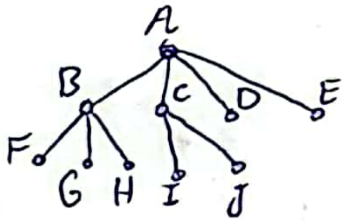
پایان ترم DS   اعمال محمدی   ۹۹۱۰۲۲۰۷

سؤال ۱) درخت DFS یک گراف را داریم ، اول حالتی که یک رأس برشی دارد را چک می کنیم.

اگر فقط یک رأس برشی داشته باشیم ، آن رأس ، رأس A است زیرا در این درخت ،



رأس های E و D نمی توانند به هیچ کدام از فرزندان A ، یال داشته باشند ؛ پس

A تنها رأس برشی گراف است . (طبق گفته خود سؤال)

حالا چون A تنها رأس برشی ما است پس به تمام رئوس وصل است ؛ اگر مثلاً به رأس J

وصل نباشد و یال AJ نداشته باشیم ، رأس C هم رأس برشی خواهد بود ، پس A به تمام رئوس یال دارد و

درجه رأس A دقیقاً ۹ می باشد.   حداقل و حداکثر ۹

حالا اگر دو تا رأس برشی داشته باشیم ، با توجه به اینکه رئوس E و D فقط به A وصل هستند و اگر نباشند ، خودشان

به رئوس دیگر وصل نیستند طبق درخت DFS ، پس قطعاً رأس A ، یکی از رئوس برشی است.

حالا رأس دیگر برشی یا رأس B است یا رأس C ⇐

حالت اول : A و B برشی باشند پس برای حداقل درجه ، A حتماً باید به E و D و C و B و J و I ؛

یال داشته باشد پس حداقل درجه آن ۶ است ؛ برای حداکثر درجه ، A می تواند به ۶ تا رأس قبلی و ۲ از فرزندهای

B نیز یال داشته باشد ؛ پس حداکثر درجه آن ۸ است.

حالت دوم : A و C برشی باشند پس برای حداقل درجه ، A حتماً باید به E و D و C و B و H و G و F ؛

یال داشته باشد پس حداقل درجه آن ۷ است ؛ برای حداکثر درجه ، A می تواند به یکی از فرزندهای C نیز یال

داشته باشد به همراه ۷ تا رأس قبلی ؛ پس حداکثر آن ۸ است.

پس وقتی ۲ رأس برشی داشته باشیم ، حداقل درجه A ، ۶ است و حداکثر درجه A ، ۸ است.

سؤال ۲ پایان ترم PS — ایمان فهمی — ۹۹۱۰۲۲۰۵۷

یالهایی که ۰ هستند دو رأس کنار آن ها باید یا با هم سوییچ شوند یا اصلا سوییچ نشوند و برای یالهایی که ۱ هستند نیز حتما یکی از رئوس کناری باید سوییچ شود.

حالا پس در گراف هایی که دور داریم که تعداد ۱ ها در آن فرد است، مشکل داریم و هیچ گاه تمام یالها ۰ نمی شوند چون در یک دور، با زدن هر سوییچ، زوجیت مجموع تمام یال ها ثابت می ماند پس وقتی تعداد فرد ۱ داریم، هیچوقت به ۰ که زوجیت زوج دارد نمی رسیم.

حالا برای شرایطی که دور با تعداد فرد ۱ داریم، الگوریتم ارائه می دهیم.

اگر یالی صفر باشد، می توانیم هر دو رأس آن را یک رأس در نظر بگیریم پس تا زمانی که یال ۱ فقط داریم، این کار را انجام می دهیم و همه دو رأس یال های ۰ را یک رأس در نظر می گیریم.

حالا برای اینکه بتوانیم همه یال ها را ۰ کنیم، باید گرافمان دوبخشی باشد چون دو رأس هر یال ۱ در زوجیت سوییچ کردن تفاوت دارند و فقط در حالتی که گراف دوبخشی است و یک بخش آن را فرد بار سوییچ می کنیم و یک بخش آن را زوج بار، می توان تمام یال ها را ۰ کرد.

