https://course.acciojob.com/idle?question=ea0ba181-faaa-4ec3-89d2-c40d022b321f

MEDIUM

Max Score: 40 Points

Count components

You are given an undirected graph with `N` vertices. You have to find the number of connected components in the graph.

Note Complete the given function. The input and output would be handled by the driver code.

A set of vertices forms a connected component in an undirected graph if any vertex from the set of vertices can reach any other vertex by traversing edges.

Input Format

The first line of the input contains `N`.

and next `N` lines consists of `N` integers each of the adjacency matrix `adj`.

If `adj[u][v]` is `1`, it means there is an edge between `u` and `v`.

Output Format

Print the answer in a new line.

Example 1

Input

```
3
1 1 0
1 1 0
0 0 1
```

Output

```
2
```

Explanation

The graph has two components. [1, 2], and [3].

Example 2

Input

```
2
1 0
0 1
```

Output

```
2
```

Explanation

The graph has two components. [1], and [2].

Constraints

1 <= N <= 300

0 <= adj[u][v] <= 1

Topic Tags

- **Graphs**
- **DFS**

My code

```
// in java
import java.io.*;
import java.util.*;
class Main {
    public static void main(String args[]) throws IOException {
        BufferedReader read = new BufferedReader(new
InputStreamReader(System.in));
        int N = Integer.parseInt(read.readLine());

        ArrayList<ArrayList<Integer>> adj = new ArrayList<>();

        for(int i=0; i<N; i++)
        {
            String S[] = read.readLine().split(" ");
            ArrayList<Integer> temp = new ArrayList<>();
            for(int j=0; j<N; j++)
                temp.add(Integer.parseInt(S[j]));
            adj.add(temp);
        }

        Solution ob = new Solution();
        System.out.println(ob.components(adj,N));
    }
}

class Solution {
```

```
            static void DFS(ArrayList<ArrayList<Integer>> graph,
boolean[] visited, int start) {
            ArrayList<Integer> adj = graph.get(start);

        for (int i = 0; i < adj.size(); i++) {
            if (adj.get(i) == 1 && !visited[i]) {
                        visited[i] = true;
                DFS(graph, visited, i);
            }
        }
        }

    int components(ArrayList<ArrayList<Integer>> adj, int N) {
        boolean[] visited = new boolean[N];
            int count = 0;

            for (int i = 0; i < N; i++) {
                if (!visited[i]) {
                    DFS(adj, visited, i);
                    visited[i] = true;
                    count++;
                }
            }

            return count;
    }
};
```