

Q2. 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.

Problem Link : <https://leetcode.com/problems/combination-sum/>

Approach :

- Initially, the index will be 0, target as given and the list will be empty.
- Now there are 2 options viz to pick or not pick the current index element.
- If you pick the element, again come back at the same index as multiple occurrences of the same element is possible so the target reduces to $\text{target} - \text{arr}[\text{index}]$ (where $\text{target} - \text{arr}[\text{index}] \geq 0$).
- If you decide not to pick the current element, move on to the next index and the target value stays as it is.
- While backtracking makes sure to remove the last element.
- Keep on repeating this process while $\text{index} < \text{size of the array}$ for a particular recursion call.
- When the target value becomes 0, you have found a valid combination. Add it to your result list.
- If the $\text{target} < 0$, that means your combination sum has exceeded the target, and the combination is invalid, so you return.

Example :

Input: candidates = [2,3,6,7], target = 7

Output: [[2,2,3],[7]]

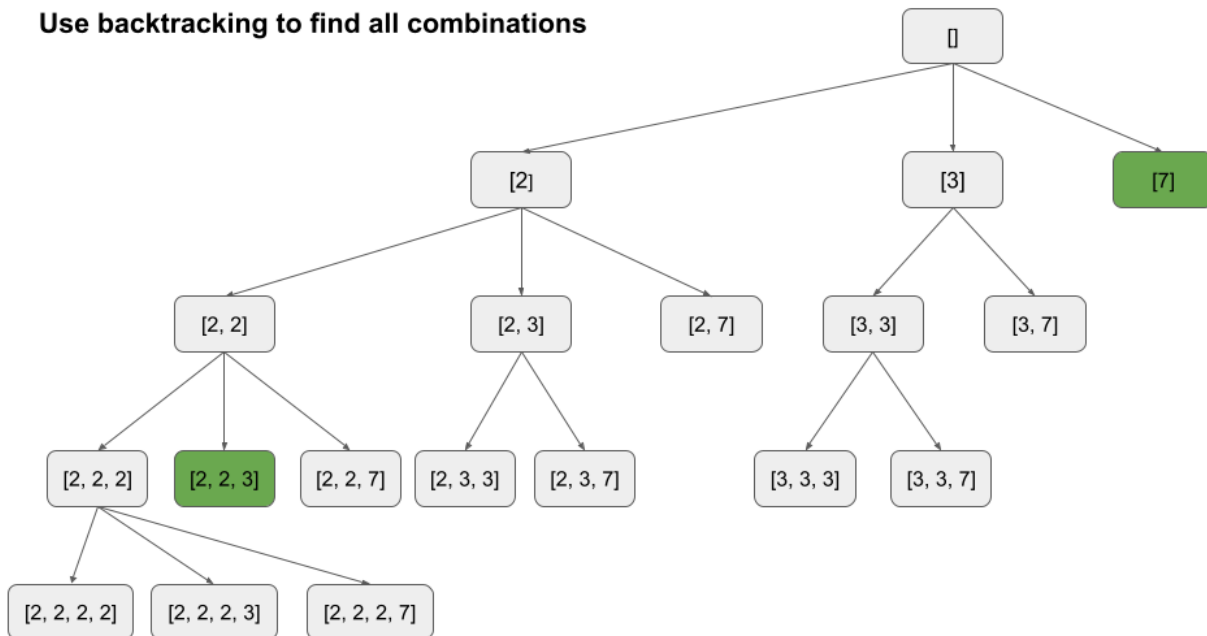
Explanation:

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.

Use backtracking to find all combinations



Solution :

```

class Solution {
    public void backtrack(List<List<Integer>> result, List<Integer> temp ,int[]
candidates , int target , int start){
        if(target < 0)
            return;
        else if(target==0){
            result.add(new ArrayList<>(temp));
        }
        for(int i=start ; i<candidates.length ; i++){
            temp.add(candidates[i]);
            backtrack(result,temp,candidates,target-candidates[i],i);
            temp.remove(temp.size()-1);
        }
    }

    public List<List<Integer>> combinationSum(int[] candidates, int target){
        List<List<Integer>> result = new ArrayList<>();
        backtrack(result,new ArrayList<Integer>(),candidates ,target ,0);
        return result;
    }
}

```