حالا برای فهمیدن اینکه گرافمان بعد از یکی کردن رأس های یال ۰، دوبخشی است یا نه، dfs می زنیم و به هر رأس گراف، یک mark با عنوان One و به همسایه های آن mark با عنوان Zero می دهیم و ... (ادامه)

حالا تک تک یال ها را چک می کنیم که دو سر آن متفاوت باشد و یکی One و یکی Zero باشد؛ اگر غیر از این بود، آنگاه گرافمان دوبخشی نیست.

پیچیدگی زمانی DFS : $O(n+m)$ اگر دوبخشی بود می توانیم یالها را ۰ کنیم با چک کردن یالها

برای صفر کردن یال های ۰ نیز از DSU استفاده می کنیم اینطوری که رئوس کناری یال های صفر را در یک مولفه قرار می دهیم و گراف را با مولفه های ایجادشده و یالهای ۱ می سازیم.

پیچیدگی زمانی : $O(m+m\log m) \simeq O(m\log m)$

پایان ترم DS   ایمان فهمی   ۹۹۱۰۲۲۰۱

سؤال ۳) گراف ساده بدون وزن G را داریم. هرکدام از یال ها به یکی سه رنگ قرمز یا آبی یا زرد رنگ شده.
از BFS با کمی تغییر با توجه به رنگ ها استفاده می کنیم. در نتیجه برای الگوریتمان داریم:
۱- نقطه شروعمان را وارد صف می کنیم. ۲- در هر مرحله عنصر جلوی صف را گرفته و مشاهده، سپس از صف خارج می کنیم. ۳- فرزندان و همسایه های آن رأس را که هنوز مشاهده نشده اند را به انتهای صف اضافه می کنیم.
حالا برای پیاده سازی این روش یک pointer به یک Linked list شامل رنگ های قرمز و آبی و زرد می زنیم و آن ها را mark می کنیم. برای مثال ۰ ← آبی ۱ ← زرد ۲ ← قرمز (برای ذخیره در صف)
حالا BFS را شروع می کنیم فقط نکته این است، هر رأس را توسط یک رنگ مشاهده می کنیم و برای هر رأس در نتیجه، یک رنگ قائل می شویم.
حالا تفاوت این است که در مراحل بعدی ۲ و ۳، عنصر صف را خارج می کنیم و همسایه های غیرهمرنگ و مشاهده نشده را به انتهای صف اضافه می کنیم.
فرق این حالت با BFS معمولی این است که مشاهده شدن هر رأس، به رنگ یالی که از طریق آن به رأس می رسیم ربط دارد و برای مثال ممکن است یک رأس را سه بار با رنگ های متفاوت مشاهده کنیم و مشکلی هم ندارد چون رنگ متناظر را همواره وارد صف می کنیم. ← جمع dist های پدرش و یالی که از پدرش به آن رسیده
حالا با این الگوریتم، در آخر برای هر رأس حداکثر ۳ تا dist و ۳ تا پدر داریم که اگر از رأس ۱ شروع کنیم، در نهایت ماکسیمم ۳ تا فاصله مختلف برای رأس آخر خواهیم داشت که مینیمم آن را به عنوان خروجی می خواهیم. و از طریق آرایه Father هم به مسیر دسترسی خواهیم داشت.

