



2 1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18

**Day9 (Recursion):****1. Combination sum-1**

[https://www.youtube.com/watch?v=OyZFFqQtu98&list=PLgUwDviBIf0p4ozDR\\_kJkONnb1wdx2Ma&index=49](https://www.youtube.com/watch?v=OyZFFqQtu98&list=PLgUwDviBIf0p4ozDR_kJkONnb1wdx2Ma&index=49)

**2. Combination sum-2****3. Palindrome Partitioning****4. Subset Sum-1****5. Subset Sum-2****6. K-th permutation Sequence****Day10: (Backtracking)****1. N queens Problem****2. Sudoku****3. M coloring Problem (Graph prob)****4. Rat in a Maze****5. Print all Permutations of a string/array****6. Word Break (print all ways)**

## 40. Combination Sum II

Medium  2457  85  Add to List  Share

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.

### Example 1:

**Input:** `candidates = [10,1,2,7,6,1,5]`, `target = 8`

**Output:**

```
[
  [1,1,6],
  [1,2,5],
  [1,7],
  [2,6]
]
```

### Example 2:

**Input:** `candidates = [2,5,2,1,2]`, `target = 5`

**Output:**

```
[
  [1,2,2],
  [5]
]
```

## 4 Combination Sum II | Leetcode | Recursion | Java | C++



Medium 2457 85 Add to List Share

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.

### Example 1:

Input: `candidates = [10,1,2,7,6,1,5]`, `target = 8`

Output:

```
[
  [1,1,6],
  [1,2,5],
  [1,7],
  [2,6]
]
```

### Example 2:

Input: `candidates = [2,5,2,1,2]`, `target = 5`

Output:

```
[
  [1,2,2]
]
```

TUF

1:32 / 32:36



$$\text{arr} = [1, 1, 1, 2, 2] \quad \text{target} = 4$$

$$\begin{array}{r} 1, 1, 2 \\ \hline 2, 2 \\ \hline \end{array}$$



## L9. Combination Sum II | Leetcode | Recursion | Java | C++



```
1 class Solution {
2
3     private void findCombinations(int ind, int[] arr, int target, Set<List<Integer>> ans, List<Integer> ds) {
4         if(ind == arr.length) {
5             if(target == 0) {
6                 ans.add(new ArrayList<>(ds));
7             }
8             return;
9         }
10
11         if(arr[ind] <= target) {
12             ds.add(arr[ind]);
13             findCombinations(ind+1, arr, target - arr[ind], ans, ds);
14             ds.remove(ds.size() - 1);
15         }
16         findCombinations(ind + 1, arr, target, ans, ds);
17     }
18     public List<List<Integer>> combinationSum(int[] candidates, int target) {
19         Set<List<Integer>> ans = new ArrayList<>();
20         findCombinations(0, candidates, target, ans, new ArrayList<>());
21         // hashSet can be converted to a list of list, then return the list
22         return ans;
23     }
24 }
```

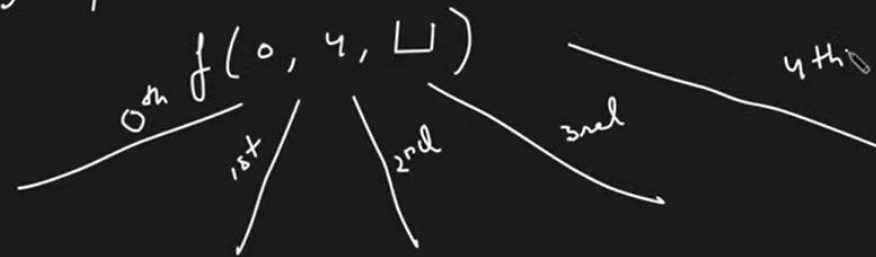


4:45 / 32:36



TUF

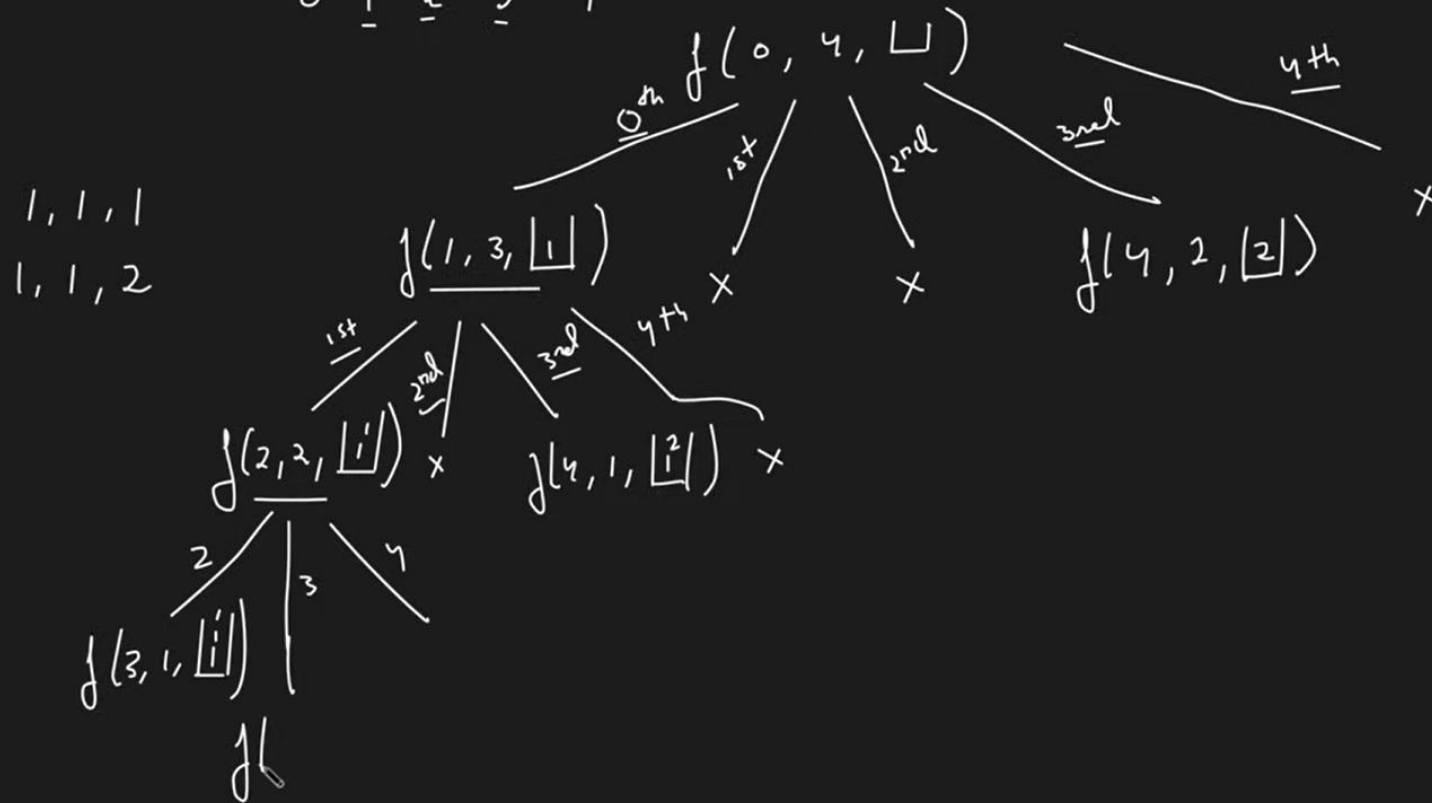
arr[] =  $\begin{bmatrix} 1, & 1, & 1, & 2, & 2 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix}$  target = 4



arr =  $\begin{bmatrix} 1 & 1 & 1 & 2 & 2 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix}$

