

[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 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
```

[f](#) [t](#) [in](#)Contest ends in 2 months

Submissions: 76

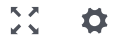
Max Score: 10

Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

Java 7



```
1 //224G1A0553
2 import java.util.Scanner;
3 public class Kruskals {
4     static int parent[],cost[][], mincost,n,i,j,ne,a,b,min,u,v;
5     public void kruskal(int n,int[][] cost) {
6         ne=1;
7         while(ne<n) {
8             min=999;
9             for(i=1;i<=n;i++) {
10                for(j=1;j<=n;j++)
11                if(cost[i][j]<min) {
12                    min=cost[i][j];
13                    a=u=i;
14                    b=v=j;
15                }
16            }
17            u=find(u);
18            v=find(v);
19            if(v!=u) {
20                System.out.println( ne+"edge("+a+", "+b+")="+min);
21                ne=ne+1;
22                mincost=mincost+min;
23                uni(u,v);
24            }
25            cost[a][b]=cost[b][a]=999;
26        }
27        System.out.println("The minimum cost of spanning tree is "+mincost); }
28     public int find (int i) {
29         while (parent[i] != 0)
30             i=parent[i];
31         return i;
32     }
33     public void uni(int i,int j) {
34         parent[j]=i;
35     }
36     public static void main(String[] args) {
37         Scanner sc=new Scanner(System.in);
38         //System.out.println("Enter the number of vertices: ");
39         n=sc.nextInt();
40         int cost[][]= new int [n+1][n+1];
41         parent=new int[n+1];
42         //System.out.println("Enter the cost matrix:");
43         for(i=1;i<=n;i++) {
44             for(j=1;j<=n;j++) {
45                 cost[i][j]=sc.nextInt();
46                 if(cost[i][j]==0)
47                     cost[i][j]=999;
48             }
49         }
50         Kruskals k = new Kruskals();
51         k.kruskal(n,cost);
52     }
53 }
```

Line: 1 Col: 13

 Upload Code as File ☐ Test against custom input

Run Code

Submit Code

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
```