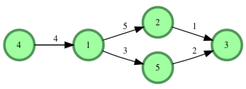
Floyd : City of Blinding Lights

locked

Given a directed weighted graph where weight indicates distance, for each query, determine the length of the shortest path between nodes. There may be many queries, so efficiency counts.

For example, your graph consists of 5 nodes as in the following:



A few queries are from node 4 to node 3, node 2 to node 5, and node 5 to node 3.

1. There are two paths from 4 to 3:
   - $4 \Rightarrow 1 \Rightarrow 2 \Rightarrow 3$ at a distance of $4 + 5 + 1 = 10$
   - $4 \Rightarrow 1 \Rightarrow 5 \Rightarrow 3$ at a distance of $4 + 3 + 2 = 9$
     In this case we choose path 2.
2. There is no path from 2 to 5, so we return -1.
3. There is one path from 5 to 3:
   - $5 \Rightarrow 3$ at a distance of 2.

Input Format

The first line has two integers n and m, the number of nodes and the number of edges in the graph.
Each of the next m lines contains three space-separated integers x y and r, the two nodes between which the *directed* edge $x \Rightarrow y$ exists, and r, the length of the edge.
The next line contains a single integer q, the number of queries.
Each of the next q lines contains two space-separated integers a and b, denoting the start and end nodes for traversal.

Constraints

$2 \leq n \leq 400$
$1 \leq m \leq \frac{n \times (n-1)}{2}$
$1 \leq q \leq 10^5$
$1 \leq x, y, \leq N$
$1 \leq r \leq 350$
The distance from a node to itself is always 0 and it is always reachable from itself.

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 for the query.

If the destination node is not reachable, return -1.

Sample Input

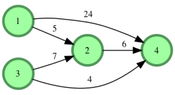
```
4 5
1 2 5
1 4 24
2 4 6
3 4 4
3 2 7
3
1 2
3 1
1 4
```

Sample Output

```
5
-1
11
```

Explanation

The graph given in the test case is shown as :



The shortest paths for the 3 queries are :

- $1 \Rightarrow 2$: The direct Path is shortest with weight 5
- -1: There is no way of reaching node 1 from node 3
- $1 \Rightarrow 2 \Rightarrow 4$ The indirect path is shortest with weight (5+6) = 11 units. The direct path is longer with 24 units length.

Current Buffer (saved locally, editable) C++

```
#include <iostream>
#include <vector>
#include <list>
#include <climits>
using namespace std;

struct Graph{
  vector<vector<int>> adjMatrix;
  Graph(int nodeCount){
    adjMatrix.resize(nodeCount+1);
   for(int row=0;row<=nodeCount;row++){
     adjMatrix[row].resize(nodeCount+1);
     for(int col=0;col<=nodeCount;col++){
       adjMatrix[row][col]=10000000;
     }
   }
  }

  void connect(int from,int to,int weight){
    adjMatrix[from][to]=weight;
  }

  void floyd(vector<vector<int>> &dist,int nodeCount){
     for(int row=0;row<=nodeCount;row++){
     for(int col=0;col<=nodeCount;col++){
       dist[row][col]=adjMatrix[row][col];

     }
   }
    for(int k=0;k<=nodeCount;k++){
      for(int i=0;i<=nodeCount;i++){
        for(int j=0;j<=nodeCount;j++){
          if(dist[i][j]>dist[i][k]+dist[k][j]){
            dist[i][j]=dist[i][k]+dist[k][j];
          }
        }
      }
    }
  }

};
int main(){
  int nodeCount;
  int edgeCount;
  cin>>nodeCount;
  cin>>edgeCount;
  int from,to,weight;
  Graph g(nodeCount);
  for(int i=0;i<edgeCount;i++){
    cin>>from>>to>>weight;
    g.connect(from-1,to-1,weight);
  }
  vector<vector<int>> dist;
  dist.resize(nodeCount+1);
  for(int row=0;row<=nodeCount;row++){
    dist[row].resize(nodeCount+1);
   }
  g.floyd(dist,nodeCount);

  int queriesNumber;
  cin>>queriesNumber;
  int fromQ,toQ;
  for(int i=0;i<queriesNumber;i++){
    cin>>fromQ>>toQ;
    if(fromQ==toQ){
      cout<<0;
    }
    else if(dist[fromQ-1][toQ-1]!=10000000){
    cout<<dist[fromQ-1][toQ-1]<<endl;
    }
    else{
      cout<<-1<<endl;
    }
  }
  return 0;
}
```

Line: 1 Col: 1

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

Submissions: 85
Max Score: 100
Difficulty: Hard

Rate This Challenge:

More