Пътна проверка

locked

Problem | Submissions | Leaderboard | Discussions

Даден ви е претеглен неориентиран граф G с N на брой върха, M на брой ребра. От вас се иска да намерите дали дадена поредица от върхове X_1,X_2,...,X_n образува път в графа и ако да, да изведете дължината на пътя.

Input Format

От първия ред на стандартния вход се въвеждат N и M. На следващите M реда се въвеждат по 3 числа - x, y, w. Всеки такъв ред определя ребро с дължина w, свързващо възлите x и y. На следващия ред се въвежда K. На последния ред се въвеждат K на брой числа - X_1, X_2 и т.н. до X_K

Constraints

0 < N <= 1000

0 < M <= 10000

0 < K <= 1000

0 <= a, b, X_i < N

0 < w <= 100

Output Format

Ако дадената последователност от върхове образува път, то се извежда дължината му, а в противен случай -1.

Sample Input 0

```
4 5
0 1 1
0 2 2
2 3 5
2 1 3
0 3 1
3
0 1 2
```

Sample Output 0

```
4
```

Submissions: 109
Max Score: 50
Difficulty: Easy

Rate This Challenge:

More

Current Buffer (saved locally, editable) C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <list>
using namespace std;
struct Pair{
  int index;
  int weight;
};
struct Node{
  list<Pair> neighbours;
  bool hasNeighbour(int index){
    for(auto it:neighbours){
      if(it.index==index){
        return true;
      }
    }
    return false;
  }
  int lengthToNeighbour(int index){
    for(auto it:neighbours){
      if(it.index==index){
        return it.weight;
      }
    }
    return -1;
  }
  void addNeighbour(int index,int weight){
    neighbours.push_back({index,weight});
  }
};
struct Graph{
  vector<Node> nodes;
  Graph(int nodeCount=0){
    nodes.resize(nodeCount);
  }
  void connect(int from,int to,int weight){
    nodes[from].addNeighbour(to,weight);
    nodes[to].addNeighbour(from,weight);
  }
};

int main() {
    int nodeCount;
  cin>>nodeCount;
  int edgeCount;
  cin>>edgeCount;
  int from,to,weight;
  Graph g(nodeCount);
  for(int i=0;i<edgeCount;i++){
    cin>>from>>to>>weight;
    g.connect(from,to,weight);
  }
  int pathSize;
  cin>>pathSize;
  int startNode;
  cin>>startNode;
  int currentNode;
  int result=0;
  bool path=true;
  for(int i=0;i<pathSize-1;i++){
    cin>>currentNode;
    if(g.nodes[startNode].hasNeighbour(currentNode)){
      result+=g.nodes[startNode].lengthToNeighbour(currentNode);
      startNode=currentNode;
    }
    else{
      path=false;
    }
  }
  if(path){
    cout<<result;
  }
  else{
    cout<<"-1";
  }
    return 0;
}
```

Line: 1 Col: 1

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