I participated in three different contests ranging from easy (div3) all the way to division1. The most common occurring issue I had was converting an initial solution into something that met the run time requirement. An area I aim to improve is studying/understanding tools for improving efficiency. What I mean by this is a better understanding of certain data structures. I had a very hard time converting a solution into a data structure that utilizes the problem structure efficiently.

Problem #1. Hash Table Related

Given a string containing digits from `2-9` inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

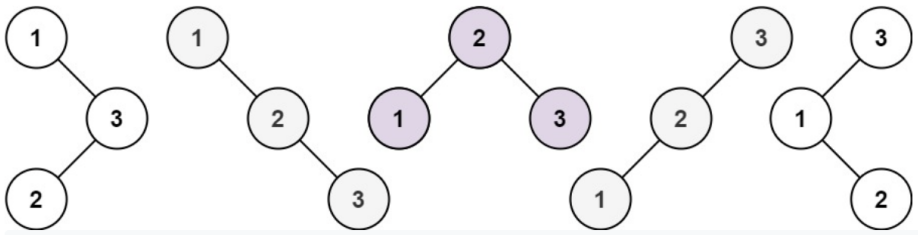
A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

https://leetcode.com/problems/letter-combinations-of-a-phone-number/

Problem #2 Binary Tree Related

Given an integer `n`, return *all the structurally unique **BST**'s (binary search trees), which has exactly* `n` *nodes of unique values from* `1` *to* `n`. Return the answer in **any order**.

Example 1:




```
Input: n = 3
Output: [[1,null,2,null,3],[1,null,3,2],[2,1,3],[3,1,null,null,2],[3,2,null,1]]
```

https://leetcode.com/problems/unique-binary-search-trees-ii/

Problem #3 Graph Related

133. Clone Graph

Medium 4100 2001 Add to List Share

Given a reference of a node in a **connected** undirected graph.

Return a **deep copy** (clone) of the graph.

Each node in the graph contains a value (`int`) and a list (`List[Node]`) of its neighbors.

```
class Node {
    public int val;
    public List<Node> neighbors;
}
```

Test case format:

For simplicity, each node's value is the same as the node's index (1-indexed). For example, the first node with `val == 1`, the second node with `val == 2`, and so on. The graph is represented in the test case using an adjacency list.

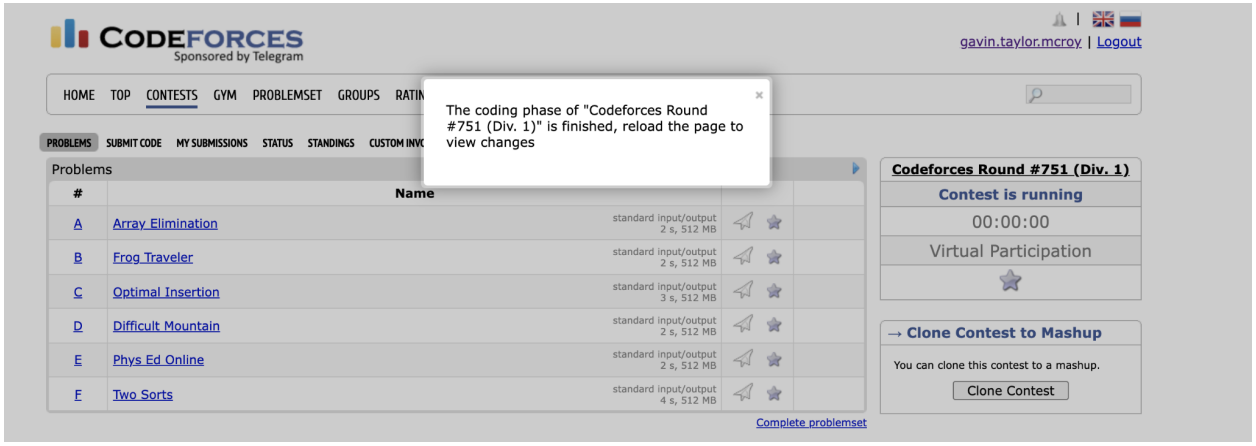
An adjacency list is a collection of unordered **lists** used to represent a finite graph. Each list describes the set of neighbors of a node in the graph.

The given node will always be the first node with `val = 1`. You must return the **copy of the given node** as a reference to the cloned graph.

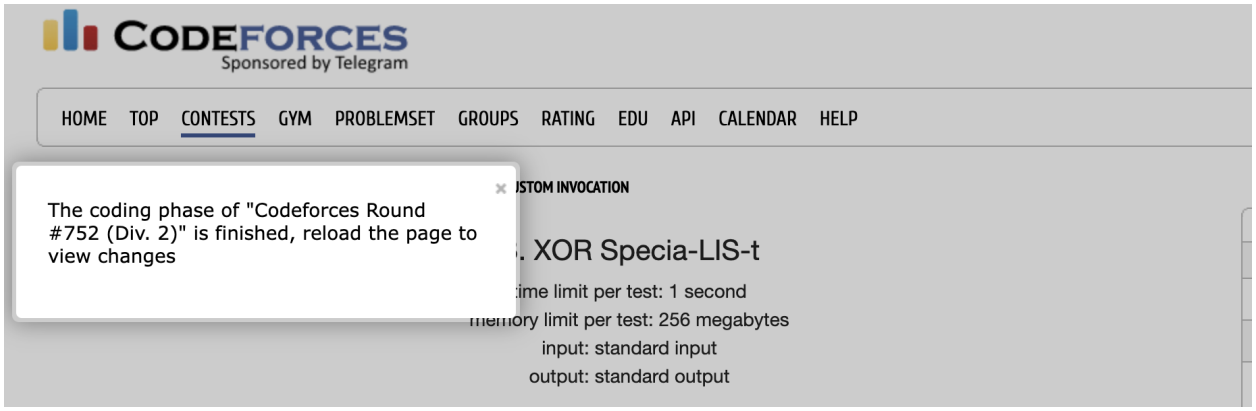
https://leetcode.com/problems/clone-graph/

Competitions completed:

Division 1 (Got destroyed)



Division 2 (Could come up with brute force algos but not fast ones)



Division 3 (Not bad)