پیچیدگی زمانی: دسترسی به رنگ یال $O(1)$ است و کل الگوریتم هم مشابه BFS است که ماکسیمم ۳ برابر می شود ⟸ پیاده سازی با ماتریس مجاورت: $O(n^2)$ پیاده سازی با لیست مجاورت: $O(n+m)$

درستی الگوریتم: چون G ساده و بدون وزن و بدون جهت است، اجرای الگوریتم مشکلی ندارد.
در صورت همبند نبودن، اگر رأس ابتدایی و انتهایی در مؤلفه همبندی یکسان نباشند، جوابی ندارد سؤال

پایان ترم DS   ایمان محمدی   99102207

سؤال ۴) گراف وزن دار وجهت دار G شامل n رأس و m یال داریم

دقیقا گراف G' با رأس و یال و وزن های روی یال مشابه تعریف می کنیم بدون هیچ کم یا اضافه کردنی

حالا می خواهیم برای الگوریتمان، به ازای هر یال بین دو رأس G، یک یال بین یک راس از G و یک رأس

از G' با $\frac{1}{10}$ وزن داشته باشیم، برای شبیه سازی بهتر مثالی می زنم ⟸ مثلاً به ازای یال از A به B در G

یک یال با $\frac{1}{10}$ وزن از A به B' داشته باشیم

شبیه سازی شکلی ⟸

حالا از الگوریتم دایسترا استفاده می کنیم و در گراف کلی ای که تعریف کردیم، از رأس اول در گراف G، به رأس آخر و مقصد در گراف G' دایسترا می زنیم و جواب مورد نظر را بدست می آوریم.

جواب مورد نظر حتماً جواب درست است زیرا تمام یال های بین G و G'، از G به G' هستند و امکان برگشت به G از G' نداریم، و دقیقاً طبق شرایط سؤال

پیچیدگی زمانی: الگوریتم مورد استفاده ما دایسترا است پس پیچیدگی زمانی الگوریتمان مشابه دایسترا است؛ صرفا تعداد رئوس با توجه به اضافه شدن G'، دو برابر شده و تعداد یال ها نیز ۳ برابر حالت عادی است با توجه به اضافه شدن یال های G' و یال های G به G'

البته تغییر تعداد یال و رأس ها (۳ و ۲ برابر شدن آن ها) تأثیری در پیچیدگی زمانی ندارد و داریم:

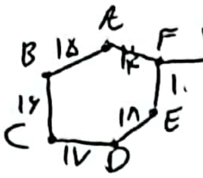
با پیاده دایسترا با هرم فیبوناچی داریم $O(n\log n+m)$

درستی الگوریتم: با توجه به عدم امکان برگشت از G' به G، فقط یک بار از یال با وزن $\frac{1}{10}$ عبور می کنیم و همچنین تمامی وزن یال ها مثبت است.

پایان ترم DS   ایمان مهمی   ۹۹۱۰۲۲۰۷

سؤال ۵) گزاره اول نادرست است و با مثال نقض این را نشان می دهیم:

گراف هایی تک راس با یک یال وصل شده اند به مجموعه ی همبندی، مثال نقض اند ⟸



مثلا در گراف G، GFEDC و GFABCD و ... درخت پوشای بیش کمینه هستند ولی فقط GFEDCB درخت پوشای کمینه است.

ب) این گزاره صحیح است و با توجه به الگوریتم کراسکال برای ساختن درخت پوشای کمینه می دانیم که اول تمام یال‌ها به صورت صعودی از بالا sort می شوند و سپس تک تک اضافه می شوند. تا زمانی که دوری اضافه نشده است و کل راس‌ها تکمیل شده باشند.

حالا از برهان خلف استفاده می کنیم و فرض می کنیم درخت پوشای کمینه G وجود دارد که پوشای بیش کمینه نیست، پس درخت G ای وجود دارد که بیش کمینه است که بزرگترین یال آن از بزرگترین یال G کوچک‌تر است. (بزرگترین یال G: A و بزرگترین یال G: B)

در کراسکال، زودتر از A به B رسیدیم پس در درخت G، یا A تشکیل دوری داده یا ~~[illegible]~~ هنوز به (n-1) یال نرسیده بودیم. پس درختی که بزرگترین یال آن B است، پوشا نیست پس هر درخت پوشای کمینه یک درخت پوشای بیش کمینه نیز هست.