Bonus: BDZ

locked

Problem | Submissions | Leaderboard | Discussions

Submissions: 50
Max Score: 100
Difficulty: Medium

Rate This Challenge:

More

Дойде време за празници и всички започнаха да си тръгват от София чрез любимото БДЖ. За да са по-ефективни, БДЖ решили да направят всички влакове да са еднопосочни и да няма циклични линии - така ще могат влаковете да са съсредоточени върху изкарването на хората от София. Градовете, които имат спирки, са N на брой и са номерирани с числата от 1 до N, а БДЖ има M на брой линии между тях. В един такъв момент човек си задава логичния въпрос - "По колко начина мога да стигна от град A до град B?" Напишете програма, която по дадени начален и краен град, намира броят на възможните пътища, започвайки от началния и завършвайки в крайния град.

Input Format

На първия ред на входа се въвеждат **N** и **M** - броят градове и броят линии. Следват **M** реда с по две числа - еднопосочна линия на БДЖ между два града. На последния ред има две числа - началния и крайния град.

Constraints

1 ≤ N ≤ 100 000

1 ≤ M ≤ 200 000

1 ≤ start, end ≤ N

Гарантирано е, че няма цикъл.

Output Format

Изведете едно число - броят различни пътища от началния до крайния град. Тъй като това число може да е прекалено голямо, изведете остатъка му по модул 1000000007.

Sample Input 0

```
4 6
2 1
4 3
1 3
4 1
2 3
2 4
2 3
```

Sample Output 0

```
4
```

Explanation 0

Различните пътища от 2 до 3 са:

2->3

2->4->3

2->4->1->3

2->1->3

Current Buffer (saved locally, editable) C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <list>
using namespace std;
  int result=0;
struct Node{
  list<int> neighbours;
  bool hasNeighbour(int index) {
    for (auto i : neighbours) {
      if (i == index) {
        return true;
      }
    }
    return false;
  }

  void addNeighbour(int index){
    neighbours.push_back(index);
  }
};
struct Graph{
  vector<Node> nodes;
  Graph(int nodeCount=0){
    nodes.resize(nodeCount);
  }
  void connect(int from,int to){
    if(!nodes[from].hasNeighbour(to)){
      nodes[from].addNeighbour(to);
    }
  }
  void countPaths(int start,int end,vector<bool> visited){
    visited[start] = true;
    if (start == end){
        result++;
      }
    else
    {
         for(auto it:nodes[start].neighbours){
            if (!visited[it]){
                countPaths(it, end, visited);
            }
         }
    }
   visited[start] = false;
}

};

int main() {
  int townsNumber;
  scanf("%d",&townsNumber);
  int roadsNumber;
  scanf("%d",&roadsNumber);
  int from,to;
  Graph g(townsNumber);
  for(int i=0;i<roadsNumber;i++){
      scanf("%d",&from);
      scanf("%d",&to);
    g.connect(from-1,to-1);
  }
  int start;
  int end;
  scanf("%d",&start);
 scanf("%d",&end);
  vector<bool> visited;
  visited.resize(townsNumber,false);


  g.countPaths(start-1,end-1,visited);
  printf("%d",result);
    return 0;
}
```

Line: 1 Col: 1

Upload Code as File Test against custom input Run Code Submit Code