Problem 1245: Tree Diameter

Problem Information

Difficulty: Medium
Acceptance Rate: 61.18%
Paid Only: Yes
Tags: Tree, Depth-First Search, Breadth-First Search, Graph, Topological Sort

Problem Description

The **diameter** of a tree is **the number of edges** in the longest path in that tree.

There is an undirected tree of `n` nodes labeled from `0` to `n - 1`. You are given a 2D array `edges` where `edges.length == n - 1` and `edges[i] = [ai, bi]` indicates that there is an undirected edge between nodes `ai` and `bi` in the tree.

Return _the**diameter** of the tree_.

Example 1:

Input: edges = [[0,1],[0,2]] **Output:** 2 **Explanation:** The longest path of the tree is the path 1 - 0 - 2.

Example 2:

Input: edges = [[0,1],[1,2],[2,3],[1,4],[4,5]] **Output:** 4 **Explanation:** The longest path of the tree is the path 3 - 2 - 1 - 4 - 5.

Constraints:

* `n == edges.length + 1` * `1 <= n <= 104` * `0 <= ai, bi < n` * `ai != bi`

Code Snippets

C++:

```
class Solution {
public:
int treeDiameter(vector<vector<int>>& edges) {

}
};
```

Java:

```
class Solution {
public int treeDiameter(int[][] edges) {

}
}
```

Python3:

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