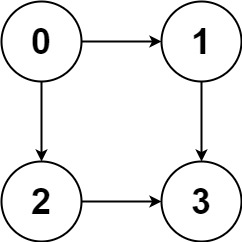
All Paths From Source to Target [LeetCode](https://leetcode.com/problems/all-paths-from-source-to-target/description/)

Given a directed acyclic graph (**DAG**) of n nodes labeled from 0 to n - 1, find all possible paths from node 0 to node n - 1 and return them in **any order**.

The graph is given as follows: graph[i] is a list of all nodes you can visit from node i (i.e., there is a directed edge from node i to node graph[i][j]).

Example:



Output: {{0, 1, 3}, {0, 2, 3}}

Example 1:

A diagram of a network

Description automatically generated

Output: {{0, 4}, {0, 3, 4} . {0, 1, 4}, {0, 1, 3, 2}, {0, 1, 2, 3, 4}}

**Approach 1: Function to find all paths from source to target using BFS traversal**

* **Explanation:**
  + The **allPathSourceToTargetBFS** function uses BFS traversal to explore paths from the source to the target.
  + It maintains a queue (**bfsQueue**) to iteratively explore neighbors.
  + Paths are extended during traversal, and once the target is reached, the path is added to the result.
* **Time Complexity:**
  + **The time complexity is O(V + E), where V is the number of vertices, and E is the number of edges.**
    - **Each vertex and edge are visited once during BFS traversal.**
* **Space Complexity:**
  + **The space complexity is O(V + E), where V is the number of vertices, and E is the number of edges.**
    - **The queue (bfsQueue) and the result paths contribute to space usage.**

**Approach 2: Function to find all paths from source to target using DFS traversal**

* **Explanation:**
  + The **allPathSourceToTargetDFS** function uses DFS traversal to explore paths from the source to the target.
  + It maintains a recursive DFS function (**dfsTraversal**) to explore neighbors.
  + Paths are extended during traversal, and once the target is reached, the path is added to the result.
* **Time Complexity:**
  + **The time complexity is O(V + E), where V is the number of vertices, and E is the number of edges.**
    - **Each vertex and edge are visited once during DFS traversal.**
* **Space Complexity:**
  + **The space complexity is O(V + E), where V is the number of vertices, and E is the number of edges.**
    - **The recursive call stack and the result paths contribute to space usage.**

**Conclusion**The program effectively finds all paths from the source to the target in a directed graph using both BFS and DFS traversal approaches. **These methods provide different perspectives on graph exploration, and the time and space complexity of both approaches are reasonable for typical graph sizes.**