ساختمان داده‌ها (۲۲۸۲۲)

مدرس: حسین بومری

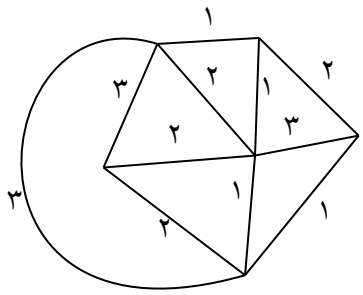
[بهار ۹۹]

جلسه ۲۸: درخت پوشای کمینه و مجوعه های مجزا

نگارنده: مبین معدنی

در این جلسه قصد داریم که الگوریتم های پیدا کردن درخت پوشا ی در درخت های وزن دار را پیدا کنیم

حال در این جلسه با استفاده از گراف روبه‌رو الگوریتم هایی که در جلو تر گفته میشود را بررسی میکنیم



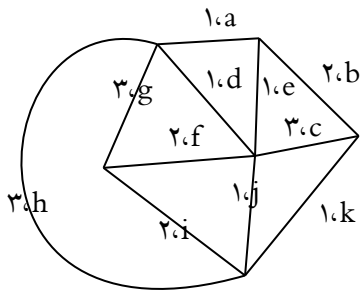
۱ ایده ی اول

یکی از ایده هایی که سر کلاس مطرح شد ، این بود که هر سری دورهای درون گراف اصلی را پیدا کنیم و سپس راسی را که دارای بیشترین وزن است را حذف کنیم ، که برای این کار باید به تعداد E مرتبه الگوریتم DFS را صدا بزنیم.

۲ ایده ی دوم

۱.۲ بیان الگوریتم

ایده ی دوم الگوریتم $kruskal$ است که در آن ابتدا گراف را خالی در نظر بگیریم و سپس بر اساس ترتیب مرتب شده ی کوچک به بزرگ یال ها ، از یال کوچک تر اضافه میکنیم و سپس هرگاه یالی را که موجب ایجاد دور میشود را اضافه نمیکنیم و سراغ یال بعدی میرویم. برای مثال برای گراف ذکر شده ابتدا یال های e, d, j, k را اضافه میکنیم ولی یال a را به علت اینکه موجب تشکیل دور میشود را در نظر نیمگیریم



۲.۲ تحلیل زمانی

مرتبه ی زمانی الگوریتم ذکر شده برابر است با :

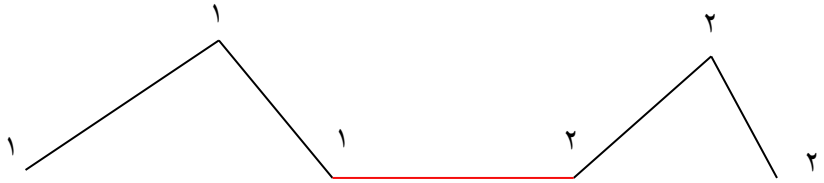
مرتب سازی یال ها

مرتب سازی یال ها : $e \log e$

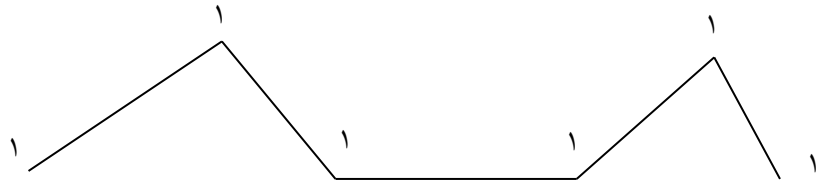
بررسی دورهای گراف

به ازای هر یال چک کنیم که دور ایجاد میشود یا خیر : $O(e)$ و برای تمامی راس های میشود $eO(e) = e^{۲}$

راه دیگر شماره گذاری مولفه های هم بندی در گراف تشکیل شده در هر مرحله است، و وقتی که دو سر یالی که قرار است اضافه شود از دو شماره ی همبندی متفاوت بود یال را اضافه میکنیم و تمامی شماره های مولفه ی همبندی جدید را هم نام میکنیم.

