0040. 组合总和 II

ITCharge 大约 2 分钟

- 标签：数组、回溯
- 难度：中等

题目链接

- 0040. 组合总和 II - 力扣

题目大意

描述：给定一个数组 `candidates` 和一个目标数 `target` 。

要求：找出 `candidates` 中所有可以使数字和为目标数 `target` 的组合。

说明：

- 数组 `candidates` 中的数字在每个 中只能使用一次。
- $1 \leq candidates.length \leq 100$。
- $1 \leq candidates[i] \leq 50$。

示例：

- 示例 1：

```py
输入：candidates = [10,1,2,7,6,1,5], target = 8,
输出：
[
[1,1,6],
[1,2,5],
[1,7],
[2,6]
]
```

- 示例 2：

```py
输入：candidates = [2,5,2,1,2], target = 5,
输出：
[
[1,2,2],
[5]
]
```

解题思路

思路 1：回溯算法

跟「0039. 组合总和 」不一样的地方在于本题不能有重复组合，所以关键步骤在于去重。

在回溯遍历的时候，下一层递归的 `start_index` 要从当前节点的后一位开始遍历，即 `i + 1` 位开始。而且统一递归层不能使用相同的元素，即需要增加一句判断 `if i > start_index and candidates[i] == candidates[i - 1]: continue` 。

思路 1：代码

```py
class Solution:
    res = []
    path = []
    def backtrack(self, candidates: List[int], target: int, sum: int, start_index: int):
        if sum > target:
            return
        if sum == target:
            self.res.append(self.path[:])
            return

        for i in range(start_index, len(candidates)):
            if sum + candidates[i] > target:
                break
            if i > start_index and candidates[i] == candidates[i - 1]:
                continue
            sum += candidates[i]
            self.path.append(candidates[i])
            self.backtrack(candidates, target, sum, i + 1)
```

```
            sum -= candidates[i]
            self.path.pop()

    def combinationSum2(self, candidates: List[int], target: int) ->
List[List[int]]:
        self.res.clear()
        self.path.clear()
        candidates.sort()
        self.backtrack(candidates, target, 0, 0)
        return self.res
```

思路 1：复杂度分析

- **时间复杂度**：$O(2^n \times n)$，其中 n 是数组 `candidates` 的元素个数，2^n 指的是所有状态数。
- **空间复杂度**：$O(target)$，递归函数需要用到栈空间，栈空间取决于递归深度，最坏情况下递归深度为 $O(target)$，所以空间复杂度为 $O(target)$。