طراحی الگوریتم

پاسخنامه تمرین چهارم - الگوریتم‌های گراف

فاطمه کرمی و هستی کریمی

۱. فاصله‌ها ۱۵ نمره

یک درخت با n راس و عدد k به شما داده شده است. تعداد جفت گره‌هایی را پیدا کنید که فاصله بین آن‌ها دقیقاً برابر k باشد.

پاسخ :

این مسئله را می‌توان با استفاده از برنامه‌نویسی پویا حل کرد. در نظر بگیرید که درخت را آویزان کرده و آن را ریشه‌دار کنیم. برای هر رأس v از درخت، مقادیر $d[v][lev]$ را محاسبه می‌کنیم (که در آن $0 \leq lev \leq k$). $d[v][lev]$ تعداد رأس‌های زیر درخت v را ذخیره می کند که که فاصله آن‌ها تا v برابر lev است. توجه داشته باشید که $d[v][0] = 0$ می‌باشد.

سپس پاسخ مسئله را محاسبه می‌کنیم. پاسخ برابر با مجموع دو مقدار زیر برای هر رأس v است:

- تعداد مسیرهای به طول k که در زیر درخت v شروع و در v تمام می‌شوند. بدیهی است که این مقدار برابر $d[v][k]$ است.
- تعداد مسیرهای به طول k که در زیر درخت v شروع و در زیر درخت v پایان می‌یابند. این مقدار برابر با مجموع مقادیر زیر برای هر راس u فرزند v است:

$$0.5 \cdot \sum_{x=1}^{k-1} d[u][x-1] \cdot (d[v][k-x] - d[u][k-x-1]).$$

در نهایت مجموع این مقادیر را برای همه رأس‌ها محاسبه کرده و پاسخ مسئله را در پیچیدگی زمانی $O(n \cdot k)$ به دست می‌آوریم.

## ۲. جهت‌دهی گراف	۱۵ نمره

یک گراف بی‌جهت با n راس و m یال به شما داده شده است. یال‌های این گراف را به گونه‌ای جهت‌دار کنید که از هر رأس بتوان با پیمودن مسیری به هر رأس دیگر گراف رسید.

پاسخ :

فرض کنید گراف ما حداقل یک یال برشی دارد (یالی که با حذف آن تعداد مولفه های همبندی افزایش می یابند.) می دانیم که این یال در هیچ دوری حضور ندارد و در نتیجه اگر دو سر آن را رئوس A و B در نظر بگیریم بین این دو تنها یک مسیر وجود دارد و هر طور که این یال را جهت دهی کنیم یا دیگر از A به B دسترسی نخواهیم داشت و یا برعکس. در نتیجه تمام یال ها باید حداقل در یک دور حضور داشته باشند. حال برای نحوه جهت دهی از الگوریتم DFS استفاده می کنیم و اثبات می کنیم که اگر یال های دیده شده در درخت DFS را از سمت پدر به بچه جهت بدهیم و یال های $backedge$ را در جهت عکس، در این صورت گرافی جهت دار و قویا همبند خواهیم داشت. ابتدا مشخص است که می‌توان از ریشه (راس دلخواه X) به هر رأس دیگری رسید. حالا باید ثابت کنیم که از هر رأس v می‌توان به رأس u (که پدر v در درخت DFS است) دسترسی پیدا کرد. اگر این را ثابت کنیم، با استقرا می‌توانیم نشان دهیم که از هر رأس می‌توان به ریشه رسید و گراف به یک گراف قویاً همبند تبدیل می‌شود. حال به سراغ اثبات دسترسی از v به u می‌رویم. فرض کنید یال v به u را از گراف حذف می‌کنیم. از آنجا که این یال برشی نیست، یک مسیر دیگر از v به u وجود دارد. واضح است که باید یک یال معکوس از زیردرخت v به یکی از والدهای v وجود داشته باشد(چون یالی که حذف کردیم حداقل در یک دور بوده است). حال می‌توانیم از v به ابتدای این یال برویم، از آن استفاده کنیم و سپس در امتداد درخت به سمت پایین حرکت کنیم تا به u برسیم.

## ۳. مسیرپوشی	۲۰ نمره

یک گراف جهت‌دار با n رأس و m یال وزن‌دار با وزن‌های مثبت به شما داده می‌شود. الگوریتمی با پیچیدگی زمانی $O((n+m)\log(n))$ طراحی کنید که برای دو رأس مشخص a و b، بررسی کند هر رأس دیگر گراف در همه کوتاه‌ترین مسیرها از a به b حضور دارد، فقط در برخی از آن‌ها حضور دارد یا اصلاً در هیچ‌کدام حضور ندارد.

