LeetCode 797 - All Paths From Source to Target - BFS

Report Issue

Video Solution

In the following video, we'll solve the All Paths From Source to Target problem that we previously solved using DFS. However, this time we will use BFS to do so.

▶ Clarifying Notes

Implementation

C++ Java Python3 Copy

```cpp
class Solution {
public:
    vector<vector<int>> allPathsSourceTarget(vector<vector<int>>& graph) {
        vector<vector<int>> paths;
        if (graph.size() == 0) {
            return paths;
        }

        vector<int> path;
        queue<vector<int>> q;
        path.push_back(0);
        q.push(path);

        while (!q.empty()) {
            vector<int> currentPath = q.front();
            q.pop();
            int node = currentPath.back();
            for (int nextNode : graph[node]) {
                vector<int> tmpPath(currentPath.begin(), currentPath.end());
                tmpPath.push_back(nextNode);
                if (nextNode == graph.size() - 1) {
                    paths.push_back(tmpPath);
                } else {
                    q.push(tmpPath);
                }
            }
        }
```

Complexity Analysis

- Time Complexity: $O(2^V \cdot V)$. Here, V represents the number of vertices.
    - For a graph with V vertices, there could be at most $2^{V-1} - 1$ possible paths to go from the starting vertex to the target vertex. We need $O(V)$ time to build each such path.

- Therefore, a loose upper bound on the time complexity would be $(2^{V-1} - 1) \cdot O(V) = O(2^V \cdot V)$.
- Since we have overlapping between the paths, the actual time spent on the traversal will be lower to some extent.

- Space Complexity: $O(2^V \cdot V)$. The queue can contain $O(2^V)$ paths and each path will take $O(V)$ space. Therefore, the overall space complexity is $O(2^V \cdot V)$.