target = 4

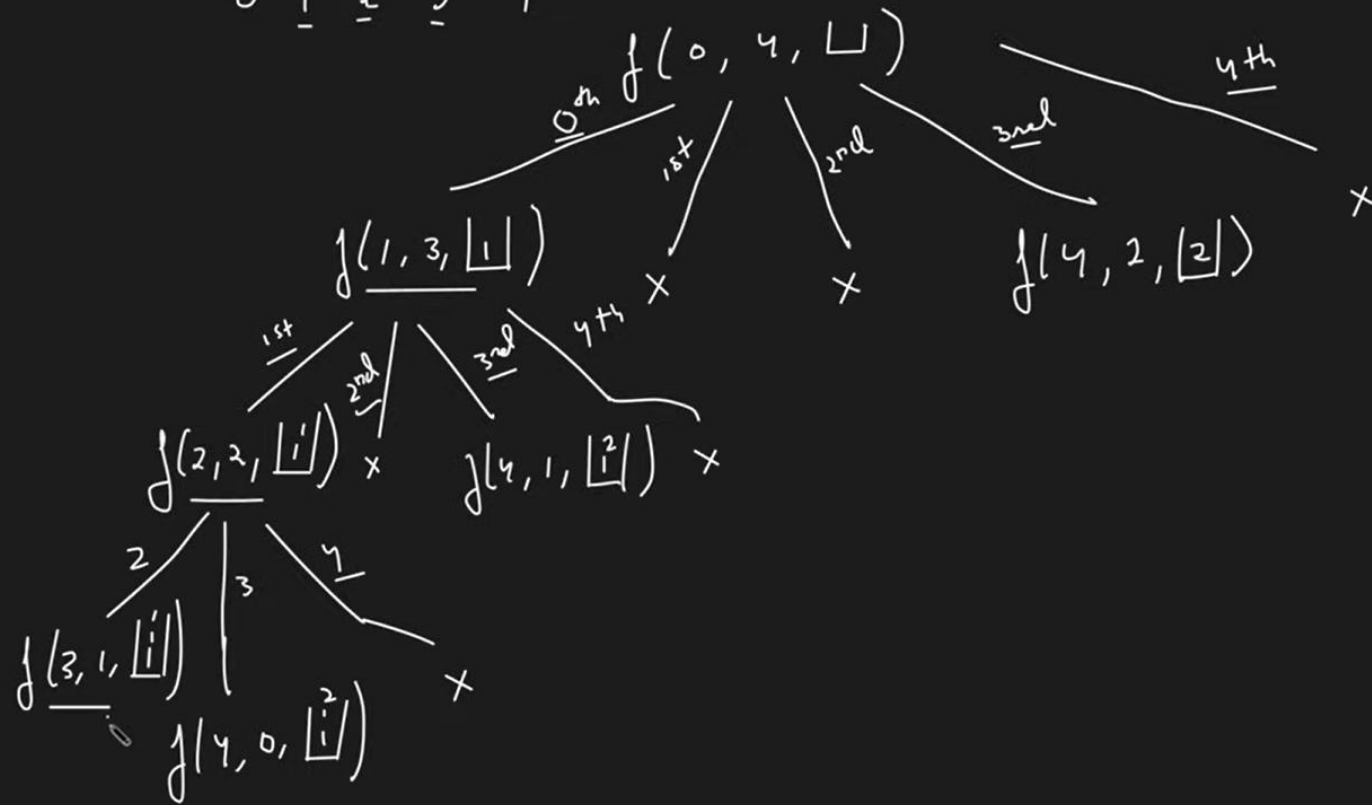
1, 1, 2



$$\text{arr} = \begin{bmatrix} 1 & 1 & 1 & \underline{3} & 2 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix}$$