پاسخ :

می‌دانیم به ازای هر رأس v در گراف، در صورتی این رأس می‌تواند در حداقل یکی از کوتاه‌ترین مسیرها بین a و b باشد که شرط زیر برقرار باشد:

$$dist_{a\to v} + dist_{v\to b} = dist_{a\to b}$$

در صورتی که شرط بالا برای رأس v برقرار نباشد این رأس در هیچ‌کدام از کوتاه‌ترین مسیرها بین a و b حضور ندارد.

همچنین اگر رأس v در همه کوتاه‌ترین مسیرها بین a و b حضور داشته باشد شرط زیر برقرار است:

$$count_{a\to v} \times count_{v\to b} = count_{a\to b}$$

در شرط بالا $count_{i\to j}$ نشان‌دهنده تعداد کوتاه‌ترین مسیرها از رأس i تا رأس j است.

ابتدا برای به دست آوردن $dist_{a\to v}$ به ازای تمام v ها، یک بار الگوریتم دایکسترا را با شروع از رأس a روی گراف اجرا می‌کنیم. سپس برای به دست آوردن $dist_{v\to b}$ به ازای تمام v ها یک بار الگوریتم دایکسترا را روی گراف برعکس شده (گرافی که جهت یال‌های آن برعکس گراف اصلی است) با شروع از رأس b اجرا می‌کنیم. حال می‌توان رئوسی که در هیچ‌کدام از کوتاه‌ترین مسیرها از a به b حضور ندارند را با توجه به شرط اول تشخیص داد. این بخش در زمان $O(m\log(n))$ (انجام ضرب و جمع از $O(1)$ است) انجام می‌شود.

حال برای تشخیص رئوسی که در همه کوتاه‌ترین مسیرها بین a و b حضور دارند باید باید $count_{a\to v}$ و $count_{v\to b}$ را به ازای همه رئوس محاسبه کنیم. برای این کار پس از اجرای دایکسترا با شروع از رأس a روی گراف اصلی، گراف جدیدی تشکیل می‌دهیم که همه رئوس گراف اصلی را دارد، اما فقط شامل یال‌هایی‌ست که بین دو رأس u و v هستند، وزن w دارند و این شرط را برآورده می‌کنند:

$$dist[u] + w = dist[v]$$

حال رئوس این گراف را به ترتیب $dist[v]$ می‌چینیم. می‌دانیم هیچ یالی در این ترتیب از رأس‌های جلوتر به رأس‌های عقب‌تر برنمی‌گردد (چون رئوس جلوتر فاصله بیشتری از s دارند) و همه یال‌ها رو به جلو هستند. آرایه $count$ را به این صورت تشکیل می‌دهیم: $count[v]$ نشان‌دهنده $count_{a\to v}$ است و این آرایه از سمت چپ به این شکل پر می‌شود:

$$count[s] = 1$$
$$count[v] = \sum_{u}(count[u])$$

به طوری که هر i سر دیگر یکی از یال‌های ورودی به رأس v در گراف جدید است. این آرایه در زمان $O(m+n)$ پر می‌شود.

به همین ترتیب مقادیر $count_{v\to b}$ را نیز با استفاده از گراف معکوس به دست می‌آوریم و سپس به ازای هر رأس شرط دوم را چک می‌کنیم.

به این ترتیب به ازای هر رأس در گراف مشخص می‌شود که آیا در همه کوتاه‌ترین مسیرها از a به b حضور دارد، فقط در برخی از آن‌ها حضور دارد یا اصلاً در هیچ‌کدام حضور ندارد.

پیچیدگی زمانی کل این الگوریتم برابر با اجرای دایکسترا و اجرای الگوریتم DP ذکر شده است که در مجموع $O(m+n) + O(m\log(n)) + O(n\log(n)) = O((m+n)\log(n))$ زمان می‌گیرد.

۴. درخت کریسمس ۲۰ نمره

درختی داریم که هر رأس آن با یک بازه مشخص می‌شود. رأس i با دو مقدار l_i و r_i معرفی شده و بازه‌ی $[l_i, r_i]$ را مشخص می‌کند. برای تزئین درخت کریسمس، قصد داریم از هر رأس یک گوی شیشه‌ای آویزان کنیم که عددی روی آن نوشته شده باشد. این عدد برای رأس i باید در بازه‌ی $[l_i, r_i]$ قرار داشته باشد، یعنی $l_i \le a_i \le r_i$.

