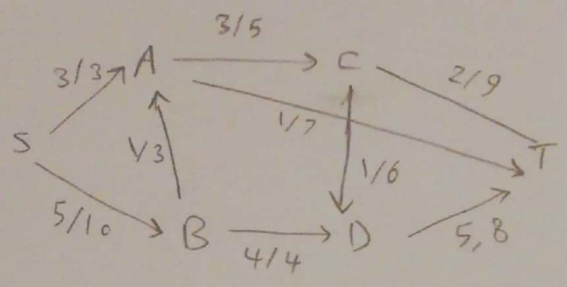
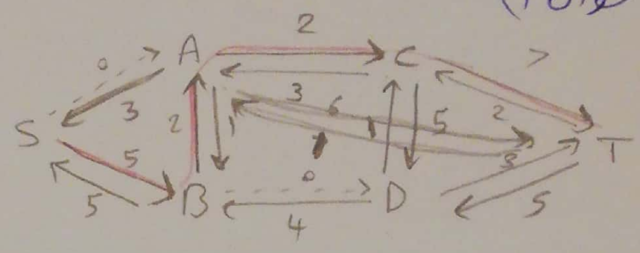
DA_HW5 network_flow ۸۱۰۱۰۱۴۹۲ علیرضا کرمی

سوال ۱)



network flow graph



Residual graph

الف) برای اینکه ببینیم که جریان feasible هست یا نه، باید چک کنیم که

۱) به ازای هر یال e ، $f(e) \le c(e)$ (جریان گذرنده از آن کمتر از ظرفیت آن باشد) ✓ به ازای تمام یال‌ها برقرار است

2) به ازای هر جفت u و v ، $f(u,v) = -f(v,u)$ ✓ طبق دو گراف بالا این شرط نیز برقرار است.

3) به ازای تمام راس‌های u به غیر از S و T داشتیم $\sum_{u \to e} f(e) = \sum_{u \leftarrow e} f(e)$ ، یعنی داشته باشیم که مجموع جریان ورودی به آن راس برابر با مجموع جریان خروجی از آن راس باشد. ✓

	جریان ورودی به آن	جریان خروجی	
A	3+1=4	3+1=4	✓
B	5	1+4=5	✓
C	3	2+1=3	✓
D	1+4=5	5	✓

و باید طبیعتاً داشته باشیم که جریان خروجی از S برابر جریان ورودی به T باشد که این هم برقرار است

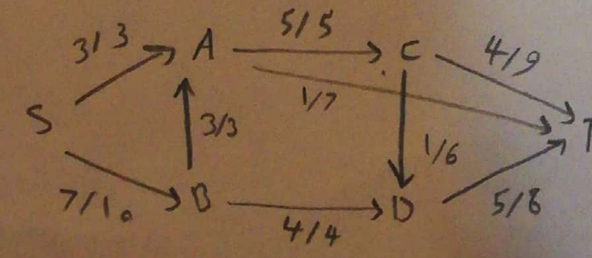
جریان خروجی از S : 3+5=8 جریان ورودی به T : 2+1+5=8

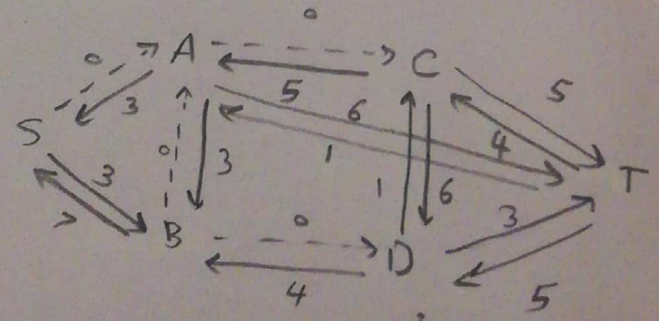
ب) خیر، جریان بیشینه نیست. چون همانطور که در residual graph می‌بینید، هنوز مسیری از S به T (با رنگ قرمز) وجود دارد.

ج) همانطور که مسیر را با رنگ قرمز مشخص کردم، ابتدا bottleneck مسیر را حساب کرده و به این میزان flow به آن مسیر اضافه می‌کنیم.

