دانشکده‌ی علوم ریاضی

آنالیز الگوریتم‌ها — ۳ اردیبهشت۹۲

جلسه‌ی ۱۵: پیدا کردن کوتاه‌ترین مسیر بین تمام زوج رأس‌ها

مدرّس: دکتر شهرام خزائی — *نگارنده: حسین اورعی*

۱ مقدمه

همان‌طور که در جلسات قبل دیدیم، الگوریتم‌های یافتن کوتاه‌ترین مسیر در یک گراف با یک رأس شروع[۱] (SSSP) به صورت‌های زیر می‌باشد:

- اگر طول یال‌ها نامنفی باشد، از الگوریتم دایکسترا استفاده می‌کنیم که پیچیدگی آن $O(m \log n)$ می‌باشد.
- اما اگر روی طول یال‌ها شرطی نداشته باشیم، از الگوریتم بلمن-فورد استفاده می‌کنیم که پیچیدگی آن $O(mn)$ می‌باشد.

در این جلسه می‌خواهیم کوتاه‌ترین مسیر بین تمام زوج رأس‌ها[۲] (APSP) را بیابیم. اگر مسأله را از APSP به SSSP کاهش دهیم، پیچیدگی الگوریتم‌های بالا به صورت زیر خواهد بود:

	$m = \Theta(n^2)$	$m = \Theta(n)$
n بار اجرای الگوریتم دایکسترا	$O(n^3 \log n)$	$O(n^2 \log n)$
n بار اجرای الگوریتم بلمن-فورد	$O(n^4)$	$O(n^3)$

در این جلسه الگوریتم‌های فلوید-وارشال[۳] و جانسون[۴] را برای مسأله APSP بررسی می‌کنیم. الگوریتم‌های فلوید-وارشال و جانسون به ترتیب دارای پیچیدگی‌های $O(n^3)$ و $O(mn \log n)$ می‌باشند.

۲ الگوریتم فلوید-وارشال

الگوریتم فلوید-وارشال از نوع برنامه‌ریزی پویاست. در این الگوریتم ابتدا رأس‌های گراف را از ۱ تا n شماره‌گذاری می‌کنیم و سپس زیرمسأله‌ها را به صورت زیر تعریف می‌کنیم:
کوتاه‌ترین مسیر از s به t که رأس‌های میانی متعلق به مجموعه $\{1, 2, ..., k\}$ باشد. به عبارت دیگر هیچ گره میانی در کوتاه‌ترین مسیر، دارای اندیس بزرگتر از k نباشد.

تعریف ۱ (k-مسیر) *مسیری است که اندیس تمام گره‌های میانی حداکثر k باشند.*

فرض کنیم $\mathrm{dist}(i, j, k)$ طول کوتاه‌ترین k-مسیر از گره i به گره j باشد.

---
[۱] Single Source Shortest Path (SSSP)
[۲] All-Pairs Shortest Path (APSP)
[۳] Floyd-Warshal
[۴] Johnson

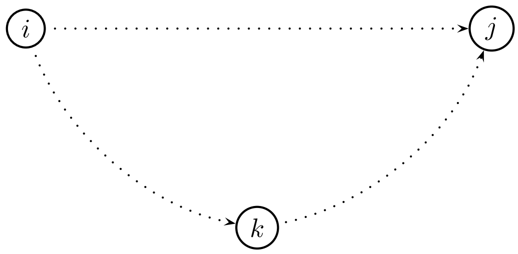
واضح است که:

$$\mathrm{dist}(i,j,۰)=\begin{cases} ۰ & \text{if}: i=j \\ l_{ij} & \text{if}: (i,j)\in E \\ \infty & \text{if}: (i,j)\notin E \end{cases}$$

لم ۱ *فرض کنید گراف وزن‌دار G بدون دور منفی باشد. فرض کنید P کوتاه‌ترین k -مسیر از گره i به j باشد. دراین صورت P به یکی از دو صورت زیر است:*

- *P کوتاه ترین $(k-۱)$-مسیر از گره i به گره j است.*
- *P از ترکیب مسیرهای $P_۱$ و $P_۲$ حاصل می شود که $P_۱$ یک $(k-۱)$-مسیر از i به k و $P_۲$ یک $(k-۱)$-مسیر از k به j است.*

برهان. اگر در مسیر P گرهی با اندیس k وجود نداشته باشد، P کوتاه ترین $(k-۱)$-مسیر از گره i به گره j است. اما اگر گرهی با اندیس k وجود داشته باشد، P از ترکیب مسیرهای $P_۱$ و $P_۲$ حاصل می‌شود که $P_۱$ یک $(k-۱)$-مسیر از i به k، و $P_۲$ یک $(k-۱)$-مسیر از k به j است. این مطلب در شکل زیر و با توجه به اینکه اندیس‌ها متمایزند به خوبی دیده می‌شود:



■

بنابراین خواهیم داشت:

$$\mathrm{dist}(i,j,k)=\min\{\mathrm{dist}(i,j,k-۱),\mathrm{dist}(i,k,k-۱)+\mathrm{dist}(k,j,k-۱)\}$$

و الگوریتم به صورت زیر خواهد بود:

Algorithm 1

for $i=1$ **to** n **do**
  for $j=1$ **to** n **do**
    $\mathrm{dist}(i,j,0)\leftarrow\begin{cases} 0 & : i=j \text{ if} \\ l_{ij} & : (i,j)\in E \text{ if} \\ \infty & : (i,j)\notin E \text{ if} \end{cases}$

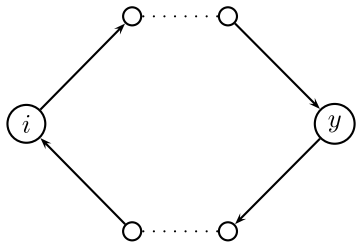
for $k=1$ **to** n **do**
  for $i=1$ **to** n **do**
    for $j=1$ **to** n **do**
      $\mathrm{dist}(i,j,k)\leftarrow\min\{\mathrm{dist}(i,j,k-1),\mathrm{dist}(i,k,k-1)+\mathrm{dist}(k,j,k-1)\}$

return $\mathrm{dist}(i,j,n)$ for all i , j

پیچیدگی این الگوریتم $O(n^۳)$ می‌باشد. حافظه مصرفی نیز $O(n^۳)$ است، اما می‌توان آن را به $O(n^۲)$ کاهش داد.

لم ۲ *گراف G دور منفی دارد، اگرو فقط اگر برای یک رأس i پس از اجرای الگوریتم فوق داشته باشیم:* $\mathrm{dist}(i,i,n) < ۰$.

برهان. ابتدا فرض کنید گراف، دور منفی دارد. همچنین فرض کنید در دور منفی، y رأس با بیشترین اندیس و i نیز یک رأس دلخواه در این دور باشد:



با توجه به لم رابطه بازگشتی الگوریتم داریم: $\mathrm{dist}(i,i,y) \leq \mathrm{dist}(i,y,y-۱) + \mathrm{dist}(y,i,y-۱)$.
از طرفی چون دور، منفی می‌باشد: $\mathrm{dist}(i,y,y-۱) + \mathrm{dist}(y,i,y-۱) < ۰$. بنابراین: $\mathrm{dist}(i,i,y) < ۰$. همچنین داریم: $\mathrm{dist}(i,i,k) \leq \mathrm{dist}(i,i,k-۱)$. لذا $\mathrm{dist}(i,i,n) < ۰$.
برای اثبات طرف دیگر لم فرض کنید برای یک رأس i داشته باشنم: $\mathrm{dist}(i,i,n) < ۰$. بنابراین با شروع از رأس i یک دور منفی خواهیم داشت. ■

۳ الگوریتم جانسون

۱.۳ مقدمه

الگوریتم جانسون، یک بار از الگوریتم بلمن-فورد و n بار از الگوریتم دایکسترا استفاده می‌کند. بنابراین اگر طول یال‌ها منفی باشد باید طول یال‌ها را مثبت کنیم. برای این کار از *طول‌گذاری مجدد* استفاده می‌کنیم:

لم ۳ *فرض کنید در گراف G مسیر دلخواه P از رأس s به رأس t دارای طول l باشد. به هر رأس گراف، یک وزن p_v نسبت دهید و طول یال $uv \in E$ را از l_{uv} به $l'_{uv} = l_{uv} + p_u - p_v$ تغییر دهید. در این صورت طول P از l به $l + p_s - p_t$ تغییر پیدا می‌کند.*

برهان. فرض کنیم l' طول جدید مسیر P باشد. واضح است که با توجه به رابطه

$$l' = \sum_{uv \in P} (l_{uv} + p_u - p_v)$$

خواهیم داشت: $l' = l + p_s - p_t$. ■

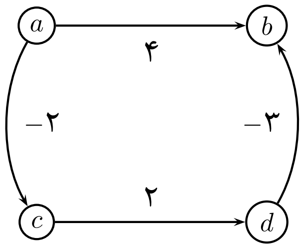
نتیجه ۱ *اگر P کوتاه‌ترین مسیر از s به t باشد، پس از طول‌گذاری مجدد نیز کوتاه‌ترین مسیر از s به t باقی خواهد ماند.*

برهان. با توجه به لم قبل چون همه مسیرها از s به t به یک اندازه تغییر می‌کنند، بنابراین P پس از طول‌گذاری مجدد نیز کوتاه‌ترین مسیر از s به t باقی خواهد ماند. ■