زیبایی درخت کریسمس بدین صورت تعریف می‌شود: برای هر یال (u, v) درخت، مقدار $|a_u - a_v|$ محاسبه می‌شود. زیبایی کل درخت برابر با مجموع این مقادیر برای تمام یال‌های درخت است.

اکنون یک درخت به شما داده می‌شود. وظیفه شما این است که به مناسبت نزدیک بودن کریسمس، بیشینه زیبایی ممکن این درخت را محاسبه کنید.

پاسخ :

برای حل این مسئله، از این نکته استفاده می‌کنیم که یک تخصیص بهینه برای a (اعداد نوشته‌شده روی گوی‌ها) وجود دارد که در آن برای هر رأس v، مقدار a_v باید یا برابر l_v باشد یا برابر r_v.

ابتدا فرض کنید یک تخصیص دلخواه برای a داریم. حال اگر برای رأس v، مقدار a_v بین l_v و r_v باشد (برابر هیچ یک نباشد)، می‌توانیم آن را طوری تغییر دهیم که زیبایی درخت بهبود یابد.

فرض کنید p تعداد رأس‌های u مجاور با v باشد که $a_u > a_v$. همچنین، q تعداد رأس‌های u مجاور با v باشد که $a_u < a_v$. حالات زیر را در نظر بگیرید:

- اگر $p > q$: در این حالت می‌توانیم مقدار a_v را به l_v کاهش دهیم و نتیجه بهتری بگیریم.
- اگر $p < q$: در این حالت می‌توانیم مقدار a_v را به r_v افزایش دهیم و نتیجه بهتری بگیریم.
- اگر $p = q$: در این حالت تغییر مقدار a_v به l_v یا r_v یا نتیجه را بهبود می‌دهد یا تغییری در زیبایی درخت ایجاد نمی‌کند.

بر اساس نتیجه فوق، می‌توانیم از برنامه‌ریزی پویا برای یافتن پاسخ استفاده کنیم. تعریف می‌کنیم:

$dp[v,0]$: حداکثر زیبایی زیردرخت v زمانی که $a_v = l_v$ باشد.

$dp[v,1]$: حداکثر زیبایی زیردرخت v زمانی که $a_v = r_v$ باشد.

مقدار $dp[v,j]$ بر اساس فرزندان v محاسبه می‌شود. برای هر یک از فرزندان v، مانند u، سهم u به $dp[v,j]$ اضافه می‌شود:

$$dp[v,0] += \max(dp[u,0] + |l_v - l_u|, dp[u,1] + |l_v - r_u|)$$

$$dp[v,1] += \max(dp[u,0] + |r_v - l_u|, dp[u,1] + |r_v - r_u|)$$

واضح است که پاسخ برابر است با:

$$\max(dp[v,0], dp[v,1])$$

پیچیدگی زمانی این راه‌حل $O(n)$ است که n تعداد رئوس درخت است.

۵. آزمون ماز　　　　۱۵ نمره

پروفسور مک‌گوناگل برای به چالش کشیدن هری پاتر، یک ماز طراحی کرده است که شامل n اتاق است. بین هر دو اتاق، دو راهرو (در دو جهت) وجود دارد. در هر یک از این راهروها یک روح قرار گرفته است که شکست دادن آن نیازمند درجه سختی مشخصی است. درجه سختی هر مسیر در ماز برابر است با مجموع درجه سختی ارواح موجود در راهروهای آن مسیر.

اما مشکل اینجاست که پیوز، روح بدعنق هاگوارتز، در هر مرحله یکی از اتاق‌ها و تمام راهروهای متصل به آن را محو می‌کند. پیش از هر بار که پیوز این کار را انجام می‌دهد، هری باید مجموع درجه سختی کم‌چالش‌ترین مسیر بین هر دو اتاق باقی‌مانده را در ماز محاسبه کند. کم‌چالش‌ترین مسیر می‌تواند از هر اتاق باقی‌مانده‌ای در آن مرحله عبور کند.

الگوریتمی طراحی کنید که هری بتواند این کار را در زمان $\mathcal{O}(n^3)$ انجام دهد. فرض کنید ترتیب محو کردن اتاق‌ها توسط پیوز از ابتدا مشخص است.

