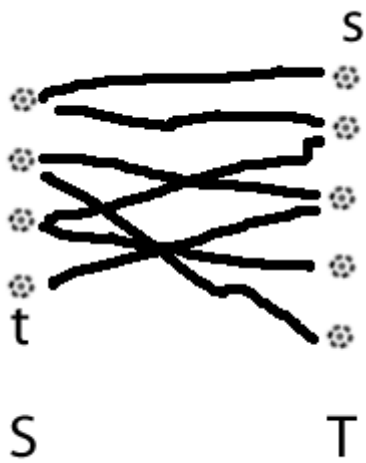
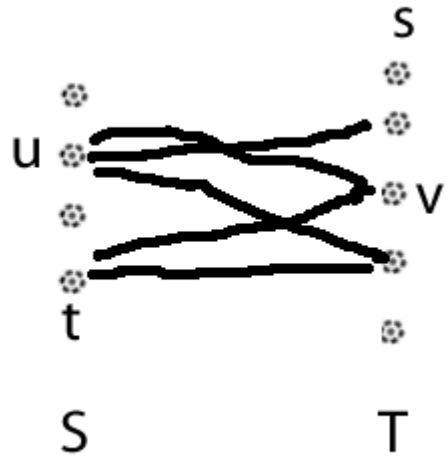
به نام خداوند بخشایشگر مهربان

تمرین دوم الگوریتم پیشرفته محمد تنهایی ۸۷۲۰۲۴۴۶ نرم افزار

۱. الف) چون که فرض کرده ایم که یال ها تنها بین دو مجموعه وجود دارند ، گرافی به شکل ۱ داریم ، در این گراف بین رئوس داخل S یالی وجود ندارد ، برای رئوس T نیز همین طور است . حال اثبات می کنیم که حداکثر طول مسیر از $O(\sqrt{n})$ است . طولانی ترین مسیر زمانی به وجود می آید که هر بار راسی را ملاقات کنیم که قبلاً به آن راس نرفته ایم . زیرا اگر از راسی مانند v به راس u برویم که قبلاً ملاقات شده است . یک حلقه به وجود می آید (در شکل ۲ مشخص است) و می دانیم که می توانیم یک حلقه را از یک مسیر حذف کنیم (چون که بعد از طی حلقه به یک راس که قبلاً در مسیر بوده باز می گردیم) . بنابراین طول مسیر حداکثر می تواند به اندازه تعداد رئوسی باشد که می توانیم بدون تکرار در مسیر بازدید کنیم . این تعداد برابر $\min(\sqrt{n}, n-\sqrt{n}-2)$ است که برابر $\sqrt{n}$ می شود.



شکل ۱ : گراف مورد نظر ، با معذرت به خاطر شکل بد ترسیم شده!!!



شکل ۲ : یک مسیر که یک راس بیش از یک بار در آن مشاهده شده است ، می توان کل دوری که به وجود می آید را بدون تاثیر در مسیر حذف کرد.

ب) اگر یک مسیر از u به v داشته باشیم که شامل : $<v_0,\dots,v_k>$ آنگاه طبق اثبات الف حداکثر مقدار k برابر $\sqrt{n}$ خواهد بود .

(طبق نتیجه تمرین 26.4-7) زمانی که h(u)<|V| باشد آنگاه $h[u] \le \delta(u,t)$ به عبارت بهتر $h[u] < \sqrt{n}$ و زمانی که h(u)>=|V|=n باشد $h[u] - n < \sqrt{n}$ اگر تعداد relabel های قابل انجام را بخواهیم محاسبه کنیم باید تعداد relabel ها در زمان h(u)<|V| و h(u)>=|V| محاسبه کنیم : در زمان h(u)<|V| حداکثر $\frac{\sqrt{n}}{2}$ relabel انجام می شود ، زیرا $h[u] < \sqrt{n}$. تعداد relabel ها در زمان h(u)>=|V| نیز برابر $\frac{\sqrt{n}}{2}$ است ، زیرا $h[u] - n < \sqrt{n}$ و می دانیم که h(u)>=|V| بنابراین h[u] حداکثر $\sqrt{n}$ واحد زیاد می شود. بنابراین تعداد relabel ها برابر است با $O(\sqrt{n})$

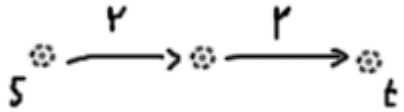
ج) از اثبات قبلی متوجه شدیم که حداکثر تغییر ارتفاع هر راس برابر با $2\sqrt{n}$ است .بنا بر این تعداد sp ها برابر است با $2\sqrt{n}$ زیرا طبق لم اثبات شده در کلاس با هر sp ارتفاع هر راس مربوطه ۲ واحد زیاد می شود در نتیجه یک یال e(u,v) می تواند $\sqrt{n}$ بار از راس u و $\sqrt{n}$ بار از راس v به صورت sp ارضا شود.تعداد کل sp ها برابر است با $\sqrt{n} * E$ است.

