All Contests > DAA_LAB > Krushkals_algorithm

Krushkals_algorithm

Problem Submissions Leaderboard Discussions

Find Minimum Cost Spanning Tree of a given connected undirected graph using Kruskal's algorithm. Use Union-Find algorithms in your program

Input Format

7 0 28 999 999 999 10 999 28 0 16 999 999 999 14 999 16 0 12 999 999 999 999 999 12 0 22 999 18 999 999 999 22 0 25 24 10 999 999 999 25 999 999 999 14 999 18 24 999 999

Constraints

No Constraints

Output Format

1edge(1,6)=10 2edge(3,4)=12 3edge(2,7)=14 4edge(2,3)=16 5edge(4,5)=22 6edge(5,6)=25 The minimum cost of spanning tree is 99

Sample Input 0

```
7
0 28 999 999 999 10 999
28 0 16 999 999 999 14
999 16 0 12 999 999 999
999 999 12 0 22 999 18
999 999 999 22 0 25 24
10 999 999 999 25 999 999
999 14 999 18 24 999 999
```

Sample Output 0

```
1edge(1,6)=10
2edge(3,4)=12
3edge(2,7)=14
4edge(2,3)=16
5edge(4,5)=22
6edge(5,6)=25
The minimum cost of spanning tree is 99
```

Contest ends in 9 days

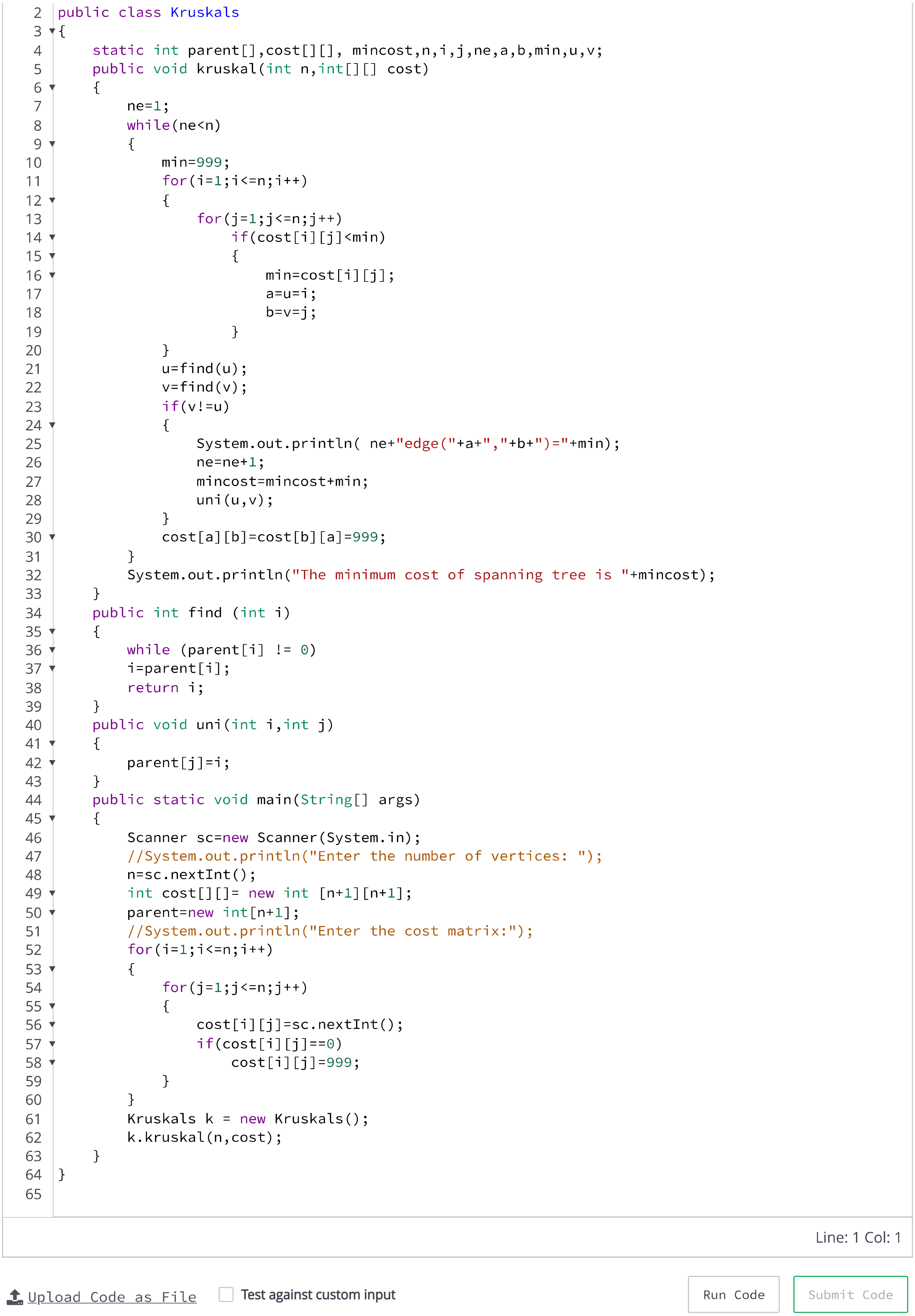
Submissions: 97
Max Score: 10
Difficulty: Medium

Rate This Challenge:
☆☆☆☆☆

More

Java 7

```
import java.util.Scanner;
```



```
public class Kruskals
{
    static int parent[],cost[][], mincost,n,i,j,ne,a,b,min,u,v;
    public void kruskal(int n,int[][] cost)
    {
        ne=1;
        while(ne<n)
        {
            min=999;
            for(i=1;i<=n;i++)
            {
                for(j=1;j<=n;j++)
                    if(cost[i][j]<min)
                    {
                        min=cost[i][j];
                        a=u=i;
                        b=v=j;
                    }
            }
            u=find(u);
            v=find(v);
            if(v!=u)
            {
                System.out.println( ne+"edge("+a+","+b+")="+min);
                ne=ne+1;
                mincost=mincost+min;
                uni(u,v);
            }
            cost[a][b]=cost[b][a]=999;
        }
        System.out.println("The minimum cost of spanning tree is "+mincost);
    }
    public int find (int i)
    {
        while (parent[i] != 0)
        i=parent[i];
        return i;
    }
    public void uni(int i,int j)
    {
        parent[j]=i;
    }
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        //System.out.println("Enter the number of vertices: ");
        n=sc.nextInt();
        int cost[][]= new int [n+1][n+1];
        parent=new int[n+1];
        //System.out.println("Enter the cost matrix:");
        for(i=1;i<=n;i++)
        {
            for(j=1;j<=n;j++)
            {
                cost[i][j]=sc.nextInt();
                if(cost[i][j]==0)
                    cost[i][j]=999;
            }
        }
        Kruskals k = new Kruskals();
        k.kruskal(n,cost);
    }
}
```

Testcase 0 ✓

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
7
0 28 999 999 999 10 999
28 0 16 999 999 999 14
999 16 0 12 999 999 999
999 999 12 0 22 999 18
999 999 999 22 0 25 24
10 999 999 999 25 999 999
999 14 999 18 24 999 999
```

Your Output (stdout)

```
1edge(1,6)=10
2edge(3,4)=12
3edge(2,7)=14
4edge(2,3)=16
5edge(4,5)=22
6edge(5,6)=25
The minimum cost of spanning tree is 99
```

Expected Output

```
1edge(1,6)=10
2edge(3,4)=12
3edge(2,7)=14
4edge(2,3)=16
5edge(4,5)=22
6edge(5,6)=25
The minimum cost of spanning tree is 99
```