min (5,2,2,7) = 2 ⟶ مقدار flow که می‌توان در این مسیر اضافه کرد

$\Rightarrow$

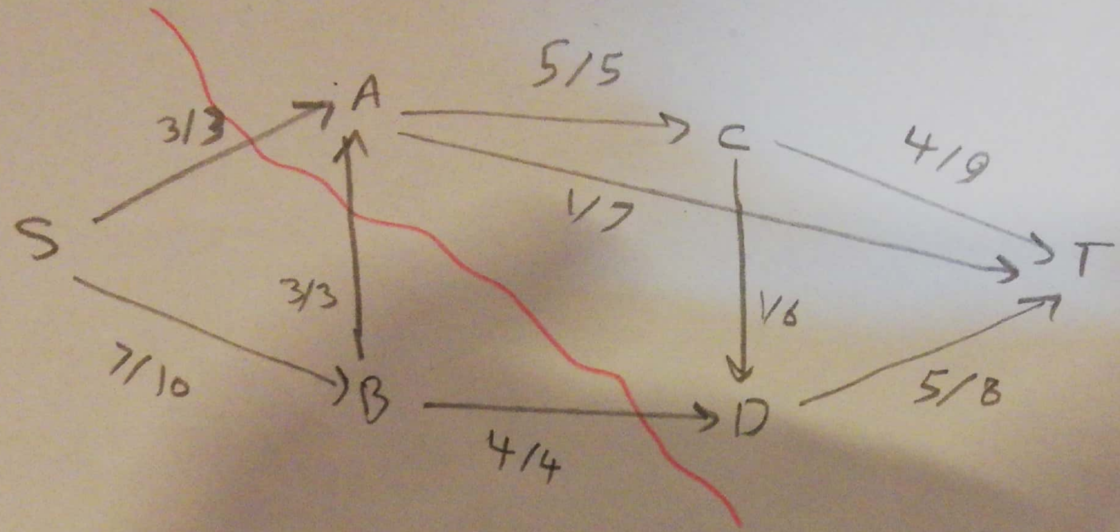




همانطور که می‌بینید، دیگر هیچ مسیری از S به T در residual graph نداریم و flow ما بیشینه و برابر ۱۰ شده است.

ادامه سوال ۱)

د)



$$\Longrightarrow c(S,T) = \sum_{u\in S,\, v\in T} c[u,v] = 3+3+4 = 10$$

$$\Longrightarrow f(S,T) = \sum_{u\in S,\, v\in T} f[u,v] - \sum_{u\in S,\, v\in T} f[v,u] = 3+3+4 = 10$$

$$\Longrightarrow c(S,T) = f(S,T)$$

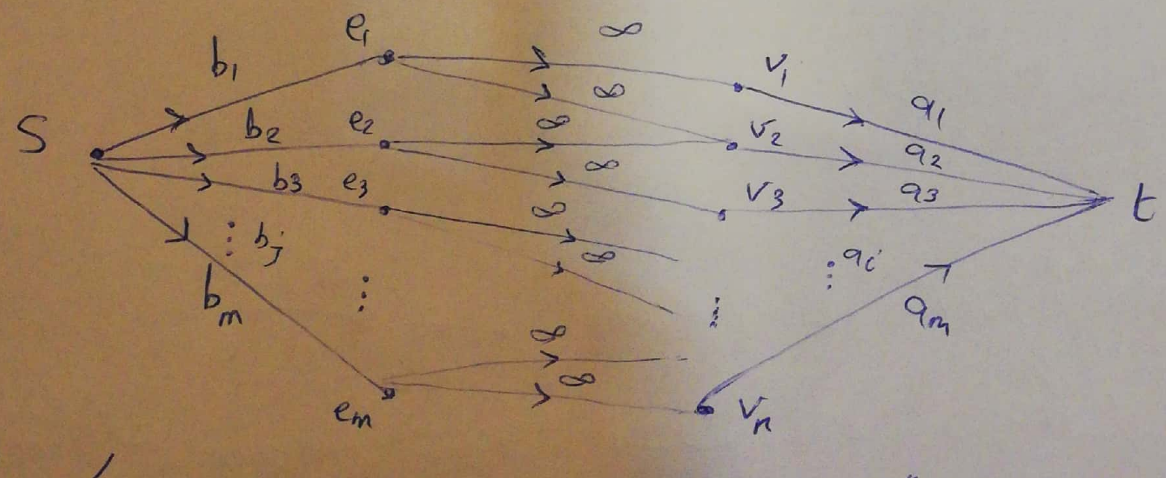
۲) طبق راهنمایی‌ها سر کلاس حل تمرین، برای حل این سوال باید دنبال mincut باشیم، یعنی باید دنبال یک cut با minimum capacity باشیم. حال در مسئله ما، ما دنبال minimum کردن چه چیزی هستیم؟ دنبال این هستیم که تا جایی که می‌توانیم، ارزش یال‌ها یا همان cost راس‌هایی که استفاده می‌کنیم را کم کنیم و از طرفی ارزش یال‌هایی که استفاده می‌کنیم را زیاد کنیم (و یا به عبارتی، ارزش یال‌هایی که استفاده نمی‌کنیم را کم کنیم) تا به این صورت، با کاهش $\sum a_i$ های استفاده شده و افزایش $\sum b_i$ های استفاده شده، مقدار $\sum b_i - \sum a_i$ را به maximum مقدار ممکن برسانیم. حال از این رو، یک گراف با این مشخصات درست می‌کنیم.

الف) ابتدا به ازای تک تک یال‌های e، یک node می‌گذاریم و یک node برای source هم درست می‌کنیم و از آن source به تمام e_i ها، یال می‌زنیم با ظرفیت b_i

ب) به ازای تک تک راس‌های v در گراف، از همه آنها به node مربوط به sink، یال می‌زنیم، با ظرفیت a_i

ج) از تمام یال‌های e به دو سر آن (راس‌های v_1 و v_2) یال می‌زنیم با ظرفیت بی‌نهایت (این کار را می‌کنیم که min cut ما هیچ وقت آن یال‌ها را کات نکند، تا هر یال با راس‌های متصل به آن حتما در یک سمت کات قرار بگیرند. بعدا مفصل توضیح می‌دهم)

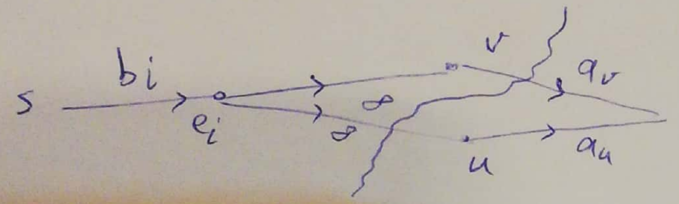
در نهایت گراف جدید ما چنین شکلی خواهد داشت: (هزینه ساخت این گراف از اردر چندجمله‌ای $O(n+m)$) تعداد راس‌ها، تعداد یال‌ها



حال به این می‌پردازیم که کات ما، چگونه باید باشد. اولا کات ما نباید از یال‌هایی با ظرفیت ∞ عبور کند، چرا که با این کار، مقدار min cut ما (منظور capacity خروجی از آن) ∞ خواهد شد در صورتی که این قطعا غلط است، چون می‌توانستیم تمام یال‌های با ظرفیت b_i را در cut بگذاریم و اینگونه هزینه mincut ما

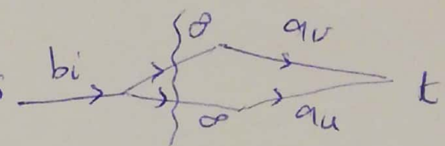
(ادامه سوال ۴)

برابر با $\sum b_i$ خواهد بود که قطعا از ∞ کمتر است. پس تناقض است. پس min cut ما یا از یال‌های با وزن b_i می‌گذرد یا از یال‌های با وزن a_i می‌گذرد. از طرفی یک نکته جالب هم وجود دارد و آن اینکه اگر دو راس در سمت چپ cut بودند، حتما یال مربوط به آن نیز در سمت چپ cut باشه یعنی اینطوری نباشد که یک یال و یک راس متصل به آن در سمت چپ cut باشه اما راس دیگر متصل به یال در سمت راست cut باشه. اگر یال سمت چپ cut بود و حداقل یکی از راس‌های متصل به آن سمت راست cut باشه قطعا cut ما از یال‌های میانی عبور کرده (که capacity آنها ∞ بود) پس به تناقض خواهیم رسید و این امکان پذیر نیست (دلیل گذاشتن capacity برابر با ∞ نیز دقیقا همین بود که بین یال و دو راس متصل به آن هیچ وجه کات نخورد) مثلا حالت‌های زیر غیر قابل قبول هستند:



غیر قابل قبول

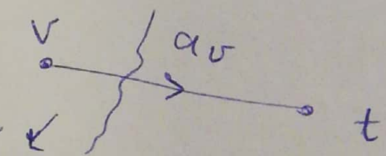
یا



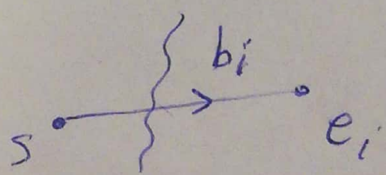
غیر قابل قبول

(چون min cut، ∞ می‌شه در هر دو حالت)

حال که روشن شد چرا نمی‌توانیم mincut را از یال‌های میانی عبور دهیم، یا mincut از a_iها عبور می‌کند (که اینگونه آن راس در سمت چپ کات می‌افتد و یعنی آن راس را انتخاب کردیم) و یا mincut از b_iها عبور می‌کند (که یعنی راس مربوط به آن یال در سمت راست کات می‌افتد و یعنی آن یال را انتخاب کردیم.



یعنی راس v انتخاب شده و ظرفیت a_v خروجی داریم



یعنی یال e_i انتخاب نشده و باز ظرفیت e_i خروجی خواهیم داشت.

همانطور که گفتیم دنبال این هستیم که ارزش یال‌های انتخاب نشده (b_iهای cut خورده) را کمینه کرده و ارزش راس‌های انتخاب شده (یعنی a_iهای کات خورده) را نیز کمینه کنیم تا $\sum b_i - \sum a_i$ های مجموعه انتخاب شده‌مان، max شود. پس باید دنبال mincut در این گراف باشیم و این دقیقا معادل پیدا کردن mincut در این گراف است.

ادامه سوال ۲) از طرفی می‌دانیم که پیدا کردن min cut، برابر با پیدا کردن max flow در گراف است.
پس کافی است در گراف ساخته شده با کمک الگوریتم Edmonds Karp که در زمان چندجمله‌ای E^2V اجرا می‌شود،
max flow را پیدا کنیم که برابر همان میزان min cut ما خواهد بود. حال برای پیدا کردن $\sum b_i - \sum a_i$ در
مجموعه انتخاب شده‌مان، کافی است که مجموع کل b_i ها را منهای b_i های انتخاب نشده کنیم و البته منهای a_i های
انتخاب شده کنیم که یعنی مجموع کل b_i ها را منهای مقدار min cut (یا همان max flow) بکنیم. در کلاس
گفتیم که min cut با همان max flow برابر است. پس طبق توضیحات ارائه شده داشتیم:

$$\sum b_i - \sum a_i \text{ پیدا کردن ماکسیمم مقدار} \longleftrightarrow \text{min cut}$$

