Practice Compete Jobs Rank Leaderboard gadhiya

Dashboard > Algorithms > Graph Theory > Prim's (MST) : Special Subtree

Badge Progress ★ ★ ★ ★

Points: 4727.88 Rank: 491

Prim's (MST) : Special Subtree

by pranav9413

Problem Submissions Leaderboard Discussions Editorial

Given a graph which consists of several edges connecting the N nodes in it.
It is required to find a subgraph of the given graph with the following properties:

- The subgraph contains all the nodes present in the original graph.
- The subgraph is of minimum overall weight (sum of all edges) among all such subgraphs.
- It is also required that there is **exactly one, exclusive** path between any two nodes of the subgraph.

One specific node S is fixed as the starting point of finding the subgraph.
Find the total weight of such a subgraph (sum of all edges in the subgraph)

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 **undirected** edge exists, r denotes the length of edge between the corresponding nodes.

The last line has an integer S, denoting the starting node.

Constraints

$2 \leq N \leq 3000$
$1 \leq M \leq (N * (N-1))/2$
$1 \leq x, y, S \leq N$
$0 <= r <= 10^5$
If there are edges between the same pair of nodes with different weights, they are to be considered as is, like multiple edges.

Output Format

Print a single integer denoting the total weight of tree so obtained (sum of weight of edges).

Sample Input 0

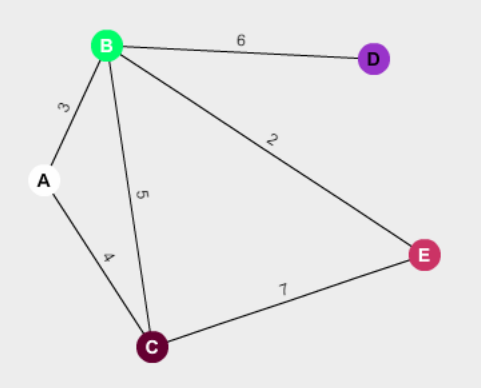
```
5 6
1 2 3
1 3 4
4 2 6
5 2 2
2 3 5
3 5 7
1
```

Sample Output 0

```
15
```

Explanation 0

The graph given in the test case is shown as :



- The nodes A,B,C,D and E denote the obvious 1,2,3,4 and 5 node numbers.
- The starting node is A or 1 (in the given test case)

Applying the Prim's algorithm, edge choices available at first are :

A->B (**WT. 3**) and A->C (**WT. 4**) , out of which A->B is chosen (smaller weight of edge).

Now the available choices are :

A->C (**WT. 4**) , B->C (**WT. 5**) , B->E (**WT. 2**) and B->D (**WT. 6**) , out of which B->E is chosen by the algorithm.

Following the same method of the algorithm, the next chosen edges , sequentially are :

A->C and B->D.

Hence the overall sequence of edges picked up by prims are:

A->B : B->E : A->C : B->D

and Total weight of the hence formed MST is : **15**

Submissions: 9394
Max Score: 60
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

Current Buffer (saved locally, editable)

Java 8

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

class DisjointSet{

    long[] rank,parent;
    int n;

    public DisjointSet(int n){
        this.n = n;
        rank = new long[n];
        parent = new long[n];
        makeset(n);
    }

    void makeset(int n){
        for(int i = 0 ; i < n ; i++){
```

```java
            parent[i] = (long) i;
        }
    }

    long find(int x){

        if(parent[x] != (long)x){
            parent[x] = find((int)parent[x]);
        }
       return parent[x];
    }

    void union(int x, int y){

        int xRoot = (int) find(x);
        int yRoot = (int)  find(y);

        if(xRoot == yRoot){
            return;
        }

        if(rank[xRoot] < rank[yRoot]){
            parent[xRoot] = yRoot;
        }
        else if(rank[xRoot] > rank[yRoot]){
            parent[yRoot] = xRoot;
        }
        else{
            parent[yRoot] = xRoot;
            rank[xRoot] = rank[xRoot] + 1;
        }
    }
}


class Vertex{

    private int v1,v2;

    public Vertex(int v1, int v2){
        this.v1 = v1;
        this.v2 = v2;
    }

    public int getV1(){
        return v1;
    }

    public int getV2(){
        return v2;
    }

}


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]);

        DisjointSet disjoint = new DisjointSet(V);

        HashMap<Integer, LinkedList<Vertex>> adj = new HashMap<Integer, LinkedList<Vertex>>();;

        long output = 0;

        //List sortedKeys=new ArrayList(yourMap.keySet());
        //Collections.sort(sortedKeys);

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

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

            int v1 = Integer.parseInt(numbers[0]);
            int v2 = Integer.parseInt(numbers[1]);
            int weight = Integer.parseInt(numbers[2]);

            Vertex vertex = new Vertex((v1 - 1),(v2 - 1));

            if(adj.containsKey(weight)){
                //list2 = (LinkedList) list1.clone();

                LinkedList<Vertex> clone = (LinkedList) adj.get(weight).clone();
                clone.add(vertex);

                adj.put(weight,clone);
            }
            else{
                LinkedList<Vertex> link = new LinkedList<Vertex>();
                link.add(vertex);
                adj.put(weight,link);
            }

        }

        List sortedKeys=new ArrayList(adj.keySet());
        Collections.sort(sortedKeys);



        for(int g = 0 ; g < sortedKeys.size() ; g++){

            LinkedList<Vertex> clone = (LinkedList) adj.get(sortedKeys.get(g)).clone();

            Iterator itr = clone.listIterator();

            while(itr.hasNext()){

                Vertex vert = (Vertex) itr.next();
                int v1 = vert.getV1();
                int v2 = vert.getV2();

                long representativeV1 = disjoint.find(v1);
                long representativeV2 = disjoint.find(v2);

                if(representativeV1 != representativeV2){
                    disjoint.union((int)representativeV1, (int)representativeV2);
                    output = output + (Integer) sortedKeys.get(g);
                }


            }

        }

        System.out.println(output);


    }
}
```

Line: 1 Col: 1

Upload Code as File □ Test against custom input

Run Code Submit Code

Join us on IRC at #hackerrank on freenode for hugs or bugs.

Contest Calendar | Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature