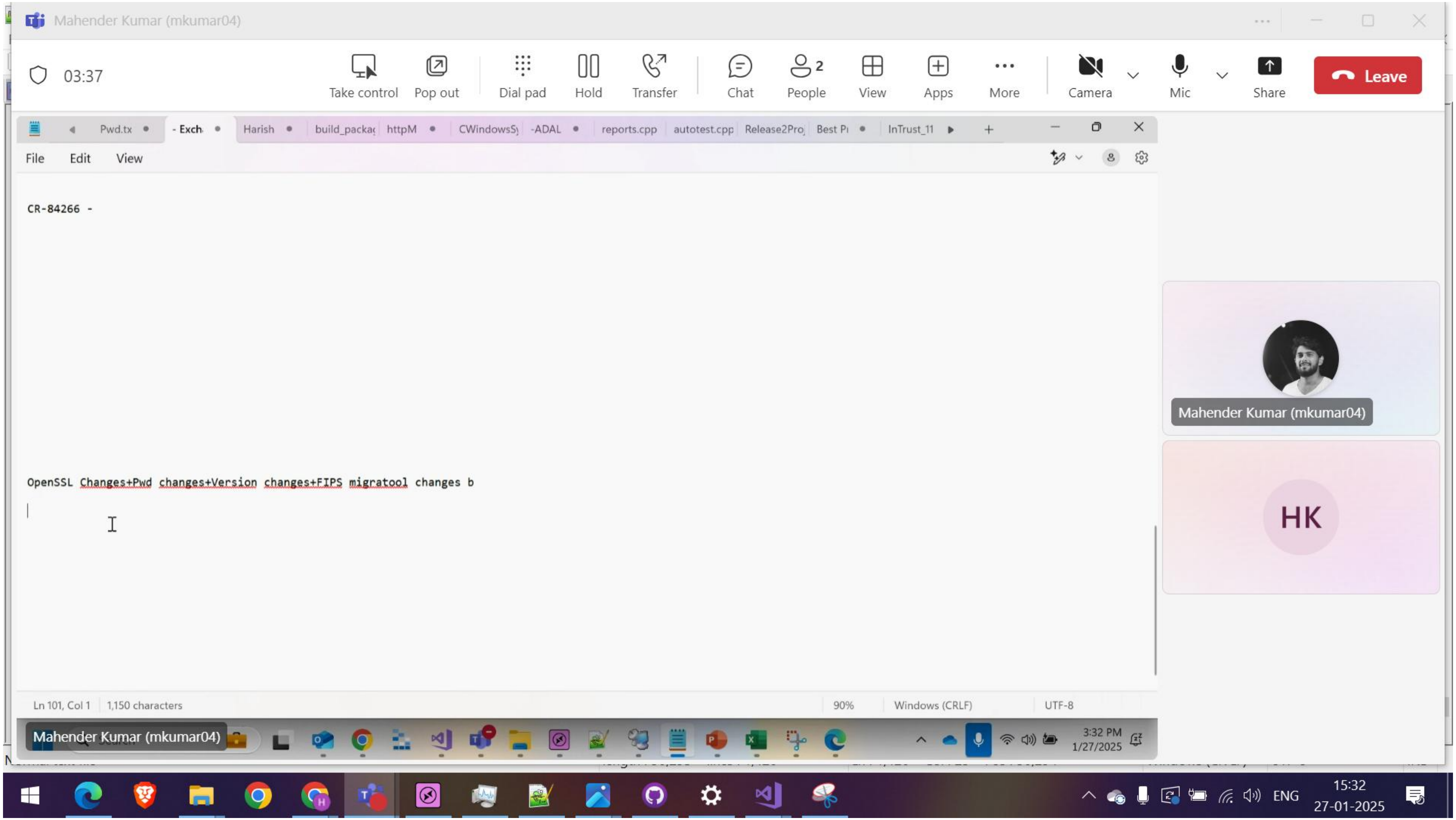
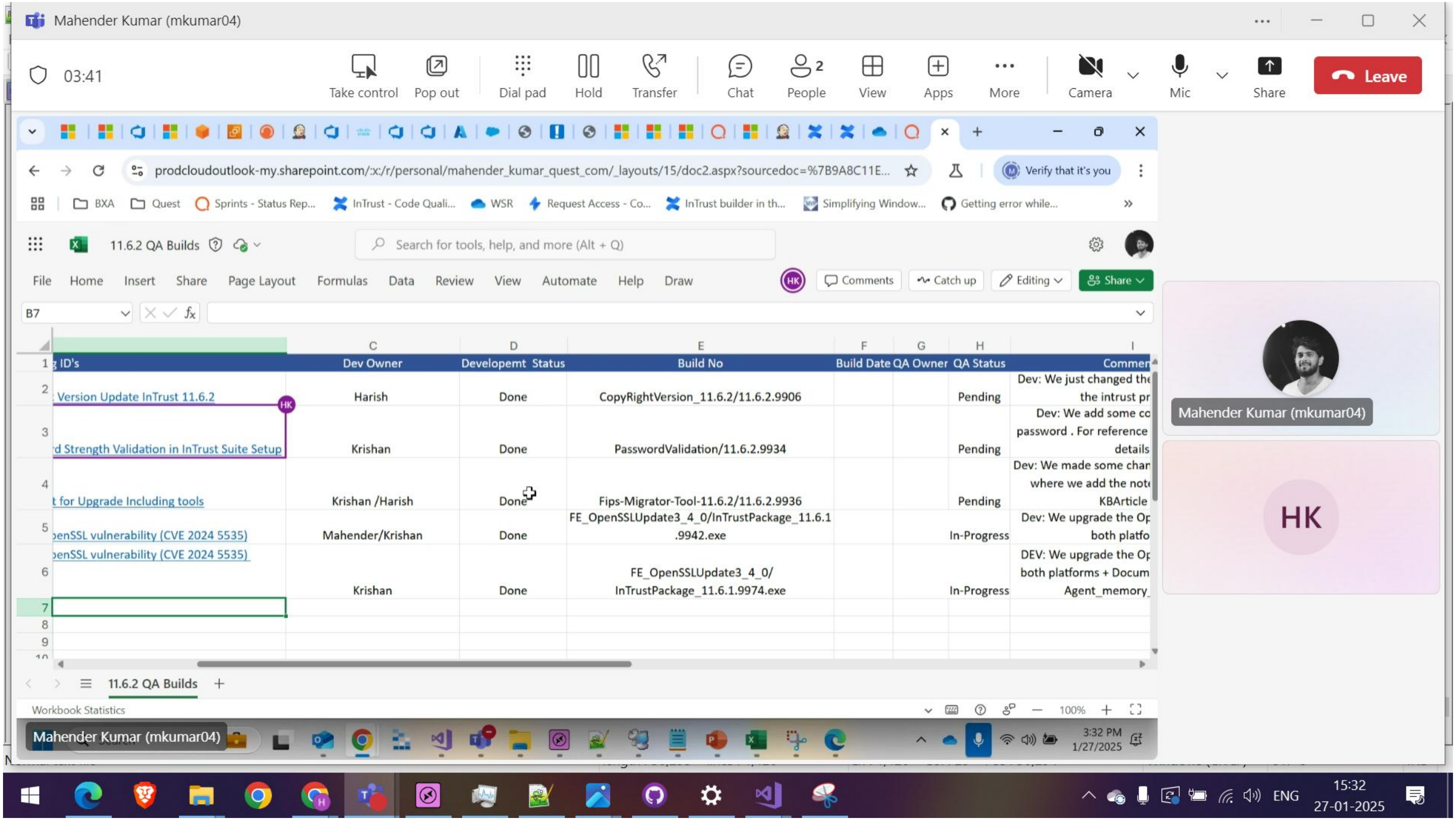
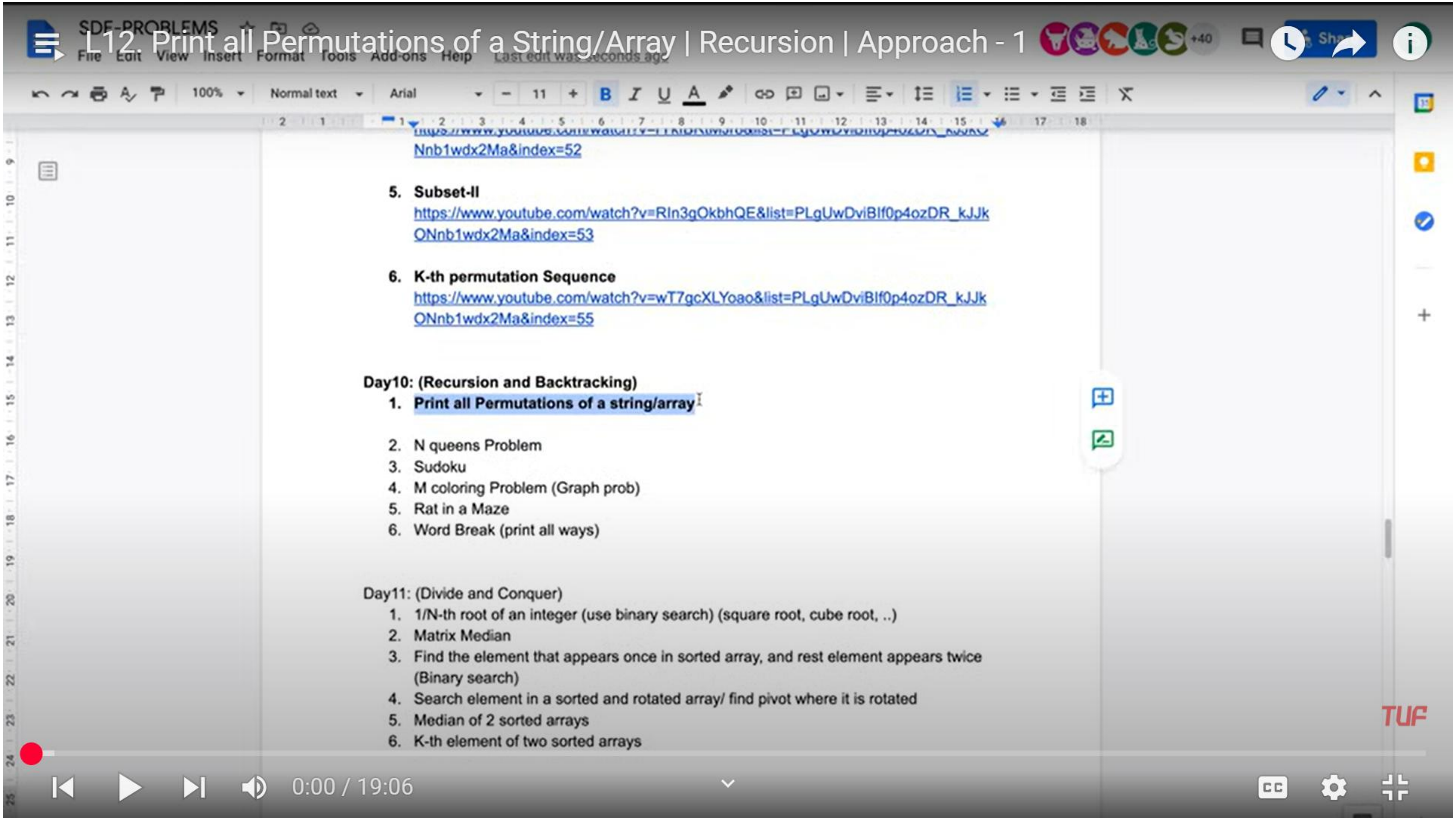


```
CR-84266 -

OpenSSL Changes+Pwd changes+Version changes+FIPS migratool changes b
```



	C	D	E	F	G	H	I
1	g ID's	Dev Owner	Developemt Status	Build No	Build Date	QA Owner	QA Status	Commen
2	Version Update InTrust 11.6.2	Harish	Done	CopyRightVersion_11.6.2/11.6.2.9906			Pending	Dev: We just changed the the intrust pr
3	'd Strength Validation in InTrust Suite Setup	Krishan	Done	PasswordValidation/11.6.2.9934			Pending	Dev: We add some co password . For reference details
4	t for Upgrade Including tools	Krishan /Harish	Done	Fips-Migrator-Tool-11.6.2/11.6.2.9936			Pending	Dev: We made some chan where we add the note KBArticle
5	penSSL vulnerability (CVE 2024 5535)	Mahender/Krishan	Done	FE_OpenSSLUpdate3_4_0/InTrustPackage_11.6.1 .9942.exe			In-Progress	Dev: We upgrade the Op both platfo
6	penSSL vulnerability (CVE 2024 5535)	Krishan	Done	FE_OpenSSLUpdate3_4_0/ InTrustPackage_11.6.1.9974.exe			In-Progress	DEV: We upgrade the Op both platforms + Docum Agent_memory_
7								
8								
9								


L12. Print all Permutations of a String/Array | Recursion | Approach - 1
SDE-PROBLEMS
File Edit View Insert Format Tools Add-ons Help Last edit was seconds ago
Share
100%
Normal text
Arial
11
TUF
0:00 / 19:06

Nnb1wdx2Ma&index=52

5. **Subset-II**
   https://www.youtube.com/watch?v=RIn3gOkbhQE&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=53

6. **K-th permutation Sequence**
   https://www.youtube.com/watch?v=wT7gcXLYoao&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=55

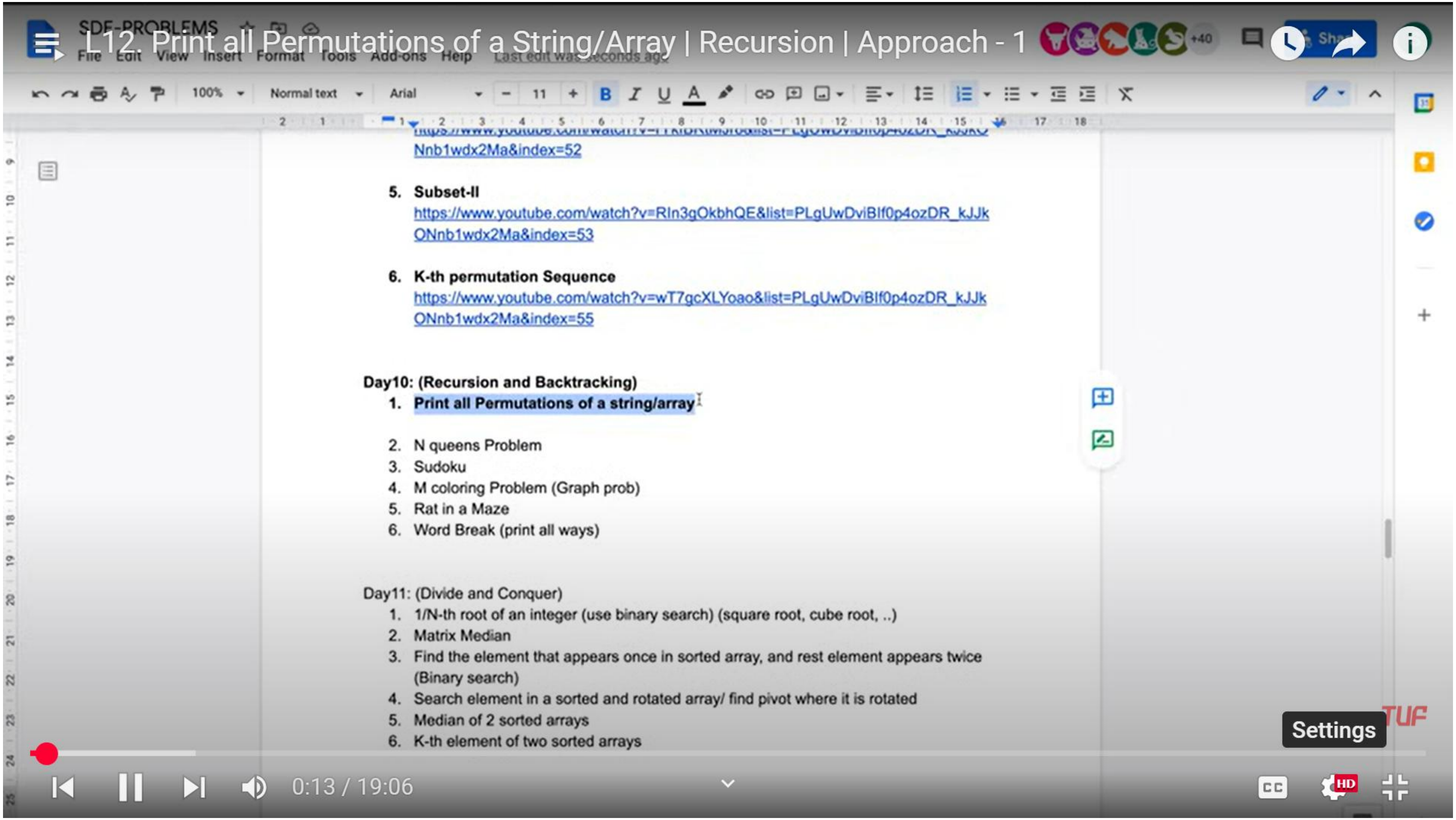
Day10: (Recursion and Backtracking)

1. **Print all Permutations of a string/array**

2. N queens Problem
3. Sudoku
4. M coloring Problem (Graph prob)
5. Rat in a Maze
6. Word Break (print all ways)

Day11: (Divide and Conquer)

1. 1/N-th root of an integer (use binary search) (square root, cube root, ..)
2. Matrix Median
3. Find the element that appears once in sorted array, and rest element appears twice (Binary search)
4. Search element in a sorted and rotated array/ find pivot where it is rotated
5. Median of 2 sorted arrays
6. K-th element of two sorted arrays


L12. Print all Permutations of a String/Array | Recursion | Approach - 1
SDF-PROBLEMS
File Edit View Insert Format Tools Add-ons Help
Last edit was seconds ago
+40
Share
100%
Normal text
Arial
11
TUF
Settings
0:13 / 19:06
HD

Nnb1wdx2Ma&index=52

5. **Subset-II**
https://www.youtube.com/watch?v=RIn3gOkbhQE&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=53

6. **K-th permutation Sequence**
https://www.youtube.com/watch?v=wT7gcXLYoao&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=55

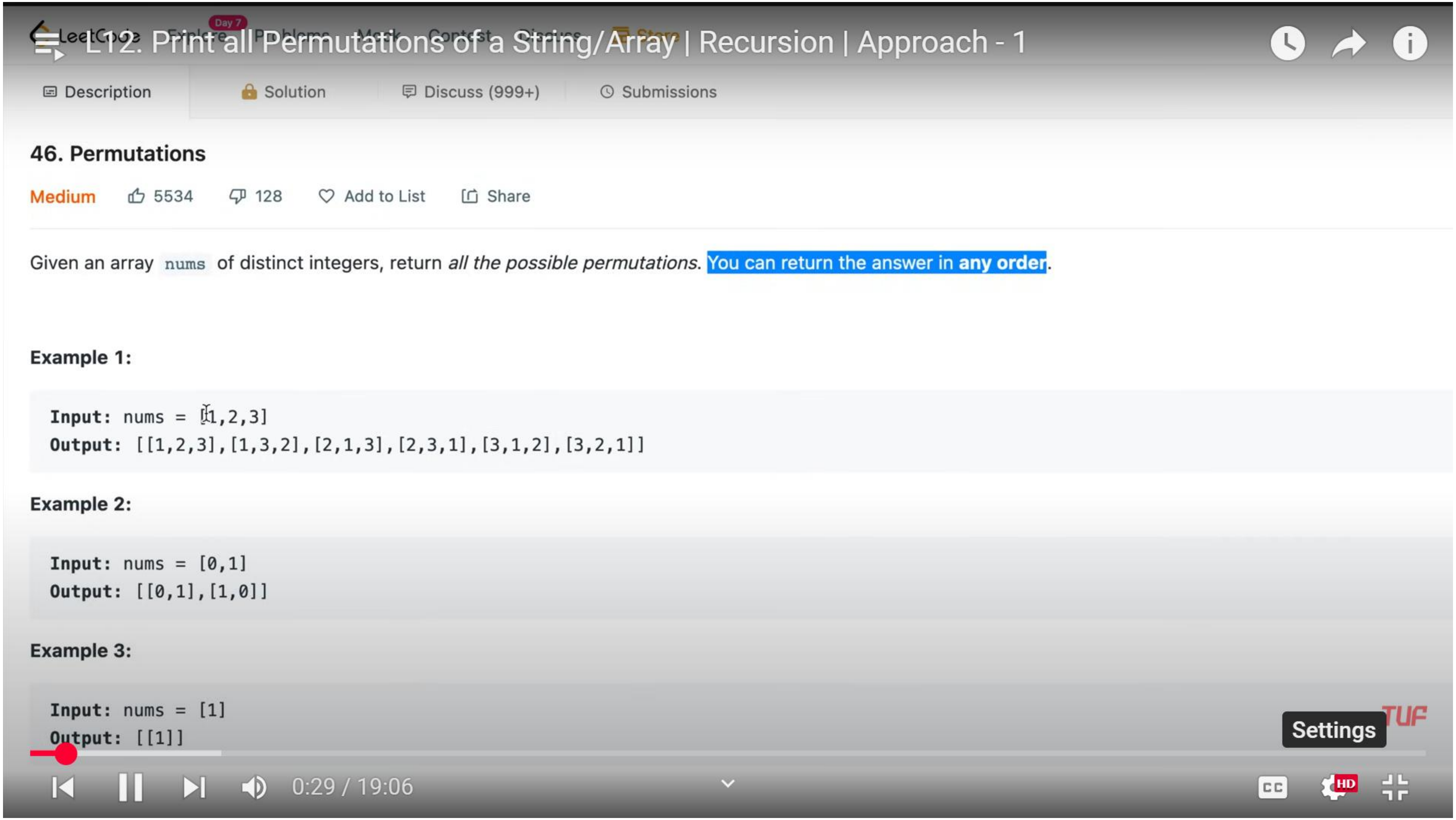
Day10: (Recursion and Backtracking)

1. **Print all Permutations of a string/array**

2. N queens Problem
3. Sudoku
4. M coloring Problem (Graph prob)
5. Rat in a Maze
6. Word Break (print all ways)

Day11: (Divide and Conquer)

1. 1/N-th root of an integer (use binary search) (square root, cube root, ..)
2. Matrix Median
3. Find the element that appears once in sorted array, and rest element appears twice (Binary search)
4. Search element in a sorted and rotated array/ find pivot where it is rotated
5. Median of 2 sorted arrays
6. K-th element of two sorted arrays



46. Permutations

Given an array `nums` of distinct integers, return *all the possible permutations*. You can return the answer in **any order**.

Example 1:

```
Input: nums = [1,2,3]
Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Example 2:

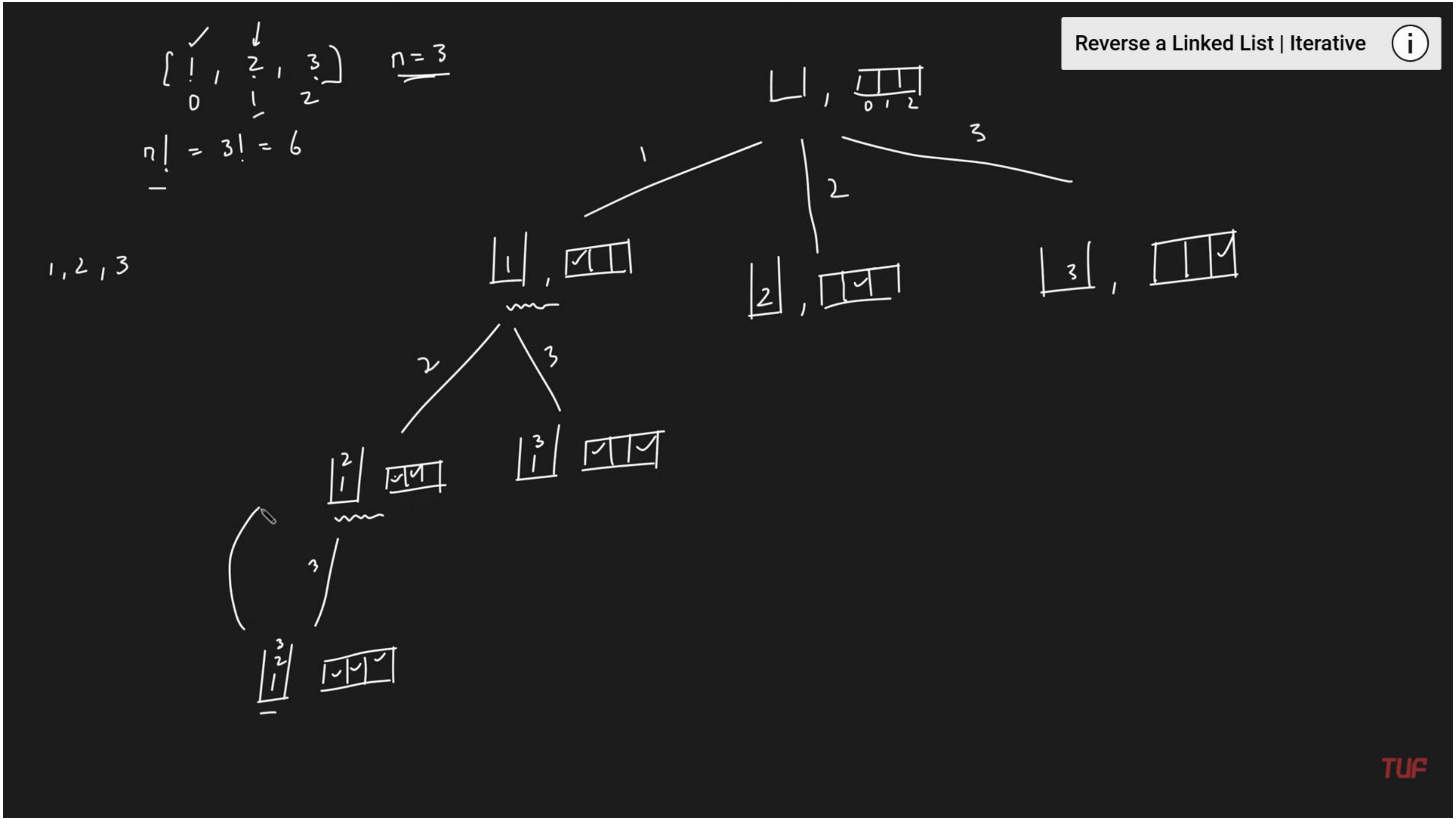
```
Input: nums = [0,1]
Output: [[0,1],[1,0]]
```

Example 3:

```
Input: nums = [1]
Output: [[1]]
```

[1, 2, 3]

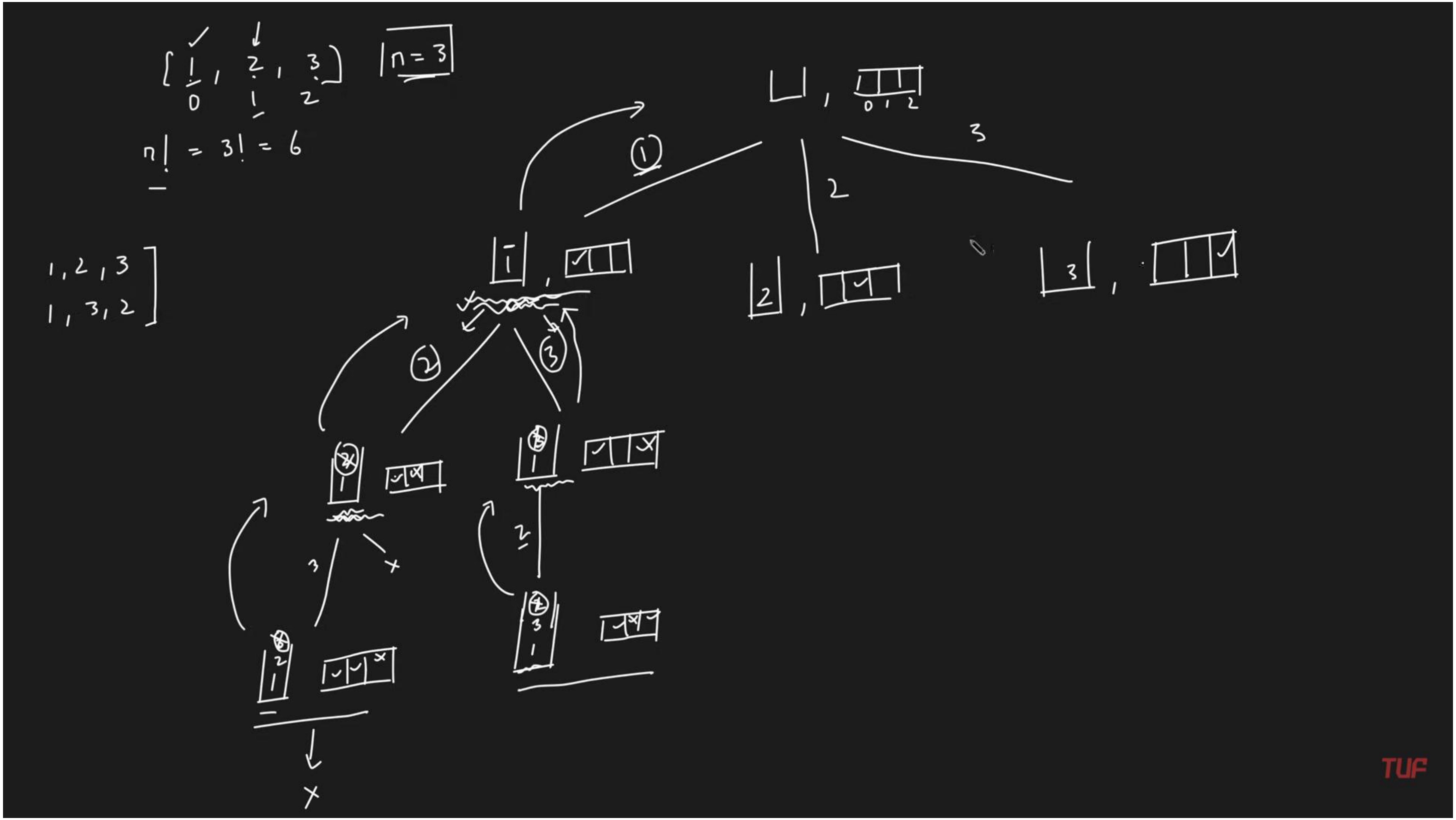
Let's give back to the community :)

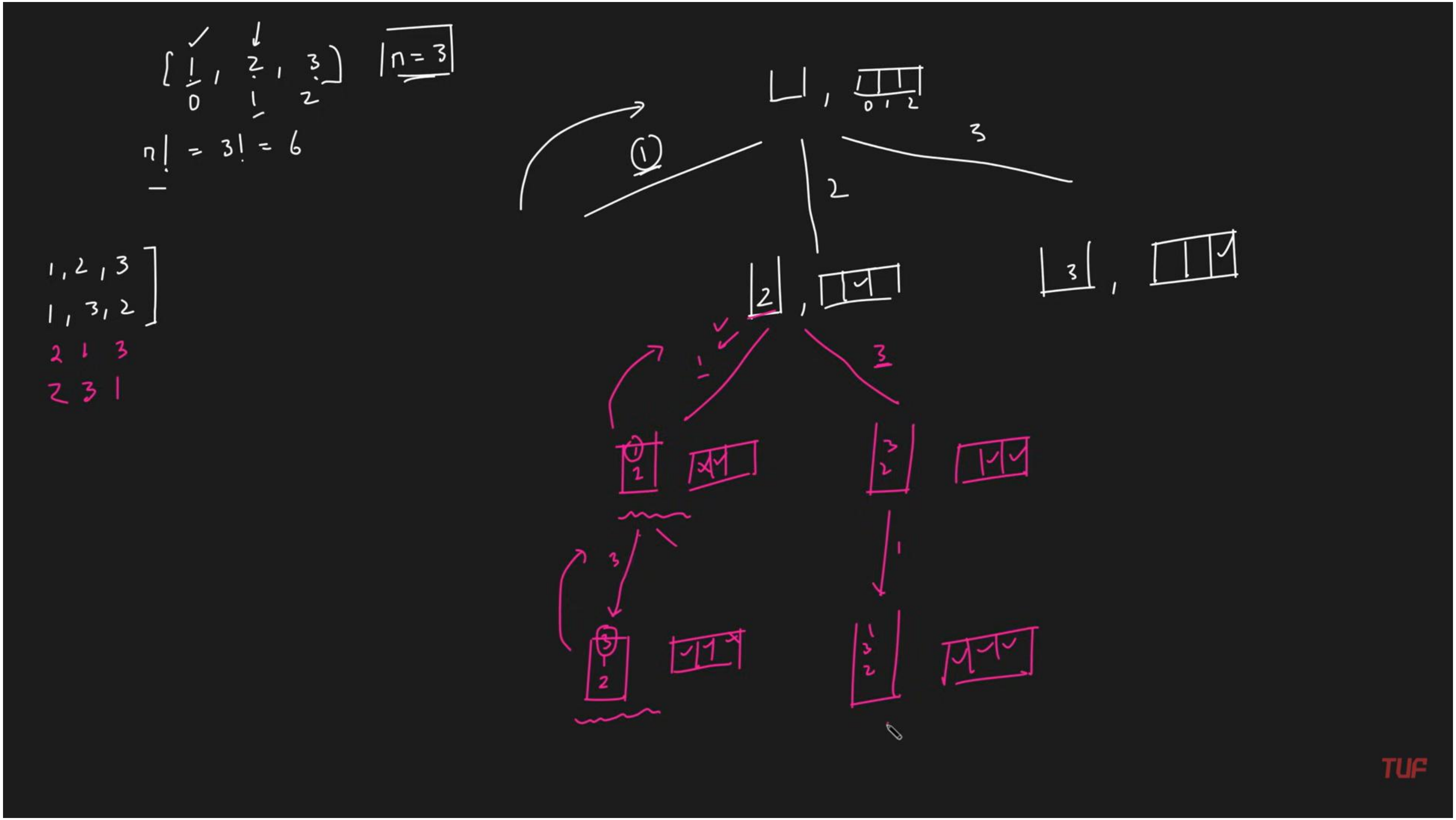


$[1, 2, 3]$ $n = 3$

$n! = 3! = 6$

1, 2, 3


[1, 2, 3]
0 1 2
n = 3
n! = 3! = 6
1, 2, 3
1, 3, 2
[ ], [| |]
0 1 2
1
2
3
[1], [✓| |]
[2], [|✓|]
[3], [| |✓]
2
3
[1, 2], [✓|✓|]
[1, 3], [✓| |✓]
3
2
×
[1, 2, 3], [✓|✓|✓]
[1, 3, 2], [✓|✓|✓]
×
TUF



[1, 2, 3]

0 1 2

n = 3

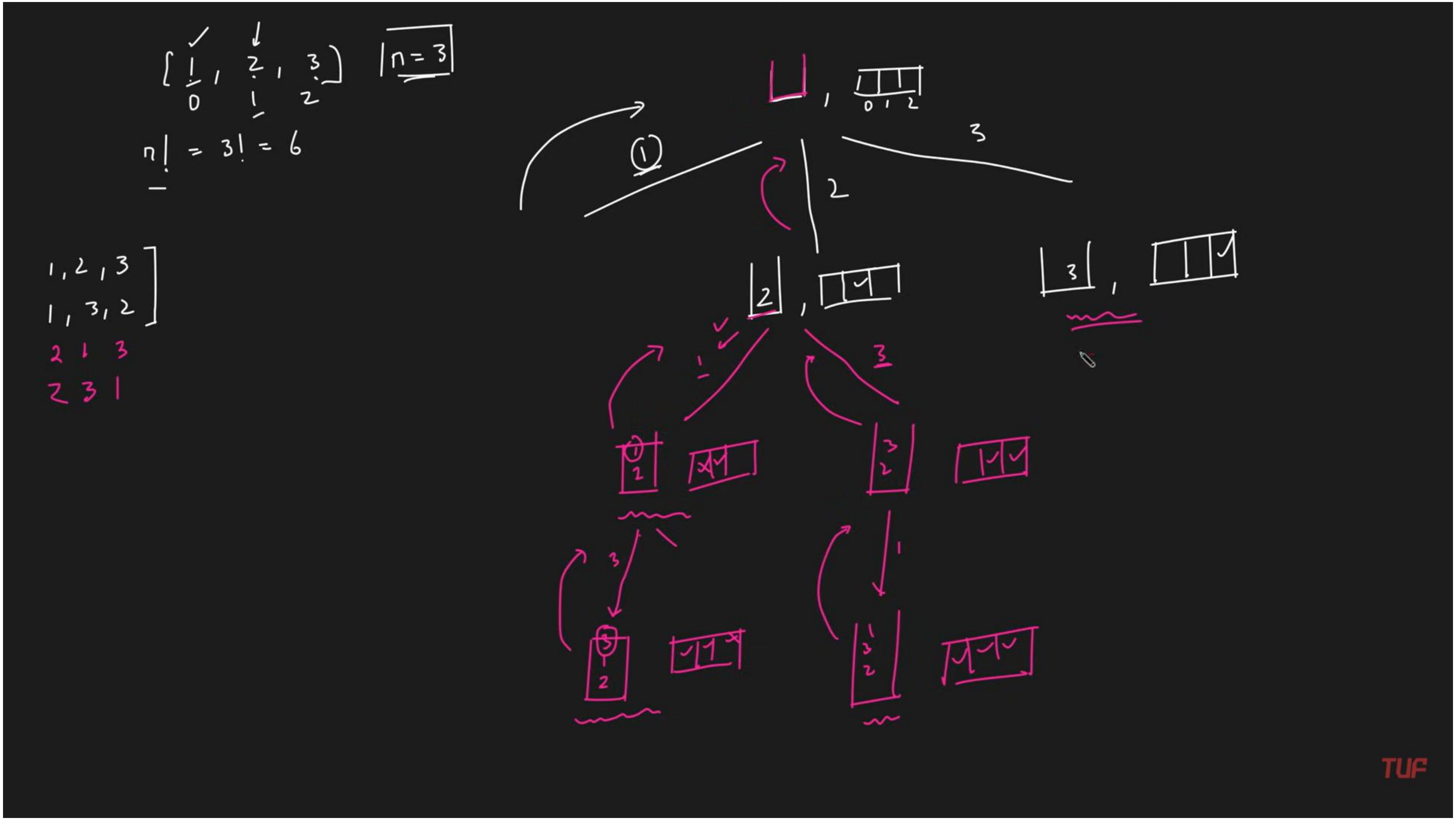
n! = 3! = 6

1, 2, 3

1, 3, 2

2 1 3

2 3 1



$$\begin{array}{ccc} 1, & 2, & 3 \\ 0 & 1 & 2 \end{array}$$

$\boxed{n = 3}$

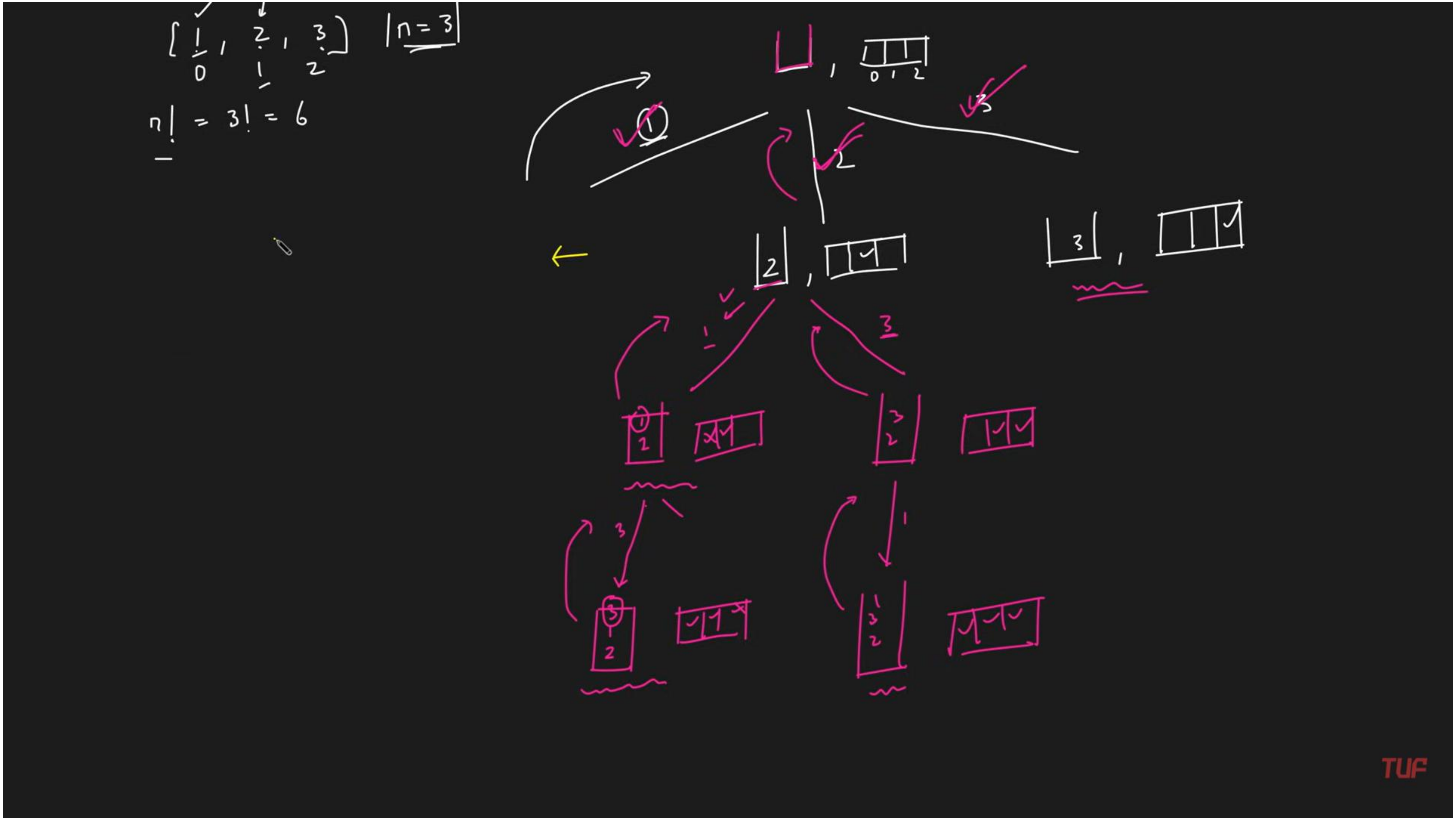
$n! = 3! = 6$

1, 2, 3

1, 3, 2

2 1 3

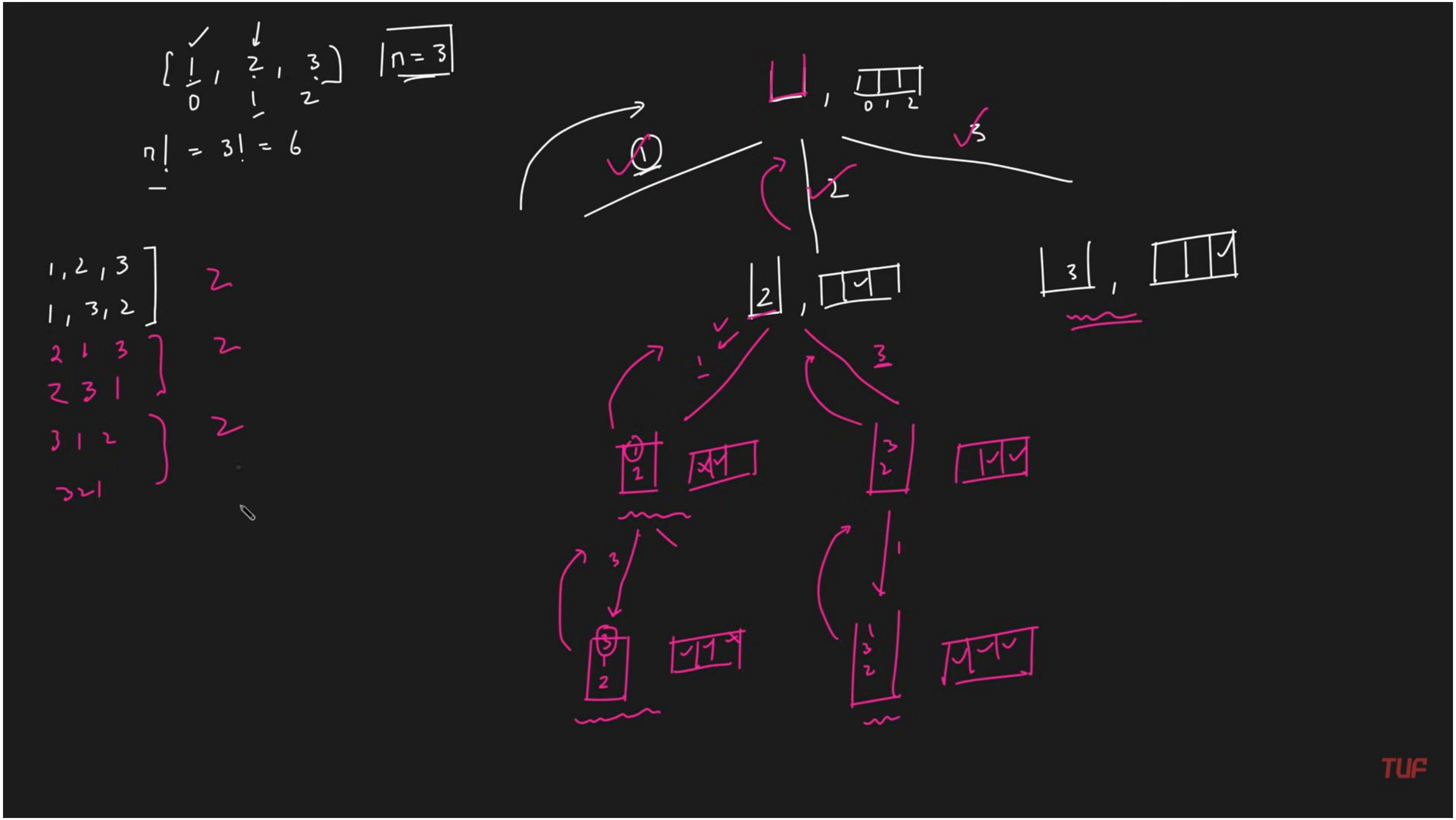
2 3 1



[1, 2, 3] n = 3

0 1 2

$n! = 3! = 6$

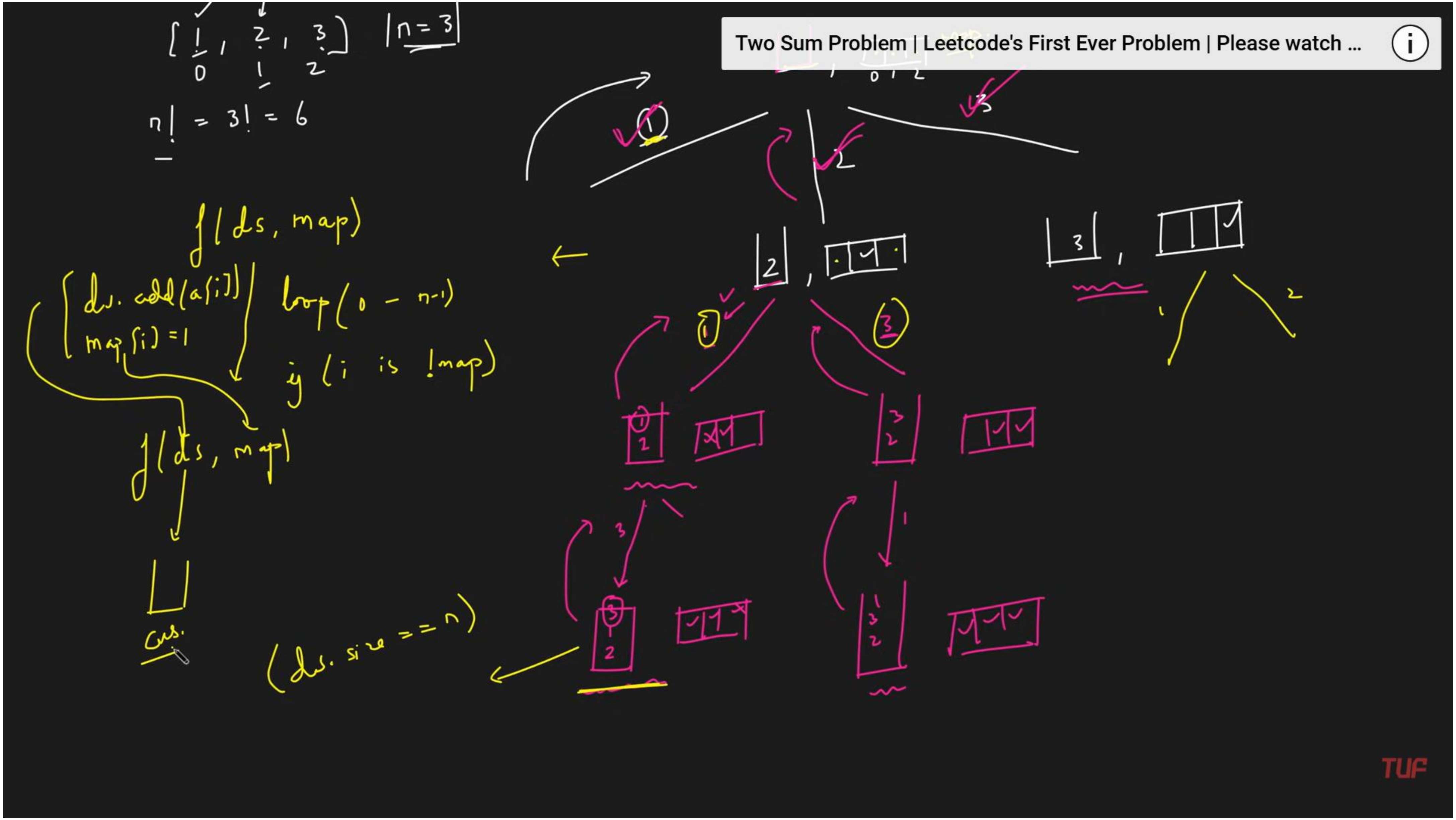


[1, 2, 3]
0 1 2

n = 3

n! = 3! = 6

1, 2, 3
1, 3, 2   2
2 1 3
2 3 1   2
3 1 2   2
3 2 1



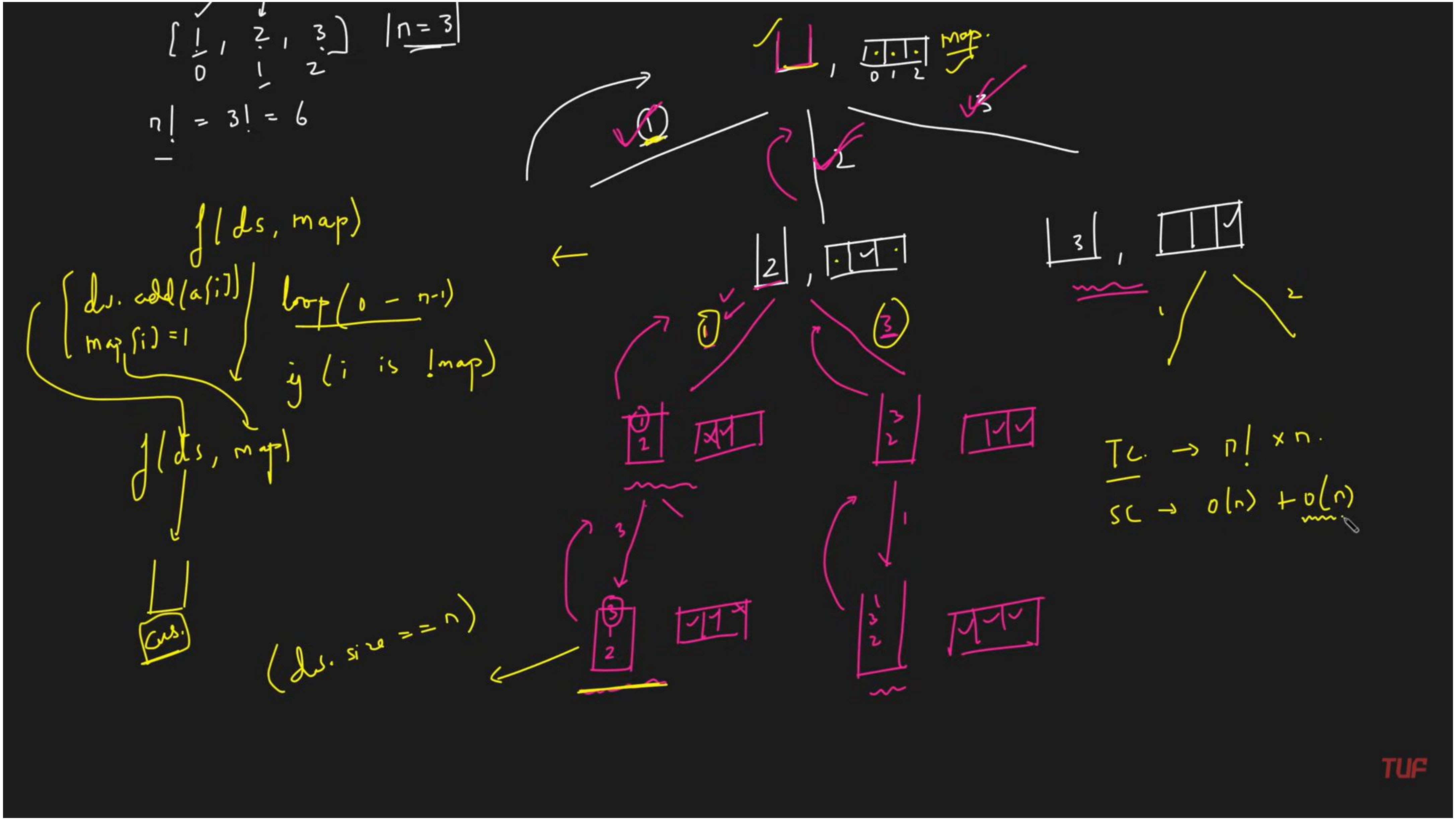
Two Sum Problem | Leetcode's First Ever Problem | Please watch ...

[1, 2, 3]   n = 3
0   1   2

n! = 3! = 6

f(ds, map)