خلاصه این اثبات

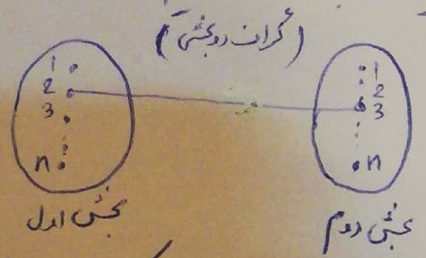
الف) اگر min cut داشته باشیم، از یال‌های ∞ که نمی‌گذرد، پس از a_i ها یا b_i ها می‌گذرد. این min cut ما قطعا
b_i های نداشته را کمینه کرده (با عبور از آنها) و از طرفی ارزش a_i های داشته را نیز کمینه کرده است. پس قطعا ارزش
مقدار $\sum b_i - \sum a_i$ مجموعه انتخابی ما بیشینه خواهد بود.

ب) در صورتی که مقدار ارزش $\sum b_i - \sum a_i$ مجموعه انتخابی ما بیشینه باشد، یعنی یال‌های نداشته و ارزش راس‌های
داشته را کمینه کردیم که این دقیقا معادل پیدا کردن min cut در گراف ارائه شده است.

هزینه ساخت گراف $O(V+E)$ و هزینه اجرای الگوریتم Edmonds Karp برای پیدا کردن max flow $O(E^2V)$ هر دو از ارد
چندجمله‌ای هستند.

۳) می‌دانیم درصورتی که بتوانیم راس‌ها را به تعدادی دور افراز کنیم، یعنی هر راس در یک دور وجود دارد و این یعنی هر راس تنها یک یال خروجی و یک یال ورودی دارد. از طرفی می‌دانیم درصورتی که به ازای هر راس، تنها یک یال خروجی و یک یال ورودی داشته باشیم، یعنی n راس و n یال داریم که تمامی راس‌ها در دور شرکت دارند (اگر از راس v_1 یک مسیری را شروع کنیم، انتهای این مسیر حتما باید به v_1 بازگردد، چون حتما باید یک یال ورودی به آن هم داشته باشیم). این ایده که به ازای هر راس تنها یک یال ورودی داریم و به ازای هر راس تنها یک یال خروجی داریم، ما را به ایده‌ی matching می‌رساند. پس می‌آییم و مسئله را به این شکل، به مسئله matching تبدیل می‌کنیم:

یک گراف دوبخشی در نظر می‌گیریم. در هر دو بخش، تمام رئوس را قرار می‌دهیم. حال به ازای تمام یال‌های جهت‌دار $e\ (u_1, v_1)$، یعنی از u_1 به v_1 یال داریم، می‌آییم از u_1 در بخش اول به v_1 در بخش دوم، یال جهت‌دار می‌کشیم.



فرض کنید به ازای یال از راس 2 به راس 3 خواهیم داشت

به ازای تک‌تک یال‌های در گراف اصلی این کار را می‌کنیم. درصورتی که بتوانیم یک maximum bipartite matching از سایز n پیدا کنیم، یعنی تک‌تک راس‌های ما در matching شرکت کردند، به طوری که به ازای هر راس، الان هم یک یال خروجی داریم و هم یک یال ورودی داریم، پس می‌توان نتیجه گرفت که دور خواهیم داشت. یعنی ادعا کردیم اگر و تنها اگر یک matching از سایز n داشته باشیم، راس‌های گراف را می‌توان به دورهایی افراز کرد. حال به اثبات ادعای‌مان می‌پردازیم.

درصورتی که یک matching از سایز n داشته باشیم، تک‌تک یال‌های matching را از گراف اصلی انتخاب می‌کنیم و بقیه را انتخاب نمی‌کنیم (به این ترتیب گراف جدیدی تنها با یال‌های استفاده‌شده در matching ساخته شده که زیرمجموعه گراف اصلی است). می‌دانیم با توجه به matching با سایز n در گراف دوبخشی، به ازای تک‌تک راس‌ها، فقط یک یال ورودی و به ازای تک‌تک راس‌ها فقط یک یال خروجی داریم. از آنجایی که به ازای تک‌تک رئوس فقط یک یال ورودی و یک یال خروجی داریم، پس هر راس تنها در یک دور شرکت کرده است و توانستیم راس‌ها را در تعدادی دور افراز کنیم. و این دورها با هم در هیچ راسی اشتراک ندارند.