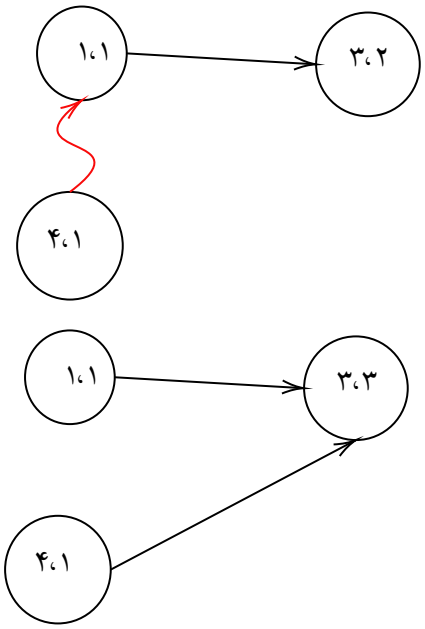


همانند شکل کشیده شده ، به علت اینکه دو طرف یال قرمز دارای شماره های متفاوت است یال را میتوانیم اضافه کنیم و سپس بعد از اضافه شدن تمامی شماره ها را یکی میکنیم و گراف زیر بدست میاید:



پس حالا تحلیل الگوریتم گفته شده برابر است با : به ازای اضافه کردن هر یال $O(v)$ کار انجام خواهیم داد ، پس در کل به اندازه ی $vO(v)$ کار انجام میدهیم که میشود $v^{۲}$ و به اندازه ی eبار چک میکنیم که در یک مولفه ی همبندی هستند یا خیر

راه دیگر استفاده از داده ساختار $unionfind$ است. فرض میکنیم که اعضای مجموعه ای که داریم ، هر کدام دارای یک شماره هستند و داده ساختار ما دارای دو دستور $union$ و $find$ است . در دستور $find$ یک موجود گرفته میشود و شماره ی راس آن داده میشود. دستور $union$ دو عنصور میگیرد و آن ها را در یک مجموعه قرار میدهد و درون خود دستور $find$را دوبار صدا میزند. اگر شماره ها یکسان بودند کاری انجام نمیدهد ولی در غیر اینصورت بررسی میکند کدام تعداد کدام مجموعه کوچک تر است. و مجموعه ای را که کمتر است را به بیشتر وصل میکند. و مقدار تعداد بیشتر را آپدیت میکند. برای مثال: ابتدای دستوری $union$ را روی ۳ ,۱ صدا میزنیم و در انتها ۱ به ۳ وصل میشود و طول ۳ از ۱ به ۲ افزایش میابد. حال دوباره دستور $union$ را روی ۱۴ صدا میزنیم. حال دستور $find(۱)$ به ما عدد۳ را برمیگرداند و دستور $find(۴)$ عدد ۴ را، حال با توجه به اینکه طول دسته ی ۳ برابر ۲ میباشد و بلند تر از دسته ی ۴ است. دسته ی ۴ را به دسته ی ۳ وصل میکند. و طول دسته ی ۳ یکی اضافه میشود.



با توجه به نوع ساختار و نوع اتصال دسته ها به یکدیگر ارتفاع درختی که از اتصال اعضا به یکذیگر به دست میاید حداکثر $\log n$ میشود.(زیرا که همیشه اتصال دو عضو منجر به اتصال عضو کوچک تر به راس عضو بزرگ تر میشود)(دقیقه ۵۸ جلسه ۲۸ کلاس) بنابراین دستور $find$ در $\log n$انجام میشود و دستور $union$ در $\log n$ انجام میشود. بنابر این با استفاده از داده ساختار ذکر شده میتوانیم یالی را که قرار است به درخت کمینه اضافه کنیم را ، دو راس آن را در $find$ صدا بزنیم و اگر در یک مولفه نبوند یال را اضافه میکنیم و در آخر دستور $union$ را صدا میزنیم. پس در کل $e \log v$ میشود.

جمع بندی تحلیل زمانی

: در کل سه مدل بررسی دورهای گراف داشتیم و مرتبه های زمانی آنها به ترتیب برای کل الگوریتم پیدا کردن درخت پوشای کمینه برابر زیر میشود.

استفاده از DFS برای پیدا کردن دور ها :

$$= O(e \log e) + o(e^۲)$$

استفاده از نام گذاری مولفه های همبندی:

$$= O(e \log e) + o(v^۲ + e)$$

استفاده از داده ساختار $unionfind$:

$$= O(e \log e) + o(e \log v)$$