```
f(ds, map)
    ds.add(a[i])
    map[i] = 1
    loop(0 - n-1)
        if (i is !map)
            f(ds, map)
(ds.size == n)
    ans.
```



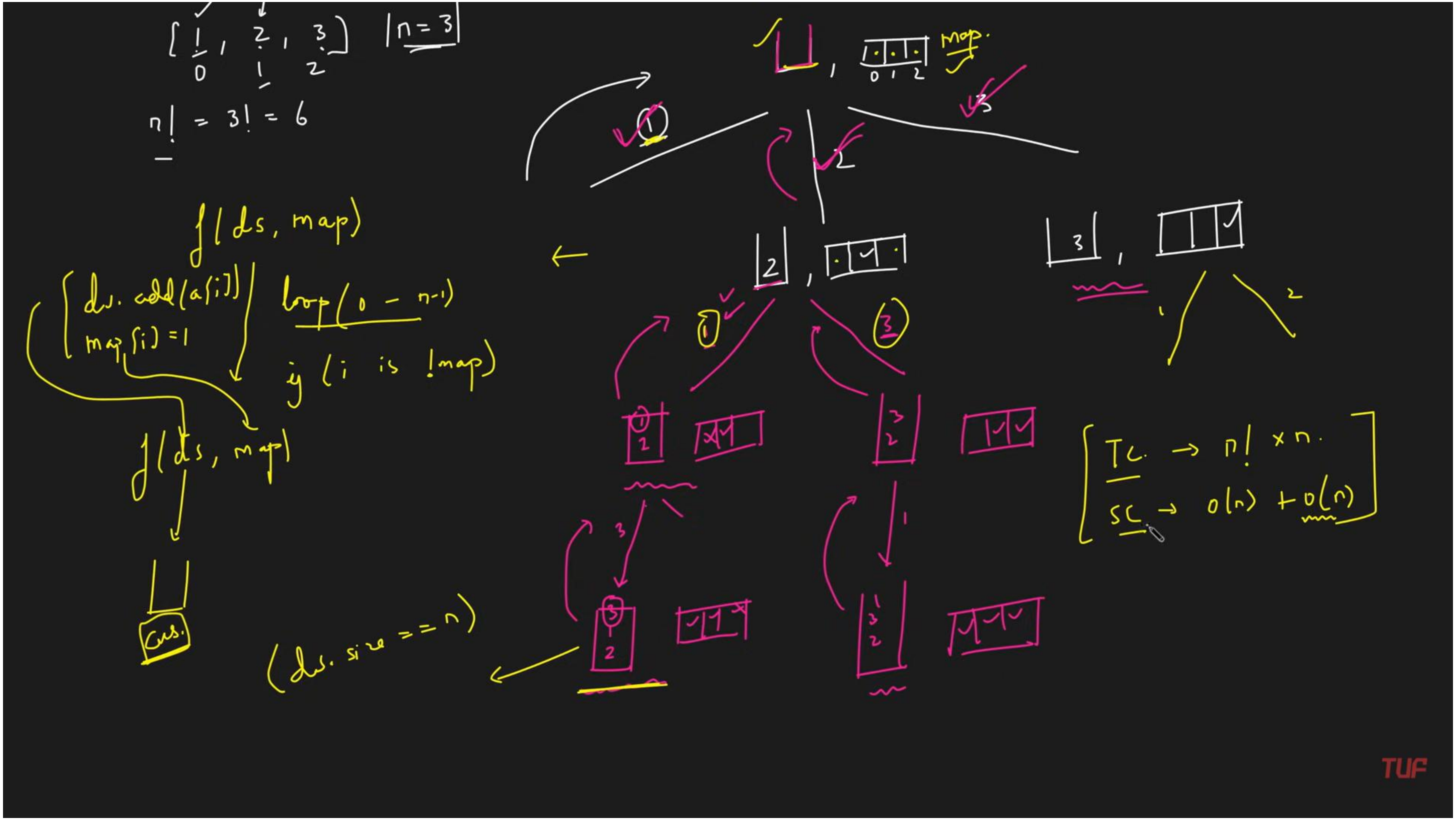
[1, 2, 3] |n=3|
0 1 2

n! = 3! = 6

