In lab-12(1)

**Kruskal (MST): Really Special Subtree**

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Comparator;

import java.util.HashMap;

import java.util.PriorityQueue;

import java.util.Scanner;

public class Solution {

    static int[] sets;

    static int findSet(int i)

    {

        if (sets[i] == i) {

            return i;

        }

        return findSet(sets[i]);

    }

    static boolean sameSet(int a , int b)

    {

        return findSet(a) == findSet(b);

    }

    static void mergeSet(int a, int b)

    {

        int i = findSet(a);

        int j = findSet(b);

        sets[i] = j;

    }

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        int N = in.nextInt();

        int M = in.nextInt();

        int[][] graph = new int[M][3];

        for (int i = 0; i < graph.length; i++) {

            for (int j = 0; j < 3; j++) {

                graph[i][j] = in.nextInt();

            }

        }

        Arrays.sort(graph, new Comparator<int[]>() {

            public int compare(int[] o1, int[] o2) {

                return o1[2] - o2[2];

            }

        });

         sets = new int[N+1];

        for (int i = 0; i < sets.length; i++) {

            sets[i] = i;

        }

        int sum = 0;

        for (int[] t : graph) {

            if (!sameSet(t[0], t[1])) {

                sum += t[2];

                mergeSet(t[0], t[1]);

            }

        }

        System.out.println(sum);

    }

}

In lab-12(2)

# Prim's (MST) : Special Subtree

import java.io.\*;

import java.util.\*;

public class SubTree {

    public static void main(String[] args) {

        Scanner in=new Scanner(System.in);

        int n=in.nextInt();

        int m=in.nextInt();

        int adm[][]=new int[n][n];

        int MX=Integer.MAX\_VALUE;

        for(int i=0;i<n;i++){

            Arrays.fill(adm[i],MX);

        }

        for(int i=0;i<m;i++){

            int x=in.nextInt()-1;

            int y=in.nextInt()-1;

            int w=in.nextInt();

            adm[x][y]=adm[y][x]=Math.min(adm[x][y],w);

        }

        int s=in.nextInt()-1;

        boolean[]hv=new boolean[n];

        hv[s]=true;

        int[]mw=new int[n];

        Arrays.fill(mw,MX);

        for(int i=0;i<n;i++){

            mw[i]=adm[s][i];

        }

        int sum=0;

        for(int i=1;i<n;i++){

            int ind=-1;

            for(int j=0;j<n;j++){

                if(!hv[j]&&mw[j]<MX){

                    if(ind==-1)ind=j;

                    else{

                        if(mw[j]<mw[ind])ind=j;

                    }

                }

            }

            sum+=mw[ind];

            hv[ind]=true;

            for(int j=0;j<n;j++){

                mw[j]=Math.min(mw[j],adm[j][ind]);

            }

        }

        System.out.println(sum);

        in.close();

    }

}