1h 1m left

BETA Can't read the text? Switch theme

ALL

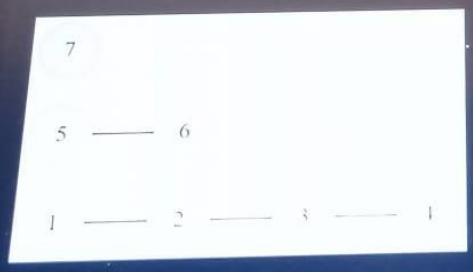
1. Social Connections

A popular social media platform provides a feature to connect people online. Connections are represented as an undirected graph where a user can see the profiles of those they are connected to.

There are *connection_nodes* users numbered 1 to *connection_nodes*, and *connection_edges* connections where the i^{th} pair connects nodes *connection_from[i]* and *connection_to[i]*. The *queries* array contains node numbers. Find the number of users whose profiles are visible to *query[i]*. Report an array of integers where the i^{th} value is the answer to the i^{th} query.

Example

connection_nodes = 7
connection_edges = 4
connection_from = [1, 2, 3, 5]
connection_to = [2, 3, 4, 6]
queries = [1, 3, 5, 7].

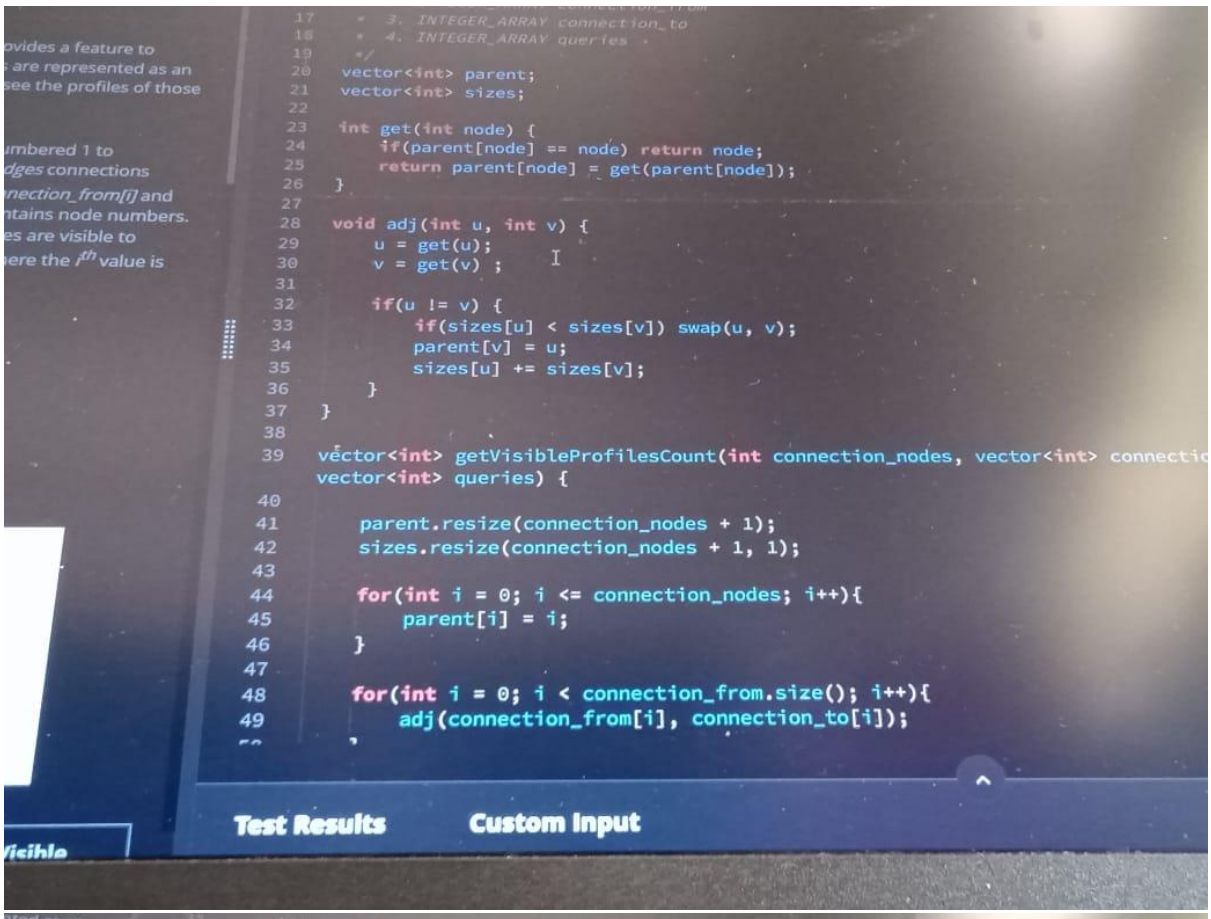


	Visible	Number of Visible

Language C++20

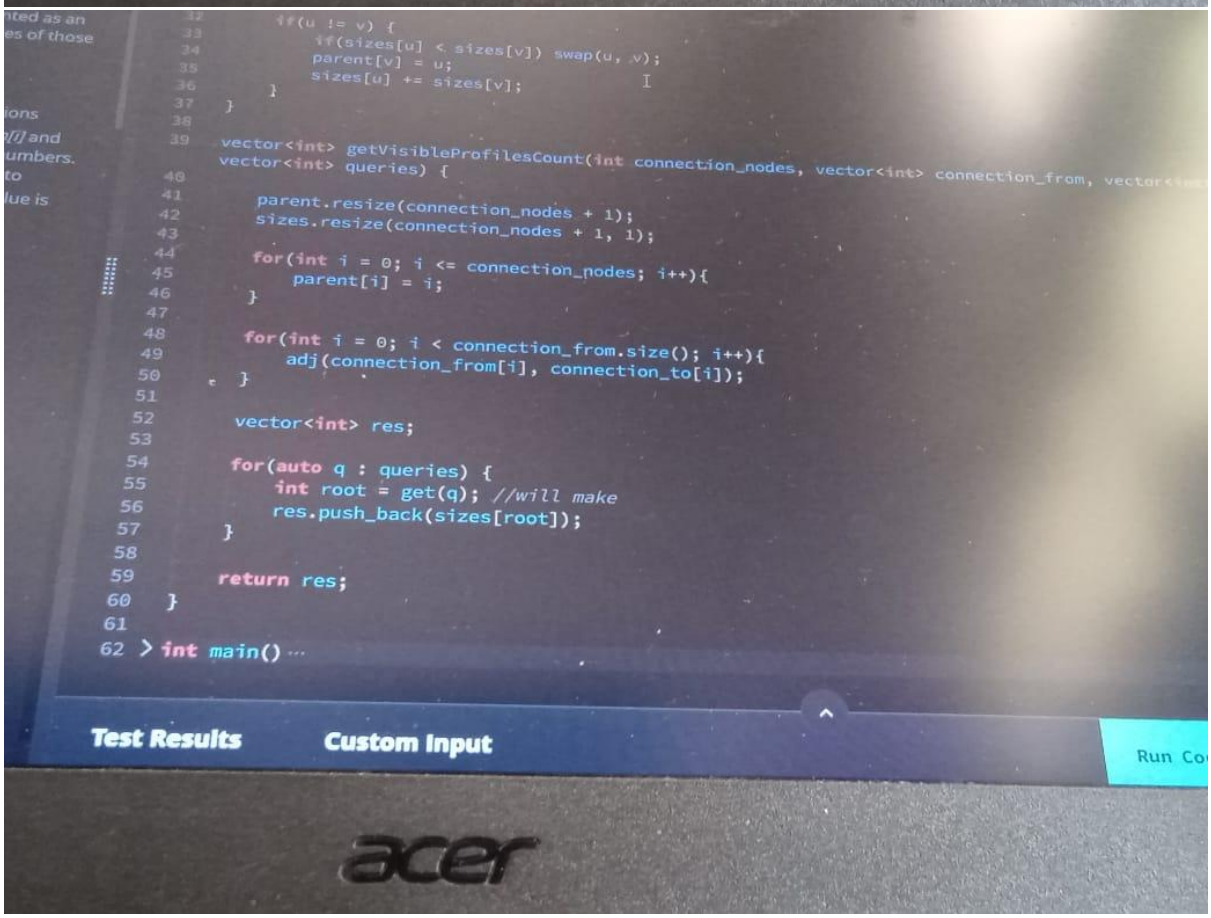
```
 *  1. INTEGER connection_nodes
 *  2. INTEGER_ARRAY connection_from
 *  3. INTEGER_ARRAY connection_to
 *  4. INTEGER_ARRAY queries
 */
vector<int> parent;
vector<int> sizes;

int get(int node) {
    if(parent[node] == node) return node;
    return parent[node] = get(parent[node])
}

void adj(int u, int v) {
    u = get(u);
    v = get(v) ;

    if(u != v) {
        if(sizes[u] < sizes[v]) swap(u,
        parent[v] = u;
        sizes[u] += sizes[v];
    }
}

vector<int> getVisibleProfilesCount(i
vector<int> queries) {

    parent.resize(connection_nodes +
    sizes.resize(connection_nodes +

    for(int i = 0; i <= connection_
        parent[i] = i;
    }

