# **Leetcode 39 – Combination Sum**

## Problem Understanding

**Leetcode 39. Combination Sum**  
Given an array of **distinct positive integers** candidates and an integer target, return **all unique combinations** of candidates where the **chosen numbers sum to target**.  
You may use the **same number unlimited times**.

### Key Constraints:

* All numbers are **positive**.
* You can reuse elements **multiple times**.
* **Order of numbers in a combination doesn’t matter**.
* You must avoid duplicate combinations in the final output.

## Optimized Java Solution (Backtracking)

import java.util.\*;

public class Solution {

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

List<List<Integer>> result = new ArrayList<>();

backtrack(0, candidates, target, new ArrayList<>(), result);

return result;

}

private void backtrack(int start, int[] candidates, int target,

List<Integer> path, List<List<Integer>> result) {

if (target == 0) {

result.add(new ArrayList<>(path));

return;

}

if (target < 0) return;

for (int i = start; i < candidates.length; i++) {

path.add(candidates[i]); // choose

backtrack(i, candidates, target - candidates[i], path, result); // explore

path.remove(path.size() - 1); // un-choose

}

}

}

## Dry Run Using Table

**Input:** candidates = [2, 3, 6, 7], target = 7  
(sorted for easier understanding, though not required)

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Path | Remaining Target | Action |
| 1 | [2] | 5 | continue |
| 2 | [2, 2] | 3 | continue |
| 3 | [2, 2, 2] | 1 | continue |
| 4 | [2, 2, 2, 2] | -1 | backtrack |
| 5 | [2, 2, 3] | 0 | ✅ add to result |
| 6 | [2, 3] | 2 | continue |
| 7 | [2, 6] | -1 | backtrack |
| 8 | [3] | 4 | continue |
| 9 | [3, 3] | 1 | continue |
| 10 | [3, 6] | -2 | backtrack |
| 11 | [6] | 1 | continue |
| 12 | [7] | 0 | ✅ add to result |

### Final Output:

[

[2, 2, 3],

[7]

]

## Time / Space Complexity

|  |  |  |
| --- | --- | --- |
| Metric | Value | Why |
| Time | O(2^T) | T = target, exponential paths |
| Space | O(T) | Max depth of recursion stack |
| Output Size | Depends on input | Each valid combination is a list of integers |

## Alternate Approaches

|  |  |
| --- | --- |
| Approach | Notes |
| Backtracking (standard) | Simple, handles reuse well |
| DP (2D array) | Overkill here, better for "how many ways" questions |
| Sorting + Greedy | Doesn’t apply, as you need exact target sums |