Problem 39: Combination Sum

Problem Information

Difficulty: Medium
Acceptance Rate: 75.63%
Paid Only: No
Tags: Array, Backtracking

Problem Description

Given an array of **distinct** integers `candidates` and a target integer `target`, return _a list of all**unique combinations** of _`candidates` _where the chosen numbers sum to_`target` _._ You may return the combinations in **any order**.

The **same** number may be chosen from `candidates` an **unlimited number of times**. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

The test cases are generated such that the number of unique combinations that sum up to `target` is less than `150` combinations for the given input.

Example 1:

Input: candidates = [2,3,6,7], target = 7 **Output:** [[2,2,3],[7]] **Explanation:** 2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used multiple times. 7 is a candidate, and 7 = 7. These are the only two combinations.

Example 2:

Input: candidates = [2,3,5], target = 8 **Output:** [[2,2,2,2],[2,3,3],[3,5]]

Example 3:

Input: candidates = [2], target = 1 **Output:** []

Constraints:

* `1 <= candidates.length <= 30` * `2 <= candidates[i] <= 40` * All elements of `candidates` are **distinct**. * `1 <= target <= 40`

Code Snippets

C++:

```
class Solution {
public:
vector<vector<int>> combinationSum(vector<int>& candidates, int target) {

}
};
```

Java:

```
class Solution {
public List<List<Integer>> combinationSum(int[] candidates, int target) {

}
}
```

Python3:

```
class Solution:
def combinationSum(self, candidates: List[int], target: int) ->
List[List[int]]:
```