SDE-1 Problems - by Kushagra

Q1: (Topic - Cycle detection in Directed graph, Topo sort)

At FLIPKART, becoming a successful SDE involves completing a series of modules, each designed to build upon the last. These modules are labeled from 0 to `numModules - 1`, but there's a catch—you can't just complete them in any order. Some modules require you to finish others first, much like prerequisites.

You're given a list of these dependencies, where each pair `[ai, bi]` indicates that module `bi` must be completed before module `ai`. For example, if you see `[0, 1]`, it means you need to finish module 1 before you can start module 0.

Basic question:-

If you can complete all the courses, then return a message to your mentor - "Modules Completed Successfully" otherwise if you find that there are dependencies in a way that it can't be completed then return your mentor with a message "Modules Incomplete"

Example 1:

Input: `numModules` = 2, prerequisites = [[1,0]]

Output: Modules Completed Successfully

Explanation: There are a total of 2 modules to take.

To take module 1 you should have finished module 0. So it is possible.

Example 2:

Input: `numModules` = 2, prerequisites = [[1,0],[0,1]]

Output: Modules Incomplete

Explanation: There are a total of 2 modules to take.

To take course 1 you should have finished modules 0, and to take modules 0 you should also have finished modules 1. So it is impossible.

Breaking Case:

Solution:- Figure out the cycle in the graph

Follow-up:-

Your task is to figure out the correct order to complete all the modules. If you can map out this sequence, you'll successfully navigate through all the required knowledge and skills to become an SDE-1. If it is impossible to navigate through, then consult your mentor with an empty response for the courses to be completed.

Example 1:

Input: `numModules = 2`, `dependencies = [[1, 0]]` Output: `[0, 1]` Explanation: There are 2 modules to complete. To take module 1, you need to finish module 0 first. So the correct order of modules is `[0, 1]`.

Example 2:

Input: `numModules = 4`, `dependencies = [[1, 0], [2, 0], [3, 1], [3, 2]]` Output: `[0, 2, 1, 3]` Explanation: There are 4 modules to complete. To take module 3, you need to finish both modules 1 and 2. Both of these must be completed after finishing module 0. One correct order of modules is `[0, 1, 2, 3]`. Another valid order is `[0, 2, 1, 3]`.

Example 3:

Input: `numModules = 1`, `dependencies = []` Output: `[0]` Explanation: With only 1 module and no prerequisites, you can complete it directly. The correct order is `[0]`.

Solution:- Write the topo sort, otherwise return empty array

One more follow-up can be - if we have multiple answers, print the ones where we get the smallest modules first.

Example 2 - these two are the orders for example 2 - `[0, 1, 2, 3]`, `[0, 2, 1, 3]` but `return everytime the shortest one first - [0, 1, 2, 3]`

Solution - Can use pq (to be checked)

Links to the problem - Course Schedule 1, Course Schedule 2

Q2: (Topic - DP)

You're a key player for Real Madrid, gearing up for a critical match. Your coach hands you two advanced energy drinks: Prime and Red Bull. Each drink provides a different energy boost per hour, represented by the arrays `energyDrinkA` (Prime) and `energyDrinkB` (Red Bull) over the next `n` hours.

To maintain peak performance, you can choose to drink either Prime or Red Bull each hour. However, if you decide to switch from one to the other, you'll need to take a one-hour break, during which you won't receive any energy boost.

Your mission is to determine the best strategy to maximize your energy levels, ensuring you're at your best for every crucial moment of Real Madrid's big game.

Example 1:

Input: `Prime = [1, 3, 1]`, `Red Bull = [3, 1, 1]`

Output: `5`

Explanation: To achieve a total energy boost of 5, you can choose to drink only Prime or only Red Bull throughout the entire period.

Example 2:

Input: `Prime = [4, 1, 1]`, `Red Bull = [1, 1, 3]`

Output: `7`

Explanation: To achieve a total energy boost of 7:

- Drink Prime for the first hour.
- Switch to Red Bull for the second hour, which requires a one-hour cleanse period without any energy boost.
- Continue with Red Bull for the third hour to gain its energy boost.

Solution - Simple DP on the choices

Why won't the greedy approach work here? (ASK) - This problem has optimal substructure, meaning the solution to the problem can be constructed from optimal solutions to its subproblems. The problem is also a classic example of dynamic programming problems, where the solution involves storing intermediate results and making decisions based on previously computed values. Greedy algorithms do not handle overlapping subproblems and optimal substructure effectively.

Q4: The Enchanted Tree Challenge

In a mystical forest, you come across an enchanted tree with a peculiar ability: its leaves glow with a faint, magical light. A wise old sage sets forth a challenge:

"Discover the secret of the enchanted tree by plucking its glowing leaves layer by layer. Each time you remove the leaves, new ones will appear on the next layer, until only the tree's trunk remains. Your task is to identify and display the glowing leaves at every stage before you remove them, revealing the next hidden layer of leaves. Repeat this until the tree's secret, the lone magical root, is all that remains."

Your goal is to write a program to help you unravel the mystery by:

1. Printing the current layer of glowing leaves (leaf nodes).
2. Simulating the magical removal of these leaves.
3. Repeating the process until only the enchanted root remains.

Will you accept this challenge and uncover the magic within?

For example:

```
Unset
          1
        /  \
      2     3
    /  \
  4     5
```

Print: [4 5 3], [2], [1]

https://leetcode.com/problems/find-leaves-of-binary-tree (premium)

Fast Follow Up - **Write code for not Binary tree but N-ary tree.**

Q3: (Brute + Optimization) (ADHOC)

In 2050, Indian football players have unique registration numbers known as `capNumbers`. These numbers are crucial for tracking their performance on the field.

Imagine you're a sports analyst trying to find out which players have registration numbers that could be considered "almost equal." Two registration numbers, `x` and `y`, are deemed almost equal if you can make them identical by performing the following operation at most twice: choose either `x` or `y` and swap any two digits within that number.

Your challenge is to identify how many pairs of these `capNumbers` are almost equal. This would help determine which players have numbers that, though different, could be closely related through a few simple digit swaps. Remember, leading zeros are allowed after performing the operation.

Example 1:

Input: `capNumbers = [1023, 2310, 2130, 213]`

Output: `4`

Explanation:

The almost equal pairs of registration numbers are:

- **1023 and 2310**: By swapping the digits 1 and 2, and then the digits 0 and 3 in 1023, you get 2310.
- **1023 and 213**: By swapping the digits 1 and 0, and then the digits 1 and 2 in 1023, you get 0213, which is 213.
- **2310 and 213**: By swapping the digits 2 and 0, and then the digits 3 and 2 in 2310, you get 0213, which is 213.
- **2310 and 2130**: By swapping the digits 3 and 1 in 2310, you get 2130.

Example 2:

Input: `capNumbers = [1, 10, 100]`

Output: `3`

Explanation:

The almost equal pairs of registration numbers are:

- **1 and 10**: By swapping the digits 1 and 0 in 10, you get 01, which is 1.
- **1 and 100**: By swapping the second 0 with the digit 1 in 100, you get 001, which is 1.
- **10 and 100**: By swapping the first 0 with the digit 1 in 100, you get 010, which is 10.

Solution for lesser constraints : - BRUTE THIS

Follow up -

Constraints:

2 <= nums.length <= 5000
1 <= nums[i] < 107

Add these constraints, O(N^2*D^4) - will not work
Do a decision on every digit and then check if it is present in the list - O(N*D^4)