شهزاد ممیز 810100272

Subject:
Date

تکلیف 7 الگوریتم (دستی)

$w_e \xrightarrow{\text{کاهش}} w'_e$

e

سوال ①

در این سوال می‌خواهیم کوتاه‌ترین مسیر بین راس‌ها را حساب کنیم.
در حالتی که وزن یال e را از w_e به w'_e کاهش پیدا کند حل
سوال به این صورت است که اگر این کار در مسیر کوتاه‌ترین مسیر بین ۲ راس
باشد، مقدار و اندازه shortest path کمتر میشه. سپس یک عدد را update
می‌کنیم. اگر نبود هم، احتمال داره که این w کاهش یافته باشه و مسیر کوتاه‌ترین
مسیر shortest path رو تغییر بده.

در این حالت کوتاه‌ترین path از راس روی روی این یال، و shortest path
از w ؛ این یال، راس دوم را هم حل با توجه به این یال و تعداد [illegible]
منعم را حساب می‌کنیم.

<u>time complexity</u>: چون عملاً باید جفت راس‌ها را مجدد بررسی قرار دهیم
پس order زمانی ما از $O(n^2)$ است.

```
for (i=0 ; i<N ; i++)
    for (j=0 ; j<N ; j++)
        distance[i][j] = min(distance[i][j], distance[i][x] + we' +
        distance[y][j], distance[i][y] + we' + distance[x][j]);
```

↓

در این رابطه، منظم ما هست در ۳ حالت را برای distance[i][j]
قرار می‌دهیم و یال گفته شده را x—y در نظر می‌گیریم.

سوال ۲ -

تبدیل از A به B ← ۲ (مثلاً)

Arbitrage : یک تعداد شیء از A داشته باشیم. پس از تعدادی تبدیل به A بیشتر برسیم.

<u>output</u> : الگوریتمی که بگوید Arbitrage وجود دارد یا خیر. <u>input</u> : از A ، B ، C ، ... / نرخ تبدیل ...

در صورتی که گراف ما، دوری داشته باشد به طوری که حاصل ضرب یال‌های آن از 1 بزرگتر شود

یعنی با تبدیل کردن یال در آن دور، در نهایت تعدادی که داریم بیشتر از <u>1x</u> اولیه داشته است، و سود بی‌حسابی است!

<u>و در دریافت نکته!</u> (می‌توانیم در صورتی که نرخ تبدیل a و b با نرخ تبدیل b و a متفاوت بود گراف را جهت‌دار در نظر بگیریم.

$a \rightarrow b$ یا $b \rightarrow a$ یا $\overset{a\qquad b}{\text{———}}$ در صورتی که نرخ تبدیل یکی بود.

ضرب به جمع تبدیل شود پس از مقادیر یال‌ها لگاریتم می‌گیریم. اگه $x.y.z.t > 1 \Leftarrow$

$0 < \log x + \log y + \log z + \log t$. اگه در دور وزن هر یال را $(w_e' = -\log w_e)$

منفی آن در نظر بگیریم وزن‌ها تبدیل کنیم، یافتن دور، منفی می‌شود $(-\log x - \log y - \log z - \log t < 0$

مثلاً) می‌توانیم از الگوریتم Bellman Ford نیز استفاده کنیم و اگه حداکثر و عدم نه دیدیم،

برای check وجود نکردن دور باید یک بار دیگر برای یال‌ها بروید، محاسبات بلمن فورد را

اجرا کنیم اگه مقدار یک رأس تغییر کند این یعنی این است که دور منفی وجود داشته پس Arbitrage

رخ داده است و else (در غیر این صورت) : وجود ندارد. برای یال ۰ هم که یک حالت خاص

محسوب می‌شود به جای لگاریتم [illegible] $+\infty$ را قرار می‌دهیم.

این الگوریتم درست است زیرا Bellman Ford به تعداد لازم انجام می‌شود اگه هیچ رأسی relax نشود برای گراف‌ها حاصل جمع $-\log w_e$ ، مثبت است در کل دور پس در order

$O(E * V^2)$ ، time complexity داریم.