تعداد nsp ها نیز از روی تعداد sp ها و تعداد relabel ها قابل محاسبه است . اگر طبق روش داخل کتاب یک تابع ϕ تعریف کنیم که برابر مجموع ارتفاع رئوس دارای شار اضافی گراف متمم باشد . relabel مقدار ϕ را به اندازه $n + \sqrt{n}$

افزایش می دهد ، همچنین sp بر روی یالها مقدار ϕ را به اندازه $n+\sqrt{n}$ تغییر می دهد . بنابراین تعداد کل nsp ها برابر است با ϕ: $(n+\sqrt{n})(\sqrt{n}*E+n\sqrt{n})$

---

۲. الف) خیر هر شبکه شاره یک یال حساس افزایشی ندارد ، به عنوان نمونه شبکه ی زیر دارای یال حساس افزایشی نیست



مثال از گراف بدون یال حساس افزایشی

یک شبکه شاره ، اگر یک یال حساس افزایشی داشته باشد ، می توانیم یک Min-cut به دست آوریم ، هر یالی که به min-cut متعلق باشد یک یال حساس افزایشی است . بدیهی است که این یال باید ارضا شود (یعنی به اندازه وزن آن از آن شار عبور کند).

الگوریتم خود را به این صورت طراحی می کنیم ، فرض کنیم که X* یک شار بیشینه در گراف G(x) باشد ، حال دو مجموعه T و S را به شیوه زیر تعریف می کنیم : $S=\{u|u:s\to u\}$ و $T=\{u|u:u\to t\}$ این ۲ مجموعه را می توان به سادگی با الگوریتمی مانند BFS به دست آورد ، حال یال های حساس افزایشی ما یالهایی هستند که $E(v,u) is\ critical\ where\ v\in S\ and\ u\in T$. پیچیدگی این الگوریتم برابر است با محاسبه یک شاره بیشینه + پیچیدگی BFS که برابر O(VE) است ، پس پیچیدگی این الگوریتم در همان حد الگوریتم شاره بیشنه است.

ب) فرض کنیم که X* شار بیشینه در گراف G باشد . در چه صورت کم شدن وزن یک یال شار ماکزیمم را کاهش نمی دهد؟ تنها در صورتی که برای یالی مانند E(i,j) یک مسیر افزایشی از i به j وجود داشته باشد . در این صورت اگر ما مقدار E(i,j) را کم کنیم ، شار از طریق مسیر افزایشی جریان پیدا می کند و لزوماً شار ماکزیمم کاهش پیدا نمی کند ، بنا بر این الگوریتمی می نویسیم که این مسیر ها را شناسایی کند . حال اگر برای یک i,j یک مسیر افزایشی وجود نداشته باشد می توان ادعا کرد که یک یال حساس کاهشی پیدا کرده ایم . برای پیدا کردن کلیه مسیر ها از رئوس مختلف شروع کرده و از الگوریتمی مانند BFS استفاده می کنیم . پیچیدگی BFS برابر O(E) است و چون V راس داریم در کل پیچیدگی الگوریتم O(VE) می شود. پیچیدگی کلی الگوریتم برابر پیچیدگی محاسبه شار بیشینه + O(VE) است .

با توجه به توضیحات داده شده یال های حساس کاهشی برابر یال های حساس افزایشی نخواهند بود . s

---

۳. الف) هر گراف حداکثر دارای n(n-1)/2 تا یال است . تعداد افراز های ممکن برابر است با انتخاب $0<m<n^2$ یال از داخل n^2 تا یال $f(n)=\binom{n^2}{m}$. بنابراین تعداد افراز های یالی یک گراف از درجه نمایی است ، و نمی توان یک ضریب ثابت برای تعداد آنها به دست آورد.

ب) الگوریتم را به صورت زیر شرح می دهیم :

- مقدار هر یال را برابر 1-p(e) قرار بده.(احتمال درستی)
- یک راس را به عنوان s انتخاب کن.
- هر بار یک راس از V-s را انتخاب کن و آنرا برابر t بگذار.
- با استفاده از الگوریتم push/relabel شار بیشینه را برای این t حساب کن. و مقدار شاره را ذخیره کن.
- در نهایت tای که کمترین شار را به ما داده است انتخاب کن .
- حال یال های متعلق به min-cut به دست آمده از حل این شبکه شاره برابر است با یک افراز با بیشترین احتمال خرابی در شبکه

پیچیدگی الگوریتم برابر است با n بار اجرای الگوریتم push/relabel که برابر است با $O(n^4)$

.۴

26.4-7

اگر یک مسیر از راس u به راس v داشته باشیم ، این مسیر را $<v_0,\ldots,v_k>$ بنامیم داریم :

$h[v_0]<=h[v_1]+1<=\ldots<=h[v_k]+k$ در این مسئله k همان $\delta(u,v)$ است. پس $h[u] \le h[v] + \delta(u,v)$.

