RECURSION Concepts & Qns

Video
10

"मैं, DSA की शपथ लेता हूँ कि मैं जो पढ़ाउगा बहोत अच्छे से पढ़ाउगा।"

Facebook
Instagram] → codestorywithMIK

(Twitter) → CSwithMIK

codestorywithMIK →

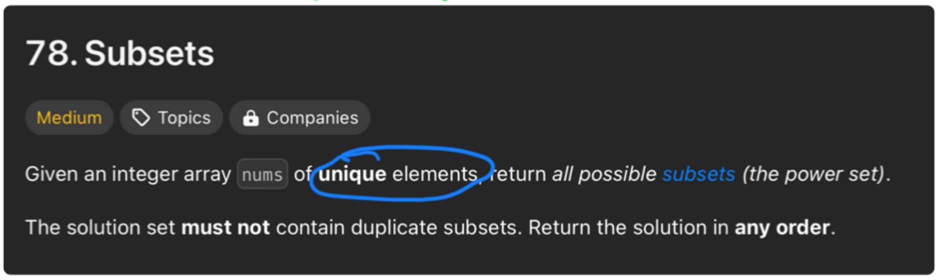
Motivation
(भाषण)

"Don't be discouraged by the complexity of data & algorithms. Break down the concepts into smaller, manageable pieces."

"That's how you can crack any tough Problem...

#CodestorywithMIK ...

MICROSOFT



78. Subsets

Medium · Topics · Companies

Given an integer array `nums` of **unique** elements, return *all possible subsets (the power set)*.

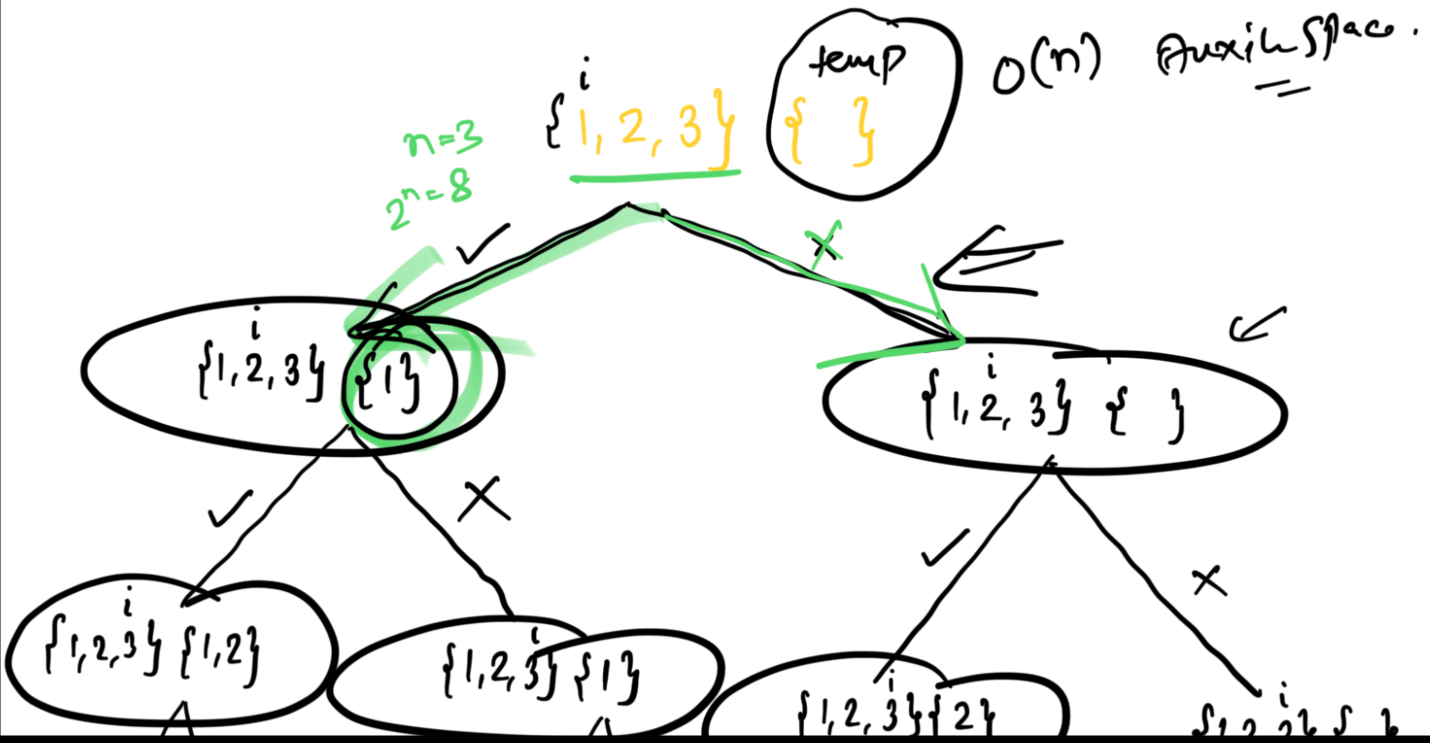
The solution set **must not** contain duplicate subsets. Return the solution in **any order**.

