Components in a graph ★

170 more points to get your next star!

Rank: **472718** | Points: **305/475**

Problem Submissions Leaderboard Editorial

RATE THIS CHALLENGE

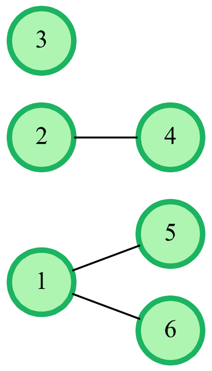
★ ★ ★ ★ ★

There are $2 \times N$ nodes in an undirected graph, and a number of edges connecting some nodes. In each edge, the first value will be between 1 and N, inclusive. The second node will be between $N + 1$ and $2 \times N$, inclusive. Given a list of edges, determine the size of the smallest and largest connected components that have 2 or more nodes. A node can have any number of connections. The highest node value will always be connected to at least 1 other node.

Note Single nodes should not be considered in the answer.

Example

$bg = [[1, 5], [1, 6], [2, 4]]$



The smaller component contains 2 nodes and the larger contains 3. Return the array $[2, 3]$.

Function Description

Complete the connectedComponents function in the editor below.

connectedComponents has the following parameter(s):

- int bg[n][2]: a 2-d array of integers that represent node ends of graph edges

Returns

- int[2]: an array with 2 integers, the smallest and largest component sizes

Input Format

The first line contains an integer n, the size of bg.

Each of the next n lines contain two space-separated integers, $bg[i][0]$ and $bg[i][1]$.

Constraints

- $1 \leq numberofnodesN \leq 15000$
- $1 \leq bg[i][0] \leq N$
- $N + 1 \leq bg[i][1] \leq 2N$

Sample Input

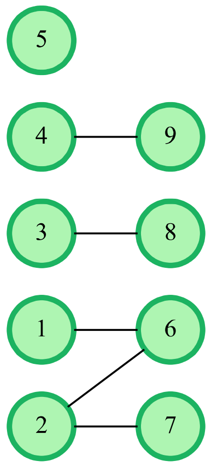
```
STDIN   Function
-----   --------
5      bg[] size n = 5
1 6     bg = [[1, 6],[2, 7], [3, 8], [4,9], [2, 6]]
2 7
3 8

4 9
```

```
2 6
```

Sample Output

```
2 4
```

Explanation



Since the component with node 5 contains only one node, it is not considered.

The number of vertices in the smallest connected component in the graph is 2 based on either $(3, 8)$ or $(4, 9)$.

The number of vertices in the largest connected component in the graph is 4 i.e. $1 - 2 - 6 - 7$.

Change Theme Language C++14

```
vector<int> componentsInGraph(vector<vector<int>>&gb) {
    int edges = 2*gb.size();
    int nodes = 2*gb.size();

    vector<int>parent(nodes+1);
    vector<int>count(nodes+1);

    for(int i=1; i<=nodes; i++){
        parent[i]=i;
        count[i]=1;
    }

    int mx=INT_MIN,mn=INT_MAX;

    for(int i=0; i<gb.size(); i++){
        int u = gb[i][0];
        int v = gb[i][1];

        int absU = absParent(parent, u);
        int absV = absParent(parent, v);

        if(absU == absV) continue;

        //union
        if(count[absU] > count[absV]){
            count[absU] += count[absV];
            count[absV] = 0;
            parent[absV] = absU;
        }
        else{
            count[absV] += count[absU];
```

Line: 150 Col: 1

Upload Code as File

Test against custom input

Run Code

Submit Code

✓ Test case 0

✓ Test case 1

✓ Test case 2

✓ Test case 3

✓ Test case 4

✓ Test case 5

✓ Test case 6

Compiler Message

```
Success
```

Hidden Test Case

Unlock this testcase for 5 hackos.

Unlock

Contest Calendar | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy | Request a Feature