

Problem 2685: Count the Number of Complete Components

Problem Information

Difficulty: Medium

Acceptance Rate: 77.65%

Paid Only: No

Tags: Depth-First Search, Breadth-First Search, Union Find, Graph

Problem Description

You are given an integer `n`. There is an **undirected** graph with `n` vertices, numbered from `0` to `n - 1`. You are given a 2D integer array `edges` where `edges[i] = [ai, bi]` denotes that there exists an **undirected** edge connecting vertices `ai` and `bi`.

Return the number of complete connected components of the graph.

A **connected component** is a subgraph of a graph in which there exists a path between any two vertices, and no vertex of the subgraph shares an edge with a vertex outside of the subgraph.

A connected component is said to be **complete** if there exists an edge between every pair of its vertices.

Example 1:

A diagram of a complete graph with 6 vertices labeled 0, 1, 2, 3, 4, and 5. Every vertex is connected to every other vertex by a straight line, representing a fully connected graph where there is an edge between every pair of vertices.

Input: n = 6, edges = [[0,1],[0,2],[1,2],[3,4]] **Output:** 3 **Explanation:** From the picture above, one can see that all of the components of this graph are complete.

Example 2:

A diagram of a graph with 6 vertices labeled 0, 1, 2, 3, 4, and 5. It contains three distinct components: a triangle formed by vertices 0, 1, and 2; a single isolated vertex 3; and a edge between vertices 4 and 5. Vertex 0 is not connected to any other vertices.

****Input:**** n = 6, edges = [[0,1],[0,2],[1,2],[3,4],[3,5]] ****Output:**** 1 ****Explanation:**** The component containing vertices 0, 1, and 2 is complete since there is an edge between every pair of two vertices. On the other hand, the component containing vertices 3, 4, and 5 is not complete since there is no edge between vertices 4 and 5. Thus, the number of complete components in this graph is 1.

****Constraints:****

* `1 <= n <= 50` * `0 <= edges.length <= n * (n - 1) / 2` * `edges[i].length == 2` * `0 <= ai, bi <= n - 1` * `ai != bi` * There are no repeated edges.

Code Snippets

C++:

```
class Solution {  
public:  
    int countCompleteComponents(int n, vector<vector<int>>& edges) {  
  
    }  
};
```

Java:

```
class Solution {  
public int countCompleteComponents(int n, int[][] edges) {  
  
    }  
}
```

Python3:

```
class Solution:  
    def countCompleteComponents(self, n: int, edges: List[List[int]]) -> int:
```