Blocked Roads

locked

Problem | Submissions | Leaderboard | Discussions

Submissions: 68
Max Score: 100
Difficulty: Medium

Rate This Challenge:

More

N-те населени места на един район, номерирани от 1 до N, са свързани с пътища. След като паднал сняг, останали проходими само M пътни отсечки, всяка от които свързва две от населените места. Изнервени граждани, които пътуват от едно населено място до друго – по работа, или просто така, атакуват многократно операторите на телефон 112 с въпроси от вида „Може ли да се стигне от селището X до селището Y в момента?“ От своя страна, почистващите служби успяват от време на време да почистят по някой от затрупаните пътища и също звънят на оператора на 112 с информация от рода: „Пътната отсечка от селището X до селището Y е проходима.“

Напишете програма, която да помага на операторите да отговарят бързо на въпросите на гражданите за актуалното състояние на пътищата.

Input Format

На първия ред на стандартния вход ще бъдат зададени числата N и M. На всеки от следващите M реда – по два номера на град, свързани с проходима пътна отсечка.

Следва ред с броя Q на обажданията – както от граждани, така и от пътните служби и Q реда със съдържанието на обажданията – вид на обаждането и двата номера на населените места, за които се отнася съответното обаждане. Ако обаждането е въпрос на гражданин – кодът е 1, а ако е от пътните служби – кодът е 2.

Constraints

N ≤ 1000

M ≤ 2N

Q ≤ 1 000 000

Output Format

На стандартния изход програмата трябва да изведе битов низ с толкова знака, колкото са въпросите на граждани за проходимост на пътната мрежа, като знакът 0 означава че отговорът на поредния въпрос е „Невъзможно е да се стигне!“, а знакът 1 – „Възможно е!“.

Sample Input 0

```
9 8
1 2
3 4
5 6
7 8
9 5
7 2
8 2
6 9
6
1 1 8
1 6 2
2 7 1
1 4 7
2 2 3
1 4 7
```

Sample Output 0

```
1001
```

Current Buffer (saved locally, editable) C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
struct Graph{
  vector<vector<int>> nodes;
  Graph(int nodeCount){
    nodes.resize(nodeCount);
    for(int i=0;i<nodeCount;i++){
      nodes[i].resize(nodeCount);
      for(int j=0;j<nodeCount;j++){
        nodes[i][j]=0;
      }
    }
  }
  void connect(int from,int to){
    nodes[from][to]=1;
  }
};

int main() {
    int nodeCount;
    int edgeCount;
  cin>>nodeCount;
  cin>>edgeCount;
  vector<int> components;
  components.resize(nodeCount);
  for(int i=0;i<nodeCount;i++){
    components[i]=i;
  }
  int from,to;
  for(int i=0;i<edgeCount;i++){
    cin>>from>>to;
    if(components[from-1]!=components[to-1]){
      int oldComp=components[from-1];
      int newComp=components[to-1];
      for(int j=0;j<nodeCount;j++){
        if(components[j]==oldComp){
          components[j]=newComp;
        }
      }
    }
  }

  int operations;
  cin>>operations;

  for(int i=0;i<operations;i++){
    int opType;
    int from;
    int to;
    cin>>opType>>from>>to;
    if(opType==1){
      if(components[from-1]==components[to-1]){
        cout<<1;
      }
      else{
        cout<<0;
      }
    }
    else if(opType==2){
      if(components[from-1]!=components[to-1]){
      int oldComp=components[from-1];
      int newComp=components[to-1];
      for(int j=0;j<nodeCount;j++){
        if(components[j]==oldComp){
          components[j]=newComp;
        }
      }
    }
    }
  }

    return 0;
}

```

Line: 1 Col: 1

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