سوال ۳

گراف G جهت دار است

بیشترین وزن یال ها را میخواهیم کمینه باشد.

s → · → · ... → t

بین shortest path ها.

```
Dijkstra(s, t, n, edge, adjList):
    A = [(0, s)]
    p[S] = [[S]]            ← مسیر
    distance[S] = 0
    maxE [-∞] x n           ← بزرگترین یال
    while (A):              ← تا زمانی که آرایه A ، null نیست
        index, node = A.pop()
        for (n', w in adjList[n]:
            if distance[n'] < distance[n] + w : continue
            elif distance[n'] > distance[n] + w:
                distance[n'] = distance[n] + w
                p[n'] = []
                A.push(distance[n'], n')
                maxE[n'] = -∞
            for path in p[n]:
                p[n'].append(path + [n'])
                maxE[n'] = max(maxE[n], edge[n][n'])
    min of max = min(maxE[t])
    index = maxE(t).index(minOfMax)
    return p[t][index];
```

وقتی که در هر shortestpath در Dijkstra وزن یال با بیشترین وزن مسیر رو ذخیره می‌کنیم. در آخر با داشتن همه shortest path ها. یالی که از بقیه یال های مسیر کمتر است را index اش را پیدا می‌کنیم که ببینیم min برای کدام بوده است. و آن مسیر را به عنوان output چاپ می‌کنیم.

دلیل صحیح بودن این است که Dijkstra، shortest path را از s به t می‌یابد. حال می‌توانیم هر مسیری که در طول مسیر، بزرگترین یال آن مسیر رو ذخیره کنیم. در نهایت بین همه shortest path ها بزرگترین یال هر مسیر save شده اند. را چک می‌کنیم و min را برمی‌داریم. اندیس این یال در آرایه را پیدا می‌کنیم.

time Complexity : $\boxed{O((V+E)\log V)}$

برای loop

سوال ۴

```
D (s, e[][], numberNode) :
    visited = [f] * numberNode        ← false کشف
    P[s] = -1                         ← پدر
    nodes = [inf] * numberNode
    nodes[s] = 0
    minheap.add[s]
    while (minheap) :
        u = minheap.pop();
        for v in neighbors of u :
            if (Node[v] > Nodes[u] + e[u][v]) :
                parent[v] = u
                Nodes[v] = Nodes[u] + e[u][v]      true
        visited[u] = T
    return P;
```

```
pathPrint ( s, d, p )      p → parent
    if (p[d] == -1)
        return (-1);
    while true :
        if (d == s) : break
        .add(d);
    print(anspath);
```

Subject:
Date

شهردار شهر ... شهروندی لیستی خواهد کند. وضعیت ۲ مسیر را می‌خواهند بسنجد

① : shortest path را باید محاسبه کند.

از ۳ گراف شامل استفاده کنیم. اگر ارا با تعمیه یکی شبیه شهر G شروع می‌کنیم.

single source می‌باشد پس باید از الگوریتم Dikstra استفاده کنیم

تعداری

با جاده‌هایی که ~~...~~ آسفالت شده‌اند آنها را علامتدار می‌کنیم. حداکثر ۲ تا از آنجا باید عبور کنند.

حالا G_1 و G_2 را ۳ کوپی وصل ... به ازای هر جاده تازه آسفالت شده یال‌های علامتدار G_2 را حذف کنیم. و این مقصد یال علامتدار G_1 را عوض می‌کنیم که به راس متناظر دیگراف متصل شود. به شیوه‌ای این‌ها را به هم وصل می‌کنیم. که تناظر برقرار بماند. حال دقیقاً همین الگوریتم را برای گراف G و G_1 انجام می‌دهیم. اگر مثلاً ترتیب G_1, G_2 را داشتیم

G G_1 داریم. زیرا حداکثر ۲ تا آسفالت تازه نیاز داریم.

