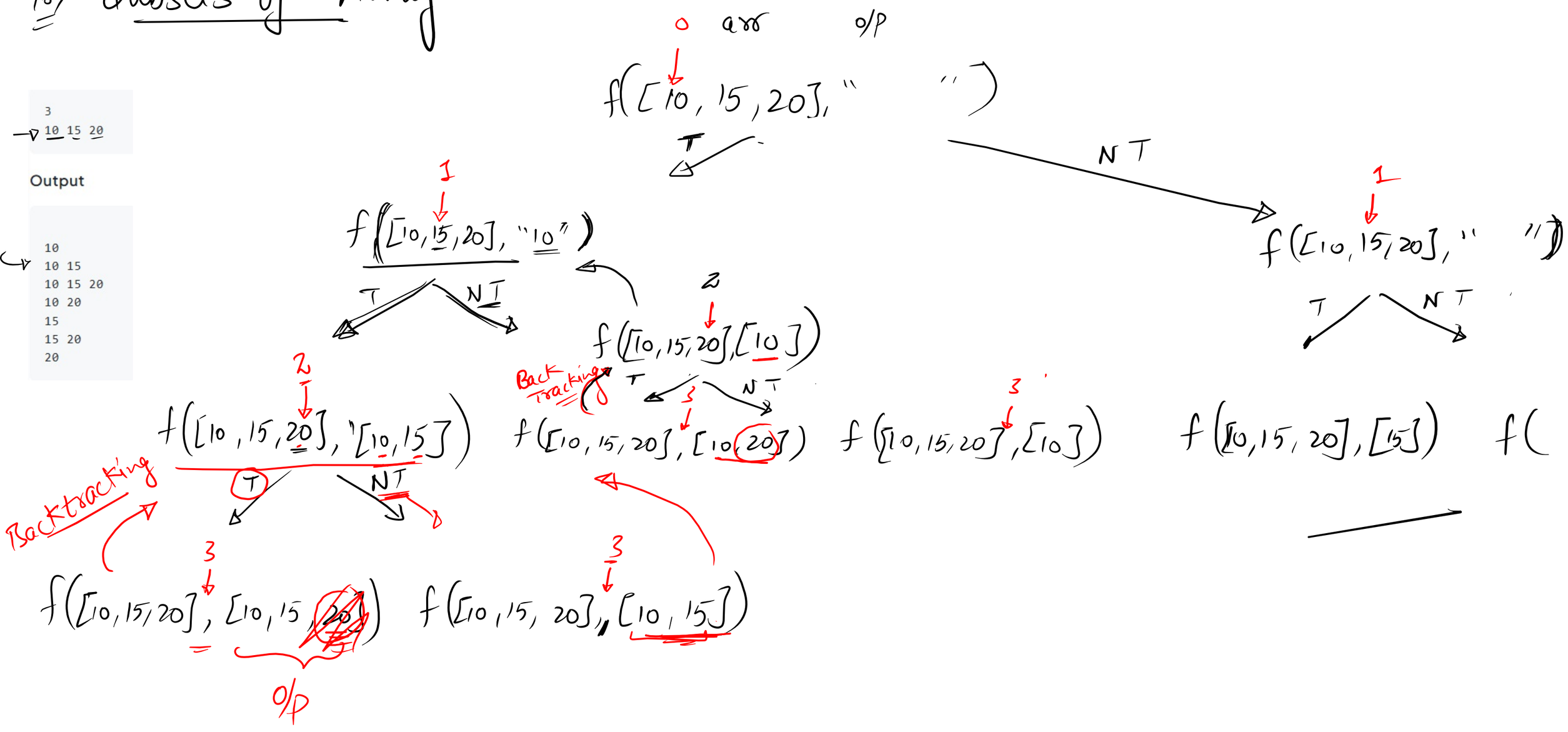
1> Subsets of Array

```
3
10 15 20
```

Output

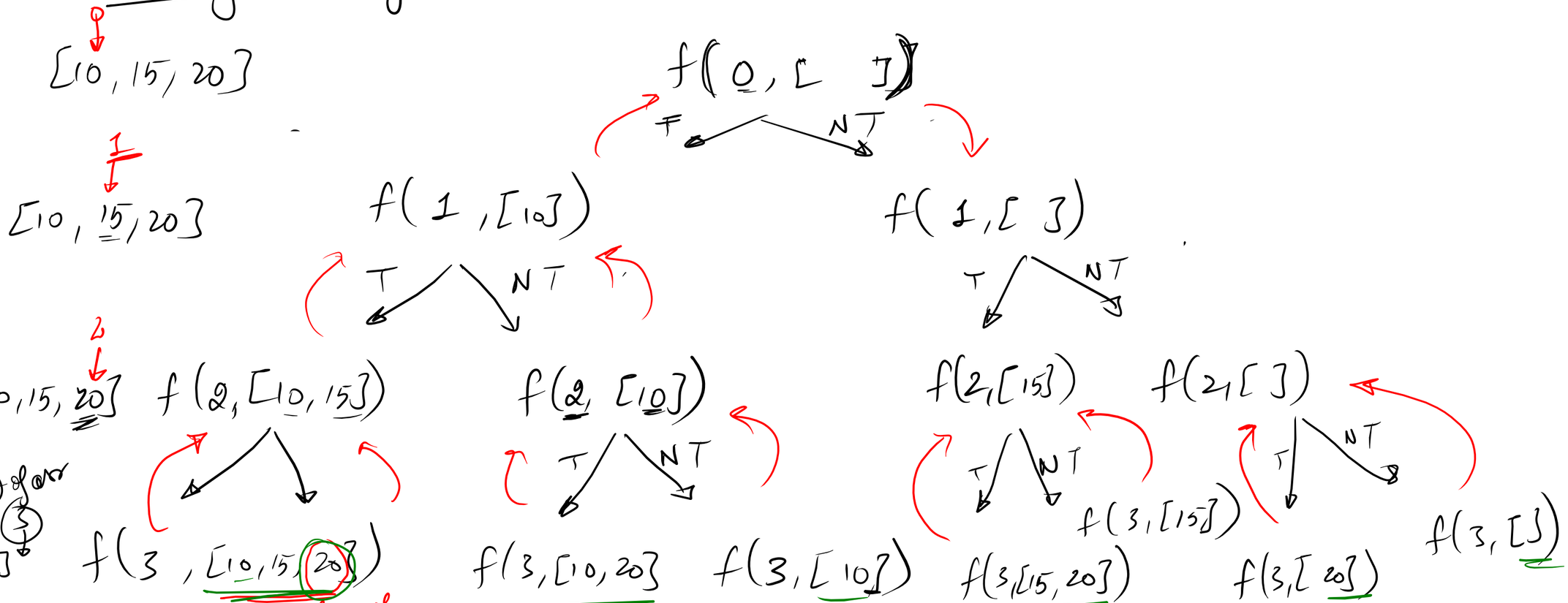
```
10
10 15
10 15 20
10 20
15
15 20
20
```



* Subset of arrays

[10, 15, 20]

f(idx, o/p)



10, 15, 20

10, 20

10

15, 20

* // Base Case

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

* process

f(idx, op, ans, arr)

