

1. Subsets of Array

3

→ 10 15 20

Output

10

10 15

10 15 20

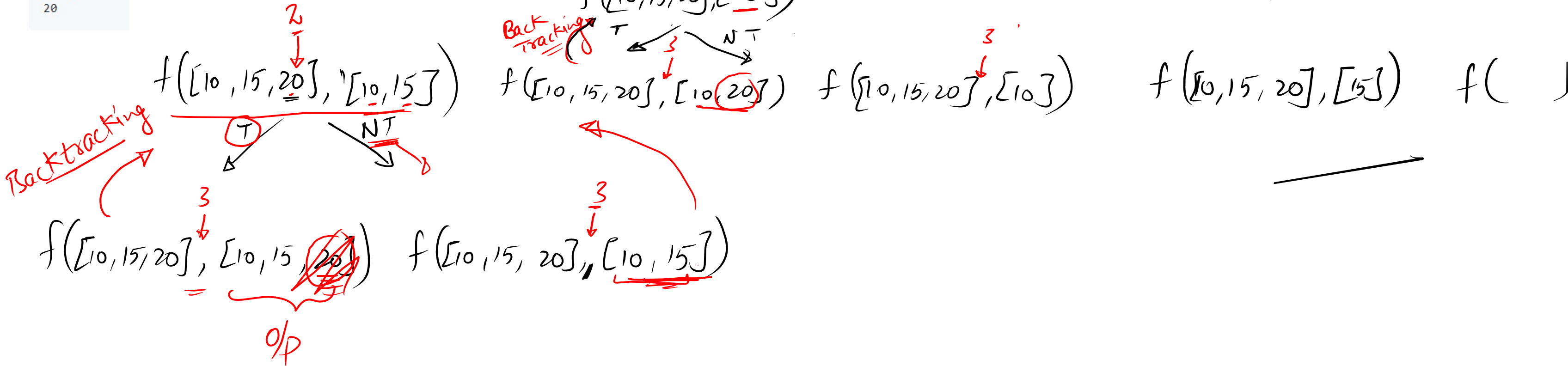
10 20

15

15 20

20

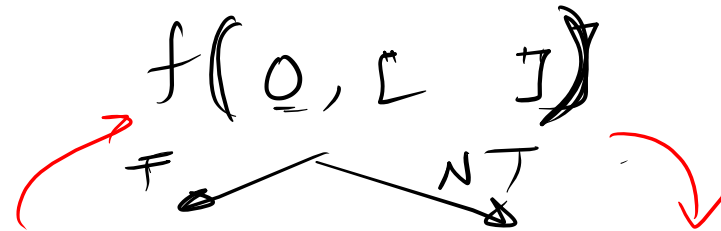
arr o/p
 $f([10, 15, 20], "")$



* subset of array

$f(\text{idx}, \text{o/p})$

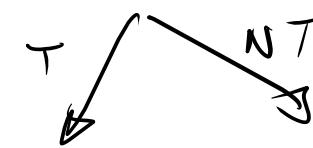
$[10, 15, 20]$



$[10, 15, 20]$

$f(1, [10, 15, 20])$

$f(1, [15, 20])$



$[10, 15, 20]$

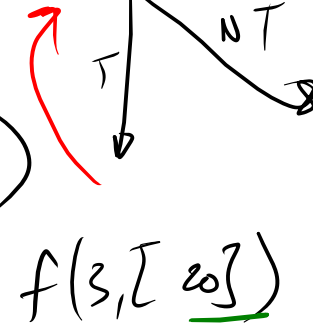
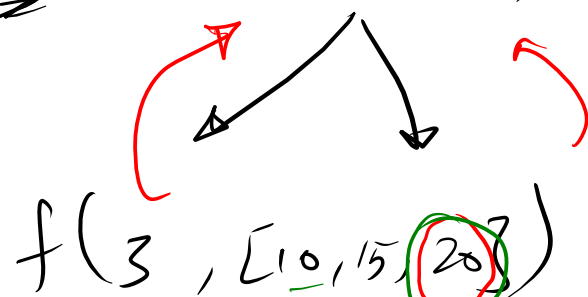
$f(2, [10, 15])$

$f(2, [15, 20])$

$f(2, [15])$

$f(2, [20])$

out of arr
[]



$f(3, [])$

$f(3, [10, 15, 20])$

$f(3, [10, 20])$

$f(3, [15, 20])$

$f(3, [15, 20])$

$f(3, [20])$

remove
Back tracking.

