

# 078. Subsets

## 078 Subsets

- Backtracking
- Bit Manipulation

### Description

Given a set of **distinct** integers, *nums*, return all possible subsets (the power set).

**Note:** The solution set must not contain duplicate subsets.

For example,

If *nums* = [1,2,3] , a solution is:

```
[
  [3],
  [1],
  [2],
  [1,2,3],
  [1,3],
  [2,3],
  [1,2],
  []
]
```

### 1. Thought line

(1) When vector& nums.empty(), result should be [ [] ].

### 2. Backtracking

```
1 class Solution {
2 private:
3     void backtrackingPowerSet(vector<vector<int>>& result, vector<int>& temp, int st, vector<int>& nums){
4         // put push action here for corner case_(1)
5         result.push_back(temp);
6         if (st>nums.size()-1) return;
7         for (int i = st; !nums.empty() && i<=nums.size()-1; ++i){
8             temp.push_back(nums[i]);
9             backtrackingPowerSet(result, temp, i+1, nums);
10            temp.pop_back();
11        }
12    }
13 public:
14     vector<vector<int>> subsets(vector<int>& nums) {
15         vector<vector<int>> result;
16         vector<int> temp;
17         if (nums.empty()) return result;
18         backtrackingPowerSet(result, temp, 0, nums);
19         return result;
20     }
21 };
```

### 3. Bit Manipulation

