *multi mode* یک مسئله بسیار وحشتناکی است ببینید شما یادتان نرود یک مسئله *RCPSP* خودش به تنهایی چه طور مسئله‌ای است ؟ قبلاً نوشتیم *strongly NP- hard* یعنی *RCPSP* خودش یک مسئله *Strongly NP- hard* است حالا حساب کنید که یک مسئله *multi mode* تشکیل شده از چند هزار تا یا بعضاً چند صد هزار تا *RCPSP* است که ما در یکی این هم مشکل داریم . حالا با یکی‌اش می‌توانستیم یک کاری بکنیم اما دیگه برای *MRCPSP* از حل *exact* اش حتی برای یک مسئله‌ای به این سادگی هم به صورت کامل نا امید می‌شویم. خوش بین مطلق نیستیم که بتوانیم این را حل کنیم همین مثال کوچک را اگر بخواهیم در کلاس حل کنیم اصلا غیر ممکن است با ۲۱۸۷ تا مسئله حل کنیم که اصلا شدنی نیست. پس مسئله اش بسیار حشن است که معمولاً برای حلش از روش‌های هوش محاسباتی استفاده میشود. روش‌های ابتکاری ، فرا ابتکاری نه فقط بتواند یک جواب قانع کننده بدهد دیگه جواب *exact* ایی را ما قطعاً براش انتظار نداریم ولی خوب *RCPSP*را چرا حالا یک کارش می‌کنیم ۴ تا روش را هم براش یاد گرفتیم. ولی این یکی را کاریش نمی‌توان کرد.

مسئله ۸: مسئله *Robust-Scheduling* (زمانبندی پایدار)

Robust یعنی پایدار.

این داده‌های مسئله از کجا می‌آید؟ شما از کجا می‌دانید که فعالیت ۶ ، ۷ روز طول می‌کشد؟ اینجا یک کانال به طول ۱۰، به عرض ۰.۵ متر و به عمق ۱ متر حفر کنید حالا یک کارگر یا با ۵ تا کارگر می‌خواهند این را انجام بدهند از کجا می‌گویند که انجام این کار ۷ روز طول می‌کشد؟ مثلاً یک آزمایشی را انجام بدهیم یا نه از داده‌های قبلی ، مثلاً کسی که کارش کلاً جوشکاری است خوب یک برآوردی دارد که به طور تجربی می‌تواند بگوید که چقدر طول می‌کشد تا این کار را انجام دهد ولی خوب هر طوری هم این موضوع را بگویید قطعاً نمی‌توانید دقیق بگوید. درست می‌گم ؟ چرا ؟ چون پروژه تعریف چی دارد؟ ***Unique*** دارد. ببینید ما یک عملیات ***(Operation)*** داریم و یک پروژه داریم. و فرقش در این است که عملیات یک کار تکراری است مثلاً شما نانوایی می‌روید مثلاً هر فردی که آنجا هستند هر کدام کار مشخصی دارند ولی این کار را روزی صد بار ، هزار بار (یک کار ثابتی را انجام می‌دهد) مثلاً کسی که آن چونه ها را وزن می‌کند دیگه احتمالاً به صدم ثانیه هم می‌تواند به شما بگوید که درست کردن این چونه چقدر طول می‌کشد. یا وزن کردنش چقدر طول می کشد یا آن شاتر اصلی می‌تواند دقیقاً به شما بگوید که پخت نان فرآیندش چند دقیقه است. چون یک کار ثابتی است. اما در پروژه کار کار ثابتی نیست حتی آن کار پی‌کنی هم کار ثابتی نیست. چرا ؟ چون صخره‌ای که این زیر است را که ندیده است. درست است ؟

