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

We are given a directed acyclic graph!

So we recursively traverse all possible path and backtrack to get all possible paths.

Since directed acyclic graph we won't need a visited array as there will be no case of revisiting a node.

Code:

class Solution {

public List<List<Integer>> allPathsSourceTarget(int[][] graph) {

List<List<Integer>> res = new ArrayList<>();

List<Integer> sub = new ArrayList<Integer>();

sub.add(0);

helper(graph, 0, res, sub);

return res;

}

public void helper(int[][] graph, int start, List<List<Integer>> res, List<Integer> sub){

if(start == graph.length-1){

res.add(new ArrayList<>(sub));

}

else

for(int n : graph[start]){

sub.add(n);

helper(graph, n, res, sub);

sub.remove(sub.size()-1);

}

return;

}

}

Github Link :<https://lnkd.in/ecwtJeaz>