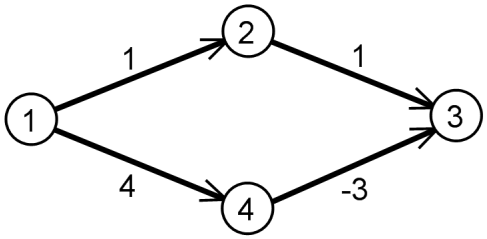
درس طراحی الگوریتم
دانشکده ریاضی. دانشگاه صنعتی خواجه نصیرالدین طوسی. پاییز ۱۴۰۳
کتاب مرجع: طراحی الگوریتم. جان کلاینبرگ و اوا تاردش

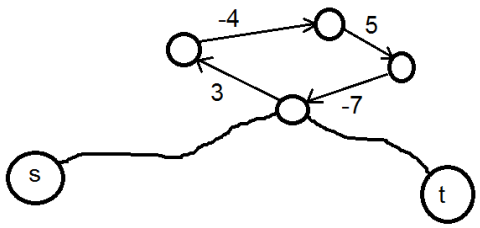
۱ الگوریتم بلمن ـ فورد برای کوتاهترین مسیر در گراف Bellman-Ford algorithm

به مسئله کوتاهترین مسیر در گراف برمیگردیم. اگر یادتان باشد، الگوریتم دایکسترا برای گرافهایی که یال با طول منفی داشتند قابل استفاده نبود. به شکل زیر توجه کنید.



در این درس میخواهیم الگوریتم بلمن فورد را معرفی کنیم که با وجود یال با طول منفی در گراف کوتاهترین مسیر را پیدا میکند. البته این الگوریتم به شرطی درست کار میکند که دور با طول منفی در گراف وجود نداشته باشد. اینجا فرض بر این است که گراف جهت دار است. دور منفی، یعنی دوری که مجموع طول یالهایش منفی باشد.

وقتی دور منفی در گراف $G = (V, E)$ وجود داشته باشد، اگر مسیر بین s و t به این دور دسترسی داشته باشد، آنگاه مسیر میتواند بطور مکرر از این دور استفاده کند و طول مسیر از s تا t را به دفعات کاهش دهد. در واقع در این حالت هیچ کران پایینی برای طول مسیر وجود نخواهد داشت.



لذا فرض میگیریم که دور منفی در گراف ورودی وجود ندارد.

۲ برنامه نویسی پویا برای کوتاهترین مسیر در گراف

فرض کنید میخواهیم طول کوتاهترین مسیر از همه رئوس گراف جهت دار $G = (V, E)$ به راس $t \in V$ را پیدا کنیم. برعکس الگوریتم دایکسترا که یک مبدا را مشخص میکرد، اینجا یک مقصد معین می‌شود.

ابتدا تعریف زیر را می آوریم.

تعریف: فرض کنید $OPT(i, s)$ طول کوتاهترین مسیر از s به t باشد که از حداکثر i یال استفاده میکند.

مشاهده: اگر گراف ورودی، دور منفی نداشته باشد، آنگاه کوتاهترین مسیر از هر راس به راس دیگر حداکثر $n - 1$ یال دارد.

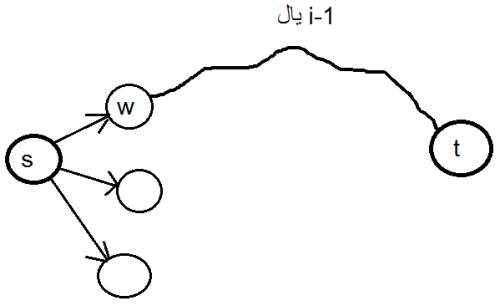
با توجه به مشاهده بالا، با فرض اینکه G دور منفی ندارد، جواب مسئله برای طول کوتاهترین مسیر از s به t برابر با $OPT(n - 1, s)$ است.

حال همانند مسائل قبل که با تکنیک برنامه سازی پویا حل کردیم، از تفکر بازگشتی و استقرایی استفاده میکنیم.

برای $OPT(i, s)$ دو حالت وجود دارد.