ادامه حل سوال در صفحه‌ی بعد

ادامه سوال ۳)

حال در صورتی که بتوانیم رأس‌های گراف را به تعدادی دور (یال‌ها و ... که با هم اشتراکی نداشته باشند) افراز کرده باشیم، به طوری که هر راس تنها در یک دور قرار گرفته باشد، به هر راس تنها یک یال وارد شده و تنها یک یال خارج شده. (یعنی ما آمدیم از روی گراف اولیه، یال‌ها را در گراف جدید به گونه‌ای انتخاب کردیم که به تعدادی دور تقسیم شده و هر راس تنها در یک دور است. حال در این گراف جدید میاییم و matching متناظر آن را می‌سازیم.) یعنی به ازای هر یال در گراف جدید که از u به v است، یک یال از u در بخش اول به راس v در بخش دوم در گراف دوبخشی‌مان وصل می‌کنیم. چون تمام راس‌ها تنها یک ورودی و یک خروجی دارند، با انتخاب تمام یال‌های بین دو بخش، به matching با سایز n می‌رسیم.

پس داشتیم: matching با سایز n $\Longleftrightarrow$ افراز راس‌های گراف به تعدادی دور

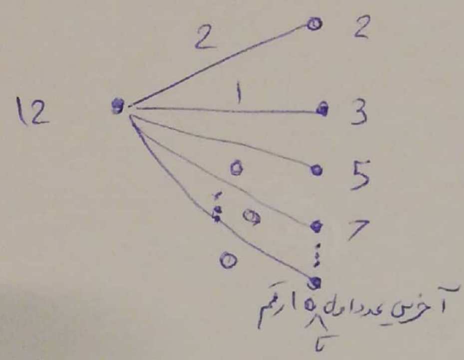
خود matching را هم که در slide های درس می‌دانیم می‌شود با max flow مدل‌سازی کرد. پس مسئله حل شد.

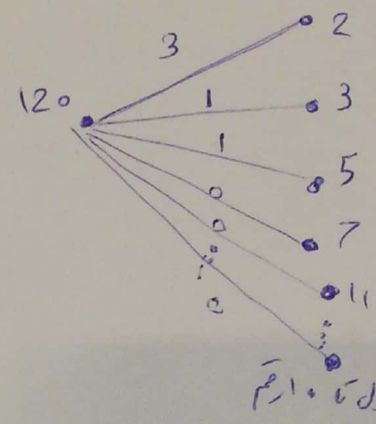
(راهش این بود به source با یال با capacity ۱ به تمام راس‌های بخش اول یال می‌زدیم. یال‌های بین دو بخش را هم capacity ۱ می‌گذاشتیم. یک راس sink هم داشتیم و از تمام راس‌های بخش دوم گراف به آن با capacity برابر ۱، یال می‌زدیم. اگر flow خارج شده از source، برابر n بود، یعنی یک matching از سایز n داریم.)

برای پیدا کردن max flow بعد از تبدیل کردن به آن matching (که اردر چندجمله‌ای است. البته به یاد داریم تبدیل خود گراف به matching هم از اردر چندجمله‌ای بود یعنی $O(V+E)$) از الگوریتم Edmonds karp استفاده می‌کنیم که از اردر VE^2 و چندجمله‌ای نسبت به ورودی است ($O(nm^2)$)

۴) به ازای هر کدام از این n عدد، ابتدا می آییم و آنها را به فاکتورهای اولیه اش تقسیم می کنیم. برای مدل سازی آن با گراف، به این شکل عمل می کنیم. به ازای هر کدام از n عدد، به ازای تمام فاکتورهای عدد اول تا ۱۰ رقم را جلوی آن راس قرار می دهیم

خود عدد هم با یک راس نمایش می دهیم. حال از این راس به یک یک راس ها که نماینده عددهای اول بودند، یال وصل می کنیم و ظرفیت آن را برابر با توان مشارکت آن عدد اول در ساخته شدن عددمان قرار می دهیم. مثلا اگر عدد ما 120 یا 12 باشد داریم:

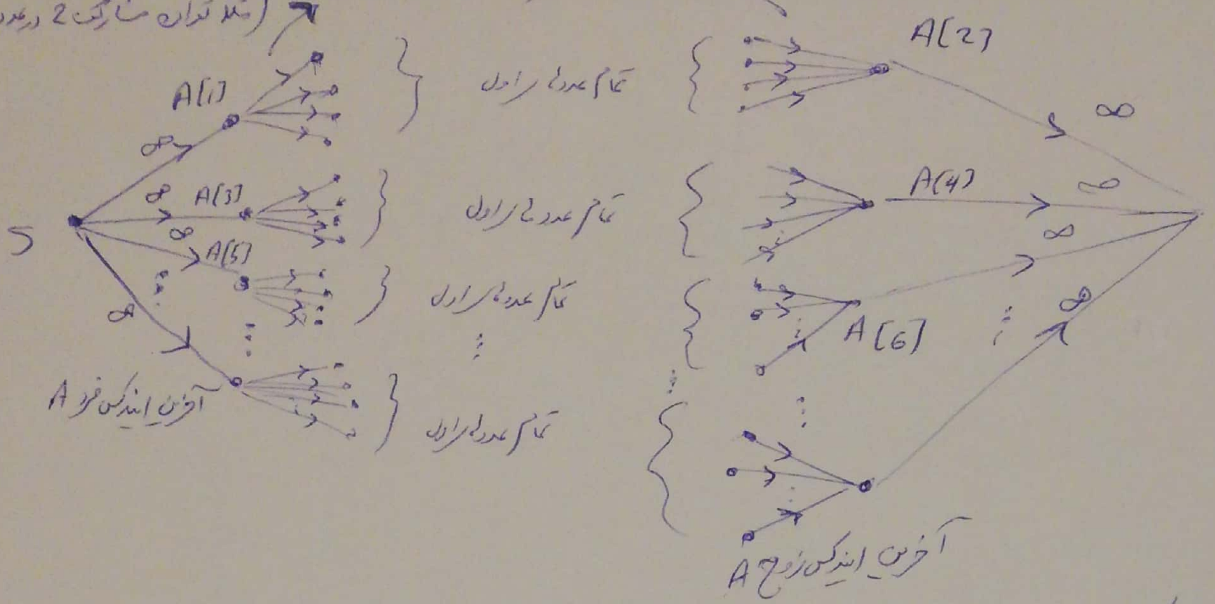




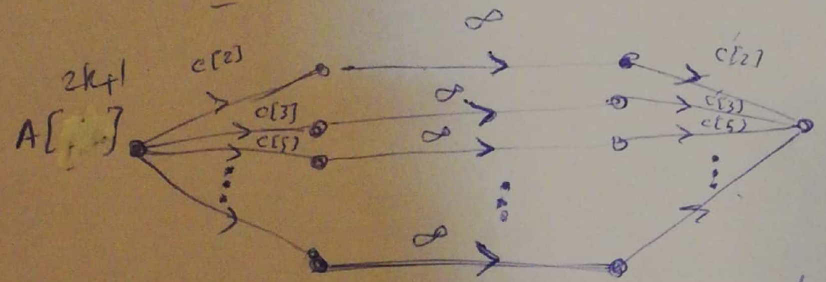
حال به ازای یک یک عددهای با اندیس فرد و یک یک عددهای با اندیس زوج باید این کار را بکنیم. حال به این صورت عمل می کنیم. تمام عددهای با اندیس زوج را در یک دسته قرار می دهیم و تمام عددهای با اندیس فرد را در یک دسته دیگری قرار می دهیم (منظور این است که به ازای آنها راسی می گذاریم که نماینده آن عدد است). حال یک راس source می گذاریم و از آن به تمام راس های عددهای دسته با اندیس فرد، یال جهت دار می زنیم (جهتش از S به آنها) و capacity آن یال ها را ∞ می گذاریم. حال به ازای یک یک راس های عددهای با اندیس فرد (دقت کنید گفتم به ازای یک یکشان، و نه همه با هم) می آییم راس های عددهای اول تا ۱۰ رقم را می گذاریم و از آن راس ها به راس های عدد اول مربوط به خودشان یال می زنیم با capacity به اندازه توان مشارکت آن عدد اول در ساخت این عدد با اندیس فرد. جهت یال از راس با اندیس فرد به راس های نماینده عدد اول است. حال شبیه اینکار را برای عددهای راس دسته با اندیس زوج انجام می دهیم، با این تفاوت ها که جهت یال ها از عددهای اول به آن عدد با اندیس زوج است و از عددهای با اندیس زوج به sink، یال جهت دار می زنیم

ادامه سوال ۴) یعنی تا اینجا خواهیم داشت

capacity > این یال‌ها توان مشترک آن عدد اول است
(مثلا توان مشترک 2 در عدد ۸ برابر 3 است)

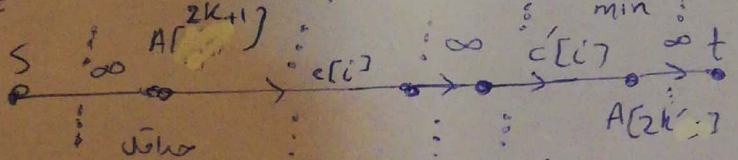


حال به ازای تک تک جفت‌های m ، می‌دانیم که هر کدام لذا آنها یک مجموعه عددهای اول دارند، می‌آییم هر کدام از راس‌های عدد اول از عدد با اندیس فرد را به راس متناظر از آن دسته عددهای اول از عدد با اندیس زوج وصل می‌کنیم از جهت یال که هم از راس عدد اول عدد اندیس فرد به راس‌ای عدد اول اندیس زوج و با ظرفیت ∞ . یعنی الان:



مثلا برای جفت i ام از m جفت داریم:

یک نکته خیلی مهم اینکه هر دفعه باید بر عدد اول تقسیم کنیم و نه ب.م.م آنها (یا حاصل ضرب چند عدد اول مشترک) برای اینکه محاسبات (مراحل تقسیم) را به بیشترین حد بگذاریم. پس بقیه اونهایی که در m جفت گفته شده می‌تونیم یال نگذاریم یا یال با ظرفیت 0 بگذاریم. اگر دقت کنید، درصورتی که این جفت i ام از m جفت را بر عددی مشترک تقسیم کنیم، به تعداد توان مشترک از هر کدام از عددهای اول می‌توان این تعداد تقسیم داشته باشیم چون مثلا داریم (برای جفت i)



پس bottleneck در این مسیر برابر است با $\min(c[i], c'[i])$ ، یعنی هر دو به این تعداد از آن عدد اول، توان دارند، پس به این تعداد تقسیم می‌تونه صورت بگیره. مثلا برای ۱۲ و ۸ ، عدد ۱۲، ۲ تا از ۲ داره و عدد ۸ ، ۳ تا از ۲ داره. (اگه به فاکتورهای اول تقسیم کنیم، این توان‌ها مشترکشون هست) (چون $12 = 2^2 \times 3$ و $8 = 2^3$) و min اینها می‌شه ۲ ، یعنی حداقل ۲ تا ۲ مشترک دارن، پس دو بار حداکثر می‌تونیم تقسیم کنیم، به اندازه flow برابر ۲ اضافه کنیم

ادامه سوال ۴) به این شکل، به اندازه $\min(c(i), c'(i))$ از $c(i)$ و $c'(i)$ باقی مانده کم می‌کنیم، چون به این اندازه flow اضافه کردیم.

خلاصه راه‌حل: به ازای هر عدد، دسته عددهای اولش را گذاشتیم و به اندازه توان آن که در عدد، ظرفیت یال گذاشتیم. بعد از اینکه یک جفت را تقسیم بر عددهای مشترک کردیم، توان مشترک آن عدد(های) اول از هر دو کم می‌شود و در تقسیم‌های بعد انگار با عدد جدیدی مواجه هستیم. حال اگر بیاییم به کمک edmonds karp (VE^2)، max flow را در اردر چند جمله‌ای پیدا کنیم، معادل آن است که حداکثر تعداد تقسیم‌های ممکن را پیدا کنیم. می‌دانیم که تعداد عددهای اول تا ۱۰ رقم، عددی ثابت است پس در اردر تاثیر نمی‌گذارد. هزینه ساخت گراف جدید ساخته شده هم از اردر $O(A(n+m))$ است که چندجمله‌ای است.

تعداد اعداد اول که مقدار ثابت است

با ظرفیت بی‌نهایت

می‌دانیم که هر مسیری که وجود دارد در network flow graph آن گراف جدید، از s به $A[2k+1]$ از آن به فاکتور عدد اولش با ظرفیت توان مشترک آن، از آن به فاکتور عدد اول $A[2k]$ با بی‌نهایت، از آن به خود عدد $A[2k]$ با توان مشترک آن عدد اول و از آن به t با بی‌نهایت وصل است. پس اگر augmenting path باقی مانده باشد و داشته باشیم، در این مسیر، به اندازه $\min(c(i), c'(i))$ flow داشتیم که می‌شه همون bottleneck آن augmenting path. پس آن قطع می‌شود و اگر هیچ مسیر افزایشی از s به t نداریم پس max flow داریم، که می‌شه همون حداکثر تعداد تقسیم. پس داریم

یعنی بیشتر نمی‌شه تقسیم کرد

max flow در گراف جدید $\Longleftrightarrow$ حداکثر تعداد تقسیم ممکن

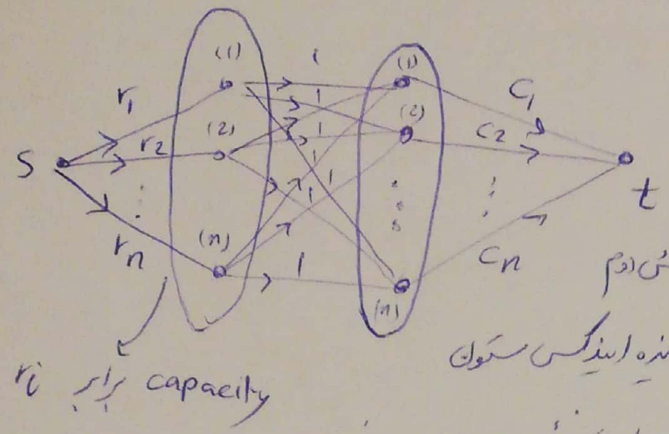
اگر هم حداکثر تقسیم را کرده باشیم پس توان مشترکی مشترک بین جفت‌ها نداریم، پس در گرافی که مدل‌سازی کردیم augmenting path باقی نمونده پس max flow داریم.

از اون سمتی که گفتم

خلاصه: پس اگر حداکثر تقسیم داشته باشیم، تقسیم باقی نمی‌مونه ← augmenting path نداریم ← max flow داریم

اگر max flow داشته باشیم و augmenting path نداشته باشیم ← تقسیمی باقی نمی‌مونه و حداکثر تقسیم رو داشتیم.

هزینه ساخت گراف از اردر $O(A(n+m)) = O(n+m)$ است (همان $O(V+E)$) و هزینه اجرای الگوریتم Edmonds Karp برای پیدا کردن max flow هم از اردر $O(E^2V)$ است و هزینه آن در کل از اردر چندجمله‌ای خواهد بود.



سوال 5) سوال را اینگونه مدل سازی کنیم.