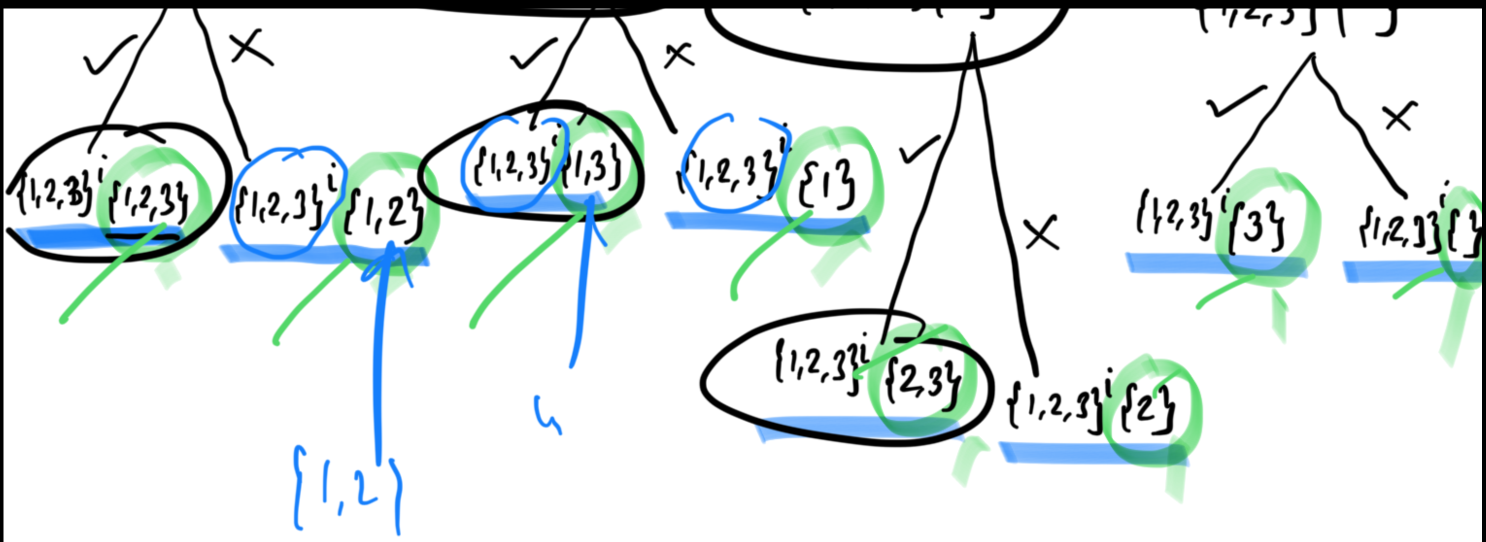
Example :- nums = {1, 2, 3}

take
not-take

Output = [{ }, {1}, {2}, {3}, {1,2}, {1,3}, {2,3}, {1,2,3}]

Options → Recursion
Hint.





(•) Tree Diagram
- ↳ options
- ↳ Base Case
- ↳ System stack space → $O(n)$
- ↳ Auxi → $O(n)$ to store subsets
- ↳ T.C = (2^n)

$2^n * n$

Space

→ Every element has 2 options.

→ we've n elements.

Recursion Magic·

(Trust)

Backtracking: Choose karo, Explore karo and Us choice ko undo kardo phir explore karo

```
Solve (nums , i , temp)    {
    if (i >= nums.size()) {
        result.push_back (temp);
    }

    temp.push_back (nums [i]);      // Take
    Solve (nums, i+1 , temp);       // Trust
    temp.pop_back ();               // not-take
    Solve (nums, i+1, temp);        // Trust

}
```