- کوتاهترین مسیر از s به t با حداکثر i یال، دقیقا از i یال استفاده میکند. با فرض اینکه $\ell(s, w)$ طول یال (s, w) است، در این حالت میتوانیم بنویسیم

$$OPT(i, s) = \min_{(s,w)\in E} \{\ell(s, w) + OPT(i - 1, w)\}$$



- کوتاهترین مسیر از s به t با حداکثر i یال، از تعداد کمتر از i یال استفاده میکند. در این حالت میتوانیم بنویسیم

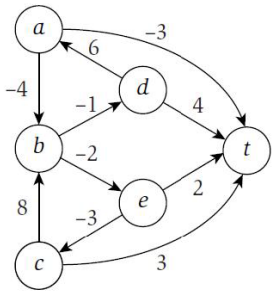
$$OPT(i, s) = OPT(i - 1, s)$$

پس در کل می‌توانیم بنویسیم

$$OPT(i, s) = \min\{OPT(i - 1, s), \min_{(s,w)\in E} \{\ell(s, w) + OPT(i - 1, w)\}\}$$

میتوانیم نتیجه بگیریم که حل زیر مسئله $OPT(i,s)$ با استفاده از حل زیرمسائل کوچکتر $OPT(i-1,s)$ و $OPT(i-1,w)$ برای همه w ها قابل انجام است.

طرز کار الگوریتم با مثال زیر روشن میشود. جدول OPT در قسمت سمت راست شکل است. هر سطر مربوط به یک راس گراف است. راس t راس مقصد است. ستونها از چپ به راست برای $i=0$ تا $i=n-1$ است. دقت کنید ستون آخر جواب الگوریتم است با فرض اینکه گراف دور منفی ندارد.



	i=0	i=1	i=2	i=3	i=4	i=5
t	0	0	0	0	0	0
a	∞	t −3	t −3	b −4	b −6	b −6
b	∞	∞	e 0	e −2	e −2	e −2
c	∞	t 3	t 3	t 3	t 3	t 3
d	∞	t 4	a 3	a 3	a 2	a 0
e	∞	t 2	c 0	c 0	c 0	c 0

عددی که در هر خانه جدول نوشته شده، طول کوتاهترین مسیر از راس مربوطه به t است که حداکثر از i یال استفاده میکند.

علاوه بر این، در گوشه سمت چپ بالای هر خانه، اولین راس بعدی در کوتاهترین مسیر نوشته شده است (به رنگ قرمز). با استفاده از این اطلاعات میتوان کوتاهترین مسیر از هر راس به t را پیدا کرد.

۳ زمان اجرا و فضای مصرفی الگوریتم

زمان اجرا. برای محاسبه هر خانه از جدول OPT حداکثر n خانه دیگر مورد دسترسی قرار میگیرد. جدول به تعداد $(n-1)\times n$ خانه دارد. پس یک کران بالای بدیهی برای زمان اجرا $O(n^3)$ است.

میتوانیم یک کران بالای بهتر برای زمان اجرا بدست بیاوریم. دقت کنید در واقع برای محاسبه خانه $OPT(i,s)$ به تعداد درجه خروجی راس s عمل مقایسه انجام میدهیم. درجه خروجی یعنی تعداد یالهایی که از راس خارج میشود. محض یادآوری

$$OPT(i,s)=\min\{OPT(i-1,s),\min_{(s,w)\in E}\{\ell(s,w)+OPT(i-1,w)\}\}$$

فرض کنید $d_{out}(s)$ درجه خروجی راس s باشد. لذا برای محاسبه یک ستون جدول، زمان مورد نیاز حداکثر

$$\sum_{s\in V} d_{out}(s)=m$$

خواهد بود. دقت کنید m تعداد یالهای گراف است. لذا برای هر ستون به اندازه $O(m)$ زمان صرف می‌کنیم و چون n ستون داریم پس $O(nm)$ یک کران بالا برای زمان اجراست.

فضای مصرفی الگوریتم. فضای مصرفی الگوریتم برابر با فضای مورد نیاز برای نگهداری جدول OPT و خود گراف است. جدول OPT اندازه ش $O(n^2)$ است. خود گراف را هم میتوان در فضای $O(n+m)$ نگهداری کنیم. لذا فضای مورد نیاز الگوریتم $O(n^2+m)$ است که همان $O(n^2)$ است چون $m \leq n^2$.

توجه کنید جدول OPT بصورت ستون به ستون از چپ به راست پر می‌شود. علاوه بر این مقدار هر ستون تنها وابسته به ستون قبلی و گراف ورودی است. لذا اگر هدف تنها محاسبه طول کوتاهترین مسیر باشد، نیازی نیست که ستونهای قدیمی را نگه داریم و تنها کافی است که دو ستون آخر را نگه داریم. در این حالت فضای مصرفی برابر با $O(n+m)$ خواهد بود.

