

دانشگاه صنعتی خواجه نصیرالدین طوسی

به نام خداوند جان و خرد

نام و نام خانوادگی: محمد حسین کریمی

شماره دانشجویی: 9731833

پروژه گرگ و گوسفند

تابستان 1400

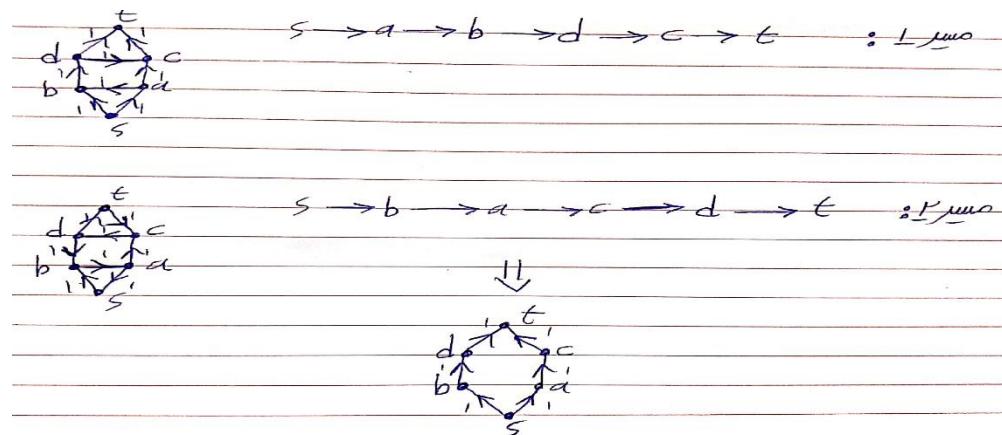
توضیح کلی

از الگوریتم max flow استفاده کردیم. اگر در گراف گفته شده 2 مسیر مجزا بین دو یال s و t که راس s برابر t برای sink و گنجایش تمامی یال هارا برابر با 1 در نظر می گیریم حال در مسئله گفته شده ایا s , t بین s, t بزرگ تر مساوی 2 است یا خیر اگر هست مسیر ها را چاپ می کند.

توضیح برنامه

از الگوریتم فولد فالکرسون استفاده می کنیم به این صورت که در هر مرحله یک مسیر افزایشی بین s و t پیدا کرده و این جریان را عبور می دهد. در الگوریتم گفته شده به ازای هر یالی که میزان جریانی از آن رد می شود یالی با جهت برعکس بین 2 راس قرار می دهیم زیرا ممکن است راه بهینه این باشد که این جریان اصلاً عبور نکند. (ظرفیت یال برعکس را برابر با جریان عبوری از آن قرار می دهیم. چون در ابتدا تمامی ظرفیت ها اعداد صحیح هستند و با هربار پیدا شدن مسیر افزایشی و عبور جریان آن اندازه \min ظرفیت یالهای موجود در مسیر جریان عبور می کند نتیجه می شود که هر مسیر افزایشی یک میزان صحیح جریان عبور می دهد و حداقل برابر 1 واحد است.

پس یعنی هر بار الگوریتم گفته شده یک واحد به جریان عبوری اضافه می کند حال ما چون می خواهیم ببینیم که ایا جریان بیشینه بزرگ تر مساوی 2 است یا نه کافی است 2 بار الگوریتم را اجرا کنیم. برای پیدا کردن مسیر افزایشی از bfs استفاده می کنیم حال اگر bfs مسیری بین s و t پیدا کند که به جریان یک واحد اضافه می کنیم و به مرحله بعد می رویم در غیر این صورت الگوریتم متوقف می شود چون نمی توان به شار افزود. حال برای پیدا کردن 2 مسیر در bfs هرگاه از راسی به راس دیگر می رویم راس پدر آن را ذخیره می کنیم تا بتوانیم بعد از رسیدن به t مسیر را بدست آوریم. حال دو مسیر مختلف بین s, t داریم اما ممکن است در مسیر 1 یالی داشته باشیم و در مسیر 2 از برعکس آن استفاده کرده باشیم مثال:



حال اگر یال هایی که خودشان در یک مسیر و معکوس شان در مسیر دیگر استفاده شود را حذف می کنیم که عبارت است از: $(a,b),(b,a),(d,c),(c,d)$

یال های باقی مانده تشکیل 2 مسیر مجزا یالی بین 2 راس s و t را می دهند. حذف یال های گفته شده به این خاطر است که اصلا از آن جریانی رد نمی شود (جریان ها یک دیگر را خنثی می کنند).

نکته مهم این است که در هنگام وارد کردن رئوس باید دقت شود که دور ایجاد نشود. بطور مثال:

```
Enter N:  
3  
Enter M:  
3  
Enter edges:  
1 2  
2 3  
1 3  
Enter S:  
1  
Enter T:  
3  
The edges of two paths are:  
0
```

bfs متده است

ابتدا یک آرایه برای راس ها در نظر می گیریم که مقدار اولیه انها `false` است یعنی ان راس ملاقات نشده است. سپس با استفاده از `linked list` یک صف ایجاد می کنیم و تمام رئوس را در آن اضافه می کنیم. و همه مقادیر انها را `true` می کنیم و راس والد را برابر 1 قرار می دهیم. سپس تا اتمام صف با استفاده از bfs چک می کنیم اگر مسیری بین مبدا و مقصد پیدا کردیم که الگوریتم متوقف شده و راس والد را `true` می کنیم. اگر به مسیری نرسیدیم ان را `false` می کنیم.

hastwodisjointpath متده است

ابتدا یک گراف باقی مانده ایجاد می کنیم و پر می کنیم ان را با ظرفیت یالها. در این گراف ظرفیت باقی مانده یالها را نمایش می دهد که اگر صفر باشد یعنی ان یال وجود ندارد. سپس با استفاده از bfs به دنبال کم ترین ظرفیت می رویم یا به عبارتی دیگر پیدا کردن بیشترین شار. در اولین for بین ظرفیت یال باقی مانده و شار فعلی کمترین را انتخاب می کنیم و در نهایت ظرفیت باقی مانده یالها را اپدیت کرده و یال را بر عکس می کنیم.

پیچیدگی زمانی

پیچیدگی زمانی الگوریتم برابر با اجرای 2 بار bfs در یک گراف جهت دار است که تعداد یال هایش حداقل 2 برابر تعداد یال های اولیه اش است. در bfs پس الگوریتم گفته شده در $O(n+m)$ به جواب می رسد.

اجرای برنامه

<pre>Enter N: 5 Enter M: 7 Enter edges: 1 2 1 3 2 3 3 5 2 4 4 5 5 5 Enter S: 1 Enter T: 5 The edges of two paths are: 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 Process finished with exit code 0</pre>	<pre>Enter N: 6 Enter M: 8 Enter edges: 1 2 1 3 2 3 3 5 2 4 4 6 5 6 5 4 Enter S: 1 Enter T: 6 The edges of two paths are: 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 Process finished with exit code 0</pre>
--	--