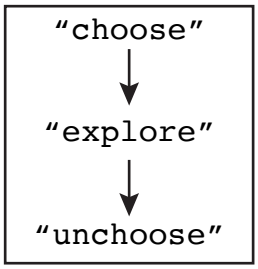
78. Subsets
Medium

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.



"choose"

Select one number (nums[i]) by adding it to the 'subset' stack that holds the current branch.

"explore"

Recursively call the 'backtrack' function which will carry on the recursion and pass along the next index, which references the next number in the nums list.

"unchoose"

Remove the recently added number and go back to step 1 to explore another sub-branch.

The termination condition of the comparison between the length of the branch and the length of the original list to permute will
1) add a copy of the current subsets to the 'results' list
and
2) stop the recursion.

```
class Solution:
    def findOrder(self, nums: List[int]) -> List[List[int]]:
        answer = []
        subset = []

        def backtrack(i):
            print(subset)
            if i >= len(nums):
                answer.append(subset.copy())
                return
            subset.append(nums[i])
            backtrack(i + 1)
            subset.pop()
            backtrack(i + 1)

        backtrack(0)

        return answer                    i = 0
```

Example 1:
input:
nums = [1,2,3]
output:
[[1, 2, 3], [1, 2], [1, 3], [1], [2, 3], [2], [3], []]

print(subset)	**calls** *1st*	**recursive level** *2nd*	*3rd*	*4th*	**subset.copy() is appended to 'answer'**
[]	●				
[1]		●			
[1, 2]			●		
[1, 2, 3]				●	answer.append(subset.copy())
[1, 2]				●	answer.append(subset.copy())
[1]			●		
[1, 3]				●	answer.append(subset.copy())
[1]				●	answer.append(subset.copy())
[]		●			
[2]			●		
[2, 3]				●	answer.append(subset.copy())
[2]				●	answer.append(subset.copy())
[]			●		
[3]				●	answer.append(subset.copy())
[]				●	answer.append(subset.copy())