جزوه جلسه بیست‌ویکم داده ساختارها و الگوریتم

۱۴ آذر ۱۴۰۰

فهرست مطالب

۱ مقدمات کوتاه ترین مسیر در گراف

الگوریتم BFS که در جلسات قبل بررسی کردیم، تا حدودی کوتاه ترین مسیر در گراف بدون وزن (گرافی که تفاوتی بین رئوس وجود ندارد) را در لایه های مختلف نشان می داد. حال در ادامه به بررسی الگوریتم های دیگر پیدا کردن کوتاه ترین مسیر در گراف می پردازیم.

۱.۱ کاربردهای کوتاه ترین مسیر

- مسیریابی جاده‌ای در برنامه هایی مانند waze یا google map
- مسیریابی شبکه در routerها برای رسیدن یک پکت از مبدا به مقصد

۲.۱ تعریف دقیق مسئله و برخی تعاریف مهم

- گراف وزن دار: در این گراف، به هر یال یک عدد حقیقی تحت عنوان وزن آن نسبت داده می شود. پس در نمایش گراف، تابع وزن یا w را نیز نشان می دهیم.
$G(V, E, W)$, $W: E \to R$

- مسیر: مسیر دنباله‌ای از رئوس است که هر یال بین ۲ راس متوالی، عضو E باشد. مسیر P در شکل زیر نمایش داده شده است.
$P=<v_0, v_1, ..., v_k>$, $(v_i, v_{i+1}) \in E$
طبق تعریف بالا مشکلی با تکراری بودن یال یا راس نداریم.

- مسیر ساده: مسیری که در آن راس تکراری نداریم.
در مسئله کوتاه ترین مسیر کار کردن با این تعریف سخت است، پس تعریف قبلی را در نظر می گیریم.

- وزن مسیر: وزن یک مسیر (w) برابر است با مجموع وزن یال های مسیر.
$w(P) = \sum_{i=0}^{k-1} w(v_i, v_{i+1})$

- وزن کوتاه ترین مسیر: بین دو راس u و v وزن کوتاه ترین مسیر را $\delta(u, v)$ می نامیم که برابر است با مینیمم وزن مسیرهایی که از u به v وجود دارد. همچنین دو راس که مسیری به هم ندارند وزن مینیمم آنها را ∞ در نظر می گیریم.
حالتی را در نظر بگیرید که دور منفی در مسیر داشته باشیم (دور منفی دوری است که جمع وزن یالهای آن منفی است). پس میتوان با هر با دور زدن طول مسیر را حداقل یک واحد کاهش داد و در نهایت می توان به $-\infty$ رسید. برای

رفع این مشکل به جای در نظر گرفتن مینیمم طول مسیرها، اینفیمم آن ها را در نظر می گیریم.

یکی از کاربرد های دور منفی در تبدیل ارز یا رمزارز می باشد. به این صورت که هر یال (u,v) با وزن w برابر است با نرخ تبدیل یک u به v می باشد. حال اگر از یک راس شروع به دور زدن کنیم و وزن یال ها را در هم ضرب کنیم در صورتی که مقدار نهایی برابر یک نشود، نشانگر وجود یک مشکل در وضعیت ثبات قیمت رمزارز هاست.
برای شهود بهتر نسبت به رمزارز میتوان وزن یالها را برابر لگاریتم نرخ تبدیل قرار داد و در صورتی جمع وزن ها صفر نشود، وجود یک مشکل گزارش می شود.

$$ln(w_1) + ln(w_2) + ... + ln(w_k) \neq 0$$

۳.۱ دسته بندی مسائل کوتاه ترین مسیر

۱. کوتاه ترین مسیر بین یک جفت راس

۲. کوتاه ترین مسیر از یک مبدا به بقیه راس‌ها (Single Source Shortest Path) یا درخت کوتاه ترین مسیر

۳. کوتاه ترین مسیر بین هر جفت راس (All Pair Shortest Path)

در این درس الگوریتم های دوم یا SSSP رابررسی می کنیم. همچنین الگوریتم هایی که می شناسیم (مانند برنامه مسیریابی waze) در بدترین حالت مسئله SSSP را حل میکنند.

۱.۳.۱ الگوریتم های معروف برای حل مسئله **SSSP**

- BFS: برای گراف هایی با وزن مثبت و برابر کاربرد دارد و در زمان $O(n+e)$ انجام می شود.
- کوتاه ترین مسیر در گراف جهت دار بدون دور: این الگوریتم نیز در زمان $O(n+e)$ انجام می شود.
- الگوریتم دکسترا (Dijkstra): این الگوریتم برای گراف هایی با وزن نامنفی در بهترین زمان $O(nlogn + e)$ کوتاه ترین مسیر را پیدا می کند.
- الگوریتم بلمن فورد: این الگوریتم در زمان $O(ne)$ کوتاه ترین مسیر را در هر گرافی پیدا می کند. به دلیل زیاد بودن زمان اجرا، استفاده از این الگوریتم زمانی توصیه می شود که نتوان از سه الگوریتم بالا استفاده کرد.

در مسائل SSSP مشابه آنچه در BFS دیدیم، میتوان درخت کوتاه ترین مسیر از یک راس را تشکیل داد.

۲.۳.۱ اسکلت بندی الگوریتم های کوتاه ترین مسیر (بدون دور منفی)

- مرحله Initialize: در این مرحله، $d[v]$ را برای هر راس برابر طول کوتاه ترین مسیر از مبدا s به v تعریف می کنیم.
- مرحله Main: در طول اجرای الگوریتم اجازه میدهیم $d[v]$ بیشتر مساوی مقدار واقعی خود باشد و در انتها با آن برابر شود.
  به این کار آسان کردن شرایط، یا relax کردن میگویند. البته در اصل عمل متضاد relax انجام می شود و شرایط را سخت تر می کنیم!
  relax کردن یال (u, v) با وزن $w(u, v)$ به شکل زیر می باشد:
  اگر $d[v] > d[u] + w(u, v)$ باشد، آنگاه می توان مسیر به v را بهبود بخشید و قرار داد: $d[v] \leftarrow d[u] + w(u, v)$

ترتیب اجرای عملیات relax روی رئوس مهم است و در صورت عدم رعایت، زمان اجرای الگوریتم می تواند چند جمله ای نباشد.

Initialize:
$\forall v \in V$: $d[v] \leftarrow -\infty$, $parent[v] \leftarrow None$ (for root: $d[s] \leftarrow 0$)

Main:
repeat:
somehow select (u, v)
if d[v] > d[u] + w(u, v):
d[v] = d[u] + w(u, v)
parent[v] = u

تفاوت الگوریتم های ذکر شده تنها در خط somehow select (u, v) می باشد.