



Practice

Compete

Jobs

Rank

Leaderboard



gadhiya

Dashboard &gt; Algorithms &gt; Graph Theory &gt; Roads and Libraries

Badge Progress



Points: 4727.88 Rank: 491

# Roads and Libraries



by torquecode

Problem

Submissions

Leaderboard

Discussions

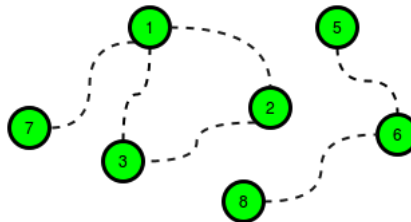
Editorial

The Ruler of HackerLand believes that every citizen of the country should have access to a library. Unfortunately, HackerLand was hit by a tornado that destroyed all of its libraries and obstructed its roads! As you are the greatest programmer of HackerLand, the ruler wants your help to repair the roads and build some new libraries efficiently.

HackerLand has  $n$  cities numbered from  $1$  to  $n$ . The cities are connected by  $m$  bidirectional roads. A citizen has access to a library if:

- Their city contains a library.
- They can travel by road from their city to a city containing a library.

The following figure is a sample map of HackerLand where the dotted lines denote obstructed roads:



The cost of repairing any road is  $c_{road}$  dollars, and the cost to build a library in any city is  $c_{lib}$  dollars.

You are given  $q$  queries, where each query consists of a map of HackerLand and value of  $c_{lib}$  and  $c_{road}$ .

For each query, find the minimum cost of making libraries accessible to all the citizens and print it on a new line.

## Input Format

The first line contains a single integer,  $q$ , denoting the number of queries. The subsequent lines describe each query in the following format:

- The first line contains four space-separated integers describing the respective values of  $n$  (the number of cities),  $m$  (the number of roads),  $c_{lib}$  (the cost to build a library), and  $c_{road}$  (the cost to repair a road).
- Each line  $i$  of the  $m$  subsequent lines contains two space-separated integers,  $u_i$  and  $v_i$ , describing a bidirectional road connecting cities  $u_i$  and  $v_i$ .

## Constraints

- $1 \leq q \leq 10$
- $1 \leq n \leq 10^5$
- $0 \leq m \leq \min(10^5, \frac{n \cdot (n-1)}{2})$
- $1 \leq c_{road}, c_{lib} \leq 10^5$
- $1 \leq u_i, v_i \leq n$
- Each road connects two distinct cities.

**Output Format**

For each query, print an integer denoting the minimum cost of making libraries accessible to all the citizens on a new line.

**Sample Input**

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

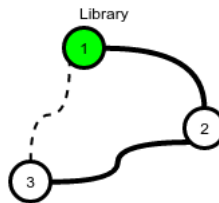
**Sample Output**

```
4
12
```

**Explanation**

We perform the following  $q = 2$  queries:

1. HackerLand contains  $n = 3$  cities connected by  $m = 3$  bidirectional roads. The price of building a library is  $c_{lib} = 2$  and the price for repairing a road is  $c_{road} = 1$ .

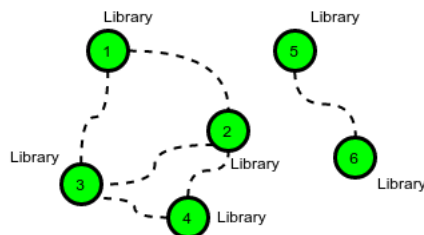


The cheapest way to make libraries accessible to all is to:

- Build a library in city **1** at a cost of  $x = 2$ .
- Repair the road between cities **1** and **2** at a cost of  $y = 1$ .
- Repair the road between cities **2** and **3** at a cost of  $y = 1$ .

This gives us a total cost of  $2 + 1 + 1 = 4$ . Note that we don't need to repair the road between cities **3** and **1** because we repaired the roads connecting them to city **2**!

2. In this scenario it's optimal to build a library in each city because the cost of building a library ( $c_{lib} = 2$ ) is less than the cost of repairing a road ( $c_{road} = 5$ ).



There are **6** cities, so the total cost is  $6 \times 2 = 12$ .



f t in

Submissions: [7792](#)

Max Score: 30

Difficulty: Medium

Rate This Challenge:

Current Buffer (saved locally, editable)  

Java 8



```
1 import java.io.*;
2 import java.util.*;
3 import java.text.*;
4 import java.math.*;
5 import java.util.regex.*;
6
7
8 class DisjointSet{
9
10     long[] rank,parent;
11     int n;
12
13     public DisjointSet(int n){
14         this.n = n;
15         rank = new long[n];
16         parent = new long[n];
17         makeset(n);
18     }
19
20     void makeset(int n){
21         for(int i = 0 ; i < n ; i++){
22             parent[i] = (long) i;
23         }
24     }
25
26     long find(int x){
27
28         if(parent[x] != (long)x){
29             parent[x] = find((int)parent[x]);
30         }
31         return parent[x];
32     }
33
34     void union(int x, int y){
35
36         int xRoot = (int) find(x);
37         int yRoot = (int) find(y);
38
39         if(xRoot == yRoot){
40             return;
41         }
42
43         if(rank[xRoot] < rank[yRoot]){
44             parent[xRoot] = yRoot;
45         }
46         else if(rank[xRoot] > rank[yRoot]){
47             parent[yRoot] = xRoot;
48         }
49         else{
50             parent[yRoot] = xRoot;
51             rank[xRoot] = rank[xRoot] + 1;
52         }
53     }
54 }
55
56
57 public class Solution {
58
59     public static void main(String[] args) {
60         Scanner in = new Scanner(System.in);
61         int q = in.nextInt();
62         for(int a0 = 0; a0 < q; a0++){
63             int n = in.nextInt();
64             int m = in.nextInt();
65             int x = in.nextInt();
```

```
67     int y = in.nextInt();
68
69     DisjointSet mySet = new DisjointSet(n);
70
71     for(int a1 = 0; a1 < m; a1++){
72         mySet.union((in.nextInt() - 1), (in.nextInt() - 1));
73     }
74
75     HashMap<Integer, Long> map = new HashMap<Integer, Long>();
76
77     for(int i = 0 ; i < n ; i++){
78
79         int representative = (int) mySet.find(i);
80
81         if(map.containsKey(representative)){
82             map.put(representative, map.get(representative) + 1);
83         }
84         else{
85             map.put(representative, (long)1);
86         }
87     }
88
89     long output = 0;
90
91     long totalValue = (long)n * (long)x;
92
93     for(long a : map.values()){
94         output = output + ((a - 1) * y) + x;
95     }
96
97     if(output > totalValue){
98         System.out.println(totalValue);
99     }
100    else{
101        System.out.println(output);
102    }
103 }
104 }
105 }
106 }
```

Line: 1 Col: 1

[Upload Code as File](#) ☐ Test against custom input

Run Code

Submit Code

Copyright © 2017 HackerRank. All Rights Reserved

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](#)