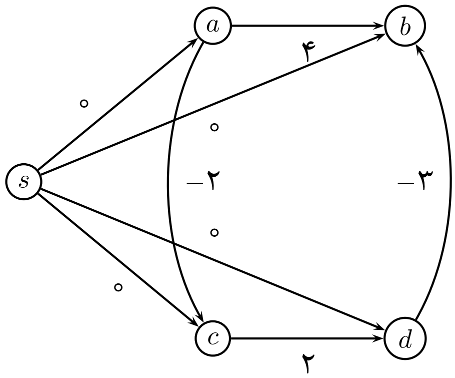
۲.۳ طول‌گذاری مجدد

در الگوریتم جانسون، ابتدا گراف G' را از روی گراف G به صورت زیر می‌سازیم:
گره جدید s را به همراه یال‌های sv (برای هر $v \in V$) به گراف G اضافه می‌کنیم. طول این یال‌ها را نیز ۰ در نظر می‌گیریم. سپس طول کوتاه‌ترین مسیر از s به هر گره $v \in V$ را p_v نامیده و آن را به عنوان وزن هر گره در نظر می‌گیریم و با استفاده از روشی که در لم قبل بیان شد، طول‌های جدیدی به یال‌ها نسبت می‌دهیم تا گراف G'' حاصل شود.

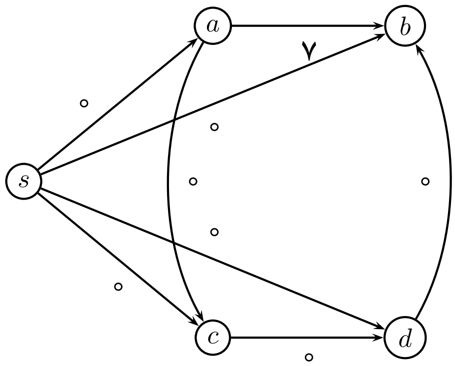
مثال ۱ *فرض کنید گراف G به صورت زیر باشد:*



در این صورت گراف G' به صورت زیر خواهد بود که در آن وزن رأس‌ها $p_a = ۰$، $p_b = -۳$، $p_c = -۲$، $p_d = ۰$ و $p_s = ۰$ می‌باشند:



پس از طول‌گذاری مجدد گراف G'' نیز به صورت زیر می‌باشد:



قضیه ۴ *در الگوریتم جانسون طول‌گذاری مجدد طول یال‌ها را نامنفی می‌کند.*

برهان. فرض کنیم u و v دو گره دلخواه باشند که در گراف G مجاورند. چون p_u و p_v به ترتیب کوتاه‌ترین مسیر از گره s به گره‌های u و v می‌باشند، داریم: $p_u + l_{uv} \geq p_v$ در نتیجه $l'_{uv} = l_{uv} + p_u - p_v \geq ۰$. ■

۳.۳ الگوریتم

الگوریتم جانسون به صورت زیر می‌باشد:

- تشکیل گراف G' از روی گراف G.
- اجرای الگوریتم بلمن-فورد با رأس مبدأ s و محاسبه p_vبرای تمام رأس‌های گراف.
- محاسبه طول‌های جدید و تشکیل گراف G''.

- (محاسبه $l'(u,v)$ ها) برای هر $u \in V$، اجرای الگوریتم دایکسترا با رأس مبدأ u و یافتن کوتاه‌ترین مسیر از u به v برای هر $v \in V$ به طول $l'(u,v)$.
- (محاسبه طول‌های واقعی $(l(u,v)$ محاسبه $l(u,v) = l'(u,v) - p_u + p_v$ تا طول یال‌ها به مقدار واقعی برگردد.

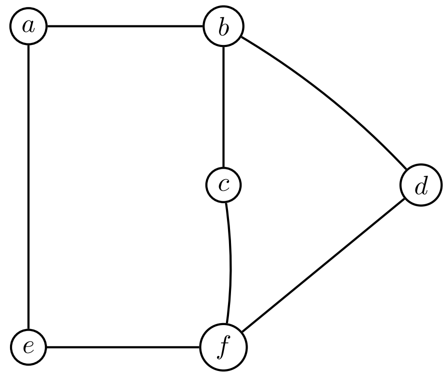
پیچیدگی مرحله اول الگوریتم $O(n)$، مرحله دوم $O(mn)$، مرحله سوم $O(m)$، مرحله چهارم $O(mn \log n)$ و مرحله پنجم نیز $O(n^۲)$ می‌باشد. در نتیجه پیچیدگی الگوریتم $O(mn \log n)$ خواهد بود.

۴ بزرگ‌ترین زیرمجموعه مستقل

تعریف ۲ *یک زیرمجموعه از رأس‌های گراف G یک زیرمجموعه مستقل گراف می‌باشد، اگر بین آن‌ها هیچ یالی نباشد.*

در مسأله بزرگ‌ترین زیرمجموعه مستقل[5] می‌خواهیم بزرگ‌ترین زیرمجموعه مستقل یک گراف را به دست بیاوریم.

