

آتش نشان ها

ریحانه حیدری 9731483

هر ایستگاه آتش نشانی را به عنوان یک راس از یک گراف در نظر می گیریم و هدف پیدا کردن تمام مسیرها از مبدا به مقصد است.

یک set به نام vertices داریم که همه ی رئوس را به طور یونیک در آن داریم.

یک map داریم که به عنوان لیست مجاورت گراف از آن استفاده می نماییم.

Graph: برای ایجاد گراف (شبیه سازی ایستگاه ها به گراف)

addEdge: لیست مجاورت رئوس داده شده را به طور دوطرفه اضافه می کنیم (چون گراف جهت دار نیست)

Visited: یک لیست Boolean که به تعداد راس هایمان است و هر راسی را دیدیم visited آن را

true می کنیم.

Dque: یک صف دوطرفه است.

getAllPaths: اگر مبدا و مقصدی که به آن داده شده یکی باشد مبدا و مقصد را به یک گراف اضافه می کنیم و برمی گرداند.

به طور کلی کارهای اولیه را انجام می دهد یک آرایه از visited می سازد یک Dque می سازد و سپس getAllpathsDFS را صدا می زند.

getAllpathsDFS: visited مبدا را true می کند در path که جواب مسئله هست آن را اضافه می نماید.

حرکت می کنیم اگر به مقصد رسیده باشیم path را به result اضافه می کند و ادامه می دهد.

اگر به مقصد نرسیده باشیم لیست مجاورت را چک میکنیم اگر لیست مجاورت با این مبدا مقصدی داشت به آن وارد می شویم و بازاء تمام مقصدهایش چک می کنیم که آیا ما تا به حال از این مقصد رد شده ایم یا نه (باتوجه به visited)

در این متد همه یحالت ها چک می شود اگر درست بود ادامه میدهد و در صورتی که مسیری وجود نداشت از بازگشت به عقب استفاده می کنیم و به یک حالت دیگر می رویم.(قسمتی که در خود getAllpathsDFS مجدد getAllpathsDFS فراخوانی شده است)

اگر از آن رد نشده ایم `getAllpaths` را این بار از آن مبدا به `target` مان ادامه می دهیم یعنی در `path` این مبدا را وارد کرده ایم و برای ادامه از تمام راس هایی که راس مبدا مان به آنها راه دارد دوباره این متد را فراخوانی کرده ایم این روند را ادامه می دهد تا تمام `path` های موجود از مبدا به مقصد رابدست آورد.

`main`: تا زمانی که به 0 0 برسیم ورودی می گیرد متد `getAllpaths` را بازاء آن گراف فراخوانی می کند که یک لیست به ما برمی گرداند که در آن همه ی مسیرها به صورت لیستی از `int` ها موجود است ک آنها را به لیستی از رشته ها تبدیل می کنیم و چاپ می نماییم.