۴ محاسبه یالهای مسیر

برای محاسبه خود مسیر، آرایه $first[]$ را نگه می‌داریم. در ابتدا داریم

$$\forall u \in V, \qquad first[u] \leftarrow u.$$

هر وقت $OPT(i,u)$ بروزرسانی شد، یعنی از مقدار قبلی اش $OPT(i-1,u)$ کمتر شد، آن موقع مقدار $first[u]$ را بروزرسانی می‌کنیم. فرض کنید $OPT(i,u) \leftarrow OPT(i-1,w)+\ell(u,w)$ در این حالت قرار می‌دهیم $first[u] \leftarrow w$.

آرایه $first[]$ یک گراف را تعریف می‌کند به آن گراف اشاره گر pointer graph گفته می‌شود. گراف G' با رئوس $V=\{1,\cdots,n\}$ را در نظر بگیرید. یال (u,w) را در گراف G' قرار می‌دهیم اگر و فقط اگر $first[u]=w$. توجه کنید که در گراف اشاره‌گر G' هر راس درجه خروجی اش دقیقا 1 است.
با این تعاریف در انتهای الگوریتم بلمن‌ـ فورد، گراف G' کوتاهترین مسیر از راس s به راس مقصد t را مشخص می‌کند. کافی است که یال خروجی از s را دنبال کنیم تا اینکه به t برسیم. اما اگر هیچ وقت به راس t نرسیم چه؟ چون درجه هر راس دقیقا 1 است، دو حالت وجود دارد:

- هیچ مسیری از s به راس مقصد t در گراف اولیه وجود نداشته. روشن است در این حالت کوتاهترین مسیر وجود ندارد.
- یک دور در گراف G' وجود دارد.

از لم زیر نتیجه می‌شود که اگر گراف ورودی G دور منفی نداشته باشد، آنگاه گراف اشاره‌گر G' دور نخواهد داشت و لذا کوتاهترین مسیرها از آن قابل استخراج است.
لم: بعد از اتمام الگوریتم بلمن فورد، اگر C یک دور در گراف اشاره‌گر G' باشد آنگاه C باید یک دور منفی در گراف G باشد.
اثبات: فرض کنید که $v_1 \to v_2 \to \cdots \to v_k \to v_1$ یک دور در گراف اشاره‌گر باشد. می‌توانیم فرض کنیم که یال $e=(v_k,v_1)$ آخرین یالی است که در دور C ایجاد شده است. فرض کنید یال e در مرحله i ام اضافه شده باشد. دقت کنید قبل از اینکه یال e اضافه شود داریم

$$OPT(i-1,v_k) > \ell(v_k,v_1)+OPT(i-1,v_1) \qquad (۱)$$

به همین ترتیب به دلیل وجود دیگر یالهای دور باید داشته باشیم:

$$OPT(i-1, v_1) \geq \ell(v_1, v_2) + OPT(i-1, v_2)$$

$$OPT(i-1, v_2) \geq \ell(v_2, v_3) + OPT(i-1, v_3)$$

$$\dots$$

$$OPT(i-1, v_{k-1}) \geq \ell(v_{k-1}, v_k) + OPT(i-1, v_k)$$

همه این نامساوی ها و (۱) را که با هم جمع بزنیم مقادیر OPT از نامساوی ها خط می‌خورند و بدست می‌آید

$$0 > \ell(v_1, v_2) + \dots + \ell(v_k, v_1)$$

در نتیجه C یک دور منفی است.

۵ چند نکته در مورد الگوریتم بلمن فورد

- با توجه به طرز کار الگوریتم، اگر مقادیر دو ستون آخر برابر باشند، نیازی به پیشروی بیشتر نیست و میتوانیم الگوریتم را همانجا خاتمه دهیم. چون مقادیر ستونهای بعدی نیز تغییری نخواهند کرد.
- اگر در گراف دور منفی وجود نداشته باشد، ستون $i = n$ حتما برابر با ستون $i = n - 1$ خواهد بود.
- اگر در گراف دور منفی وجود داشته باشد (به شرطی که این دور به t راه داشته باشد) آنگاه در ستون $i = n$ درایه‌ای وجود خواهد داشت که مقدارش کمتر از مقدار خانه متناظر در ستون قبلی است. چرا؟