    for(int i = 0; i < connection_
        adj(connection_from[i], c
    }
```

Test Results | Custom Input



```
 *  3. INTEGER_ARRAY connection_to
 *  4. INTEGER_ARRAY queries
 */
vector<int> parent;
vector<int> sizes;

int get(int node) {
    if(parent[node] == node) return node;
    return parent[node] = get(parent[node]);
}

void adj(int u, int v) {
    u = get(u);
    v = get(v) ;

    if(u != v) {
        if(sizes[u] < sizes[v]) swap(u, v);
        parent[v] = u;
        sizes[u] += sizes[v];
    }
}

vector<int> getVisibleProfilesCount(int connection_nodes, vector<int> connectio
vector<int> queries) {

    parent.resize(connection_nodes + 1);
    sizes.resize(connection_nodes + 1, 1);

    for(int i = 0; i <= connection_nodes; i++){
        parent[i] = i;
    }

    for(int i = 0; i < connection_from.size(); i++){
        adj(connection_from[i], connection_to[i]);
```

Test Results Custom Input



```
    if(u != v) {
        if(sizes[u] < sizes[v]) swap(u, v);
        parent[v] = u;
        sizes[u] += sizes[v];
    }
}

vector<int> getVisibleProfilesCount(int connection_nodes, vector<int> connection_from, vector<int
vector<int> queries) {

    parent.resize(connection_nodes + 1);
    sizes.resize(connection_nodes + 1, 1);

    for(int i = 0; i <= connection_nodes; i++){
        parent[i] = i;
    }

    for(int i = 0; i < connection_from.size(); i++){
        adj(connection_from[i], connection_to[i]);
    }

    vector<int> res;

    for(auto q : queries) {
        int root = get(q); //will make
        res.push_back(sizes[root]);
    }

    return res;
}

int main() ...
```

Test Results Custom Input Run Co

acer