Backtracking

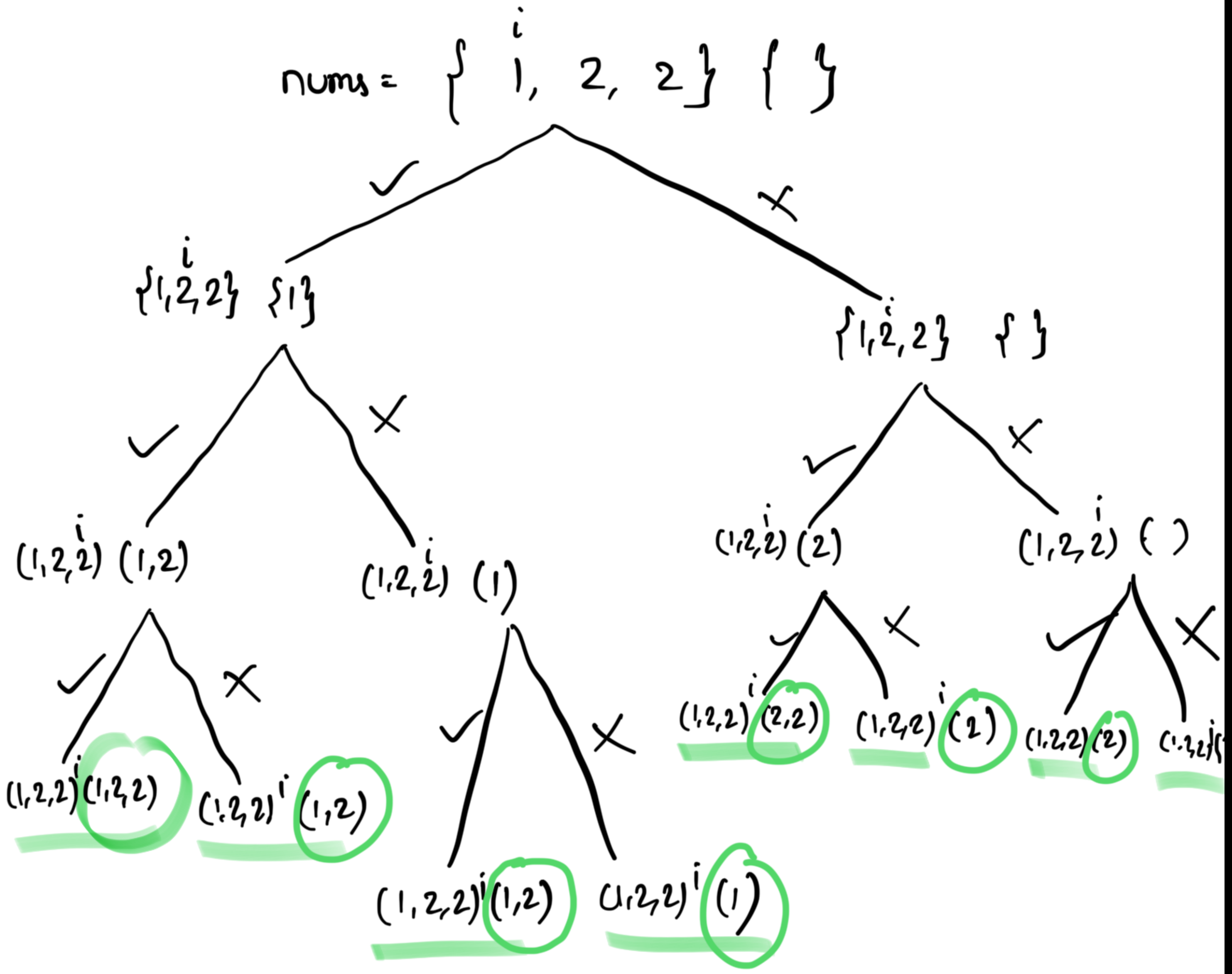
leap of faith.

Subset - II Problem discussed here,

what if we have

duplicate elements ?

duplicate elements.



(1, 2, 2)

(1,2) * duplicate

(1)

(2,2)

(2) * dupli-

```
// Approach-1
// T.C : O(2^n)
// S.C :  O(2^n*length of each subset) to store each subset
// The recursion stack space complexity is O(N) , the maximum depth of
the recursion is N, where N is the length of the input array


class Solution {
public:
    vector<vector<int>> result;

    void solve(vector<int>& nums, int idx, vector<int>& temp) {
        if(idx >= nums.size()) {
            result.push_back(temp);
            return;
        }

        temp.push_back(nums[idx]);     // Take ith element
        solve(nums, idx+1, temp);        // Explore pick element
        temp.pop_back();                  // Not take choice
        solve(nums, idx+1, temp);    // Explore
    }

    vector<vector<int>> subsets(vector<int>& nums) {
        vector<int> temp;             // Store subset
        solve(nums, 0, temp);
        return result;
    }
};
```

B

20-MAY | IGDTUW -- PODCAST add into AtoZ readme file