```
f(ds, map)
    ds.add(a[i])
    map[i] = 1
    loop(0 - n-1)
    if (i is !map)
        f(ds, map)
ans.
(ds.size == n)
```

TC → n! × n.

SC → O(n) + O(n)



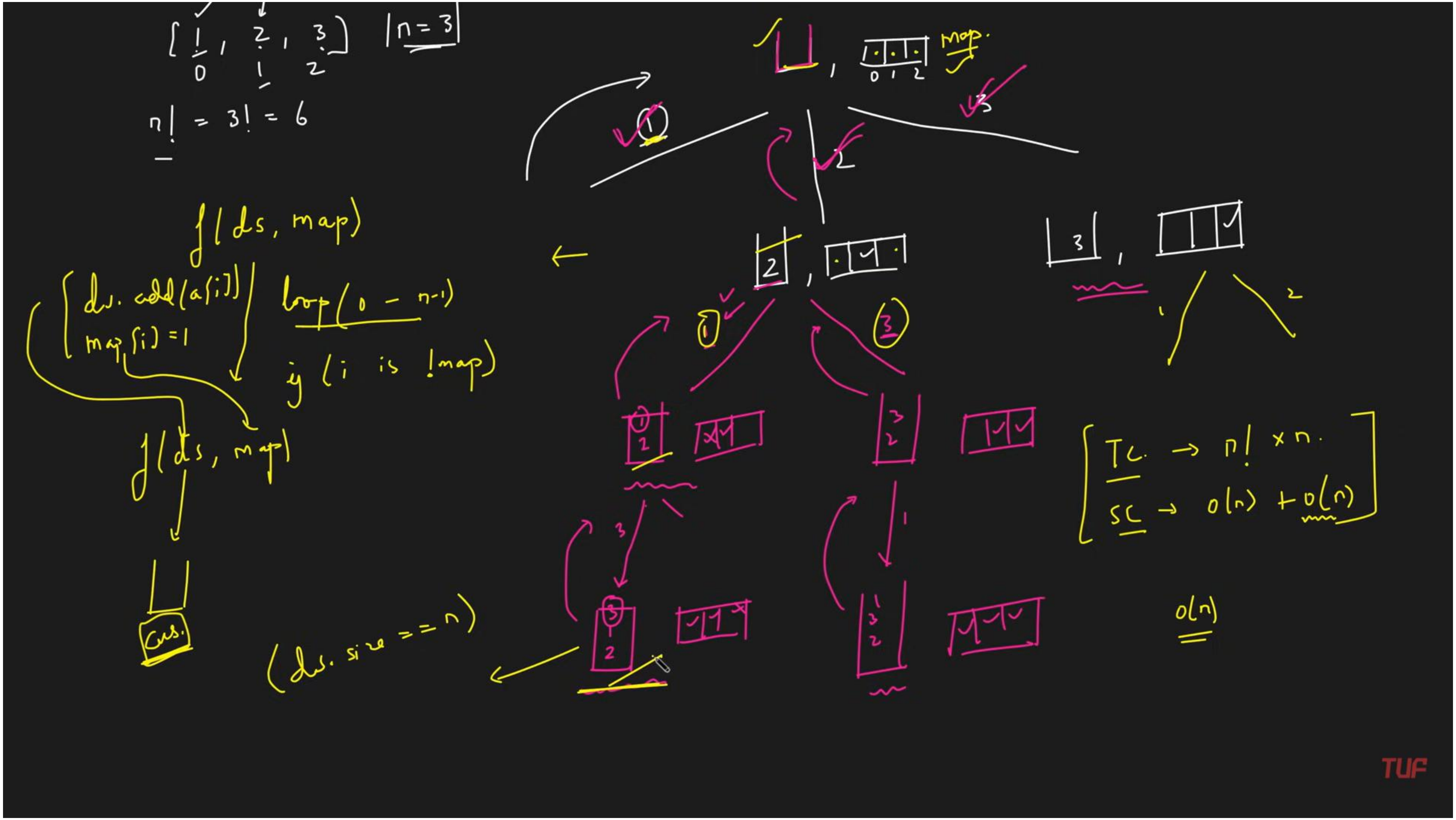
[1, 2, 3] |n = 3|
0 1 2

n! = 3! = 6

