به نام خدا

دانشگاه تهران، دانشکده مهندسی برق و کامپیوتر
تحلیل و طراحی الگوریتم‌ها

تمرین کتبی چهارم
موعد تحویل: دوشنبه ۱۹ اردیبهشت ۱۴۰۱، ساعت ۱۲:۰۰
طراح: ادیب رضایی adibrezaeish@gmail.com

---

۱. گراف $G(V,E)$ با راس هایی با وزن مثبت و یال هایی بدون وزن و راس s در ورودی به شما داده شده است. الگوریتمی با پیچیدگی $O(|E|+|V|log|V|)$ ارائه دهید که کم هزینه ترین مسیر از راس s به سایر راس هارا بدست آورد. درستی الگوریتم خود را نشان دهید. (۱۰ نمره)

پاسخ: از طریق کاهش دادن مسئله به دایکسترا آن را حل میکنیم. کافیست به ازای هر $v \in V$ و یال uv یک راس v' بین دو راس u و v اضافه کنیم به طوری که $uv'v$ مسیر جدید از u به v باشد و وزن یال $v'v$ معادل وزن راس v باشد. در صورتی که گراف بی جهت باشد باید دو یال جهت دار برای هر یال $uv \in E$ طبق الگوریتم بالا به وجود آید. همینطور اگر راس شروع s وزن دار بود در نهایت باید وزن آن را نیز در جمع نهایی مسیر مطلوب محاسبه کرد چرا که طبق الگوریتم ارائه شده آن را رد کردیم و محاسبه نکردیم. حال گراف وزن دار با وزن های مثبت و صفر روی یال های آن به وجود آمد. کافیست دایکسترا را از راس s اعمال کنیم. توجه شود که برای رسیدن به پیچیدگی $O(|E|+|V|log|V|)$ نیاز است از از صف اولویت فیبوناچی هیپ برای دایکسترا استفاده شود.

۲. درخت پوشای کمینه T را برای گراف وزن دار بی جهت $G=(V,E)$ در نظر بگیرید. میخواهیم m یال $E'=\{e_1,e_2,...,e_m\}$ را به G اضافه کنیم. برای سادگی فرض میکنیم یال ها وزن های متفاوتی دارند. الگوریتمی با $O(nm)$ ارائه دهید که درخت پوشای کمینه $G'=(V,E\cup E')$ را زمانی که درخت پوشای کمینه T برای G داده شده است پیدا کند. (۱۵ نمره)

پاسخ: سعی میکنیم ابتدا لم زیر را اثبات کنیم.

لم: درخت پوشای کمینه T را برای گراف وزن دار بی جهت $G=(V,E)$ در نظر بگیرید. میخواهیم یال $e=(u,v)$ را به G اضافه کنیم. الگوریتمی با $O(nm)$ ارائه دهید که درخت پوشای کمینه $G'=G+e$ را زمانی که درخت پوشای کمینه T برای G داده شده است پیدا کند.

درخت پوشای کمینه T در حال حاضر n-۱ یال دارد. با اضافه کردن یال $e=(u,v)$ با وزن w به گراف G دقیقا یک دور C در گراف T + e به وجود می آید. اگر وزن یال جدید اضافه شده کمتر از بیشترین وزن یال در دور C باشد آنگاه میتوان MST بهینه تر با جایگزین کردن یال جدید به جای یال قدیمی ساخت. در غیر اینصورت T همان MST بهینه باقی میماند. برای پیدا کردن یال با بیشترین وزن در دور C کافی است با اعمال DFS در درخت T از u به v یال با بیشترین وزن را بیابیم. اینکار با $O(n)$ قابل انجام است.

حال کافیست تعمیم یافته این لم را برای m یال استفاده کنیم. پیچیدگی نهایی الگوریتم از $O(nm)$ خواهد بود.

