Floyd : City of Blinding Lights

by pranav9413

Given a directed, weighted graph, consisting of N nodes and there are edges ,of specified length between some of them in the graph.

Given Q questions, inquring the shortest distance between a queried pair of nodes in the graph.

Answer all these questions as quickly as possible !

Input Format

First line has two integers N, denoting the number of nodes in the graph and M, denoting the number of edges in the graph.
The next M lines each consist of three space separated integers $x\ y\ r$, where x and y denote the two nodes between which the *directed* edge $(x->y)$ exists, r denotes the length of the edge between the corresponding edges.
The next line contains a single integer Q, denoting number of queries.
The next Q lines each, contain two space separated integers a and b, denoting the node numbers specified according to the question.

Constraints
$2 \leq N \leq 400$
$1 \leq M \leq \frac{N \times (N-1)}{2}$
$1 \leq Q \leq 10^5$
$1 \leq x, y, \leq N$
$1 \leq r \leq 350$

If there are edges between the same pair of nodes with different weights, the last one (most recent) is to be considered as the only edge between them.

Output Format

Print Q lines, each containing a single integer, specifying the shortest distance between the nodes specified for that query in the input.

If the distance between a pair of nodes is infinite (not reachable), then print -1 as the shortest distance.

Sample Input

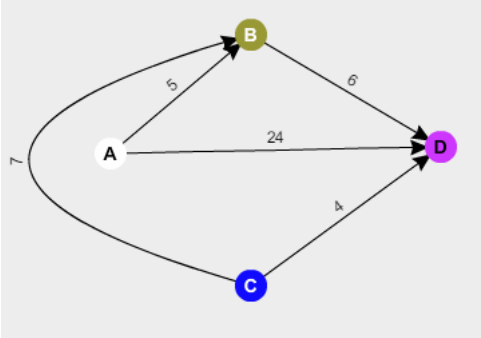
```
4 5
1 2 5
1 4 24
2 4 6
3 4 4
3 2 7
3
1 2
3 1
1 4
```

Sample Output

```
5
-1
11
```

Explanation

The graph given in the test case is shown as :



- The nodes A,B,C and D denote the 1,2,3 and 4 node numbers.

The shortest paths for the 3 queries are :

- **A->B** (Direct Path is shortest with weight 5)
- **-1** (There is no way of reaching node 1 from node 3, hence unreachable)
- **A->B->D** (Indirect path is shortest with weight (5+6) = 11 units, the direct path is longer with 24 units length)

Submissions: 5472
Max Score: 75
Difficulty: Hard

Rate This Challenge:

More

Current Buffer (saved locally, editable) Java 8

```
import java.io.*;
import java.util.*;

 class Graph{

        private int V;
        private Map<String,Integer> map;

        public Graph(int V){
            this.V = V;
            map = new HashMap<String,Integer>();

        }

        public void addEdge(String edge, int weight){
            map.put(edge, weight);
        }

        public int[][] floydAlgo(){

            int[][] output = new int[V][V];

            for(int i = 0 ; i < V ; i++){

                for(int j = 0 ; j < V ; j++){

                    if(i == j){
                        output[i][j] = 0;
                    }
                    else if(map.containsKey(i + "-" + j)){
```

```
                        output[i][j] = map.get(i + "-" + j);
                    }
                    else{
                        output[i][j] = 500000;
                    }
                }

            }


            for(int k = 0 ; k < V ; k++)
                for(int j = 0 ; j < V ; j++)
                    for(int i = 0 ; i < V ; i++)
                        if(output[i][j] > output[i][k] + output[k][j])
                            output[i][j] = output[i][k] + output[k][j];

            return output;

        }

    }


public class Solution {

    public static void main(String[] args) throws IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = br.readLine();
        String[] numbers = line.split("\\s");

        int V = Integer.parseInt(numbers[0]);
        int E = Integer.parseInt(numbers[1]);

        Graph graph = new Graph(V);

        for(int i = 0 ; i < E ; i++){
            line = br.readLine();
            numbers = line.split("\\s");
            graph.addEdge((Integer.parseInt(numbers[0]) - 1) + "-" + (Integer.parseInt(numbers[1]) - 1),
Integer.parseInt(numbers[2]));
        }

        int[][] output = graph.floydAlgo();

        int N = Integer.parseInt(br.readLine());

        for(int i = 0 ; i < N ; i++){

             line = br.readLine();
            numbers = line.split("\\s");

            if(output[(Integer.parseInt(numbers[0]) - 1)][(Integer.parseInt(numbers[1]) - 1)] == 500000){
                System.out.println("-1");
                continue;
            }

            System.out.println(output[(Integer.parseInt(numbers[0]) - 1)][(Integer.parseInt(numbers[1]) - 1)]);
        }

    }
}
```

Line: 47 Col: 13

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