**39. Combination Sum**

Medium

4902136Add to ListShare

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.

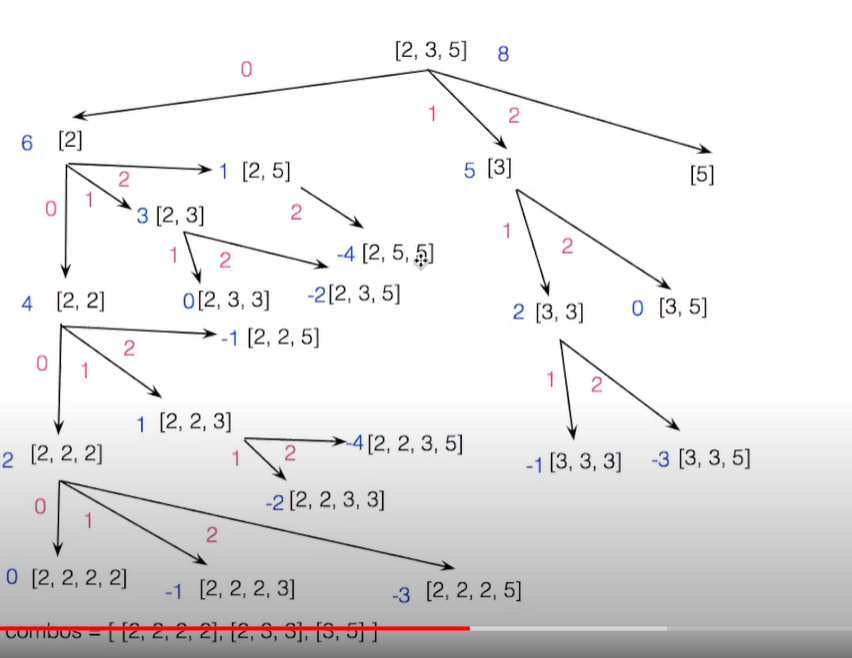
These are the only two combinations.

**Example 2:**

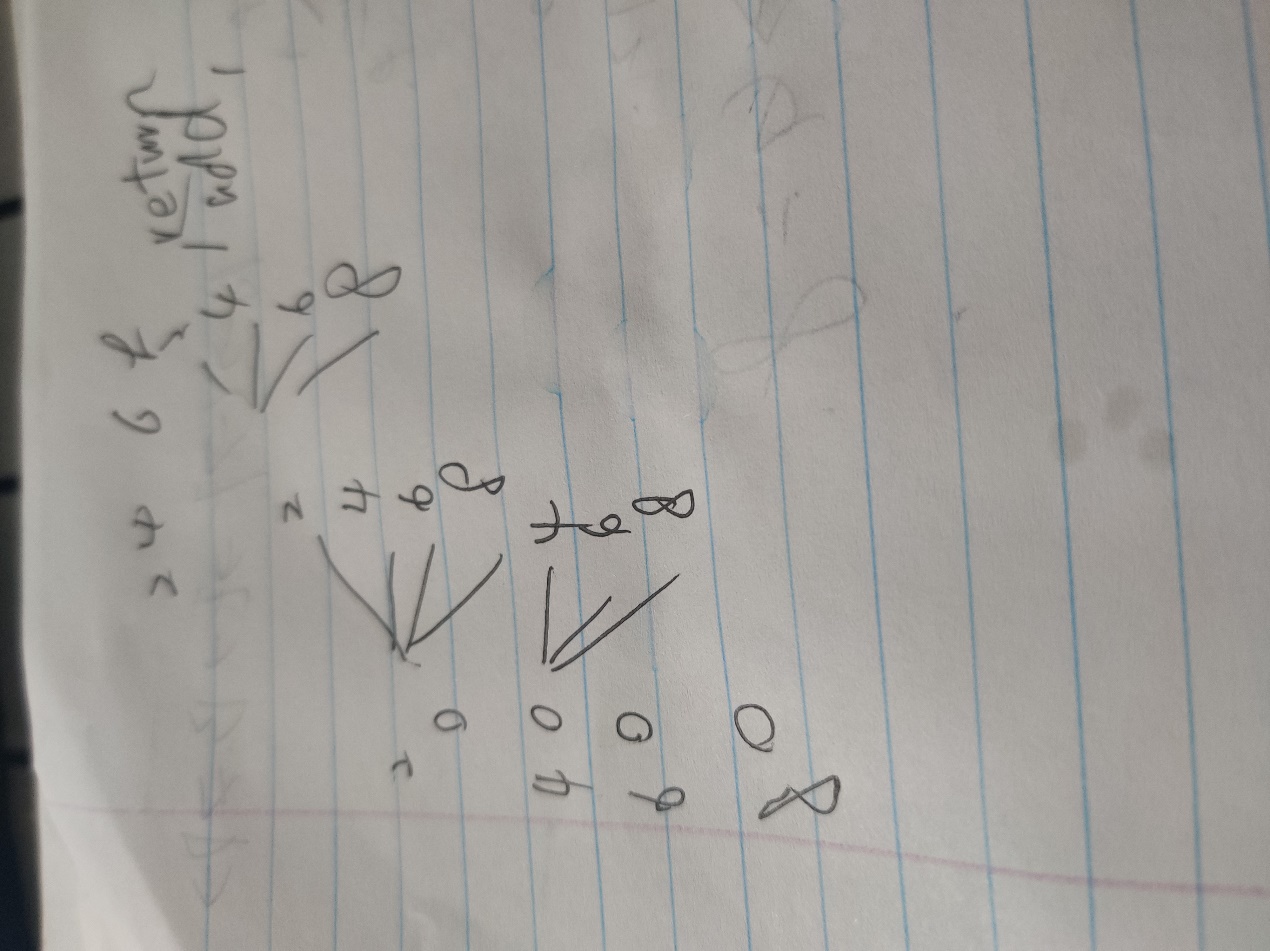
**Input:** candidates = [2,3,5], target = 8

**Output:** [[2,2,2,2],[2,3,3],[3,5]]

**思路：这题是DFS**



一路往左走，到底以后POP，如果是0，加进结果里



典型的 For loop结构

class Solution {

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

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

int index=0;

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

combinationRecursive(candidates,target,index,curr,result);

return result;

}

public static void combinationRecursive(int[] candidates, int target, int index, List<Integer> curr, List<List<Integer>> result){

if(target<0) {

return;

}

if(target==0) {

result.add(new ArrayList<>(curr)); //=0的时候return

return;

}

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

curr.add(candidates[i]);当前curr增加

combinationRecursive(candidates,target-candidates[i],i,curr,result);//target变小了，但是下一个call的还是跟着i走

curr.remove(curr.size()-1);

}

}

}