# 78 Subsets (link)

## Description

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**.

(scroll down for solution)

# Solution

*Language: cpp*

**Status: Accepted**

```cpp
#include <vector>

using namespace std;

class Solution {
public:
    void backtrack(vector<int>& nums, vector<vector<int>>& result, vector<int>& subse
        // Добавляем текущее подмножество в результат
        result.push_back(subset);

        for (int i = start; i < nums.size(); ++i) {
            // Добавляем элемент в текущее подмножество
            subset.push_back(nums[i]);
            // Рекурсивно вызываем функцию для следующего элемента
            backtrack(nums, result, subset, i + 1);
            // Удаляем последний элемент для бэктреккинга
            subset.pop_back();
        }
    }

    vector<vector<int>> subsets(vector<int>& nums) {
        vector<vector<int>> result;
        vector<int> subset;
        backtrack(nums, result, subset, 0);
        return result;
    }
};
```