اگر از راس ... در گراف شروع کنیم اگر خط که G_1 بماند، هیچ یال علامت نخورده که جاده که G_1 باشد ← ۱ یال علامتدار دیدیم، راه جاده در G_2 باشد 2 یال علامتدار دیده‌ام. به این صورت شرط حداکثر هم رعایت شده است.

① شهر پس path ها Dikstra می‌زنیم. تا کوتاه‌ترین path که بقیه رئوس را برگرداند. حال کوتاه‌ترین تا شهر دیگر را داریم. این یال‌ها را جهت‌دار رسم می‌کنیم. اندازه G

کمی از G داریم. رئوس ۳ برابر شده‌اند. و یال‌های گراف تقریباً می‌توان گفت که 3 برابر شده‌اند.

$O(V \log V) + O(E) = O(E + V \log V)$

$O(3E) = O(E)$

$O(3V \log 3V) = O(c V \log V) = O(V \log V)$

time complexity

را از کیفیت بالا محاسبه کرده‌ام.

صفحه قبل

در ادامه ۷ صفحه کد را نوشته‌ام!

Subject:
Date

تکلیف درسی ۷ . طراحی الگوریتم

شهرزاد محمدی - ۸۱۰۱۰۰۲۷۲

سوال ۵.

خانه علی ← تقاطع A (راس مبدا) $u[i]$ $v[i]$ یال جهت دار

بیمارستان ← B (راس مقصد)

خیابان ام ← تقاطع $u[i]$ را به $v[i]$ وصل می کند ، زمان عبور $t[i]$

چراغ راهنمایی هر مدت $r[i]$ قرمز ← صبر کنیم تا سبز شود.

$g[i]$ سبز ← رد می شویم و هزینه ای نمی دهیم

حال باید الگوریتمی ارائه کنیم که shortest path را حساب کند و با توجه به الگوریتم هایی که داریم روش Dijkstra که single source shortest path را حساب می کند به دست می آوریم.

← الگوریتم بدین صورت است که ما خواستیم از یک تقاطع (راس) عبور کنیم باید سبز بودن آن را چک کنیم: حالت ① این است که سبز باشد چراغ ← در این صورت $t[i]$ را اضافه می کنیم و به تقاطع بعدی می رویم. پس با توجه به اینکه چراغ های تقاطع سبز یا قرمز هستند، همه زمان از ابتدای الگوریتم طی شده در انتهای هر چراغ چه رنگی است و طول مدت قرمز بودن باید هزینه زمان صبر شده را حساب کنیم. حالت ② چراغ قرمز باشد ← مدت زمان قرمز ماندن چراغ را جمع می کنیم، وزن یال را نیز به نقطه می دهیم و به تقاطع بعدی می رسیم.

همه مسیرهای موجود از A به B را بررسی می کنیم و min را برمی داریم.

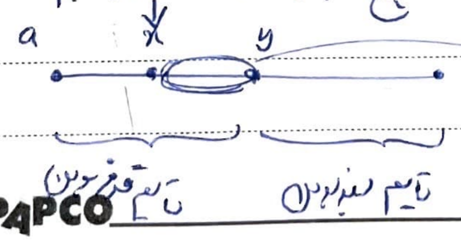
راس شروع: A روش (relax) کردن را شروع می کنیم. که از هر همسایه A، راس x با توجه به کمترین یال و چراغ، فاصله از A به x بود، مقدار کمتر را relax می کنیم و در انتها هم ~~...~~ بوده.

min heap

Fibonachi heap

روش های پیاده سازی dijkstra

در این روش اگر به تقاطع x به تقاطع y (تقاطع بعدی) برسیم، باید زمان تبدیل شدن را با زمان جدید جمع کنیم تا لحظه ای که چراغ سبز شود. انتظار هستیم.



$(y-x)$ → زمان لازم برای اینکه چراغ سبز شود وقتی در نقطه x هستیم.