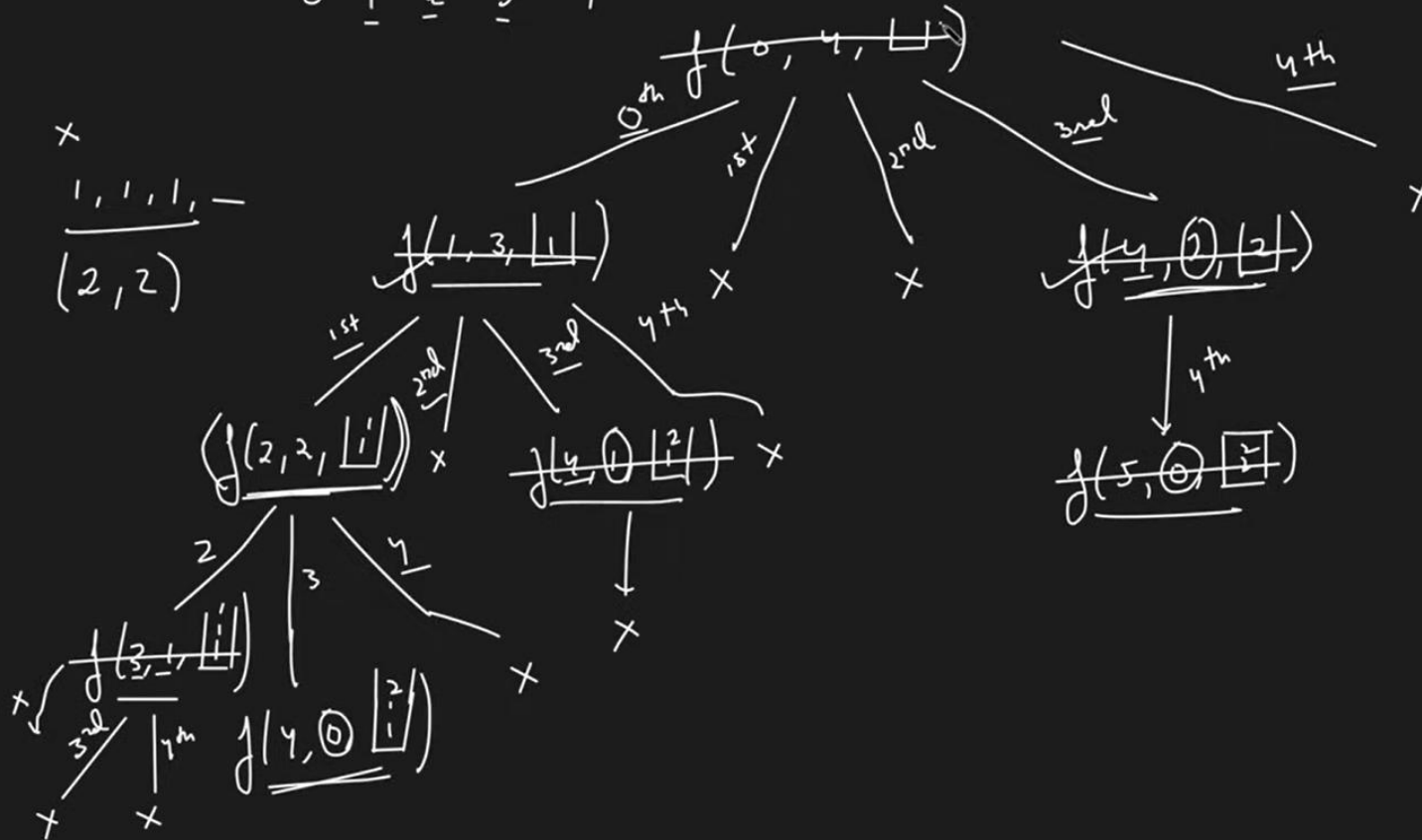
target = 4

1, 1, 2

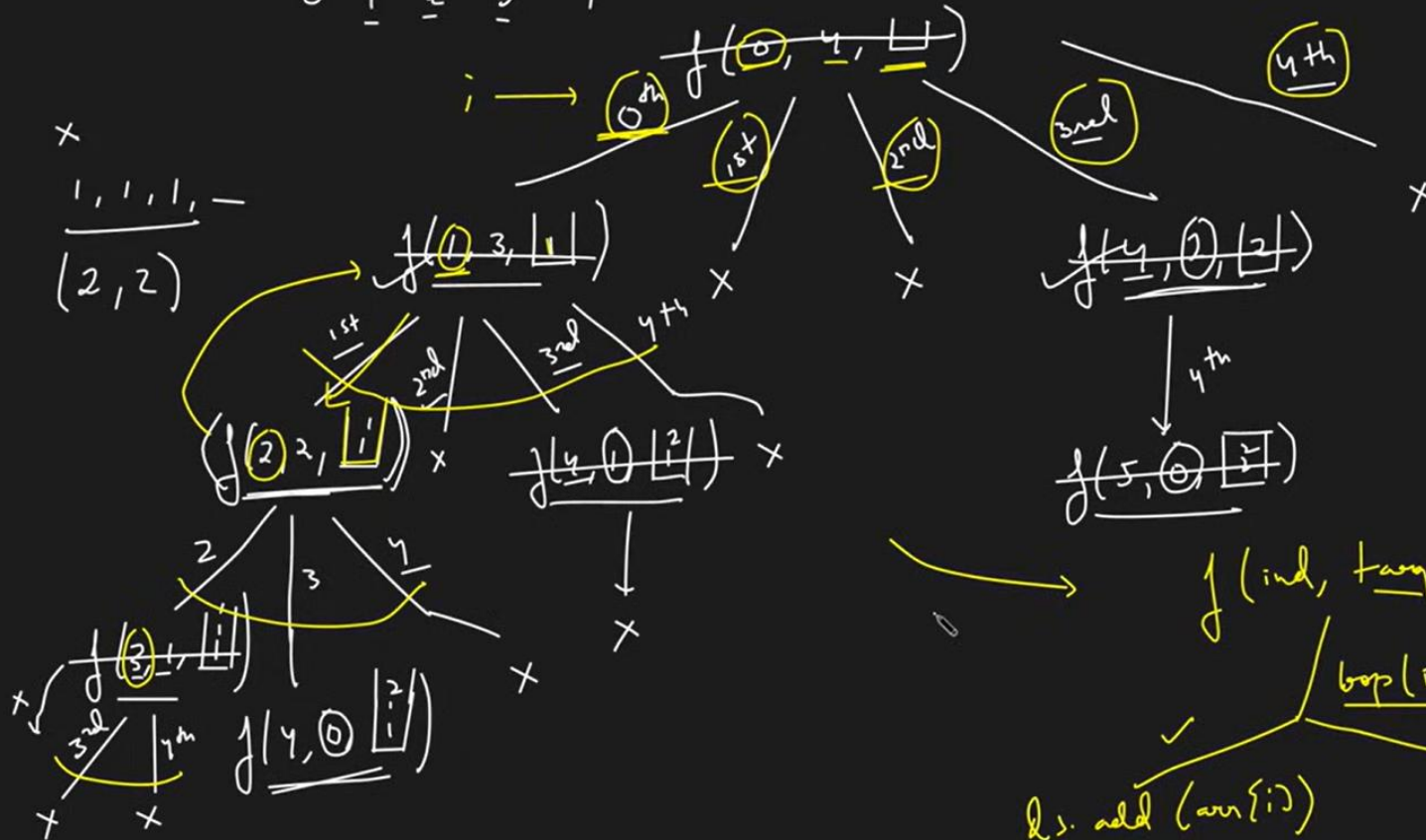


$$\text{arr} = \begin{bmatrix} 1 & 1 & 1 & 3 & 2 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix} \quad \text{target} = 4$$

