39. Combination Sum

01 February 2022 07:52 PM

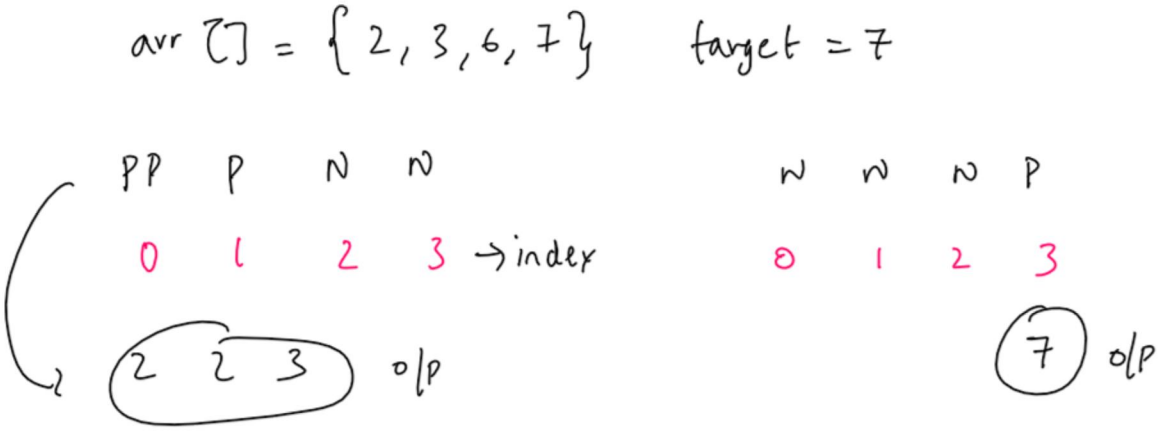
Given an array of **distinct** integers `candidates` and a target integer `target`, return *a list of all* ***unique combinations*** *of* `candidates` *where the chosen numbers sum to* `target`. You may return the combinations in **any order**.

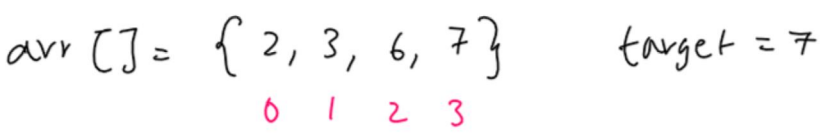
The **same** number may be chosen from `candidates` an **unlimited number of times**. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

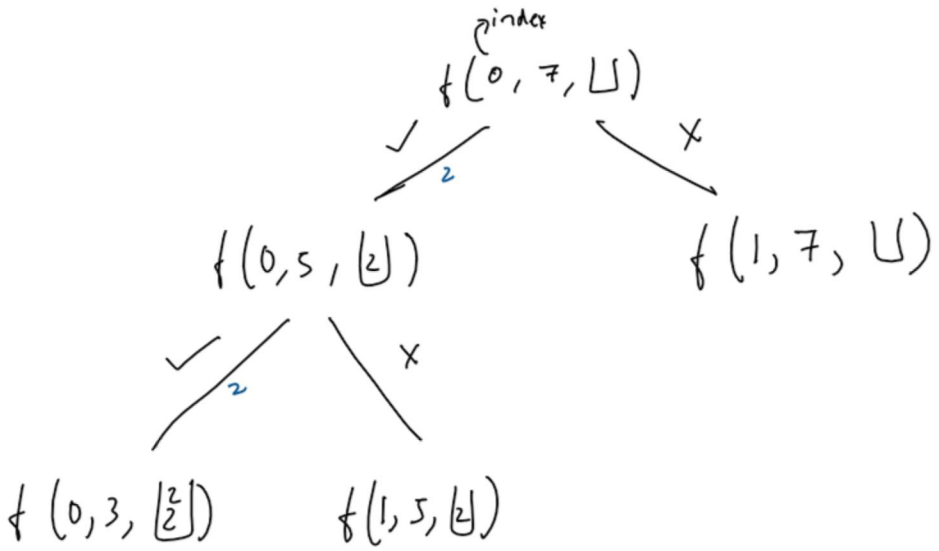
It is **guaranteed** that the number of unique combinations that sum up to `target` is less than `150` combinations for the given input.