10, 15, 20

10, 20

10

15, 20

if remove
(!op.empty())

* // Base Case

```
if (idx == arr.length) {  
    ans.add(o/p);  
    return;  
}
```

* process

Take

[
→ ~~op.add(arr[idx]);~~
→ ~~f(idx+1, op, ans)~~
→ op.remove(op.size()-1);
]

$f(\underset{\uparrow}{idx}, \underset{\uparrow}{op}, \overset{\downarrow}{\underset{\uparrow}{ans}}, arr)$

int ArrayList<Integer>

Not Take

[f(idx+1, op, ans);
]

```
public static ArrayList<ArrayList<Integer>> subsets(int[] arr, int n) {
    //Write your code here
    ArrayList<ArrayList<Integer>>ans = new ArrayList<>();
    ArrayList<Integer> output = new ArrayList<>();
    solve(0,output,ans,arr,n);
    return ans ;
}

public static void solve(int idx , ArrayList<Integer> output ,ArrayList<ArrayList<Integer>>ans ,int[] arr, int n){
    //base case
    if(idx == n){
        // ans.add(output);
        ans.add(new ArrayList<>(output));
        return ;
    }

    //process
    //TAKE
    output.add(arr[idx]);
    solve(idx+1,output,ans,arr,n);
    output.remove(output.size()-1);    //backtracking

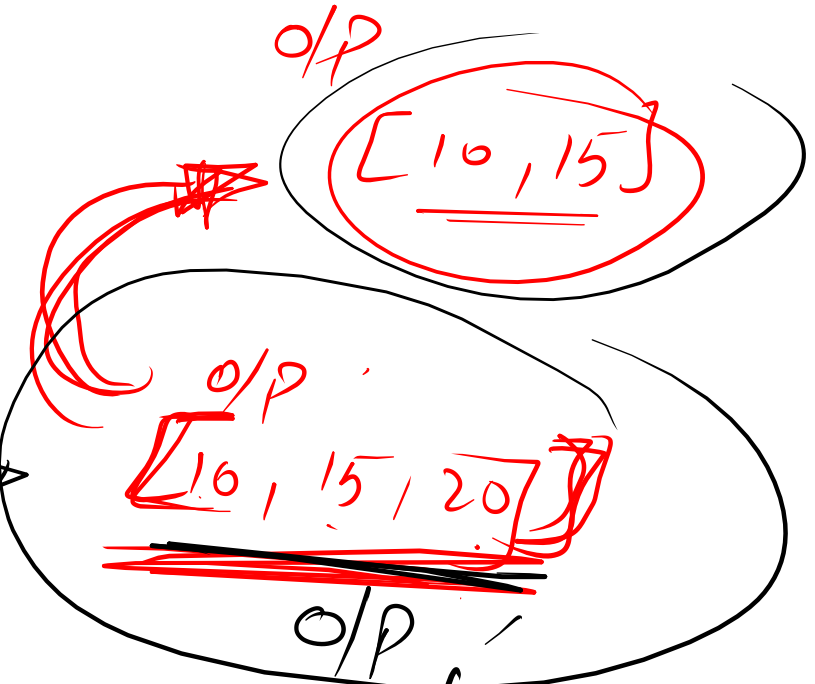
    //NOT TAKE
    solve(idx+1,output,ans,arr,n);
}
```

(*)

ans.add(op) ;

↓
reference

idx = zn



(*)

ans.add(new ArrayList<>(op)) ;

↓ copy create
new ArrayList
↓
ans

39. Combination Sum

Medium

Topics

Companies

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.

The test cases are generated such 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:

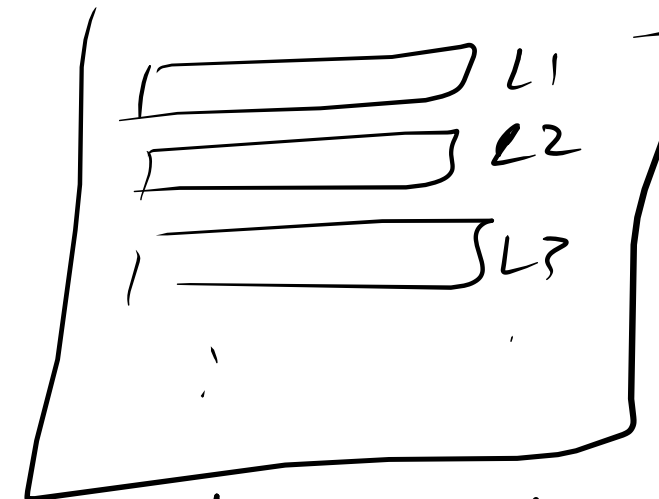
2 and 3 are candidates, and $2 + 2 + 3 = 7$. Note that 2 can be used multiple times.

