



Journey to the Moon



by amititkgp

Problem

Submissions

Leaderboard

Discussions

Editorial

The member states of the UN are planning to send **2** people to the Moon. But there is a problem. In line with their principles of global unity, they want to pair astronauts of **2** different countries.

There are N trained astronauts numbered from **0** to $N - 1$. But those in charge of the mission did not receive information about the citizenship of each astronaut. The only information they have is that some particular pairs of astronauts belong to the same country.

Your task is to compute in how many ways they can pick a pair of astronauts belonging to different countries. Assume that you are provided enough pairs to let you identify the groups of astronauts even though you might not know their country directly. For instance, if **1, 2, 3** are astronauts from the same country; it is sufficient to mention that **(1, 2)** and **(2, 3)** are pairs of astronauts from the same country without providing information about a third pair **(1, 3)**.

Input Format

The first line contains two integers, N and P , separated by a single space. P lines follow. Each line contains **2** integers separated by a single space A and B such that

$$0 \leq A, B \leq N - 1$$

and A and B are astronauts from the same country.

Constraints

- $1 \leq N \leq 10^5$
- $1 \leq P \leq 10^4$

Output Format

An integer that denotes the number of permissible ways to choose a pair of astronauts.

Sample Input 0

```
5 3
0 1
2 3
0 4
```

Sample Output 0

```
6
```

Explanation 0

Persons numbered **0, 1** and **4** belong to the same country, and those numbered **2** and **3** belong to the same country, but different from the previous one. All in all, the UN has **6** ways of choosing a pair:

- persons **0** and **2**
- persons **0** and **3**

3. persons **1** and **2**
4. persons **1** and **3**
5. persons **2** and **4**
6. persons **3** and **4**

Sample Input 1

```
4 1
0 2
```

Sample Output 1

5

Explanation 1

Persons numbered **0** and **2** belong to the same country, and persons **1** and **3** don't share countries with anyone else, so they belong to unique countries on their own. All in all, the UN has **5** ways of choosing a pair:

1. persons **0** and **1**
2. persons **0** and **3**
3. persons **1** and **2**
4. persons **1** and **3**
5. persons **2** and **3**

[f](#) [t](#) [in](#)

Submissions: 17954



Max Score: 50

Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

[More](#)

Current Buffer (saved locally, editable)  

Java 8



```
1 import java.io.*;
2 import java.util.*;
3
4 class DisjointSet{
5
6     long[] rank,parent;
7     int n;
8
9     public DisjointSet(int n){
10         this.n = n;
11         rank = new long[n];
12         parent = new long[n];
13         makeset(n);
14     }
15
16     void makeset(int n){
17         for(int i = 0 ; i < n ; i++){
18             parent[i] = (long) i;
19         }
20     }
21
22     long find(int x){
23
24         if(parent[x] != (long)x){
```

```
25     parent[x] = find((int)parent[x]);
26 }
27 return parent[x];
28 }
29
30 void union(int x, int y){
31
32     int xRoot = (int) find(x);
33     int yRoot = (int) find(y);
34
35     if(xRoot == yRoot){
36         return;
37     }
38
39     if(rank[xRoot] < rank[yRoot]){
40         parent[xRoot] = yRoot;
41     }
42     else if(rank[xRoot] > rank[yRoot]){
43         parent[yRoot] = xRoot;
44     }
45     else{
46         parent[yRoot] = xRoot;
47         rank[xRoot] = rank[xRoot] + 1;
48     }
49 }
50
51 }
52
53 public class Solution {
54
55     public static void main(String[] args) throws IOException {
56
57         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
58
59         String line = br.readLine();
60         String[] num = line.split("\\s");
61
62         int vertices = Integer.parseInt(num[0]);
63         int edges = Integer.parseInt(num[1]);
64
65         DisjointSet mySet = new DisjointSet(vertices);
66
67         for(int i = 0 ; i < edges ; i++){
68             line = br.readLine();
69             num = line.split("\\s");
70             mySet.union(Integer.parseInt(num[0]),Integer.parseInt(num[1]));
71         }
72
73
74         HashMap<Integer,Long> map = new HashMap<Integer,Long>();
75
76
77         for(int i = 0 ; i < vertices ; 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
90         Long[] arr = map.values().toArray(new Long[map.size()]);
91
92         long output = 0;
93
94         for(int i = 0 ; i < arr.length ; i++){
95
96             for(int j = i + 1 ; j < arr.length ; j++){
97                 output = output + (arr[i] * arr[j]);
```

```
98         }
99
100     }
101
102     System.out.println(output);
103
104     }
105 }
```

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](#)