۳. حسام قصد مهاجرت به کشوری را دارد که شهر های آن با جاده های دو طرفه به یکدیگر راه دارند و از هر شهر به هر شهر دیگر حداقل یک مسیر میتوان پیدا کرد. او میخواهد خانه ای بخرد که بهترین دسترسی را به سایر شهر ها دارد. شهری بهترین دسترسی را دارد که ماکسیمم مسافت کوتاه ترین مسیر آن به سایر شهرها کمترین مقدار ممکن باشد. الگوریتم بهینه ای ارائه دهید که شهر مورد نظر حسام را برای او پیدا کند. پیچیدگی و درستی الگوریتم را نشان دهید. (۱۵ نمره)

پاسخ: شهر را یک گراف وزن دار همبند بی جهت در نظر میگیریم. برای رسیدن به خواسته سوال باید کمترین فاصله بین هر دو راس را محاسبه کنیم. میدانیم الگوریتم Floyd-Warshall الگوریتم بهینه ای برای یافتن کوتاه ترین مسیر بین هر دو راس یک گراف با پیچیدگی $O(v^3)$ است. حال در جدول کم هزینه ترین مسیر هر راس به راس های دیگر برای راس $v \in V$ راسی را پیدا میکنیم که هزینه رسیدن به آن ماکسیمم است. اینکار به سادگی از $O(v)$ قابل انجام است. به طور مشابه اینکار را برای تمام راس های $v \in V$ میتوان با مرتبه $O(v^2)$ انجام داد. کافی است با یک پیمایش $O(v)$ در نتیجه پیدا کردن پرهزینه ترین مسیر ها برای هر راس کمترین آن را پیدا کنیم. در کل پیدا کردن جواب مساله در $O(v^3)$ قابل انجام است.

۴. شهردار شهر s به تازگی میخواهد به شهر دیگری سفر کند. شهر ها با جاده های یک طرفه با یکدیگر ارتباط دارند. همچنین بعضی از جاده ها به تازگی آسفالت شده اند و او میخواهد برای بررسی وضعیت اینگونه جاده ها حداکثر از ۲ تای آنها در مسیر خود بگذرد. از طرفی او عجله دارد و میخواهد در سریع ترین زمان ممکن به شهر های دیگر دسترسی پیدا کند

(آ) . الگوریتمی ارائه دهید که کمترین مسافت بین شهر s تا همه شهر های دیگر محاسبه کند و شرط داده شده برای آن برقرار باشد. مرتبه زمانی را بر اساس تعداد شهرها و جاده های شهر محاسبه کنید و درستی آن را نشان دهید. (۱۰ نمره)
پاسخ: مسئله را گراف G در نظر میگیریم که یال هایی که آسفالت دارند با رنگ قرمز مشخص شده اند.

۱. دو کپی از گراف ایجاد میکنیم. حال سه گراف $G, G1, G2$ را داریم. ۲. یال های قرمز گراف $G2$ را حذف میکنیم. هر یال با رنگ قرمز در گراف $G1$ به گونه ای تغییر میدهیم تا به راس متناظرش در گراف $G2$ به جای گراف $G1$ وصل شود. همینکار را برای گراف G تکرار میکنیم تا یال ها با رنگ قرمز به راس های متناظرش به $G1$ متصل شود.

۳. حال همه مسیرهایی که ۲ یال با رنگ های قرمز دارند در یک سر آن در $G2$ می افتد. به طور مشابه همه مسیر هایی که یک یال قرمز دارند یک سر آن در $G1$ ختم میشود.

۴. از الگوریتم دایکسترا برای پیدا کردن کوتاه ترین مسیر از راس s به سایر راس های گراف های $G, G1, G2$ استفاده میکنیم.

۵. برای هر راس در G به کوتاه ترین مسیر هایی که به $G, G1, G2$ ختم میشود نگاه میکنیم و کوتاه ترین مسیر بین آنها را برای آن راس انتخاب میکنیم. در این صورت کم هزینه ترین مسیر هایی که از حداکثر دو یال قرمز عبور کردند محاسبه میشود.

چون ضریب ثابتی از یال و راس به مسئله اضافه شده است و از الگوریتم دایکسترا استفاده شده است پیچیدگی الگوریتم متناسب است با پیچیدگی الگوریتم دایکسترا و از مرتبه $O(|E| + |V|log|V|)$ است.