پاسخ :

الگوریتم فلوید–وارشال کم‌ترین فاصله بین هر دو رأس یک گراف را با عبور از فقط i رأس ابتدایی $(0 < i \leq n)$ در زمان $O(n^3)$ محاسبه می‌کند. در نتیجه کافی‌ست ترتیب محو شدن اتاق‌ها توسط پیوز را برعکس کرده و با همین ترتیب به رئوس گراف شماره بدهیم. سپس با اجرای الگوریتم فلوید–وارشال می‌توان کم‌چالش ترین مسیر بین هر دو اتاق را در هر مرحله به دست آورد.

۶. آلیس در تالگی وود ۱۵ نمره

آلیس در حال قدم زدن در تالگی‌وود (جنگل واندرلند) متوجه مسیرهای درخشانی می‌شود که قارچ‌های جنگل را به یکدیگر متصل می‌کنند. این مسیرها به‌گونه‌ای هستند که بین هر دو قارچ a و b دقیقاً یک مسیر یکتا وجود دارد. هر قارچ یک سطح خاصیت جادویی مشخص دارد که به‌صورت عددی روی آن حک شده است.

اما جادوی واقعی در مسیرهای بین قارچ‌ها نهفته است: مقدار جادوی یک مسیر بین دو قارچ a و b برابر با اختلاف بین بزرگ‌ترین و کوچک‌ترین سطح جادویی در میان قارچ‌های این مسیر (شامل خود a و b) است.

آلیس قصد دارد مقدار کل جادوی جنگل را محاسبه کند. به این صورت که برای هر جفت قارچ در جنگل، مقدار جادوی مسیر بین آن دو را محاسبه کرده و همه این مقادیر را با یکدیگر جمع کند.

اگر جنگل تالگی‌وود n قارچ داشته باشد، الگوریتمی طراحی کنید که آلیس بتواند این محاسبه را در زمان $\mathcal{O}(m \log n)$ انجام دهد.

پاسخ :

با توجه به این که بین هر دو قارچ دقیقاً یک مسیر یکتا وجود دارد، می‌توان قارچ‌ها را رئوس یک درخت در نظر گرفت. فرض کنیم سطح جادویی قارچ i برابر با a_i است. حال باید به ازای هر دو رأس این درخت، اختلاف بزرگ‌ترین و کوچک‌ترین a_i در مسیر بین این دو رأس را یافته و مجموع این مقادیر را محاسبه کنیم. برای این کار می‌توان ابتدا مجموع بزرگ‌ترین a_i ها را در تمام مسیرها محاسبه کرد، و سپس مجموع کوچک‌ترین a_i ها در تمام مسیرها را از پاسخ کم کرد.

برای یافتن مجموع بزرگ‌ترین a_i ها در تمام مسیرها به صورت زیر عمل می‌کنیم:

ابتدا به هر یال این درخت که بین رئوس i و j است مقدار $b_{ij} = max(a_i, a_j)$ را نسبت می‌دهیم. سپس یک ساختمان داده $Disjoint - set$ تشکیل می‌دهیم که هر یک از رئوس درخت یک مؤلفه در آن است. فرض کنیم مجموع بزرگ‌ترین a_i ها در تمام مسیرها برابر S است. حال یال‌های درخت را به ترتیب b_{ij} از کوچک به بزرگ به این $Disjoint - set$ اضافه می‌کنیم و بین هر دو مؤلفه که این یال به هم متصل می‌کند $union$ می‌گیریم. با اضافه کردن هر یال با وزن b_{ij} که دو مؤلفه با اندازه‌های x و y را به هم متصل می‌کند، xy مسیر جدید تشکیل می‌شود که بزرگ‌ترین a در آن مسیر برابر با b_{ij} است (به این دلیل که به ترتیب یال‌ها را اضافه می‌کنیم). در نتیجه با اضافه کردن این یال می‌توانیم به S مقدار $x.y.b_{ij}$ را اضافه کنیم. در نهایت توانستیم در زمان $O(m\log(n))$ (یک عملیات $union$ به ازای هر یال) مجموع بزرگ‌ترین a_i ها را در تمام مسیرها محاسبه کنیم.

مجموع کوچک‌ترین a_i ها را نیز در تمام مسیرها با همین روش محاسبه کرده و در نهایت این دو را از هم کم می‌کنیم تا پاسخ نهایی به دست بیاید.