```
f(ds, map)
    [ ds.add(a[i])
      map[i] = 1
    loop(0 - n-1)
    if (i is !map)
    f(ds, map)
ans.
(ds.size == n)
```

TC → n! × n.

SC → O(n) + O(n)



[1, 2, 3] |n = 3|
0 1 2

n! = 3! = 6

```
f(ds, map)
    ds.add(a[i])
    map[i] = 1
    f(ds, map)
loop(0 - n-1)
    if (i is !map)
```

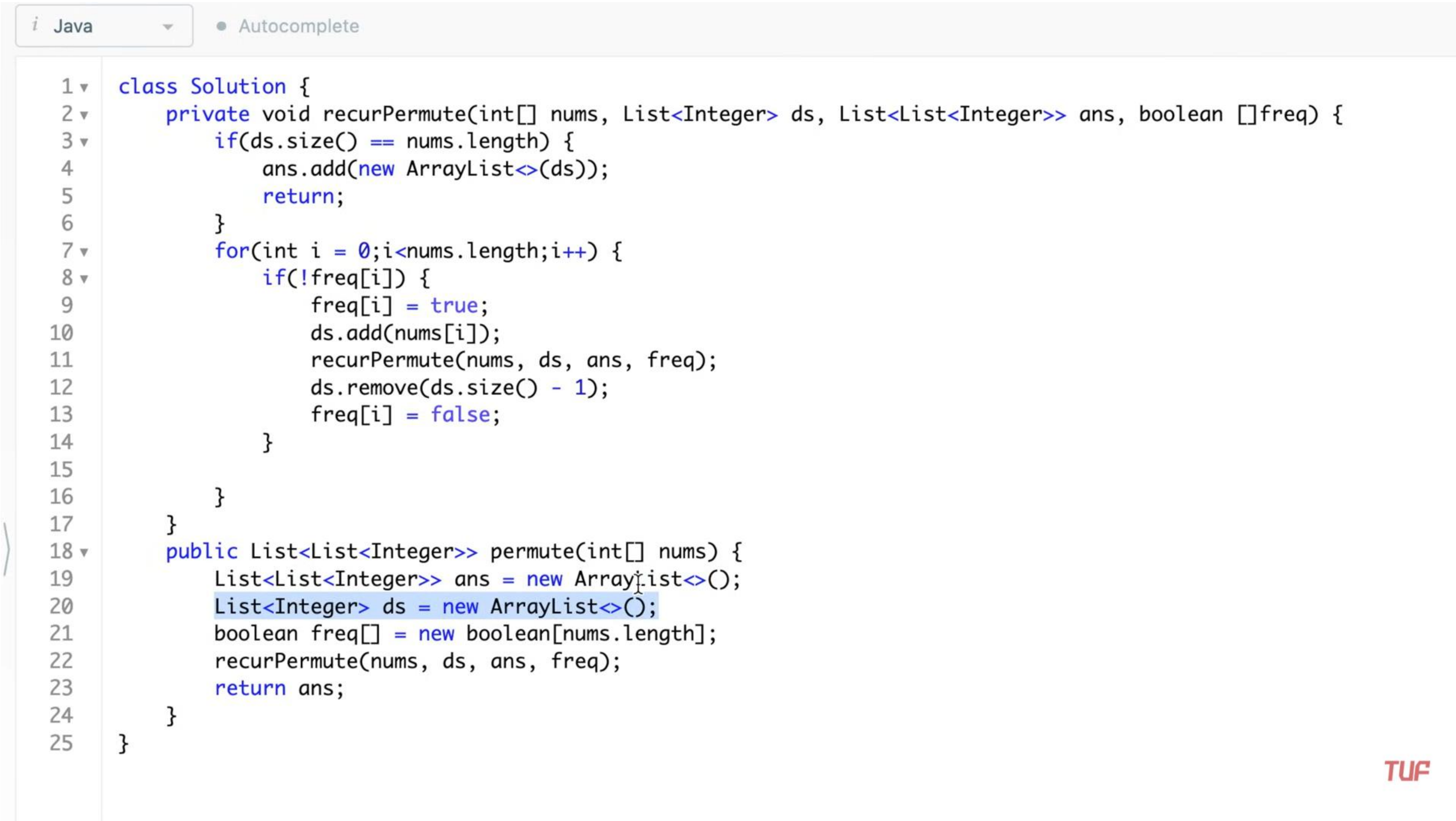
ans.

(ds.size == n)

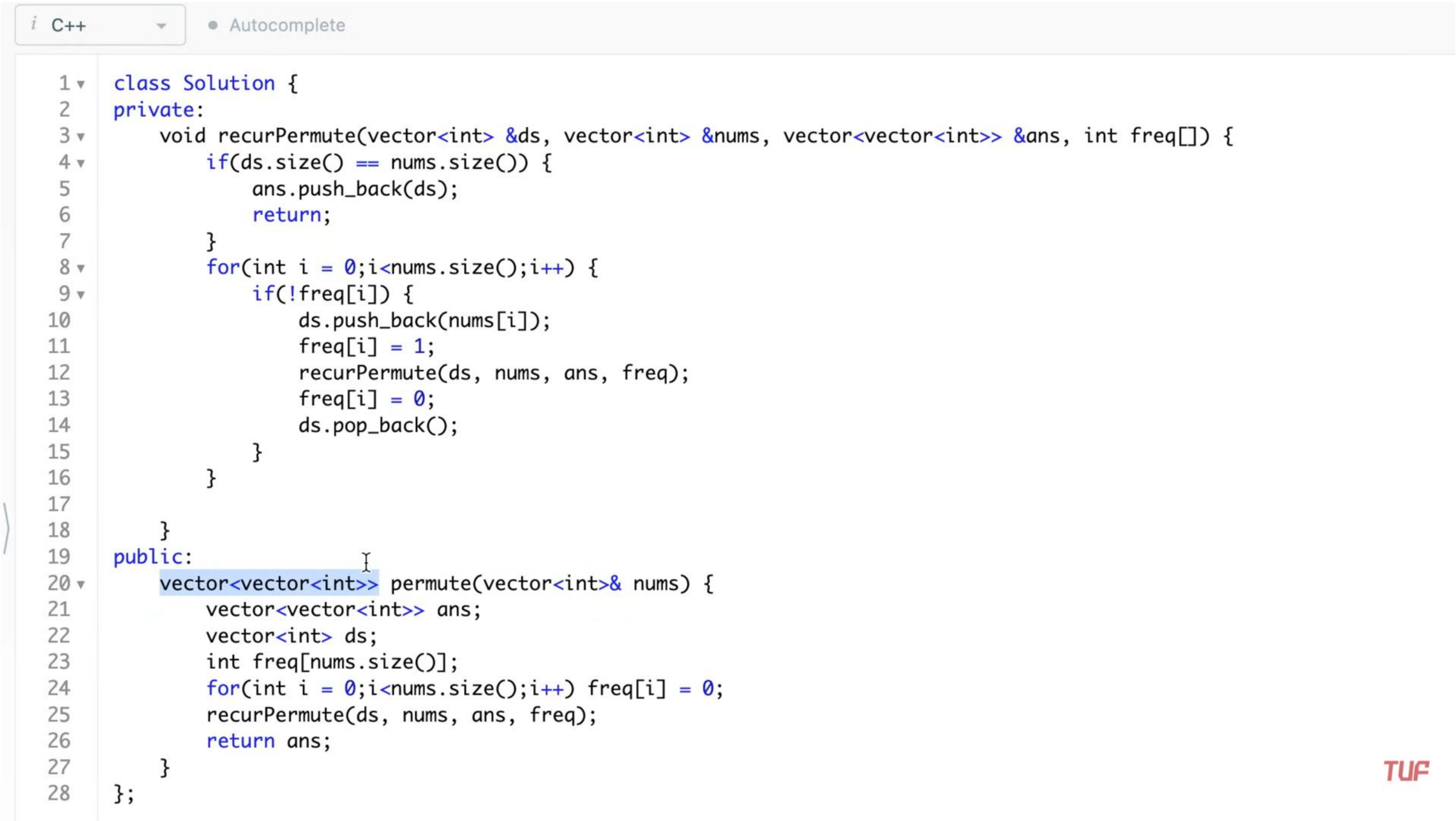
TC → n! × n.

SC → O(n) + O(n)

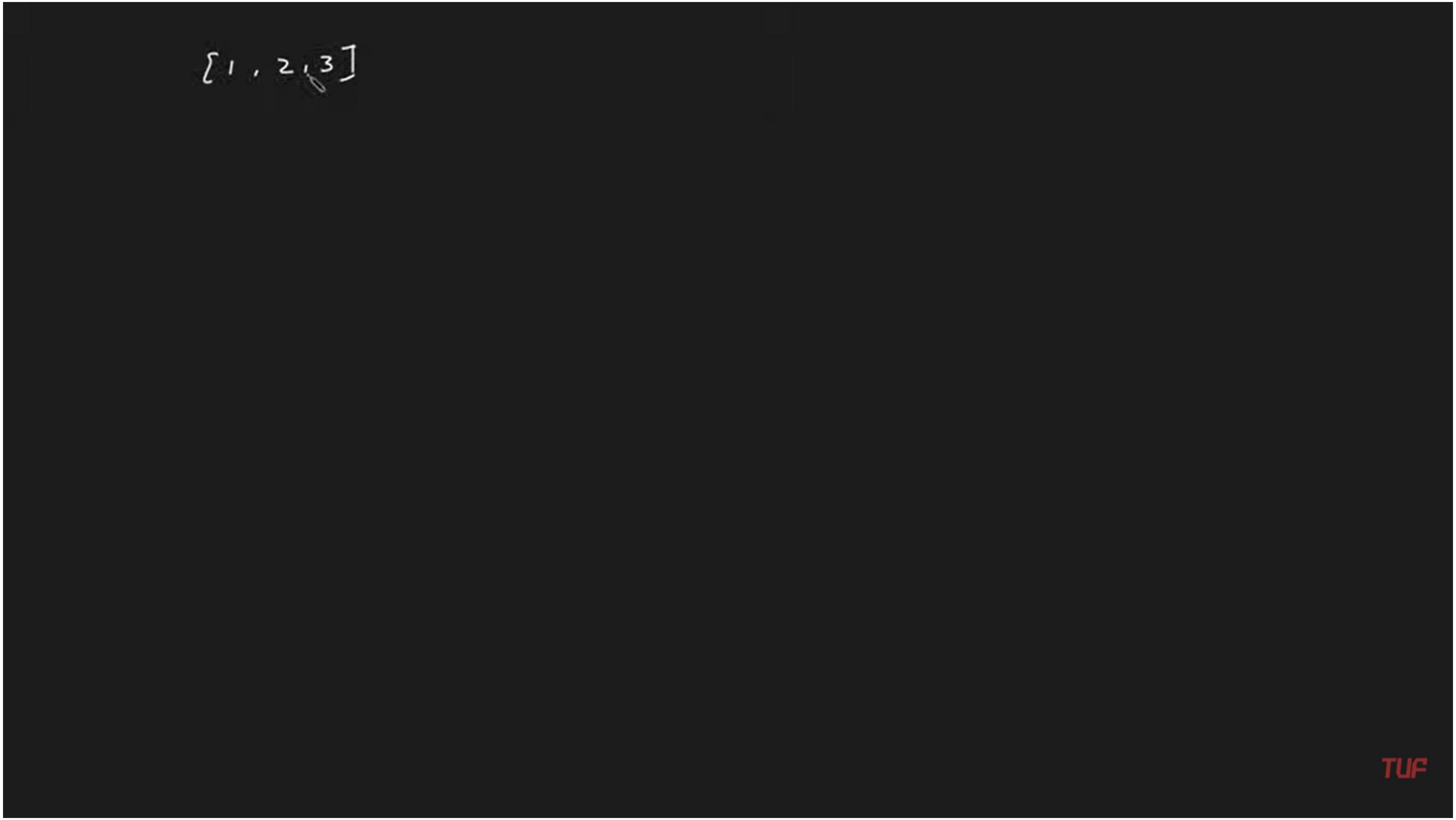
O(n)



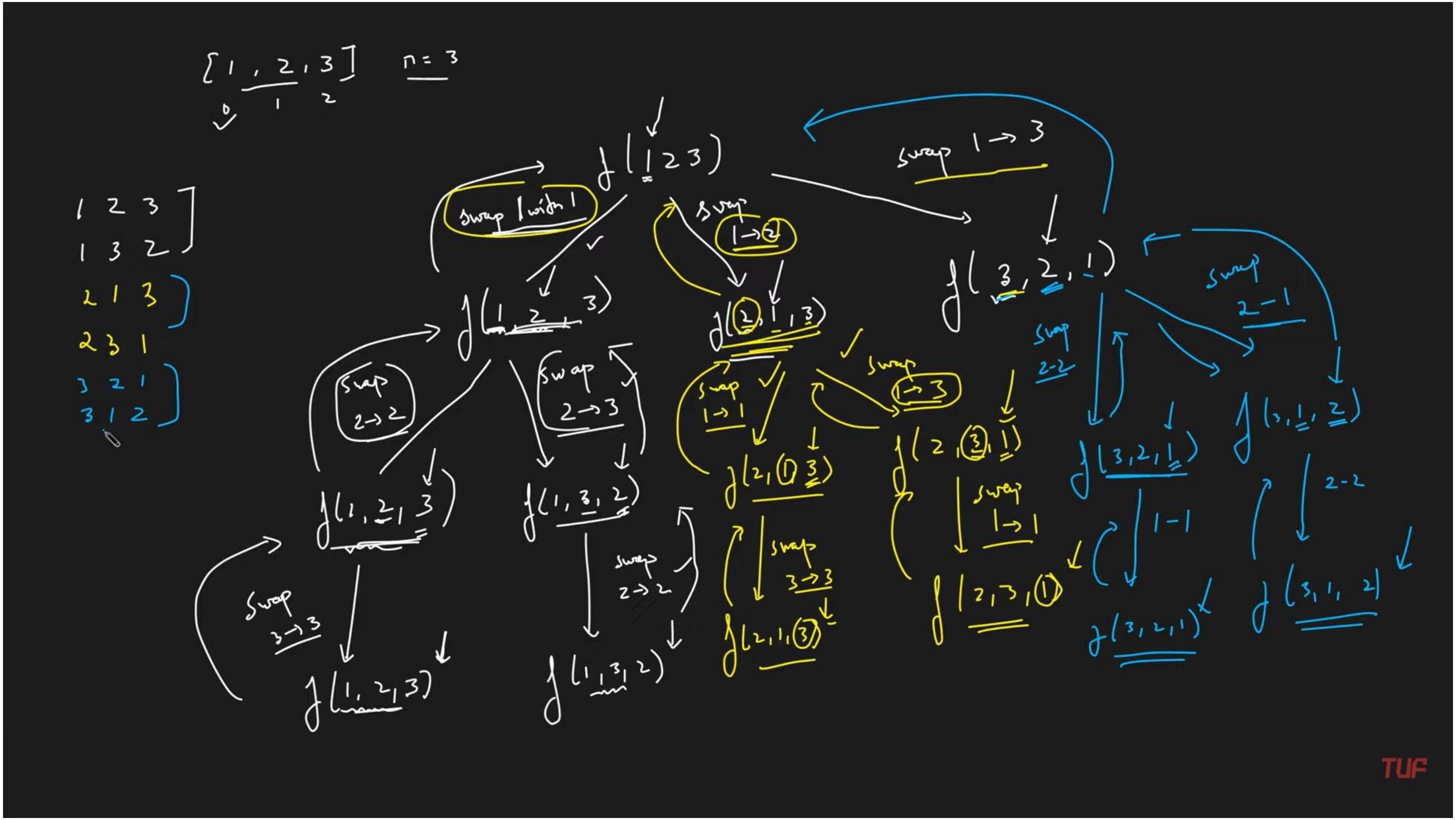
```java
class Solution {
    private void recurPermute(int[] nums, List<Integer> ds, List<List<Integer>> ans, boolean []freq) {
        if(ds.size() == nums.length) {
            ans.add(new ArrayList<>(ds));
            return;
        }
        for(int i = 0;i<nums.length;i++) {
            if(!freq[i]) {
                freq[i] = true;
                ds.add(nums[i]);
                recurPermute(nums, ds, ans, freq);
                ds.remove(ds.size() - 1);
                freq[i] = false;
            }