$$\begin{pmatrix} 1, 1, 2 \\ 2, 2 \end{pmatrix}$$



$$\left| \frac{|1, 1, 2\rangle}{(2, 2)} \right| \rightarrow \text{ang}$$



→  $f(\text{ind}, \text{target}, \text{ds}, \text{ans})$

✓  $\text{bops}(\text{ind} \rightarrow n-1) \rightarrow (i)$

$\text{ds.add}(\text{arr}[i])$

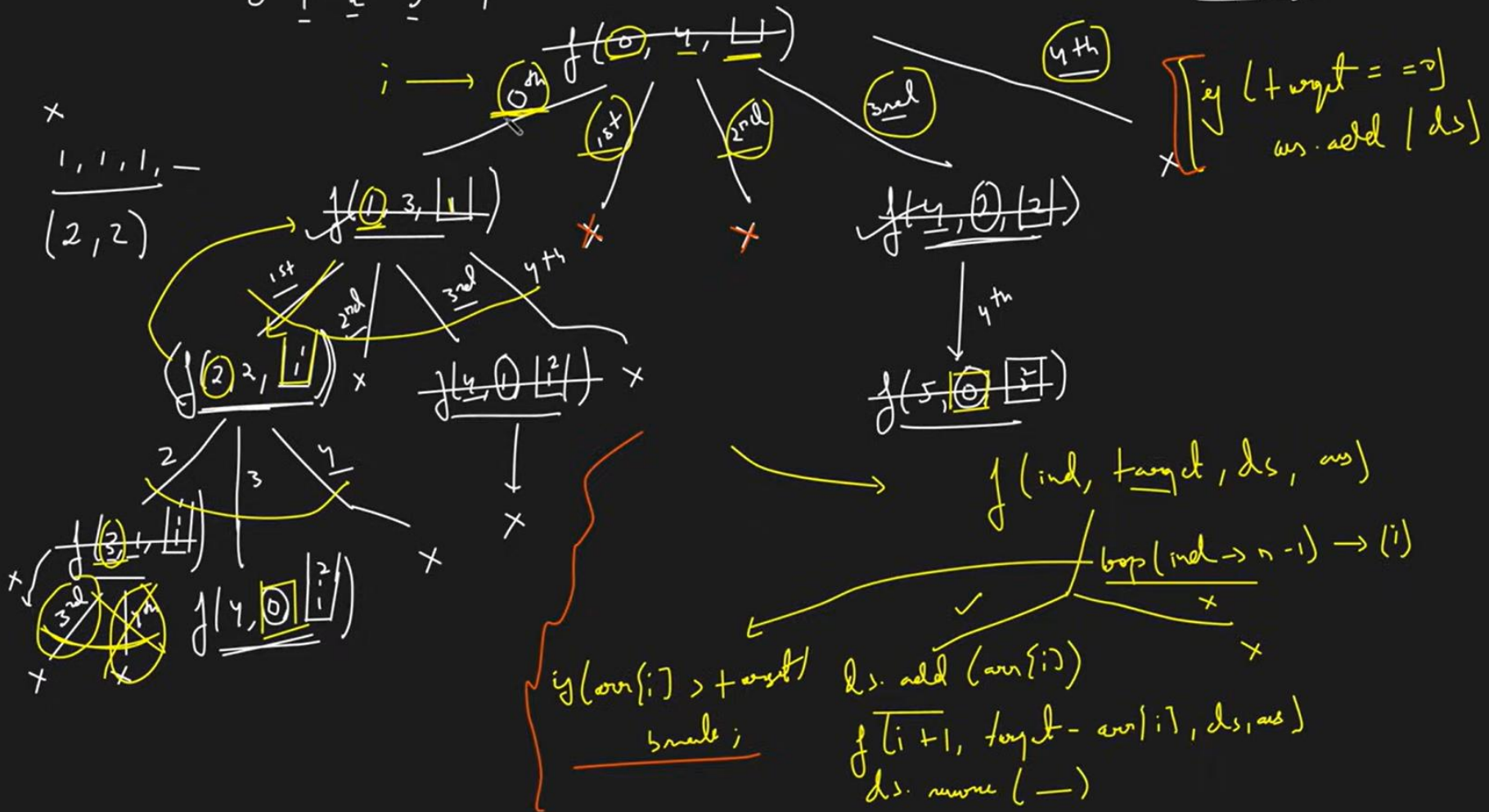
$f(i+1, \text{target} - \text{arr}[i], \text{ds}, \text{ans})$

$\text{ds.remove}(\text{—})$

arr[] = [1, 1, 1, 2, 2]      target = 4

0   1   2   3   4

ans. ← [1, 1, 2]  
              (2, 2) → ans



arr[] = [1, 1, 1, 2, 2]      target = 4

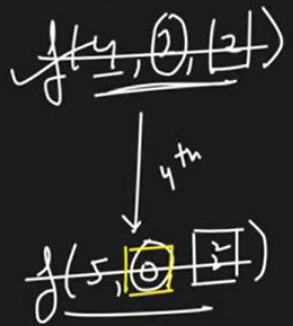
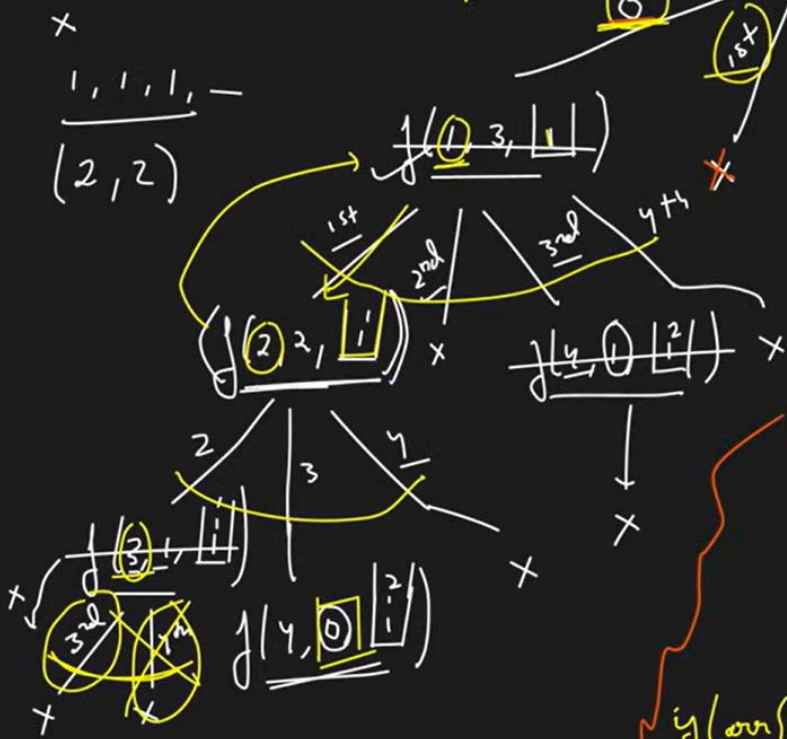
0   1   2   3   4