(ب) . شبه کدی ارائه دهید که کوتاه ترین مسیر از مبدا s را به سایر شهر ها پیدا میکند. (نیازی به در نظر گرفتن شرط گذشتن از جاده های آسفالت در شبه کد نیست) (۱۰ نمره)
پاسخ:

Algorithm 1 Dijkstra with path finding

```
function DIJKSTRA(Graph, V, source)
    sptSet ← Array of size V with false values
    parent[V] ← −1
    dist ← Array of size V with INF values
    dist[src] ← 0
    Q ← the set of all nodes in Graph
    while Q is not empty do
        u ← node in Qwith smallest dist[]
        remove u from Q
        for each neighbor v of u do
            alt ← dist[u] + dist between(u, v)
            if alt < dist[v] then
                dist[v] ← alt
                parent[v] ← u
            end if
        end for
    end while
    return parent[]
end function
```

Algorithm 2 Print Path

```
function PRINT PATH(parent, dst)
    if parent[dst] = −1 then
        return −1
    end if
    curr ← dst
    while curr ≠ src do
        push curr to path
        curr ← parent[curr]
    end while
    Print Reverse of path
end function
```

۵. در شهر گاتهام t دزد در مکان های $T = \{T_1, T_2, T_3, ..., T_t\}$ قرار دارند. شهر را با یک گراف ساده وزن دار بی جهت همبند بدون طوقه (حلقه یا loop) با n راس و m یال نمایش میدهیم. p ایستگاه پلیس در $P = \{P_1, P_2, P_3, ..., P_p\}$ نقطه شهر مستقر شده اند. سرعت فرار دزد و سرعت دنبال کردن پلیس را $V = 1$ در نظر بگیرید. در l نقطه $L = \{L_1, L_2, L_3, ..., L_l\}$ از شهر موتور ویژه وجود دارد که سرعت پلیس برای دستگیری دزد ها را ۲ برابر میکند و فقط یک بار در طول مسیر خود میتواند از آن استفاده کند. (توجه: پلیس میتواند در این نقطه ها از موتور ویژه استفاده نکند)

(آ) فرض کنید همه دزد ها به یکباره اصلاح شدند و میخواهند خود را معرفی کنند. الگوریتمی ارائه دهید که هر دزد بتواند خود را به نزدیک ترین ایستگاه پلیس معرفی کند. پیچیدگی الگوریتم را بنویسید. (۱۰ نمره)

پاسخ: کافی است یک $Multi\ Source\ dijkstra$ روی گراف حاصل از شهر گاتهام اعمال کنیم. به این صورت که یک راس s به گراف اضافه میکنیم و با یال هایی با وزن صفر به تمامی source ها که در اینجا دزد ها هستند وصل میکنیم و در گراف حاصل شده از راس جدید s الگوریتم دایکسترا را اعمال میکنیم. به این صورت تمامی source ها با مقدار اولیه صفر در صف قرار میگیرند و فاصله نزدیک ترین ایستگاه پلیس برای هرکدام از آنها طبق الگوریتم دایکسترا بدست می آید. پیچیدگی الگوریتم متناسب با پیچیدگی الگوریتم دایکسترا است و از مرتبه $O(|E| + |V|log|V|)$ میباشد.

(ب) فرض کنید دزد در نقطه T_1 قصد فرار دارد و میخواهد خود را به نقطه D برساند. در این شرایط پلیس ها از ایستگاه های پلیس خارج شده و سعی در دستگیری دزد دارند. الگوریتمی ارائه دهید که کوتاه ترین زمانی که دزد میتواند فرار کند را مستقل از اینکه پلیس چه مسیری را برای دستگیری او انتخاب میکند پیدا کند. پیچیدگی الگوریتم را بنویسید. (۱۰ نمره)

نکته: پلیس ممکن است زودتر یا همزمان به نقطه ای از شهر که دزد میرسد برسد. در جفت این حالات پلیس میتواند دزد را دستگیر کند اگر دزد نتواند با توجه به شرایط فرار کند در خروجی $Can't\ escape$ را چاپ کنید.

پاسخ: $Multi\ Source\ Dijkstra$ را دو بار اعمال میکنیم. یکبار t ایستگاه پلیس را به عنوان source ها در نظر میگیریم و یکبار موتور ها را به عنوان source در نظر میگیریم و الگوریتم را اجرا میکنیم. در این صورت کوتاه ترین زمان رسیدن پلیس ها به تمام نقاط شهر با موتور و بی موتور را میتوانیم بدست آوریم.