int ArrayList<Integer>

Take

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

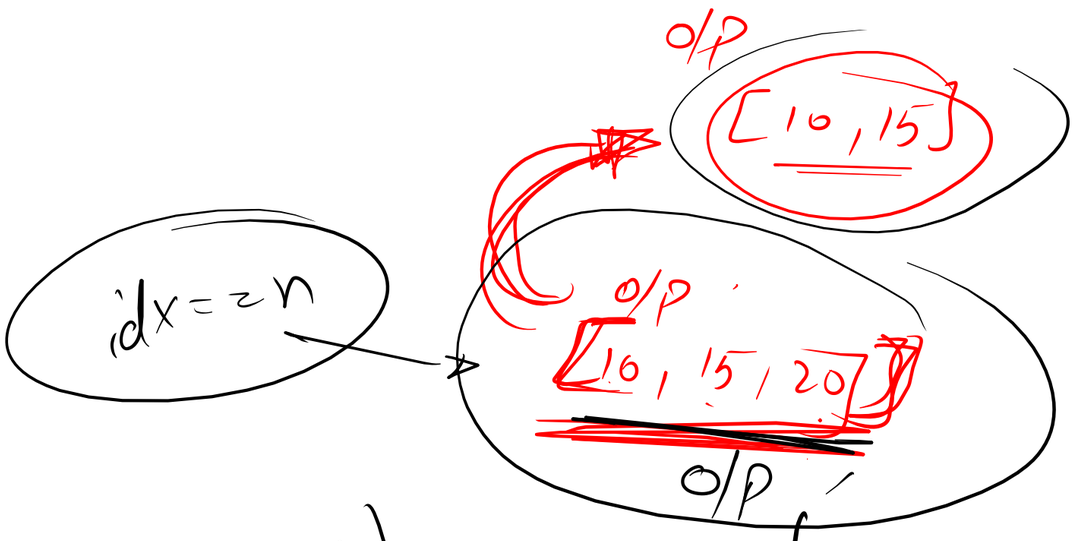
Not Take

f(idx+1, op, ans);

```java
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



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

copy create

new arraylist

ans

39. Combination Sum

Medium | Topics | Companies

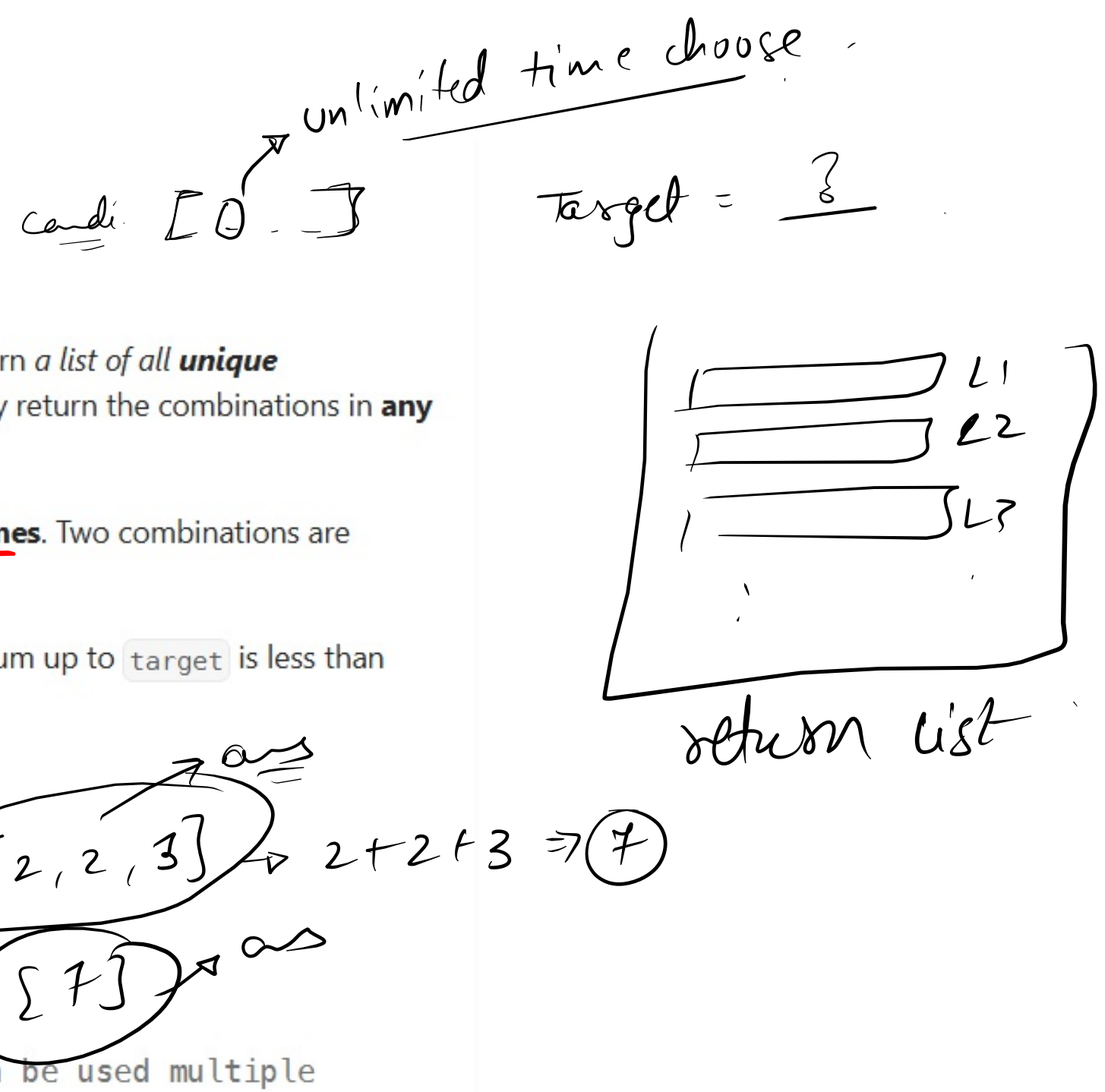
Given an array of **distinct** integers `candidates` and a target integer `target`, return *a list of all* ***unique combinations*** *of* `candidates` *where the chosen numbers sum to* `target`. You may return the combinations in **any order**.

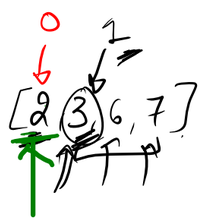
The **same** number may be chosen from `candidates` an **unlimited number of times**. Two combinations are unique if the frequency of at least one of the chosen numbers is different.

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.
```





2,2,3

f(idx 0, T 7, o/p [_])

T NT

* Base case

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

T=2

f(0, 2, [2])

T NT

f(0, 3, [2,2])

T NT

f(1, 3, [2,2])

T NT

f(2, 0, [2,2,3])

op

Base case

remove

f(0, 1, [2,2,2])

T NT

Back tracking

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

T NT

X

X

X

1-2

f(0, -1, [2,2,2,2])

Target

Base case

// Take

① op.add(arr[idx])

② f(idx, T-arr[i], op)

③ o.p.remove(op.size()-1); //BT

// NOT take

f(idx+1, T, op)

```java
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.

arr = [1, 1, 1, 2, 2]

T = 4

[1, 1, 2] ↔ [1, 1, 2] duplicate not allowed

idx T o/p

f(0, 4, [ ])

T = 4

[1, 1, 2]

[2, 2]

[

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

idx T o/p

f(0, 4, [])

0 1 2 3 4

f(1, 3, [1]) f(3, 1, [1]) f(4, 2, [2]) f(5, 2, [2]) 4-2

f(4, 2, [2])

4

f(5, 0, [2 2])

f(1, 3, [1])

1 2 3 4

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

2 3 4

f(3, 1, [1]) f(4, 0, [2 1])

4

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

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

* Base case

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