ans. ← [1, 1, 2]  
          (2, 2) → ans



if (target == 0)  
ans.add(ds)

TC → 2<sup>n</sup>



f(ind, target, ds, ans)

base(ind → n-1) → (i)

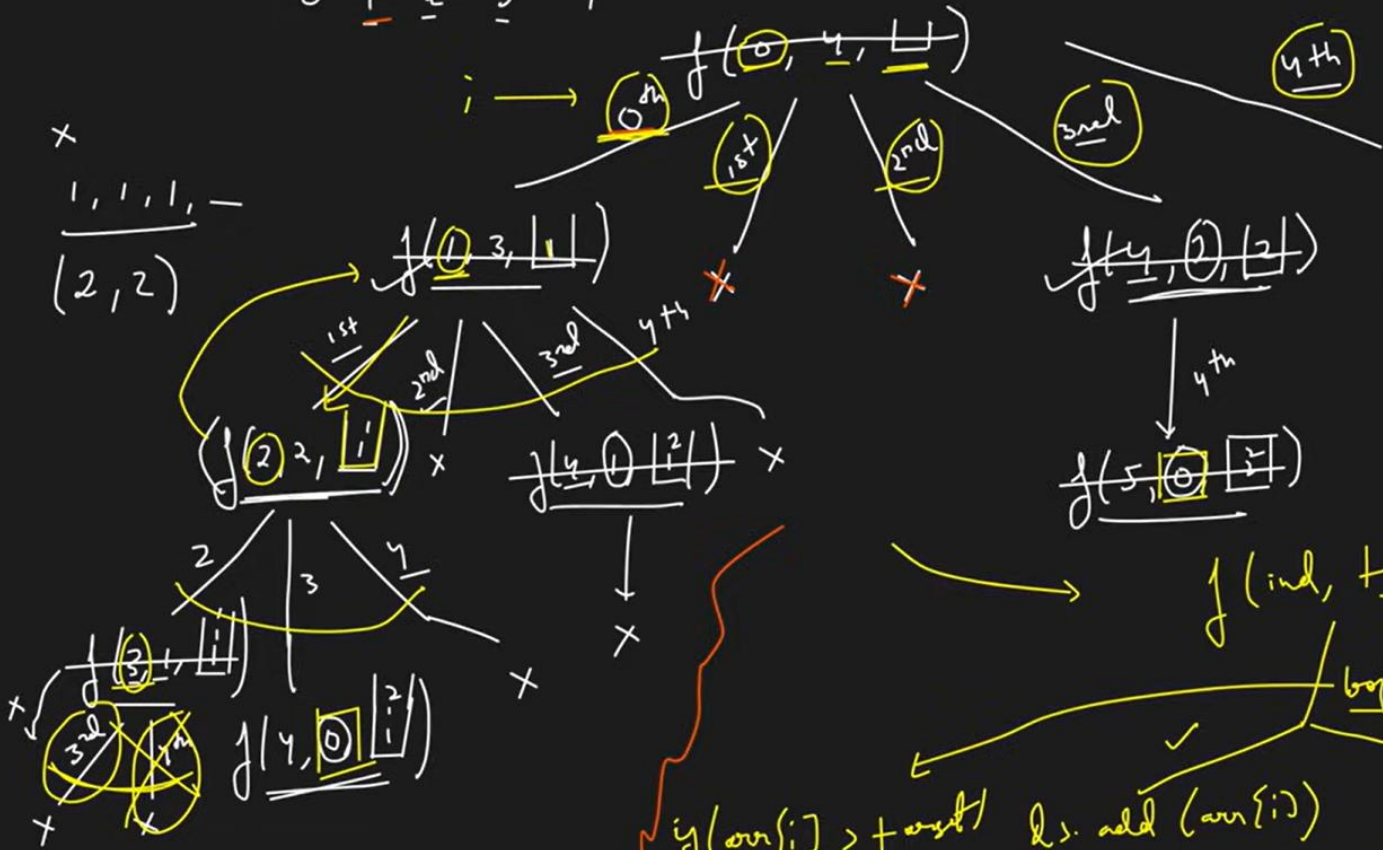
if (arr[i] > target)  
break;

ds.add(arr[i])  
f(i+1, target - arr[i], ds, ans)  
ds.remove(arr[i])

arr[] = [1, 1, 1, 2, 2]      target = 4

0   1   2   3   4

ans. ← [1, 1, 2]  
          (2, 2) → ans



if (target == 0)  
ans.add(ds)

TC →  $2^n \times k$   
SC →

f(ind, target, ds, ans)

base(ind → n-1) → (i)

if (arr[i] > target)  
    break;

ds.add(arr[i])  
f(i+1, target-arr[i], ds, ans)  
ds.remove(arr[i])

$$\underline{\text{arr}} \{ \} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 \\ 0 & 1 & 2 & 3 & 4 \end{bmatrix}$$

target = 4

Ans.

