https://course.acciojob.com/idle?question=740c2217-81de-472e-ab2f-061ad8bff051

- MEDIUM
- Max Score: 40 Points
-
-

Minimum Cost to Repair All Roads

There are `n` cities, some of which are connected by roads. All the roads are damaged at once for unknown reasons. To re-connect the cities, we must fix the roads. A specific road"s repair has a fixed cost.

These costs are represented in form of a 2D matrix called `road`, where `cost[i][j]` represents the cost to rebuild a road between city `i` and city `j`. If `cost[i][j] = 0`, then city `i` and `j` can not be connected directly.

Find the minimum cost to connect all the cities by repairing the roads.

Input Format

First line contains an integer `n` which is the number of cities.

Next `n` lines contain `n` space separated integers to make up the `cost` matrix

Output Format

Complete the function MinCost() which returns an integer

Example 1

Input

```
5
0 1 2 3 4
1 0 5 0 7
2 5 0 6 0
```

```
3 0 6 0 0
4 7 0 0 0
```

Output

```
10
```

Example 2

Input

```
6
0 1 1 100 0 0
1 0 1 0 0 0
1 1 0 0 0 0
100 0 0 0 2 2
0 0 0 2 0 2
0 0 0 2 2 0
```

Output

```
106
```

Constraints

1 <= `n` <= 500

1 <= `cost[i][j]` <= 100

Topic Tags

- Graphs
- Greedy
- Word Problems

My code

```
// n java
import java.util.*;

// class Solution {
//     public int MinCost(int[][] cost, int n) {
//         // Write your code here

//     }
// }
class Solution {
        public int minDistNode(int n, int[] dist, boolean[] visited) {
                int val = Integer.MAX_VALUE;
                int pos = 0;

                for(int i = 0; i < n; i++) {
                        if(visited[i] == false && dist[i] < val) {
                                val = dist[i];
                                pos = i;
                        }
                }

                return pos;
        }

    public int MinCost(int[][] cost, int n) {
        // Write your code here
                int parent[] = new int[n];
```

```
        int dist[] = new int[n];
        boolean visited[] = new boolean[n];

        for(int i = 0; i < n; i++) {
            dist[i] = Integer.MAX_VALUE;
            visited[i] = false;
        }

        parent[0] = -1;
        dist[0] = 0;

        for(int i = 0; i < n-1; i++) {
            // first step : find the node not visited and least dist
            int u = minDistNode(n, dist, visited);

            // second step : mark it visited
            visited[u] = true;

            // update the neighbours dist and parent
            for(int v = 0; v < n; v++) {
                if(cost[u][v] > 0 && visited[v] == false &&
cost[u][v] < dist[v]) {
                    dist[v] = cost[u][v];
                    parent[v] = u;
                }
            }
        }

        int ans = 0;
        for(int i = 1; i < n; i++) {
```

```
                ans += cost[parent[i]][i];
            }

            return ans;

    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[][] c = new int[n][n];
        for(int i = 0; i < n; i++) {
            for(int j = 0; j < n; j++)
                c[i][j] = sc.nextInt();
        }

            Solution Obj = new Solution();
        System.out.println(Obj.MinCost(c, n));
    }
}
```