الف)برای حالت $h[u] < |v|$ داریم : $h[u] \le h[t] + \delta(u,t)$ از طرفی می دانیم که $h[t] = 0$ پس در نتیجه $h[u] \le \delta(u,t)$. در حالتی که $h[u] \ge |v|$ باشد $\delta(u,t)$ وجود ندارد زیرا اگر چنین مسیری وجود داشته باشد چونکه $h[u] \le h[t] + \delta(u,t)$ نتیجه می گیریم که $\delta(u,t) \ge h[u] \ge |v|$ که تناقض است زیرا مسیری در گراف وجود ندارد که بزرگتر از |V| باشد ! بنابراین تنها از $h[u] < |v|$ می توان نتیجه گرفت که $h[u] \le \delta(u,t)$

ب) برای حالت $h[u] \ge |v|$ نیز داریم :$h[u] \le h[s] + \delta(u,s)$ می دانیم که $h[s] = |v|$ است ، بنابراین $h[u] - |v| \le \delta(u,s)$. در حالتی که $h[u] < |v|$ باشد ، درستی این رابطه بدیهی است! زیرا $\delta(u,s) \ge 0$ و طرف دیگر رابطه همیشه منفی است! عملاً در این حالت این فرمول به هیچ کار نمی آید.

نتیجه : هنگامی که $h[u] \ge |v|$ مسیری به راس t در گراف متمم وجود ندارد.(از این نتیجه در تمرین ۱ استفاده شده است) ، این نتیجه قبلاً داخل کتاب اثبات شده است.

26.5.2

لم : تعداد nsp ها حداکثر برابر است با $O(4n^2)$ ، همان طور که قبلاً (داخل کتاب) اثبات کردیم : h[u] تنها بر اثر relabeling افزایش پیدا می کند ، ارتفاع یک راس حداکثر 2n بار افزایش می یابد . و برای کل رئوس حداکثر $2n^2$ بار افزایش خواهد داشت . relabeling مجموع ارتفاع را به اندازه 2n واحد افزایش می دهد . برای هر افزایش ارتفاع در هر راس تنها یک nsp خواهیم داشت . بنابراین تعداد nsp ها برابر است با $O(4n^3)$

فرضیه : یک پیاده سازی از push/relabel در زمان $O(1) + O(VE)$ برای هر nsp قابل انجام است .

O(VE) زمان مورد نیاز برای انجام sp هاست و طبق اثبات هایی کـه در داخـل کـلاس هـم انجـام دادیـم در هـر nsp مجموع ارتفاع یک واحد کم می شود . بنابراین پیچیدگی الگوریتم از مجموع این ۲ به دست می آید.
با توجه به لم اثبات شده ، و قضیه می توانیم نتیجه بگیریم که پیچیدگی FIFO برابر اسـت بـا $O(n^3+nE)$ کـه همـان $O(n^3)$ می شود.

26-3

الف) فرض خلف : فرض کنیم که برای $E_j \in T$ و $I_k \in R_j$ نتیجه بگیریم که برای یک I_k داریم $I_k \notin T$ یا بـه عبـارت بهتر $I_k \in S$

با توجه به فرض خلف برای یک I_k داریم $E_j \in T$ و $I_k \in S$ از طرفی چون که $I_k \in R_j$ بنابراین $(I_k, E_j) = \infty$ فرض کرده ایم که min-cut دارای مقدار متناهی است ، min-cut شامل همه یال هایی است $\sum E(w,v)$ بـه طـوری که $w \in S \; and \; v \in T$. بنابراین min-cut شامل حداقل یک راس بی نهایت است و ایـن بـا فـرض متنـاهی بـودن مقدارmin-cut متناقض است . بنابراین راس I_k در داخل T قرار دارد.

ب)min-cut گراف شاره را به دو زیر گراف S,T تبدیل می کند ، طبق بند الف ، اگر یک یال p_j متعلق به T باشد آنگاه I های مرتبط با آن نیز در داخل T قرار دارند. اگر مجموع وزن یالهـای P_j را P و مجمـوع min-cut را V بنـامیم آنگـاه سود خالص برابر است با :

Net revenue = P-V

ج) ، فرض کنیم که X* یک شار بیشینه در گراف (G باشد ، حال دو مجموعه T و S را به شیوه زیر تعریف مـی کنـیم : $S = \{u|u: s \to u\}$ و $T = \{u|u: u \to t\}$ این ۲ مجموعه را مـی تـوان بـه سـادگی بـه دسـت آورد ، حـل شـار بیشنیه در گراف مفروض در زمان $(m+n)^3$ قابل انجام است ، مجموعه I که باید حمل شوند برابر است با همه I هـایی که در T-S حضور دارند ، همچنین مجموعه آزمایشات برابر است با آنهایی که در T-S حضور دارند . تعیین مجموعه S و T در زمان r قابل انجام است

پیچیدگی الگوریتم برابر است با $r+(m+n)^3$