

# 039. Combination Sum

## 039 Combination Sum

- BackTracking+array

### Description

Given a **set** of candidate numbers (**C**) (**without duplicates**) and a target number (**T**), find all unique combinations in **C** where the candidate numbers sums to **T**.

The **same** repeated number may be chosen from **C** unlimited number of times.

### Note:

- All numbers (including target) will be positive integers.
- The solution set must not contain duplicate combinations.

For example, given candidate set `[2, 3, 6, 7]` and target `7`,

A solution set is:

```
[
  [7],
  [2, 2, 3]
]
```

### 1. Thought line

### 2. BackTracking+array

```
1 class Solution {
2 private:
3     void backTrackingSum(vector<int>& nums, int target, int sum, int st, vector<vector<int>>& result, vector<int>& temp){
4         if (sum == target) {
5             result.push_back(temp);
6             return;
7         }
8         if (st>=nums.size() || sum > target || sum+nums[st]>target) return;
9         for (int i = st; i<=nums.size()-1; ++i){
10             temp.push_back(nums[i]);
11             backTrackingSum(nums, target, sum+nums[i], i, result, temp);
12             temp.pop_back();
13             while(i+1<=nums.size()-1&&nums[i+1]==nums[i]) ++i;
14         }
15     }
16
17 public:
18     vector<vector<int>> combinationSum(vector<int>& candidates, int target) {
19         vector<vector<int>> result;
20         vector<int> temp;
21         sort(candidates.begin(),candidates.end());
22         backTrackingSum(candidates, target, 0, 0, result, temp);
23         return result;
24     }
25 };
```

