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.

$$\begin{tikzcd}
\text{“choose”} \arrow[d] \\
\text{“explore”} \arrow[d] \\
\text{“unchoose”}
\end{tikzcd}$$

```
class Solution:
    def subsets_1(self, nums: List[int]) -> List[List[int]]:
        answer = []
        subset = []

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

“choose”

Select one number by adding it to a stack that holds the current branch.

“explore”

Recursively call the ‘explore_helper’ function which will carry on the recursion and pass along the stack which contains the numbers chosen in the current branch.

“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
add the current branch to the ‘results’ list
and
stop the recursion.

```
    def subsets_2(self, nums: List[int]) -> List[List[int]]:
        def backtrack(chosen, remaining, res):
            if not remaining:
                res.append(chosen[:])
                return
            d = remaining.pop(0)
            #choose
            chosen.append(d)
            #explore
            backtrack(chosen, remaining, res)
            chosen.pop()
            backtrack(chosen, remaining, res)
            #unchoose
            remaining.insert(0, d)

        res = []
        chosen = []
        explore(chosen, nums, res)
        return res
```