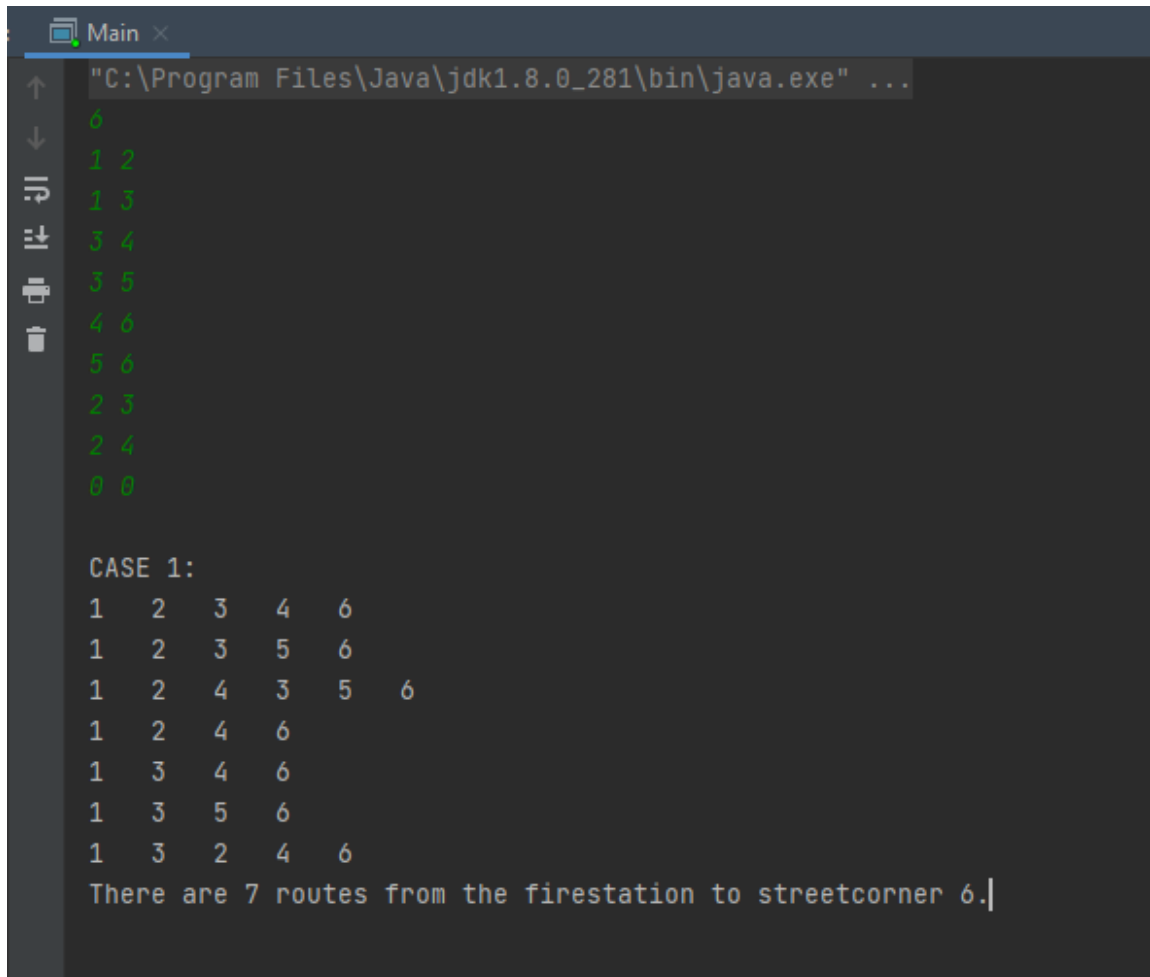
پیچیدگی زمانی:

$$O(n^n)$$

که البته باتوجه به اینکه این مسئله با روش بازگشت به عقب حل شده است برای محاسبه ی پیچیدگی زمانی از روش مونت کارلو میتوانیم استفاده کنیم.

$n + 1$ (رئوسی که با راس مبدا یال مشترک دارند) * (رئوسی که علاوه بر داشتن یال مشترک با راس مورد نظر ممکن است به مسیر مورد نظر ما ختم شوند) + ...

TEST CASES:



The screenshot shows a Java IDE window titled "Main x". The command prompt shows the execution of a Java program: `"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...`. The program's output consists of two parts. The first part is a list of 12 pairs of numbers, each pair on a new line: `0 6`, `1 2`, `1 3`, `3 4`, `3 5`, `4 6`, `5 6`, `2 3`, `2 4`, and `0 0`. The second part is labeled "CASE 1:" and shows a list of 7 routes from a firestation to streetcorner 6. Each route is a sequence of numbers separated by spaces: `1 2 3 4 6`, `1 2 3 5 6`, `1 2 4 3 5 6`, `1 2 4 6`, `1 3 4 6`, `1 3 5 6`, and `1 3 2 4 6`. The final line of output states: `There are 7 routes from the firestation to streetcorner 6.`

```

Main x
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
0 6
1 2
1 3
3 4
3 5
4 6
5 6
2 3
2 4
0 0

CASE 1:
1 2 3 4 6
1 2 3 5 6
1 2 4 3 5 6
1 2 4 6
1 3 4 6
1 3 5 6
1 3 2 4 6
There are 7 routes from the firestation to streetcorner 6.
```

```

Main x
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" ...
4
2 3
3 4
5 1
1 6
7 8
8 9
2 5
5 7
3 1
1 8
4 6
6 9
0 0

CASE 2:
1 5 2 3 4
1 5 7 8 9 6 4
1 6 4
1 6 9 8 7 5 2 3 4
1 3 2 5 7 8 9 6 4
1 3 4
1 8 7 5 2 3 4
1 8 9 6 4
There are 8 routes from the firestation to streetcorner 4.
```