در مرحله بعد دایکسترا را روی دزد T_1 اعمال میکنیم و کوتاه ترین مسیر رسیدن دزد به هر نقطه علاوه بر مقصد را بدست می آوریم و چک میکنیم آیا در هر نقطه پلیس زودتر از دزد به آن نقطه رسیده یا نه. اگر پلیس میتوانست زودتر برسد پس پلیس میتواند دزد را دستگیر کند. در غیر اینصورت کوتاه ترین مسیری را انتخاب میکنیم که دزد سریع تر نسبت به پلیس در خانه های آن حاضر میشود.

(این الگوریتم در حالت کلی وقتی $V_{police} \neq V_{thief}$ است بیان شده است. در حالت ساده تر فعلی یعنی $V_{police} = V_{thief}$ میتوان با بررسی اینکه چه کسی زودتر به مقصد میرسد هم به جواب رسید چون پلیس اگر بتواند در طول مسیر دزد را دستگیر کند حتما زودتر از او به مقصد میتواند برسد.)

پیچیدگی الگوریتم متناسب با دایکسترا است پس از مرتبه $O(|E| + |V|log|V|)$ میباشد.

۶. در نزدیکی محله سروش n روستا وجود دارد که هنوز هیچکدام از آنها برق ندارند. استاندار تصمیم گرفته که بین روستا ها سیم کشی کند که بتواند برق آنها را تامین کند. (استاندار مکان روستا ها و فاصله راه بین آنها را میداند)

(آ) الگوریتم بهینه ای ارائه دهید تا استاندار با کمترین هزینه (کمترین طول سیم کشی) بین روستا ها برق را توزیع کند. پیچیدگی الگوریتم را ذکر کنید. (۵ نمره)

پاسخ: منظور از درخت پوشای مینیمم (برای گراف همبند وزن دار) درختی است که بین درخت‌های پوشای آن گراف، مجموع وزن یال‌های آن، کمترین مقدار ممکن باشد که در مسئله فوق مطلوب است. برای به دست آوردن درخت پوشای کمینه یک گراف متصل می‌توان از الگوریتم‌های متفاوتی استفاده نمود الگوریتم های کروسکال و پریم دو الگوریتم بهینه برای بدست آوردن درخت پوشای کمینه با مرتبه زمانی $O(ElogV)$ هستند.

(ب) استاندار نگران این است که طول بزرگترین سیمی که نیاز است بین دو روستا کشیده شود آنقدری بلند شود که نیاز به برق فشار قوی داشته باشد. ثابت کنید با الگوریتم ارائه شده در قسمت (آ) طول بلندترین سیم کشیده شده بین دو روستا همواره کمینه است. (۱۵ نمره)

ثابت میکنیم هر MST یک $MBST(Minimum\ Bottleneck\ Spanning\ Tree)$ است. (MBST درخت پوشایی از گراف G است که بزرگترین یال آن در بین همه درخت های پوشا G کمینه است.)

برهان خلف: فرض کنید T یک درخت پوشای کمینه است. و همچنین فرض کنید یال (u, v) وزن بیشتری نسبت به وزن ماکسیمم یال MBST دارد. حال $V_1 \subseteq V$ را در نظر بگیرید که بدون اینکه از v بگذرند از u در درخت T قابل دستیابی هستند. به طور قرینه $V_2 \in V$ را تعریف میکنیم. حال برشی را در نظر بگیرید که V_1 و V_2 را از هم جدا میکند. طبق تعریف MST تنها یالی که میتوان به این برش اضافه کرد که درخت حاصل MST بماند یال با وزن مینیمم در این برش است پس میدانیم هیچ یالی با وزن کمتر از $w(u, v)$ در این برش وجود ندارد.

از طرفی فرض کردیم که یک MBST با وزنی کمتر از $w(u, v)$ داریم. این خلاف فرض است چون MBST خود یک درخت پوشاست و یالی در برش در نظر گرفته شده ندارد. پس فرض خلف باطل و حکم ثابت است.