        }
    }
    public List<List<Integer>> permute(int[] nums) {
        List<List<Integer>> ans = new ArrayList<>();
        List<Integer> ds = new ArrayList<>();
        boolean freq[] = new boolean[nums.length];
        recurPermute(nums, ds, ans, freq);
        return ans;
    }
}
```

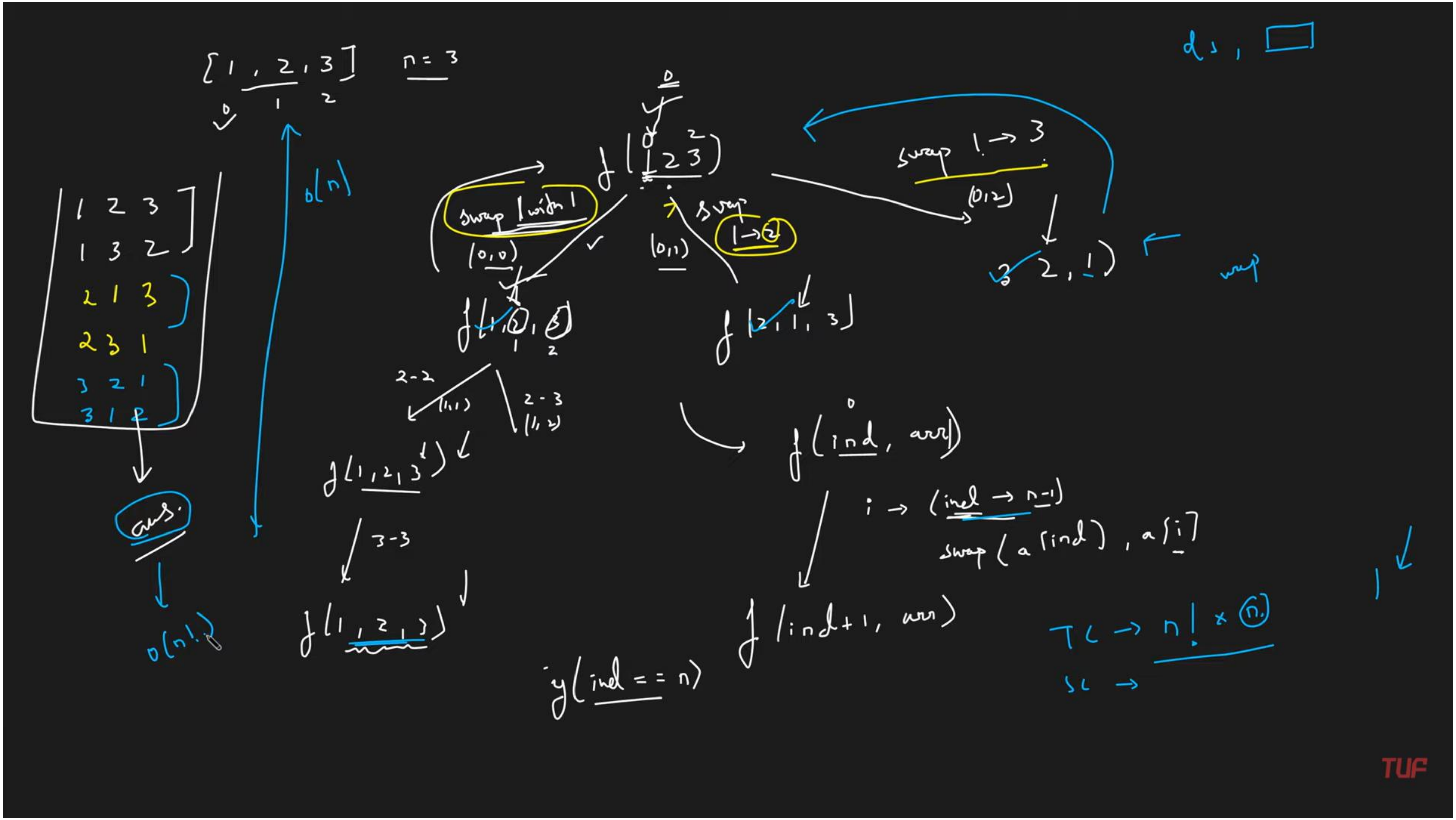


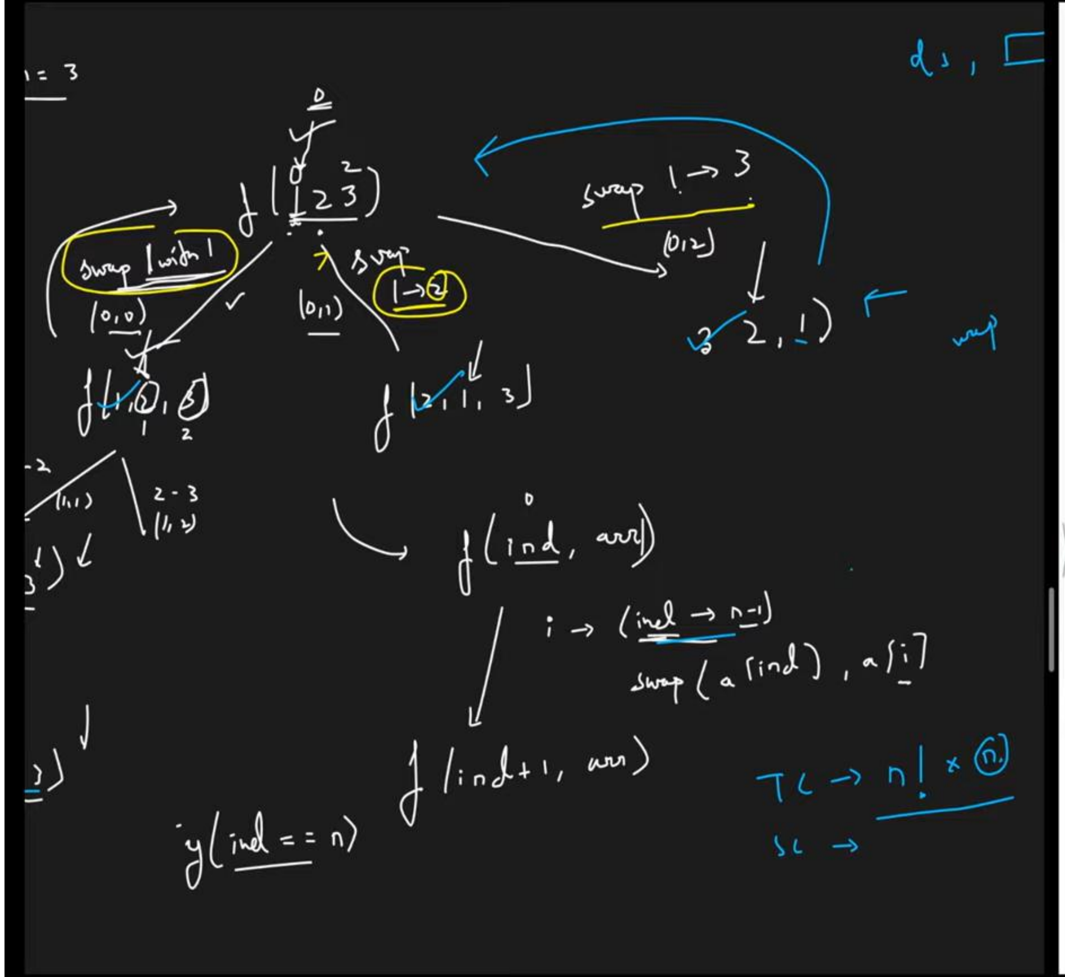
```cpp
class Solution {
private:
    void recurPermute(vector<int> &ds, vector<int> &nums, vector<vector<int>> &ans, int freq[]) {
        if(ds.size() == nums.size()) {
            ans.push_back(ds);
            return;
        }
        for(int i = 0;i<nums.size();i++) {
            if(!freq[i]) {
                ds.push_back(nums[i]);
                freq[i] = 1;
                recurPermute(ds, nums, ans, freq);
                freq[i] = 0;
                ds.pop_back();
            }
        }