اگر زیر این کانالی که می‌خواهید حفر کنید صخره باد که آن را دیگه ندیده است اگر لوله‌گذاری از آنجا رد شده باشد که نمی‌دونه و موقع حفر کردن باعث ترکاندن لوله شود و ۷ روز، ۷۰ روز شود که به اینها فکر نکرده است. پس پروژه به خاطر ذات ***unique*** و منحصر به فردش نمی‌تواند داده‌هایش دقیق باشد این یک موضوع . ثانیاً در برنامه ریزی پروژه همه چی خوش‌بینانه است یعنی ما فرض می‌کنیم که هیچ پیمانکاری بد قول نباشد مواد اولیه به موقع برسد. هوا آنی باشد که ما دوست دارید یعنی همچین یخ بندان نشود که نتوانید کار کنید یا بارندگی آنقدر شدید نباشد که نتوانید آن روز بتن‌ریزی انجام دهید. و الی آخر خوب همه چه چیز تا حدی خوش‌بینانه است ولی عملاً موقعی که می‌روید در گیر پروژه می‌شوید این اتفاق که نمی‌افته درست می‌گم؟ به هزاران دلیل متفاوت از طرف پیمانکار، از طرف تأمین کننده، از طرف کارفرما (اصلاً پول نداده که کار را انجام دهید) و حادثه (حتی یک حادثه هم می‌تواند باعث شود یک ماه درگیر کار قانونی بشوید که کارتان بخوابد. مثلاً یک کارگر افتاده و دستش شکسته است تمام است دیگه اگر شکایت کند پروژه تعطیل است شما بیمه نکردید و ...) پس می‌خواهم بگویم عدم قطعیتی که وجود دارد کار را خیلی سخت کرده است اما شما آن بالایی که گانت درست کردید خیلی خط کشی شده و ظریف و فعالیت‌ها را خیلی مرتب کنار هم چیده‌اید که این به اون تبدیل به انجام بشود . همان فعالیت ۳ را نگاه کنید. که امروز باید شروع می‌شود امروز هوا خوب نبود و یا حالا آن تیمی که باید می‌اومد تا زیرسازی را انجام بدهد نیامده نتیجه ۳ را ببینید چه خبر ؟ اگر ۳ عقب بیافتد پشت سر ۸ بعدش ۷ عقب می‌افتد خوب حالا حساب کن تا آخر ۷۰۰ تا فعالیت اینجا هست یعنی تلنگوری که به یک فعالیت زدیم یعنی کل پروژه را به هم ریخته است این یعنی اینکه زمانبندی پایدار نیست.



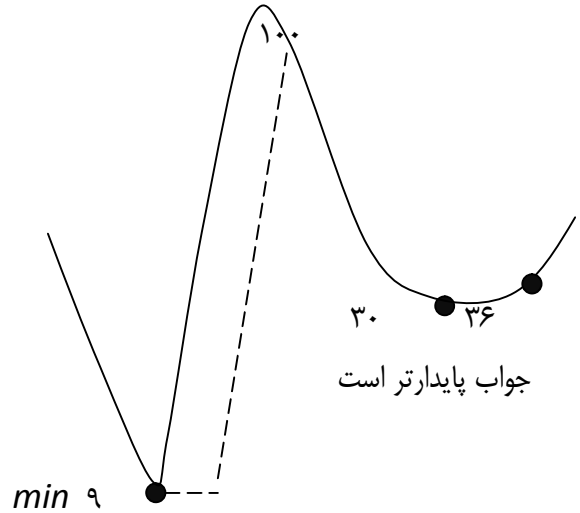
پایدار اسمش روش است دیگه یعنی باید برنامه ، ***Base plan*** با آنی که اجرا می‌شود زیاد فرقی نداشته باشد زیاد به هم نریزد راه کار چیست ؟ که این را پایدارش کنم ؟ می‌خواهم این را دوباره بچینم ولی پایدارتر. ۳، ۲، ۸ را یک خورده جلوتر می‌برم. ۶ هم همین طور ۴، ۷، ۵ . البته به شما بگویم که این ۹ روز که تمام شده بود این همه را باز چیدم که ۳۰ روز شده است به این کار ***Buffer*** دهی به فعالیت می‌گویند یه جوری که انگار یه ضربه گیر بهش دادم الان اگر ۳ عقب بیافتد دیگه خیلی نگران نباشد حالا یه روز عقب افتاده است . چرا ؟ چون من بین ۳ و ۸ ، ۳ روز ***gap*** گذاشته بودم که اگر احیاناً اتفاقی افتاد بتوانم ازش استفاده کنم یعنی از این تأخیر افتادن ۳ بقیه پروژه اصلاً آسیب نمی بیند. البته بگم تا یکی، دور روز نه خیلی زیاد. چون تا یکی، دو ، سه روز حداقل می شود این را جبران کرد و همین داستان بین *{۲,۶}* و *{۶,۴}* و *{۸,۷}* و *{۴,۵}* الی آخر این گپ ها وجود دارد. اگر من بخواهم مفهوم پایداری را بگویم الان این بالایی ناپایدار است ولی تا حدی نسب به ما پایدار است روی کاغذ کدام بهتر است؟ این خیلی بهتر است چون ۹ روز است ولی این یکی ۳۰ روز است. روی کاغذ اونی که ۹ روز است بهتر از ۳۰ روز است ولی عملاً ۳۰ روزه پایدارتر است . جوابی که روی کاغذ عالی است ولی موقع اجرا افتضاح می شود که فایده ندارد چون روی کاغذ فقط به درد پایان ترم، تئوری ، مقاله می خورد ولی وقتی ***run*** بکنی اجرایی بکنی باید یه مقدار با واقعیت نزدیک باشد.