مثال ۲ *در گراف زیر، اندازه بزرگ‌ترین زیرمجموعه مستقل، سه می‌باشد (رأس‌های $\{a, c, d\}$).*



در صورتی که گراف یک درخت باشد، با استفاده از برنامه‌ریزی پویا، برای مسأله فوق راه حل چندجمله‌ای وجود دارد:

۱.۴ بزرگ‌ترین زیرمجموعه مستقل در درخت‌ها

برای به دست آوردن بزرگ‌ترین زیرمجموعه مستقل در یک درخت، از برنامه‌ریزی پویا استفاده می‌کنیم. برای هر گره u در درخت، یک زیر مسأله را به صورت زیر تعریف کرده و آن را با $I(u)$ نمایش می‌دهیم:
بزرگ‌ترین زیرمجموعه مستقل در زیردرخت با ریشه u.
اگر این زیرمجموعه مستقل شامل گره u باشد $I(u)$ برابرست با:

$$I(u) = \sum_{w \in \{\text{ grandchildren of } u\}} I(w)$$

درغیر این صورت داریم

$$I(u) = \sum_{w \in \{\text{children of } u\}} I(w)$$

بنابراین $I(u)$ برابر با ماکزیمم دو مقدار فوق خواهد بود. واضح است که اگر رأس v یک برگ باشد داریم: $I(v) = ۱$.
تعداد زیرمسأله‌ها برابر با تعداد کل رأس‌هاست. بنابراین پیچیدگی این الگوریتم $O(m+n)$ می‌باشد.

[5]Largest Independent Subset

۵ مسأله فروشنده دوره‌گرد

فرض کنید در یک گراف کامل داده شده، مجموعه رأس‌های گراف را با استفاده از اندیس‌های $\{1,2,...,n\}$ شماره‌گذاری کنیم. همچنین فرض کنید یال ij بین رأسهای i و j دارای وزن d_{ij} باشد. در مسأله فروشنده دوره‌گرد[۶] می‌خواهیم مسیری را در گراف بیابیم که شروع و خاتمه‌اش رأس ۱ باشد، از هر رأس دقیقاً یک بار عبور کند و طول مسیر طی شده کمینه باشد. روش جستجوی کامل برای حل این مسأله به زمان $O(n!)$ نیاز دارد. در اینجا با استفاده از برنامه‌ریزی پویا پیچیدگی را به $O(n^2 2^n)$ می‌رسانیم:

برای تعریف زیرمسأله‌ها فرض کنید از رأس ۱ شروع کرده‌ایم، چند رأس را پشت سر گذاشته‌ایم و اکنون در رأس j می‌باشیم. حال لازم است هم رأس‌های طی شده را بدانیم تا دیگر آن‌ها را تکرار نکنیم و هم موقعیت رأس j را بدانیم تا بتوانیم مسیر را ادامه دهیم. بنابراین می‌توان زیرمسأله‌ها را به صورت زیر تعریف کرد:

فرض کنید برای $S \subseteq \{1,...,n\}$ که شامل گره‌های ۱ و j نیز می‌باشد، $C(S,j)$ طول کوتاه‌ترین مسیری باشد که از رأس ۱ شروع و در رأس j خاتمه می‌یابد و از تمام رأس‌های S فقط یک بار عبور می‌کند. واضح است که $C(S,j)$ برابر با طول کوتاه ترین مسیر از ۱ به i می باشد که از تمام رأس‌های $S-\{j\}$ فقط یک بار عبور کرده باشد (برای بهترین i عضو S) به اضافه d_{ij}. پس خواهیم داشت:

$$C(S.j) = \min_{i \in S, i \neq j} \{C(S-\{j\}, i) + d_{ij}\}$$

وقتی $|S| > 1$، تعریف میکنیم: $C(S,1) = \infty$ زیرا در این حالت مسیر نمی‌تواند از رأس ۱ شروع شود و به رأس ۱ نیز ختم شود.

در نهایت الگوریتم به صورت زیر خواهد بود:

Algorithm 2

$C(\{1\},1) = 0$
for $s = 2$ **to** n **do**
  for all subsets $S \subseteq \{1,...,n\}$ of size s and containing 1 **do**
    $C(S,1) = \infty$
    for all $j \in S, j \neq 1$ **do**
      $C(S.j) = \min_{i \in S, i \neq j}\{(C(S-\{j\}, i) + d_{ij}\}$

return $\min_j (C(\{1,...,n\}, j) + d_{j1}$

به وضوح پیچیدگی الگوریتم برنامه‌ریزی پویای فوق $O(n^2 2^n)$ می‌باشد.

[۶] Traveling Salesman Problem (TSP)