Problem 847: Shortest Path Visiting All Nodes

Problem Information

Difficulty: Hard
Acceptance Rate: 65.66%
Paid Only: No
Tags: Dynamic Programming, Bit Manipulation, Breadth-First Search, Graph, Bitmask

Problem Description

You have an undirected, connected graph of `n` nodes labeled from `0` to `n - 1`. You are given an array `graph` where `graph[i]` is a list of all the nodes connected with node `i` by an edge.

Return _the length of the shortest path that visits every node_. You may start and stop at any node, you may revisit nodes multiple times, and you may reuse edges.

Example 1:

![](https://assets.leetcode.com/uploads/2021/05/12/shortest1-graph.jpg)

Input: graph = [[1,2,3],[0],[0],[0]] **Output:** 4 **Explanation:** One possible path is [1,0,2,0,3]

Example 2:

![](https://assets.leetcode.com/uploads/2021/05/12/shortest2-graph.jpg)

Input: graph = [[1],[0,2,4],[1,3,4],[2],[1,2]] **Output:** 4 **Explanation:** One possible path is [0,1,4,2,3]

Constraints:

* `n == graph.length` * `1 <= n <= 12` * `0 <= graph[i].length < n` * `graph[i]` does not contain `i`. * If `graph[a]` contains `b`, then `graph[b]` contains `a`. * The input graph is always connected.

Code Snippets

C++:

```
class Solution {
public:
int shortestPathLength(vector<vector<int>>& graph) {

}
};
```

Java:

```
class Solution {
public int shortestPathLength(int[][] graph) {

}
}
```

Python3:

```
class Solution:
def shortestPathLength(self, graph: List[List[int]]) -> int:
```