نه می توانم بگم حداقل نه می توانم بگویم حداکثر. حدود ۳۰ روز است می تواند از ۳۰ روز کمتر هم بشود و یا ممکن است از ***gap*** ها استفاده نشود.

مثلاً شما ۲ روز ... گذاشته و ممکن است این ۳ روز یا ۴روز به تأخیر بیافتد و ممکن است از ۳۰ هم رد شود .

آیا این به مفهوم شناوری است ؟ نه مفهوم شناوری چیز دیگری است ***Buffer*** دهی شناوری نیست. حالا اسمش را می توانی شناوری بگذاری ولی دقیقاً مساوی آن نیست

من این موضوع را به یک زبان دیگه بگم



این یک تابع است مثلاً توی این تابع که شما اینجا می بینید کدام نقطه مینیمم تابع است؟ روی کاغذ ، روی شکل واضح است که این نقطه است این چی ؟ این هم مینیمم هست ولی این بهتر از اون است ولی کدام پایدارتر است؟ چرا اون پایدارتر است ؟ شما فرض کنید توی این پیش بینی های شما ۲۰ درصد ، ۱۰ درصد اشتباه بشود اگر این ۲۰ درصد اشتباه بشود و ۲۰ درصد جلوتر برود. کجا قرار می گیرید ؟ این نقطه . این الان ۳۰ روز بوده این میشه چند ؟ ۳۶ روز کمی بیشتر شد اما این یکی چی ؟ این روی کاغذ ۹ روز بوده اگر احیاناً ۲۰ درصد اشتباه پیش بینی کرده باشی یعنی ۲۰ درصد جلوتر بروی. نوعی را ترجیح می دهی که عملاً ۱۰۰ می شود بعداً یا نه ۳۰ ایی که ۳۶ روز است ما این ۳۰ ایی که پایدارتر است که نوسانش یکم بیشتر قبول داریم تا این ۹ ایی که به شدت لرزان است در معرض فرو ریختن است

به یک زبان دیگه همان را گفتم به یک زبان ریاضی تر گفتم ما در واقع به این می گوییم یک جواب پایدارتر و به این یکی می گوییم جواب ناپایدار. پس من تنها مفهوم ***Robust scheduling*** را گفتم والا به توضیح روش حل آن نپرداختم. یک راه بسیار ابتدای آن ***buffer*** دهی است ولی روش های بسیار متنوع دیگری هم دارد که کلاً فلسفه سختی دارد یعنی روش های حل بسیار سختی دارد که تنها به توضیح مفهوم آن اکتفا کردم و راه حلی برای این پیشنهاد نکردم.