7 is a candidate, and $7 = 7$.

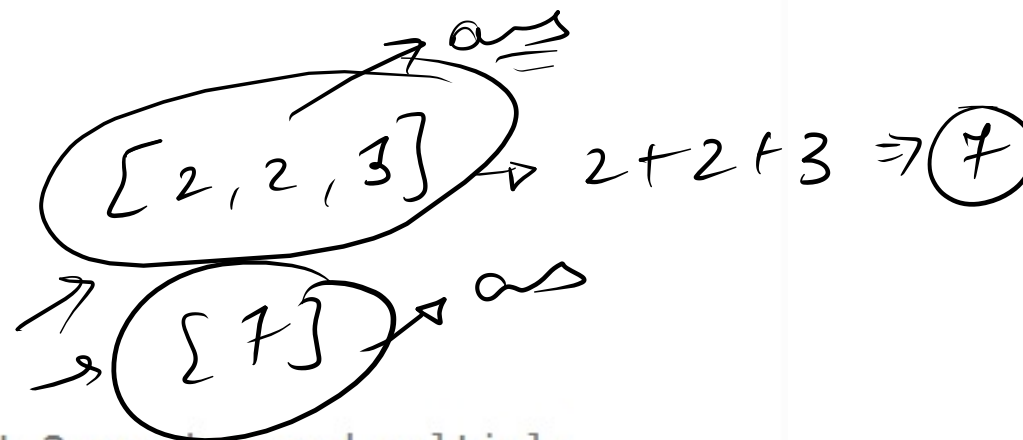
These are the only two combinations.

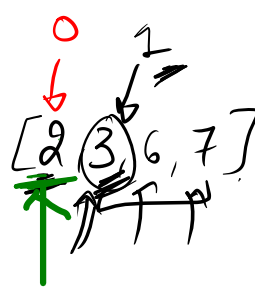
unlimited time choose
candi. `[0, ...]`

target = 3



return list





(2, 2, 3)

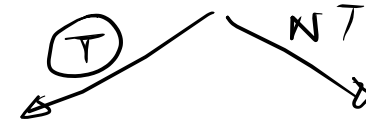
$f(\text{idx}, T, \text{o/p})$
 $f(\underline{0}, \underline{7}, [\underline{\quad}])$

* Base Case

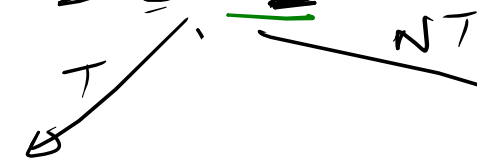
if ($T == 0$) {
 ans.add(op);
 return;
}

7-2

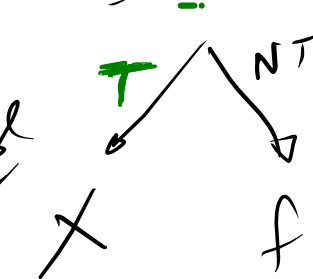
$f(\underline{0}, \underline{5}, [\underline{2}])$



$f(\underline{0}, \underline{3}, [\underline{2}, \underline{2}])$



$f(\underline{0}, \underline{1}, [\underline{2}, \underline{2}, \underline{2}])$



Backtracking

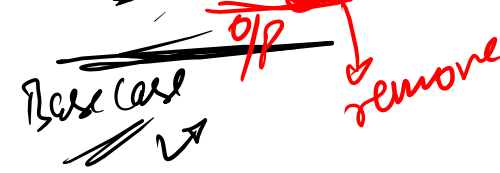
$f(\underline{1}, \underline{1}, [\underline{2}, \underline{2}, \underline{2}])$



$f(\underline{1}, \underline{3}, [\underline{2}, \underline{2}])$



$f(\underline{2}, \underline{0}, [\underline{2}, \underline{2}, \underline{3}])$



- // Take
- ① op.add(arr[idx])
 - ② $f(\text{idx}, T - \text{arr[i]}, \text{op})$

③ $\text{op.remove(op.size() - 1)}$; // BT

// NOT take
 $f(\text{idx} + 1, T, \text{op})$

1-2


$f(\underline{0}, \underline{-1}, [\underline{2}, \underline{2}, \underline{2}, \underline{2}])$

