Subsets

Question:

https://leetcode.com/problems/subsets/

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 1:

```
Input: nums = [1,2,3]
Output: [[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]
```

Example 2:

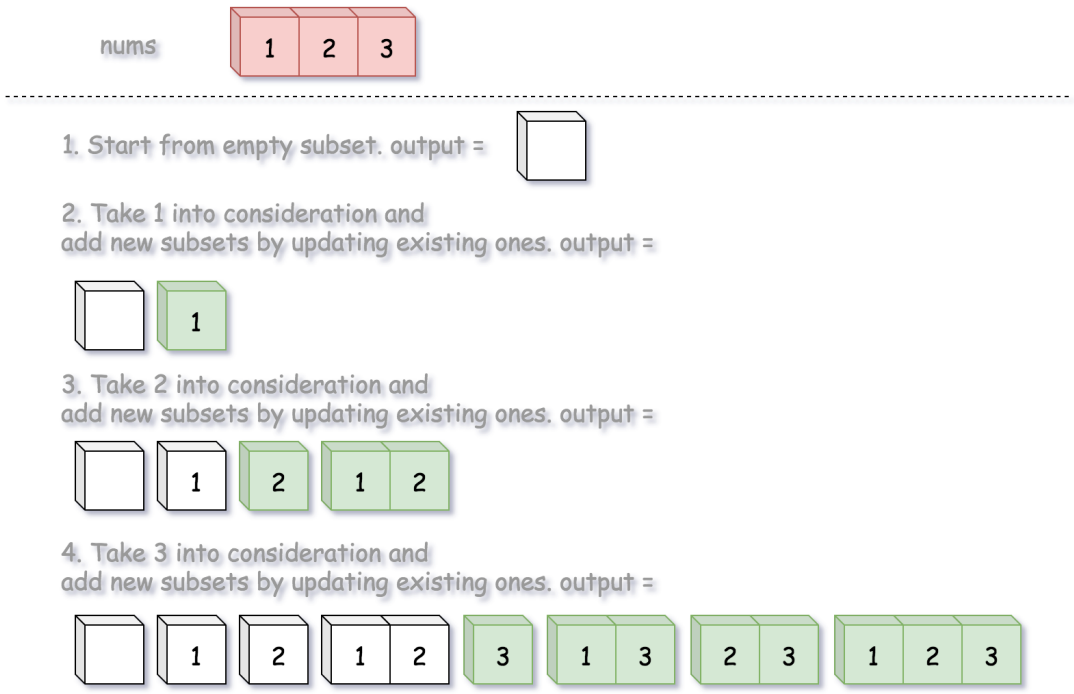
```
Input: nums = [0]
Output: [[],[0]]
```

Constraints:

- `1 <= nums.length <= 10`
- `-10 <= nums[i] <= 10`
- *All the numbers of `nums` are unique.*

Approach 1:

Start from an empty subset in the output list. At each step, take a new integer into consideration and generate new subsets from the existing ones. This is a Breath-First approach, where the output array is acting as a queue.



Solution 1:

```
def subsets(self,nums):
        out=[[]]
        for i in nums:
                out+=[j+[i] for j in out]
        return out
```

Time Complexity: O(N * 2^N)

Space Complexity: O(N * 2^N)

Approach 2:

Optimised the previous Breadth-First Search approach to a recursive approach using Depth-First Search.

Solution 2:

```
def subsets(self, nums):
        res = []
        self.dfs(sorted(nums), 0, [], res)
        return res

        def dfs(self, nums, index, path, res):
                res.append(path)
                for i in range(index, len(nums)):
                        self.dfs(nums, i+1, path+[nums[i]], res)
```

Time Complexity: O(N * 2^N)

Space Complexity: O(N)