به نام خدا

علی غفاری ثابت - ۹۸۲۳۹۰۳

توضیح کد:

۱- ابتدا کلاس Graph را برای ذخیره گراف و انجام عملیات بر روی آن ایجاد می کنیم. این کلاس تعداد رئوس (v) و یال ها را در یک گراف دو بعدی (edges) نگه می دارد. در این کلاس توابع add_edge و remove_edge به ترتیب یک یال را به گراف اضافه و حذف می کند. توابع get_v و is_edge به ترتیب تعداد رئوس گراف و موجودیت یک یال در گراف را بر می گردانند. توابع get_path و remove_path به ترتیب یک مسیر از راس مبدا به راس مقصد را به روش پیمایش DFS بر می گردانند و یک مسیر را حذف می کند. حذف مسیر به این شکل است که مسیر را مقصد به مبدا وارون می کند.

۲- تابع edge_disjoint_paths وظیفه پیدا کردن دو مسیر مستقل از راس مبدا به راس مقصد را دارد. این تابع به این شکل عمل می کند که ابتدا یک مسیر از s به t پیدا می کند و آن را از گراف حذف می کند و سپس یک مسیر دیگر از s به t پیدا می کند. (اگر هر یک از این دو مسیر پیدا نشود یعنی هیچ دو مسیر مستقلی وجود ندارد.) سپس یک گراف جدید می سازد و این دو مسیر را به آن اضافه می کند و یال هایی به صورت معکوس روی یکدیگر قرار می گرند را حذف می کند. (هر دو یال) سپس دوباره دو مسیر از مبدا به مقصد پیدا کرده و آن را چاپ می کند. دلیل معکوس کردن مسیر در تابع remove_path در کلاس Graph این است که امکان اصلاح مسیر قبلی را داشته باشیم.

۳- در بدنه اصلی کد ورودی ها (تعداد راس و یال، یال ها و راس مبدا و مقصد) را از کاربر دریافت کرده، گراف را می سازیم و ورودی ها را به تابع edge_disjoint_paths می دهیم.

پیچیدگی زمانی:

پیچیدگی پیمایش گراف به شکل DFS (تابع get_path) برابر $O(E + V)$ که در گراف شلوغ $E = V^2$. همچنین پیچیدگی زمانی تابع remove_path برابر طول مسیر است که حداکثر می تواند V باشد. در تابع edge_disjoint_paths چهار بار تابع get_path را صدا می زنیم. همچنین دو حلقه به طول های مسیر اول و مسیر دوم داریم که حداکثر می تواند V باشد. بنابراین پیچیدگی زمانی این برنامه همان $O(V^2)$ است.

سوالات:

آیا مسئله به های min cut و max flow مرتبط است؟ این مسئله در حقیقت همان مسئله max flow است که در آن وزن تمام یال های گراف برابر یک است. اگر شار بیشینه بیشتر از 1 باشد مسئله جواب دارد.

تصویر محیط اجرا:

```
ali@Ali:~/Desktop/University/3992/algorithm_design/projects/project4/individual-project-4-aghs8055/wolves_and_sheep$ python3 wolves_and_sheep.
py
Enter count of nodes and edges:(n m) 8 12
Enter edges in each line:(a b)
0 1
0 2
0 3
3 6
2 6
1 2
2 3
6 5
5 4
4 7
5 7
6 7
Enter s, t:(s t) 0 7
Path 1 is: 0 1 2 3 6 5 4 7
Path 2 is: 0 2 6 7
ali@Ali:~/Desktop/University/3992/algorithm_design/projects/project4/individual-project-4-aghs8055/wolves_and_sheep$ python3 wolves_and_sheep.
py
Enter count of nodes and edges:(n m) 4 5
Enter edges in each line:(a b)
0 1
1 2
2 3
0 2
1 3
Enter s, t:(s t) 0 3
Path 1 is: 0 1 3
Path 2 is: 0 2 3
```