Target


```
public List<List<Integer>> combinationSum(int[] candidates, int target) {  
    List<List<Integer>>ans = new ArrayList<>();  
    List<Integer>output = new ArrayList<>();  
    solve(0,target,output,ans,candidates);  
    return ans;  
}  
public void solve(int idx ,int T ,List<Integer>output , List<List<Integer>>ans, int[] arr ){  
    //base case  
    if(T == 0 ){  
        // ans.add(output);  
        ans.add(new ArrayList<>(output));  
        return ;  
    }  
    if(idx == arr.length) return ;  
  
    //process  
    // Take  
    if(arr[idx] <= T){  
        output.add(arr[idx]);  
        solve(idx,T-arr[idx],output,ans,arr);  
        output.remove(output.size()-1); //backtracking  
    }  
  
    //Not Take  
    solve(idx+1,T,output,ans,arr);  
}
```

40. Combination Sum II

Medium

 Topics

 Companies

Given a collection of candidate numbers (`candidates`) and a target number (`target`), find all unique combinations in `candidates` where the candidate numbers sum to `target`.

Each number in `candidates` may only be used **once** in the combination.

Note: The solution set must not contain duplicate combinations.

ans

$[1, 1, 1, 2, 2]$
↑ ↑ ↑

$T = 4$

$[1, 1, 2]$

$[1, 1, 2]$

duplicate
not allowed

idx T o/p.
 $f(\underline{0}, 4, \boxed{})$

$T = 4$
 $[1, 1, 2]$
 $[2, 2]$
 $[$

$[1, 1, 1, 2, 2]$ $T=4$

$[1, 1, 1, 2, 2]$

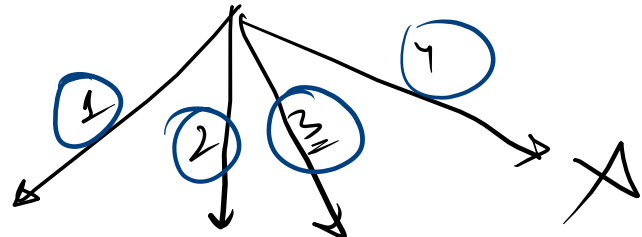
$T=4$

$f(2, 2, [1])$

$f(3, 1, [1])$ $f(4, 0, [1])$

There are

$f(1, 3, [1])$



$f(4, 1, [2])$

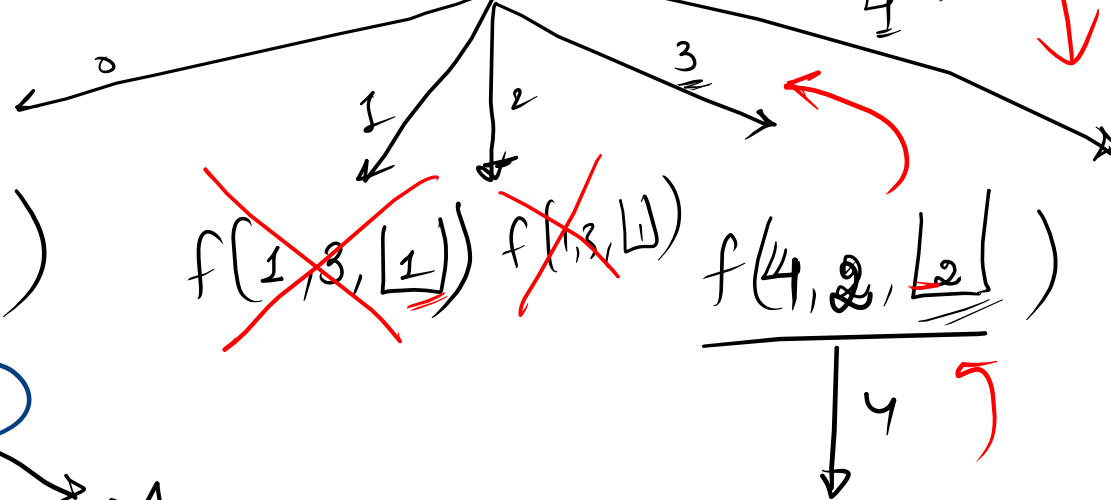
$f(4, 1, [2])$



$[1, 1, 4]$, $[2, 2]$

idx T o/p

$f(0, 4, [])$



$f(1, 3, [1])$

$f(1, 3, [1])$

$f(4, 2, [2])$

$f(5, 2, [2])$

$f(5, 0, [2])$

for (int idx = i; idx < arr.length; idx++) {
 op.add(arr[idx]);
 solve(i+1, T-arr[i]);
 op.remove(op.size()-1);
}

* Base case

```
if (T == 0) {  
    ans.add(op);  
    return;  
}
```