ریحانه حیدری 9731483

پروژه گرگ و گوسفند

شرح مسئله:

یک گرگ و یک گوسفند داریم که می خواهیم در یک گراف جهت دار و بدون دور از یک راس مثلا s به راس دیگری همچون t بروند، برای اینکه از خورده شدن گوسفند توسط گرگ جلوگیری کنیم، مسیر این دو نباید هیچ یال مشترکی داشته باشد.

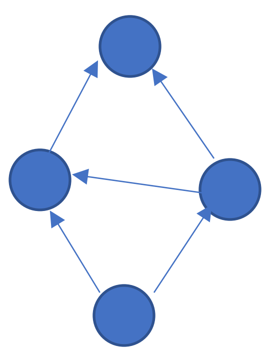
در این پروژه می خواهیم یک الگوریتم با پیچیدگی چند جمله ای ارائه دهیم که دو مسیر با این ویژگی (جدا از یال - edge-disjoint) را پیدا کند، در صورتی که چنین مسیری وجود داشت (امنیت گوسفند تضمین شود)، این مسیر ها را چاپ کند.

روش حل:

ابتدا با استفاده از dfs یک مسیر از s به t پیدا می کنیم و سپس تمامی خانه های آرایه ی mark را صفر می کنیم.

حال اگر بخواهیم دوباره از s به t dfs بزنیم اما از یالهای مسیر قبل استفاده نکنیم (آنها را از گراف حذف کرده باشیم) ممکن است دو مسیر مجزای یالی در گراف وجود داشته باشد اما مسیر اولی که در dfs پیدا شد از یال های هر مسیر در خود داشته باشدکه در نتیجه با حذف آنها دیگر مسیر از s به t باقی نمانده باشد.

مثال:



اگر راس پایین را s و راس بالا را t و راس راست و چپ را 1 و 2 درنظر بگیریم.

مسیر اول:

(s,1)(1,2)(2,t)

دو مسیر مجزا یالی:

(s,1)(1,t)

(s,2)(2,t)

برای حل این مشکل بعد از پیدا کردن بعد از پیدا کردن مسیر اول از s به t بازای هریال (u,v) در این مسیر یک یال (v,u)به گراف اضافه می کنیم(چون از ماتریس مجاورت استفاده می کنیم همانند این است که

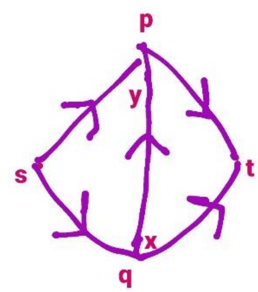
G[u][v]-=1 کرده و یال G [v][u]+= 1 بکنیم.)

حال اگر فرض کنیم دو مسیر مجزا مجزا یالی در گراف وجود داردد و مسیر اول شامل یالهای هر دو مسیر است با انجام عملیات قبل برای هر قسمتی که در مسیر پیدا شده اول است اتفاق زیر می افتد:

L مسیر پیدا شده اول

Pمسیر مجزا یالی اول

Q مسیر یالی مجزا



برای مثال در شکا بالا داریم:

L=s->x->y->t

با معکوس کردن یال های مسیر l و در نظرر گرفتن اولین نقطه ی برخورد مسیر l با p نتیجه می شود که همچنان می شود از s به y رسید و چون یالهای معکوس شده می توان از y به x رسید چون در مسیر l یالهای از x به t وجود ندارند در ادامه نیز می توان از xبه t رفت پس مسیر

s->y->x->t

وجود دارد که اگر یالهایی که در هر دو مسیر هستند اما جهت آنها متفاوت است را حذف کنیم به دو مسیر مورد نظر p,q می رسیم.

برای اثبات کلی می توان برروی تعداد تلاقی های مسیر l و در دو مسیر p,q استقرا زد و در گام های استقرا مثل مثال قبل یکی از تلاقی هارا کم کرد و پایه نیز همان مثال قبل خواهد بود.

در توضیحات قبل نیز می توان نشان داد که یالهایی که در هر دو مسیر l و مسیر پیدا شده ی بعدی

s->y->x->t دوبار تکرار شده اند(یکبار خود یال یکبار معکوسش)جز یالهای p,q نیستند زیرا نقاط تلاقی بین دو مسیر را به هم وصل میکند حال بعد از حذف یالهای گفته شده به دو مسیر مجزا یالی می رسیم که آن ها را در یک گراف نگه میداریم حال با زدن dfs بر روی این گراف و ذخیره مسیر طی شده در یک stack پس مسیر را داریم و آن را به عنوان خروجی برنامه چاپ میکنیم.

رابطه مسئله با شار بیشینه:

این مسئله وقتی یک مسیر جدید از s به t پیدا میکنیم مثل این می ماند که می توانیم یک جریان را از s به t بگذرانیم.

در این مسئله دو مسیر مجزا با این شرایط می یابیم که معادل دو شار یا جریان در مسئله ی max flow هست.(برای مثال گنجایش هر یال را میتوانیم 1 درنظر بگیریم) و میدانیم که خود مسئله ی max flow نیز معادل min cut می باشد.

تحلیل پیچیدگی زمانی:

برابر با دوبار اجرای dfs است که از O(n+m) است.

می شود که در بعضی جاها لیست mark را برابر صفر کنیم که آن هم می تواند تا n انجام شود.

مرتبه زمانی الگوریتم=O(n+m)

عملکرد برنامه:

```
DisjointPath
"C:\Program Files\Java\jdk1.8.0_281\bin
Enter number of V:
4
Enter number of E:
4
Enter Edges:
1 4
1 2
2 3
3 4
S:
1
T:
4
2 separate Edge path:
Path:
1 2 3 4
Path:
1 4

Process finished with exit code 0
```

```
DisjointPath
"C:\Program Files\Java\jdk1.8.0_281\bin\java.
Enter number of V:
4
Enter number of E:
5
Enter Edges:
1 2
1 3
2 3
2 4
3 4
S:
1
T:
4
2 separate Edge path:
Path:
1 2 4
Path:
1 3 4

Process finished with exit code 0
```