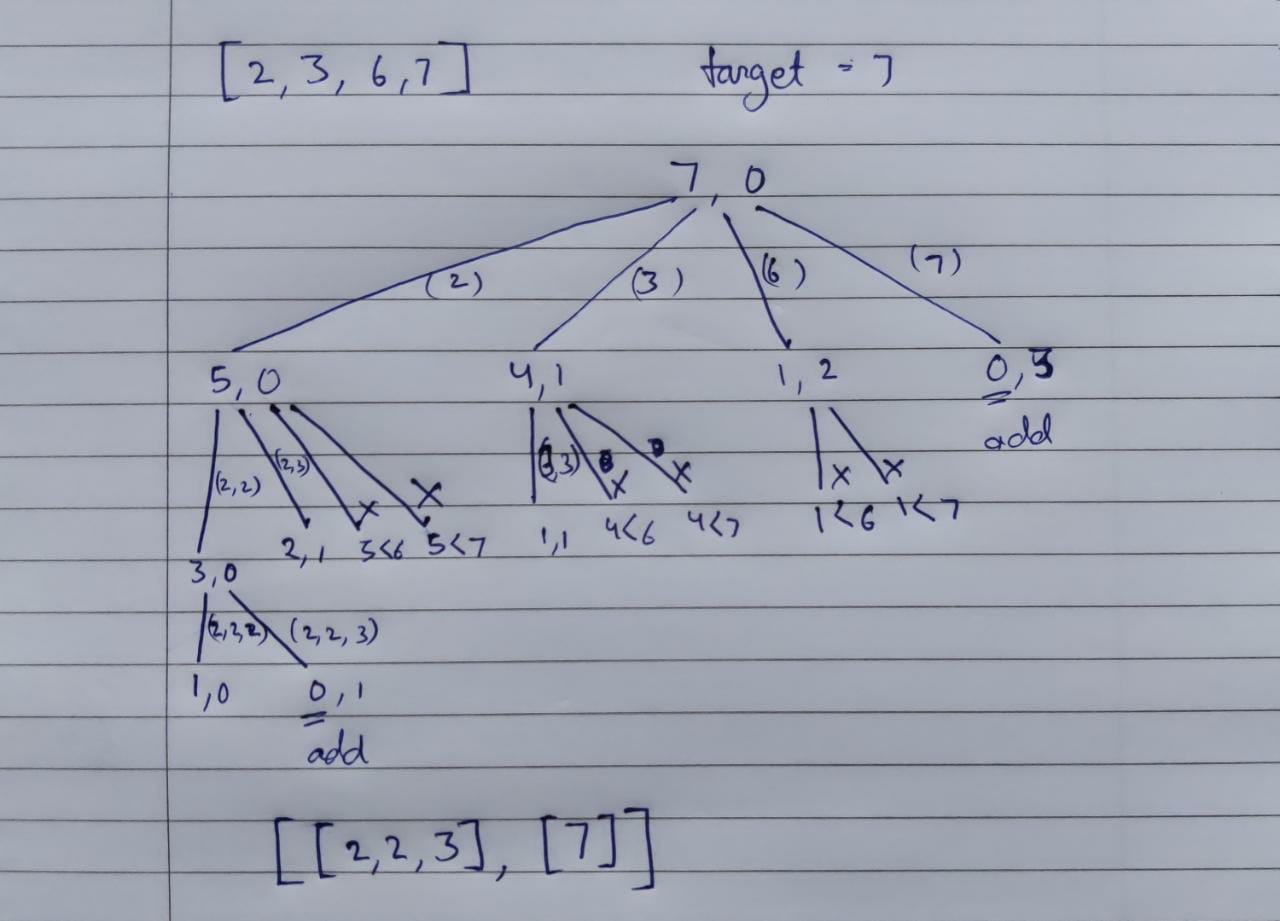
Question: https://leetcode.com/problems/combination-sum/

So we have to choose unique combinations of array elements whose sum would be equal to target.

So lets look at first how we can get combinations of numbers which will help us get target as their sum. So at every step we can choose any number from array and subtract their sum from the target if we get 0 then that is a valid target.

Since repeatation is allowed we can choose from ar[0] to ar[n], now we need to also take care of unique combinations, in order to get unique combinations what we can do is we can consider an element multiple times but when we are selecting at ith position we wont select any element before ith position, in that way we can maintain unique combinations.



Code:  
class Solution {

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

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

finder( candidates, target, 0, subArray);

return result;

}

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

public void finder(int[] cands, int target, int stIdx, List<Integer> curList){

if (target == 0) result.add(curList);

for (int i = stIdx; i < cands.length; i++) {

if (cands[i] <= target) {

List<Integer> list = new ArrayList<>(curList);

list.add(cands[i]);

finder(cands, target - cands[i], i, list);

}

}

}

}

Github Link :<https://lnkd.in/ecwtJeaz>