2024-01-27 - Handout – Backtracking

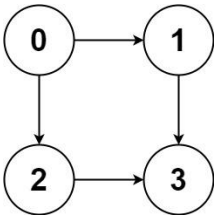
Q1. All paths from source to target

Link: https://leetcode.com/problems/all-paths-from-source-to-target/

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 1:



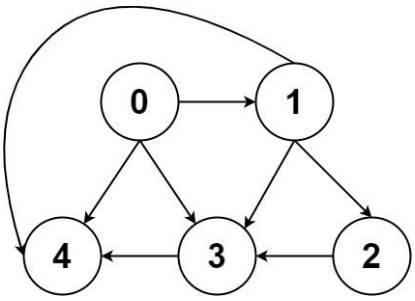
Input: graph = [[1,2],[3],[3],[]]

Output: [[0,1,3],[0,2,3]]

Explanation: There are two paths:

0 -> 1 -> 3 and 0 -> 2 -> 3.

Example 2:



Input: graph = [[4,3,1],[3,2,4],[3],[4],[]]

Output: [[0,4],[0,3,4],[0,1,3,4],[0,1,2,3,4],[0,1,4]]

Constraints:

- n == graph.length
- 2 <= n <= 15
- 0 <= graph[i][j] < n
- graph[i][j] != I (i.e., there will be no self-loops).
- All the elements of graph[i] are **unique**.
- The input graph is **guaranteed** to be a **DAG**.

Q2. Word Break II

Link: https://leetcode.com/problems/word-break-ii

Given a string s and a dictionary of strings wordDict, add spaces in s to construct a sentence where each word is a valid dictionary word. Return all such possible sentences in **any order**.

Note that the same word in the dictionary may be reused multiple times in the segmentation.

Example 1:

> **Input:** s = "catsanddog", wordDict = ["cat","cats","and","sand","dog"]
> **Output:** ["cats and dog","cat sand dog"]

Example 2:

> **Input:** s = "pineapplepenapple", wordDict = ["apple","pen","applepen","pine","pineapple"]
> **Output:** ["pine apple pen apple","pineapple pen apple","pine applepen apple"]
> **Explanation:** Note that you are allowed to reuse a dictionary word.

Example 3:

> **Input:** s = "catsandog", wordDict = ["cats","dog","sand","and","cat"]
> **Output:** []

Constraints:

- 1 <= s.length <= 20
- 1 <= wordDict.length <= 1000
- 1 <= wordDict[i].length <= 10
- s and wordDict[i] consist of only lowercase English letters.
- All the strings of wordDict are **unique**.
- Input is generated in a way that the length of the answer doesn't exceed 10^5.