Discos

locked

Problem | Submissions | Leaderboard | Discussions

Submissions: 79
Max Score: 100
Difficulty: Medium

Rate This Challenge:

More

Иванчо е кореняк софиянец. Като такъв, на него често му се налага да отговаря на приятелите си от провинцията на следния въпрос - "На какво разстояние е най-близката дискотека до общежитието ми в студентски град?". Тъй като на него му писна да отговаря на този въпрос, той реши да напише програма, която изчислява това.

Общежитията на приятелите на Иванчо, както и дискотеките, са общо N на брой и са номерирани с числата от 0 до N - 1. Можем да представим въпросните като две множества - на общежитията и на дискотеките, като броят на дискотеките е K. Иванчо има M на брой сведения за **двупосочни** пътища между двойка обекти и техните дължини.

Напишете програма, която отговаря на Q заявки. За всяка заявка получавате връх от множеството на общежитията. Вие трябва да отговорите колко е разстоянието до най-близката дискотека(връх от множеството на дискотеките).

Input Format

На първият ред на входа се въвеждат **N** и **M** - броят обекти и броят на известните на Иванчо пътища между тях. Следват **M** реда с по 3 числа на ред - двойка обекти и разстоянието между тях.

От следващия ред се въвежда **K**. Следват **K** на брой числа - номерата на дискотеките.

От следващия ред се въвежда **Q**. Следват **Q** реда с по 1 число на ред - текущото местоположение на обадилия се човек(номера на общежитието му).

Constraints

$1 \leq N \leq 100\,000$

$1 \leq M \leq 300\,000$

$1 \leq K < N$

$1 \leq Q \leq 100\,000$

В 20% от тестовете Q = 1.

В други 20% от тестовете K = 1

Output Format

Изведете **Q** реда с по 1 число на ред - разстоянието до най-близката дискотека от обадилия се човек.

Sample Input 0

```
6 8
0 1 1
1 2 2
1 3 6
0 4 5
1 4 7
1 5 8
3 5 2
4 5 2
2
4 5
4
0
1
2
3
```

Sample Output 0

```
5
6
8
2
```

Explanation 0

Отговаряме на 4 заявки - за върховете 0, 1, 2, 3.

Най-късият път от върха 0 до дискотека е 0->4 с дължина 5.

Най-късият път от върха 1 до дискотека е 1->0->4 с дължина 6.

Най-късият път от върха 2 до дискотека е 2->1->0->4 с дължина 8.

Най-късият път от върха 3 до дискотека е 3->5 с дължина 2.

Current Buffer (saved locally, editable) C++

```cpp
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
#include <list>
#include <queue>
#include <climits>
using namespace std;
struct Pair{
  int index;
  long int distance;

  bool operator<(const Pair & rhs) const {
    return distance > rhs.distance;
  }
};
struct Edge {
  int from;
  int to;
  int weight;
};
struct Node{
  list<Pair> neighbours;
  bool hasNeighbour(int index){
    for(auto neighbour:neighbours){
      if(neighbour.index==index){
        return true;
      }
    }
    return false;
  }
  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){
    if(!nodes[from].hasNeighbour(to)){
      nodes[from].addNeighbour(to,weight);
      nodes[to].addNeighbour(from,weight);
    }
  }
  void connectOne(int from,int to,int weight){


      nodes[to].addNeighbour(from,weight);
    }


  void dijkstra(int start,vector<long int> &distance){

    vector<int> parent(nodes.size(), -1);

    vector<bool> visited(nodes.size(),false);
    distance[start]=0;
    priority_queue<Pair> nextToProcess;
    nextToProcess.push({start,0});
    while(!nextToProcess.empty()){
      Pair currentPair=nextToProcess.top();
      nextToProcess.pop();
      int currentNode=currentPair.index;
      if(visited[currentNode]){
        continue;
      }
      visited[currentNode]=true;
      for(auto neighbour: nodes[currentNode].neighbours){
        int neighbourNode=neighbour.index;
         long int alternativeDistance=distance[currentNode]+neighbour.distance;

        if(alternativeDistance<distance[neighbourNode]){
          distance[neighbourNode]=alternativeDistance;
          parent[neighbourNode]=currentNode;
          nextToProcess.push({neighbourNode,alternativeDistance});
        }

      }
    }
  }
};
 int main() {

    int objectsNumber;
    int roadsNumber;
    int discosNumber;
    scanf("%d",&objectsNumber);
    scanf("%d",&roadsNumber);
     Graph g(objectsNumber+1);
    for(int i=0;i<roadsNumber;i++){
      int from,to,weight;
      scanf("%d",&from);
      scanf("%d",&to);
      scanf("%d",&weight);
      g.connect( from,  to,  weight);
    }
    scanf("%d",&discosNumber);
    for(int i=0;i<discosNumber;i++){
      int disco;
      scanf("%d",&disco);
      g.connectOne(disco,objectsNumber,0);
    }
    int queriesNumber;
    scanf("%d",&queriesNumber);
      vector<long int> distance(objectsNumber+1,LONG_MAX);
      g.dijkstra(objectsNumber,distance);
    for(int i=0;i<queriesNumber;i++){
      int query;
      scanf("%d",&query);
     printf("%d\n",distance[query]);
    }

    return 0;
  }
```

Line: 1 Col: 1

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