پاسخ امتحان چهارم - گراف

طراحی الگوریتم - بهار ۱۴۰۰

سوال اول:

فرض کنید یال بین دو راس u و v اضافه شده است. اگر این یال درون mst نباشد پس mst همان مقدار قبلی باقی می ماند و اگر جزو mst باشد باید یکی از یال های مسیر بین u و v حذف شود. هر کدام از آن‌ها را می توان حذف کرد. (زیرا $n - 1$ یال باقی می ماند که دور ندارد پس درخت است) پس سعی می کنیم یال با بیشترین وزن را حذف کنیم.

می دانیم در درخت بین هر دو راس دقیقا یک مسیر وجود دارد. مسیر بین این دو راس را با الگوریتم dfs پیدا می کنیم. (از v الگوریتم dfs ا اجرا می کنیم تا هنگامی که به u برسیم. و مسیر راس فعلی تا ریشه را در یک آرایه ذخیره می کنیم) از این مسیر بیشترین یال را پیدا می کنیم. اگر این وزن این یال از یال جدید بیشتر بود یال جدید را اضافه می‌کنیم و این یال را حذف می کنیم.

مرتبه زمانی dfs و کل الگوریتم برابر $O(n)$ خواهد بود.

سوال دوم:

الف) همه یال های گراف را در ۱- ضرب می کنیم. سپس یکی از الگوریتم های درخت پوشای کمینه (کروسکال یا پریم) را اجرا می کنیم. در نهایت کافیست جواب نهایی و وزن یال های درخت را در ۱- ضرب کنیم. به درخت پوشای بیشینه خواهیم رسید.

ب) از برهان خلف استفاده می کنیم. فرض می کنیم جواب بهینه برابر y باشد و جواب الگوریتم ما برابر x که $x < y$. حال یال های جواب بهینه را منفی می کنیم و چون $-x <- y$ پس $-y$ باید جواب کروسکال ما می بود که تناقض است. (توجه کنید که از منفی یا مثبت بودن اعداد استفاده نکردیم) پس $x = y$ و الگوریتم اثبات شده است.

سوال سوم:

این سوال با دو ایده کروسکال و دایکسترا قابل حل است. در این راه حل ایده کروسکال آن شرح می دهیم و به ایده دایکسترا نیز اشاره خواهیم کرد.

الف) طبق الگوریتم کروسکال درخت پوشای کمینه گراف را بدست می آوریم. بیشترین وزن یال در مسیر s به هر راس را توسط پیمایش dfs محاسبه می کنیم. (برای هر راس این مقدار برابر بیشینه مقدار پدرش و وزن یال به پدرش است. مقدار هر راس در آرگومان dfs به فرزندش پاس داده می شود) این مقدار برابر کمترین پهنای باند، که خواسته سوال است، خواهد بود.

مرتبه زمانی $O(m.lgn)$ برای کروسکال است و $O(n)$ برای پیمایش dfs روی درخت پس هزینه زمانی کل الگوریتم برابر $O(n + m.lgn)$ است.

در راه حلی دیگر می توانیم از ایده دایکسترا استفاده کنیم و فقط به جای فاصله پهنای کمینه هر راس را محاسبه می کنیم. وقتی یک راس به مجموعه s دایکسترا اضافه می شود باید مجاور هایش را آپدیت کند. اگر راس u اضافه شده باشد و همسایه راس v بیرون از مجموعه باشد؛ مقدار

$max(w[uv],\ width[u])$ یکی از پهنای موجود برای v است. پس آن را با مقدار فعلی $width[v]$ مینیمم می گیریم. شبه کد آن به این صورت است:

آ)

```
S = ϕ
For v in V:
    width[v] = ∞
width[S] = 0
For v in V:
    MinHeap.insert(v, width[v])   → با توجه به اینکه به جز s ، پهنای بقیه راس ها ∞ است ، با هم برابراست. این عملیات در O(n) انجام می شود.

while ! MinHeap.empty():
    u = MinHeap.popMin()   → O(log(n)) → n بار تکرار می شود و از مرتبه O(n logn) خواهد بود.
    S = S + {u}
    for all v in neighbors of u and in V-S:
        newWidth = max(w[uv], width[u])
        if width[v] > newWidth:
            width[v] = newWidth
            MinHeap.decreaseKey(v, width[v])
        } O(log(n)) → برای هر یال ما این عملیات را انجام می دهیم پس در O(m log n) انجام می شود.

⟹ T(n) = O(n logn + m logn)
```

ب) از برهان خلف استفاده می کنیم. پس s به راس v مسیری با پهنای کمتر دارد. فرض کنید پنهای بدست آمده در الگوریتم ما x باشد. یعنی طبق الگوریتم کروسکال قبل از اضافه شدن یال با وزن x راس s و v در مولفه جدا بوده اند که با این یال متصل شده اند. فرض کنید جواب بهینه پهنا برابر y باشد. پس $y < x$. پس با یال های با وزن کوچکتر مساوی y در گراف مسیری بین s و v وجود

داشته است که این با اضافه شدن یال با وزن x و در دو مولفه بودن s و v در تناقض است. پس $x = y$.

اثبات راه حل با ایده دایکسترا سخت تر است. باید مانند اثبات الگوریتم دایکسترا که در کلاس دیدیم از استقرا استفاده کنیم و همان فرض ها را بکنیم.