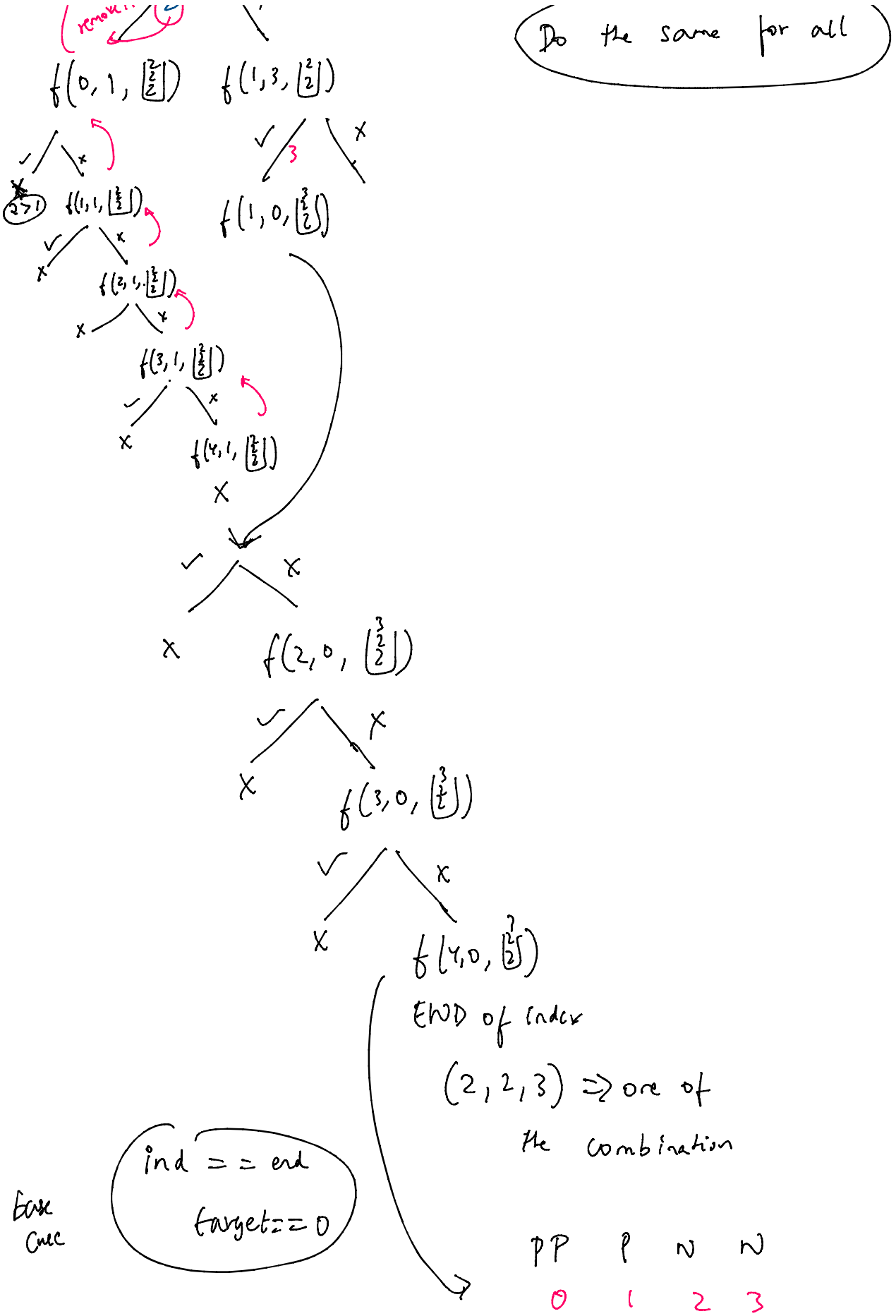
Example 1:

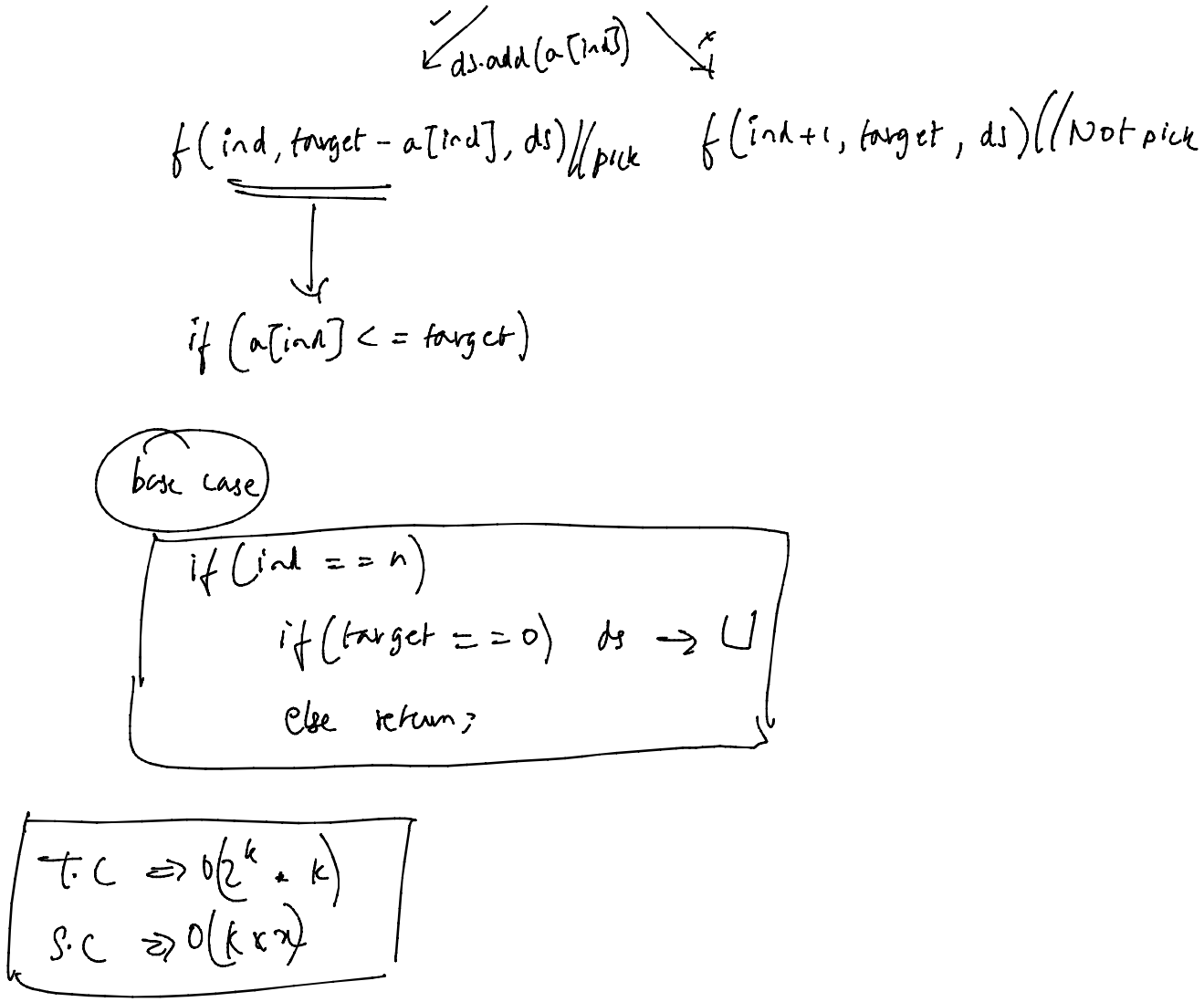
```
Input: candidates = [2,3,6,7], target = 7
Output: [[2,2,3],[7]]
Explanation:
2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used multiple times.
7 is a candidate, and 7 = 7.
These are the only two combinations.
```








Do the same for all
f(0, 1, [2/2])
f(1,3,[2/2])
f(1,1,[3/2])
f(1,0,[3/2])
f(2,1,[3/2])
f(3,1,[3/2])
f(4,1,[3/2])
f(2,0,[3/2])
f(3,0,[3/t])
f(4,0,[?/2])
END of index
(2,2,3) => one of the combination
ind == end
target == 0
Base Case
PP P N N
0 1 2 3
f(ind, target, ds)




```java
class Solution {
    private void findCombination(int ind, int[] arr, int target, List<List<Integer>> ans, List<Integer> ds){
        // base case
        if(ind == arr.length){
            if(target == 0){
                ans.add(new ArrayList<>(ds));
            }
            return;
        }
        //pick condition
        if(arr[ind] <= target){
            ds.add(arr[ind]);
            findCombination(ind, arr, target - arr[ind], ans, ds);
            ds.remove(ds.size() - 1); //backtrack
        }
        // not pick condition
        findCombination(ind + 1, arr, target, ans, ds);
    }

    public List<List<Integer>> combinationSum(int[] candidates, int target) {
        List<List<Integer>> ans = new ArrayList<>();
        // new ArrayList<>() in findCombinatin is empty ds as we start with empty ds
        findCombination(0, candidates, target, ans, new ArrayList<>());
        return ans;
    }
}
```