→  $f(0, 4, 4)$

$$\begin{array}{r} \times \\ \hline 1, 1, 1, 1, - \\ (2, 2) \end{array}$$

$\frac{3rd}{4th}$   
 $f(4, 0, 2)$

$\times \left[ \begin{matrix} \text{ig} \\ \text{wsg} \end{matrix} \right] (+ \text{wsgt} = = \text{v})$   
wsg. add (ds)

$$\begin{array}{r} \text{TC} \rightarrow 2^n \times k \\ \hline \text{SC} \rightarrow k \times n \end{array}$$

Handwritten notes on a blackboard showing a sequence of moves in a 3-disk Tower of Hanoi problem. The moves are represented by diagrams of the three disks (labeled 1, 2, 3) and their positions (1, 2, 3). The sequence starts with (2, 2) and proceeds through several steps, with some moves marked as incorrect (X) or correct (+). The final move shown is (1, 1, 1).

$f(5, \boxed{0}, \boxed{2})$

$$f(\text{ind}, \text{target}, ds, \text{ans})$$
$$\frac{\text{bop}(\text{ind} \rightarrow n-1)}{\times} \rightarrow (i)$$

$ds.add(arr[i])$   
 $f[i+1, target - arr[i], ds, ans]$   
 $ds.remove(-)$

C++ Code at 28:05

```
1 class Solution {  
2     private void findCombinations(int ind, int[] arr, int target, List<List<Integer>> ans, List<Integer> ds) {  
3         if(target == 0) {  
4             ans.add(new ArrayList<>(ds));  
5             return;  
6         }  
7  
8         for(int i = ind; i < arr.length; i++) {  
9             if(i > ind && arr[i] == arr[i-1]) continue;  
10            if(arr[i] > target) break;  
11  
12            ds.add(arr[i]);  
13            findCombinations(i+1, arr, target - arr[i], ans, ds);  
14            ds.remove(ds.size() - 1);  
15        }  
16    }  
17    public List<List<Integer>> combinationSum2(int[] candidates, int target) {  
18        List<List<Integer>> ans = new ArrayList<>();  
19        Arrays.sort(candidates);  
20        findCombinations(0, candidates, target, ans, new ArrayList<>());  
21        return ans;  
22    }  
23 }
```

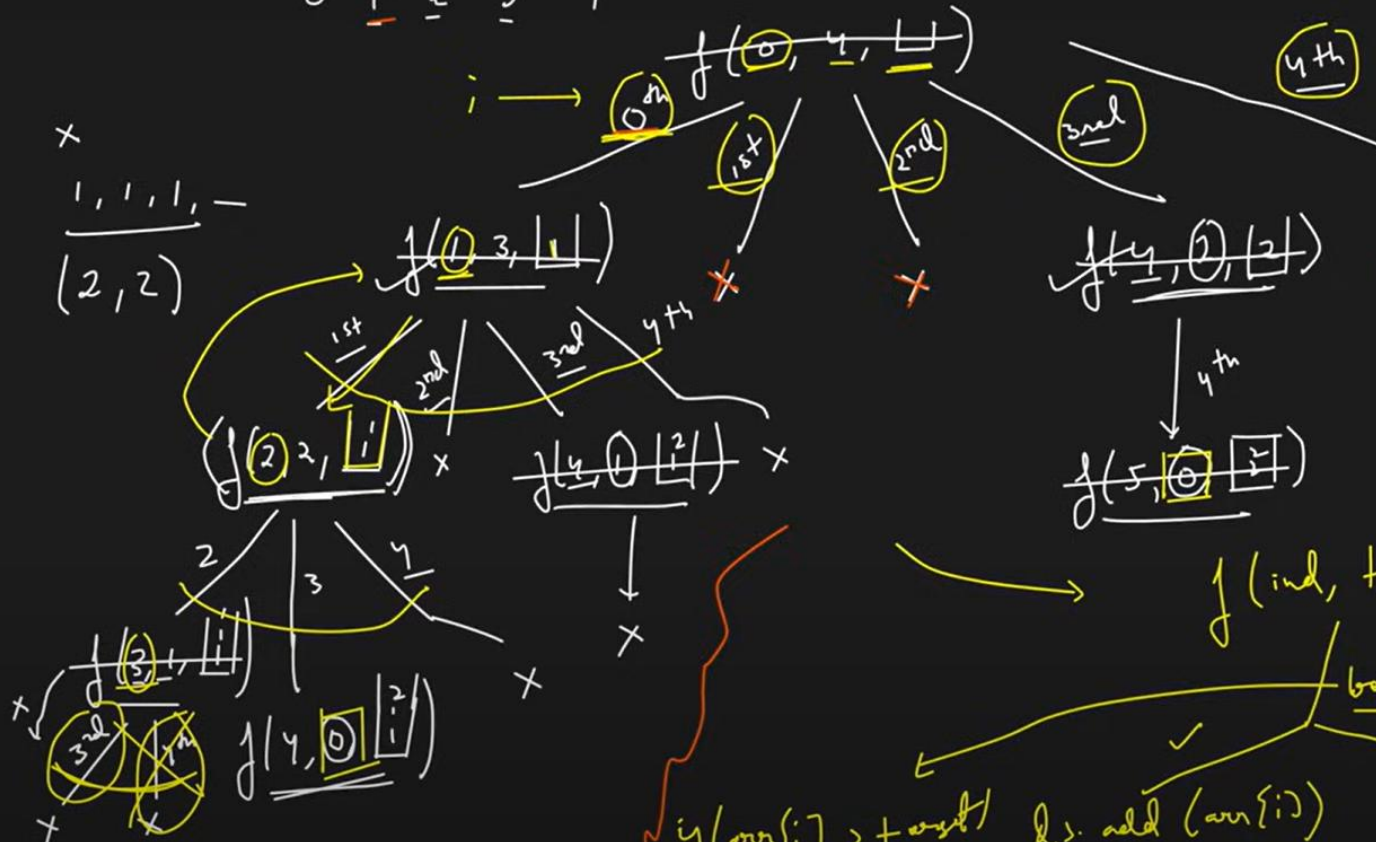
# ⇒ L9. Combination Sum II | Leetcode | Recursion | Java | C++