یک گراف دوبخشی در نظر بگیرید که بخش اول n راس داریم که هر کدام نماینده ایندکس سطر یک خانه هستند و در بخش دوم یعنی بخش راستی، n تا راس داریم که هر کدام نماینده ایندکس ستون هستند. حال یک source داریم و از آن به تک تک راس های بخش 1 یال می‌زنیم با capacity برابر با (r_i).
یک sink هم داریم که از تمام راس های بخش دوم به آن، با ظرفیت c_i، یال جهت دار می‌زنیم.
به ازای تک تک راس های بخش اول به تک تک راس های بخش دوم هم یال می‌زنیم با ظرفیت 1، هر کدام از این یال ها نماینده خانه سطر iام و ستون jام هستند. یعنی از راس iام در بخش اول به راس jام در بخش دوم یال داریم که نماینده خانه $A[i, j]$ است. ظرفیت را 1 گذاشتیم که نشان می‌دهد آیا از آن خانه شکلات برداریم یا نه. اگر برداریم، flow برابر 1 از آن می‌گذرد و از r_i و c_j مربوط به آن هم یدونه کم می‌شه. در غیر اینصورت هیچ. از اون جایی که به ازای هر راس در بخش اول، $n \geq r_i$ که r_i ظرفیت ورودی و n ظرفیت خروجی است. پس جا داریم تا r_i تا از آن راس flow رد کنیم. همینطور چون $c_j \leq n$ پس می‌توانیم حتما به اندازه c_j تا flow از آن راس های بخش دوم عبور دهیم. یعنی درصورتی که max flow را در گراف جدید پیدا کنیم، این توانایی را داریم که تمام یال های خروجی از s و تمام یال های ورودی به t را saturate کنیم و در صورتی که همه این یال ها saturate شده باشند، یعنی برای هر ردیف به همان میزان خانه و برای هر ستون به همان میزان خانه لازم انتخاب کردیم و نه کمتر. پس این جریان ممکن است (چون $r_i, c_i \leq n$) و می‌توان از باگ سیستم استفاده کرد.

یادم رفت بگم. اگر $\sum r_i \neq \sum c_i$، اصلا نمیشه باگ سیستم رو برطرف کرد و از آن استفاده کرد.

اگر مساوی بودند، چون یال های بین دو بخش n^2 تا است و از source به بخش اول، حداکثر ظرفیت $\sum r_i$ که کمتر مساوی n^2 است، و همینطور $\sum c_i$ ها هم کمتر مساوی n^2 است، پس می‌توان با یافتن max flow، همه r_i و c_i ها را saturate کرد. و این موضوع باگ satisfy می‌شود.

ادامه سوال ۵)

با مقدار $-a_i$

حال باید برای هر یال از اندیس ردیف i به اندیس ستون j، یک cost تعیین کنیم. و بین max flow های مختلف باید max flow ای را انتخاب کنیم که max_cost را داشته باشد.

البته می‌توانیم cost آن یال‌ها را $-a_i$ بگذاریم و مسئله min cost network flow یا همان mcnf را حل کنیم. یعنی max flow ای با کمترین cost. که این مسئله در زمان چندجمله‌ای حل شد.

(کلا برای روش‌های چندجمله‌ای در network flow از همان edmonds karp با اردر E^2V استفاده کردیم و این مسئله mcnf هم در زمان چندجمله‌ای قابل پیدا کردن است به کمک دایکسترا و پیدا کردن کم هزینه‌ترین مسیر ممکن و اضافه کردن آن به گراف‌مان residual graph که در کلاس حل تمرین گفتن اثباتش زیرشو نمی‌خواد ولی خب مشخصه طبق دایکسترا، داریم هر دفعه بهترین انتخاب رو و کم هزینه‌ترین مسیر ممکن رو اضافه می‌کنیم. پس در کل کم هزینه‌ترین max flow را میده که چون منفی کرده بودیم، برابر همون بیشترین شکلات خورده شده است)

از اونجایی که توضیح دادم، در صورتی که max flow داشته باشیم، تمام r_i ها و c_i ها saturate شده و می‌توانیم از باگ استفاده کنیم. از طرفی اگر بخواهیم که از باگ استفاده کنیم، باید حتما برای هر راس و هر ستون به همان تعداد خانه استفاده کرده باشیم، یعنی در گراف جدید باید این یال‌های r_i و c_i saturate شوند پس قطعا max flow ما این حالت خواهد شد و max flow بیشتر از این نداریم چون دیگه saturate شده و augmenting path نداریم دیگه. (max flow $\Longleftrightarrow$ باگ)

از طرفی هم مسیری که پیدا کردن min cost (چون cost‌ها را $-a_i$ بود) معادل همان پیدا کردن max مقدار شکلات است.

تعداد یال‌های گراف V^2+2V و تعداد راس‌های آن $2V+2$ بود. پس هزینه ساخت گراف از اردر $O(V+E)$ بود.

از طرفی می‌توانیم الگوریتم min cost max flow را به کمک الگوریتم دایکسترا (البته چون اینجا یال منفی داریم می‌توانیم بگیم بلمن فورد) اجرا کنیم حداکثر به تعداد n^2 بار (اگه $\sum r_i = n^2$ شد که حداکثر هم همین مقدار است چون داشتیم $\sum r_i \le n^2$)

پس در کل در اردر چندجمله‌ای، این min cost network flow پیدا شد (بلمن فورد هم چندجمله‌ای بود). توجه کنید که bellman ford را هر دفعه در residual graph می‌زنیم چون به یال backward هم نیاز داریم