arr[] = [1, 1, 1, 2, 2]  
 0 1 2 3 4

target = 4

ans.

← [1, 1, 2]  
 (2, 2) → ans



if (target == 0)  
 ans.add(ds)

TC →  $2^n \times k$   
 SC →  $k \times x$

f(ind, target, ds, ans)  
 loop(ind → n-1) → (i)  
 if (arr[i] > target) break;  
 ds.add(arr[i])  
 f(i+1, target - arr[i], ds, ans)  
 ds.remove(arr[i])

TUF

arr[] = [1, 1, 1, 2, 2] target = 4

ans.  $\leftarrow \left[ \frac{1, 1, 2}{2, 2} \right] \rightarrow \text{ans}$

1, 1  $\rightarrow$  0th  $f(0, 4, \_)$  1st  $f(1, 3, \_)$  2nd  $f(2, 2, \_)$  3rd  $f(3, 1, \_)$  4th  $f(4, 0, \_)$

if (target == 0)  
ans.add(ds)

TC  $\rightarrow 2^n \times k$   
SC  $\rightarrow k \times x$

1, 1, 1, 1, -  
(2, 2)

1st 2nd 3rd 4th

2 3 4

4th  $f(4, 2, 2)$

$f(\text{ind}, \text{target}, \text{ds}, \text{ans})$

base (ind  $\rightarrow n-1$ )  $\rightarrow$  (i)

if (arr[i] > target)  
break;

ds.add(arr[i])  
 $f(i+1, \text{target} - \text{arr[i]}, \text{ds}, \text{ans})$   
ds.remove(-)

```

1  class Solution {
2      private void findCombinations(int ind, int[] arr, int target, List<List<Integer>> ans, List<Integer> ds) {
3          if(target == 0) {
4              ans.add(new ArrayList<>(ds));
5              return;
6          }
7
8          for(int i = ind; i < arr.length; i++) {
9              if(i > ind && arr[i] == arr[i-1]) continue;
10             if(arr[i] > target) break;
11
12             ds.add(arr[i]);
13             findCombinations(i+1, arr, target - arr[i], ans, ds);
14             ds.remove(ds.size() - 1);
15         }
16     }
17     public List<List<Integer>> combinationSum2(int[] candidates, int target) {
18         List<List<Integer>> ans = new ArrayList<>();
19         Arrays.sort(candidates);
20         findCombinations(0, candidates, target, ans, new ArrayList<>());
21         return ans;
22     }
23 }

```

```

1  class Solution {
2      public:
3      void findCombination(int ind, int target, vector<int> &arr, vector<vector<int>> &ans, vector<int>&ds) {
4          if(target==0) {
5              ans.push_back(ds);
6              return;
7          }
8          for(int i = ind; i<arr.size(); i++) {
9              if(i>ind && arr[i]==arr[i-1]) continue;
10             if(arr[i]>target) break;
11             ds.push_back(arr[i]);
12             findCombination(i+1, target - arr[i], arr, ans, ds);
13             ds.pop_back();
14         }
15     }
16     public:
17     vector<vector<int>> combinationSum2(vector<int>& candidates, int target) {
18         sort(candidates.begin(), candidates.end());
19         vector<vector<int>> ans;
20         vector<int> ds;
21         findCombination(0, target, candidates, ans, ds);
22         return ans;
23     }
24 };

```