    }
public:
    vector<vector<int>> permute(vector<int>& nums) {
        vector<vector<int>> ans;
        vector<int> ds;
        int freq[nums.size()];
        for(int i = 0;i<nums.size();i++) freq[i] = 0;
        recurPermute(ds, nums, ans, freq);
        return ans;
    }
};
```

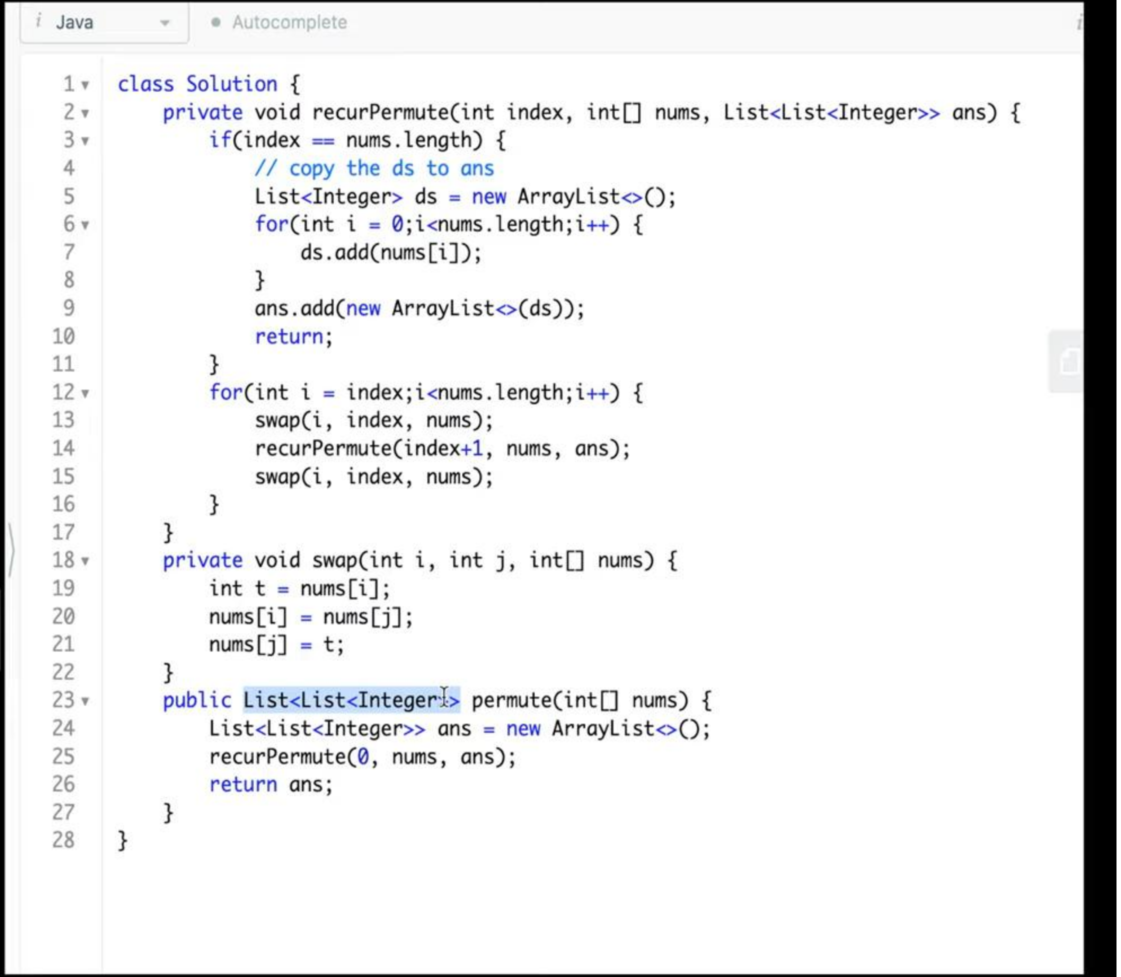


[1, 2, 3]

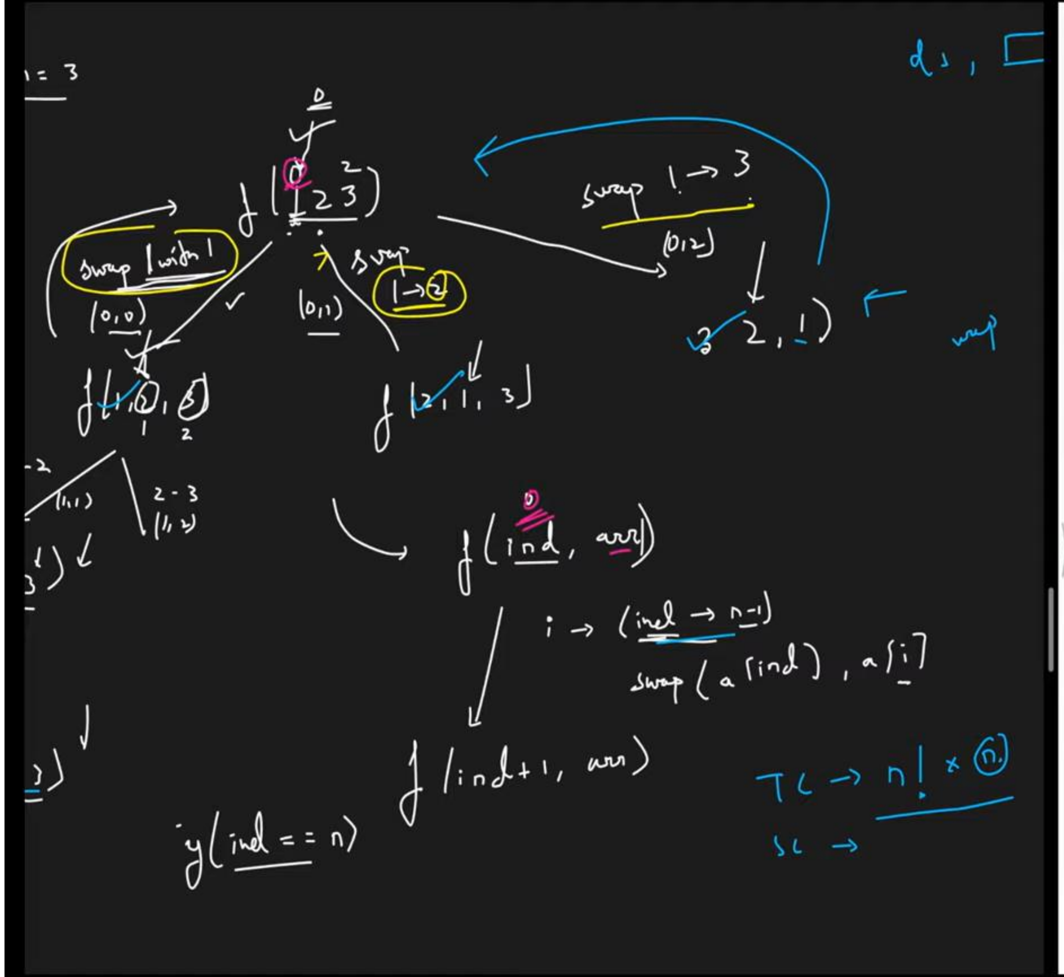

[1, 2, 3]   n = 3
0   1   2
1 2 3
1 3 2
2 1 3
2 3 1
3 2 1
3 1 2
f(1 2 3)
swap 1 with 1
swap 1 -> 2
swap 1 -> 3
f(1, 2, 3)
f(2, 1, 3)
f(3, 2, 1)
swap 2 -> 2
swap 2 -> 3
swap 1 -> 1
swap 1 -> 3
swap 2-2
swap 2 -> 1
f(1, 2, 3)
f(1, 3, 2)
f(2, 1, 3)
f(2, 3, 1)
f(3, 2, 1)
f(3, 1, 2)
swap 3 -> 3
swap 2 -> 2
swap 3 -> 3
swap 1 -> 1
1-1
2-2
f(1, 2, 3)
f(1, 3, 2)
f(2, 1, 3)
f(2, 3, 1)
f(3, 2, 1)
f(3, 1, 2)
TUF

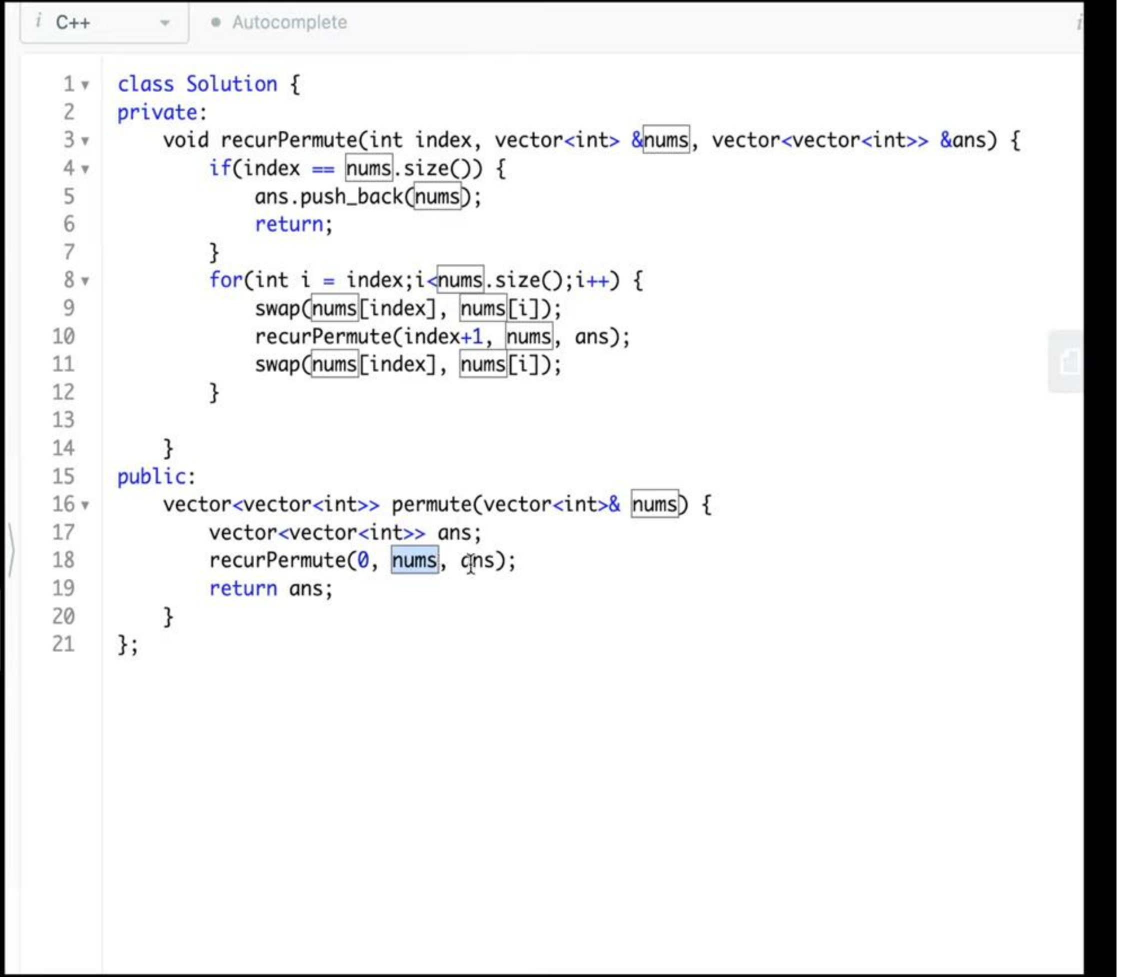

[1, 2, 3]
n = 3
0 1 2
o(n)
1 2 3
1 3 2
2 1 3
2 3 1
3 2 1
3 1 2
ans.
o(n!)
ds, □
0
f(1 2 3)
swap 1 with 1
(0,0)
swap 1→2
(0,1)
swap 1→3
(0,2)
f(1, 2, 3)
1 2
f(2, 1, 3)
3 2, 1)
swap
2-2
(1,1)
2-3
(1,2)
f(1,2,3)
3-3
f(1, 2, 3)
f(ind, arr)
i → (ind → n-1)
swap(a[ind], a[i])
f(ind+1, arr)
if(ind == n)
TC → n! × n
SC →





```java
class Solution {
    private void recurPermute(int index, int[] nums, List<List<Integer>> ans) {
        if(index == nums.length) {
            // copy the ds to ans
            List<Integer> ds = new ArrayList<>();
            for(int i = 0;i<nums.length;i++) {
                ds.add(nums[i]);
            }
            ans.add(new ArrayList<>(ds));
            return;
        }
        for(int i = index;i<nums.length;i++) {
            swap(i, index, nums);
            recurPermute(index+1, nums, ans);
            swap(i, index, nums);
        }
    }
    private void swap(int i, int j, int[] nums) {
        int t = nums[i];
        nums[i] = nums[j];
        nums[j] = t;
    }
    public List<List<Integer>> permute(int[] nums) {
        List<List<Integer>> ans = new ArrayList<>();
        recurPermute(0, nums, ans);
        return ans;
    }
}
```





```cpp
class Solution {
private:
    void recurPermute(int index, vector<int> &nums, vector<vector<int>> &ans) {
        if(index == nums.size()) {
            ans.push_back(nums);
            return;
        }
        for(int i = index;i<nums.size();i++) {
            swap(nums[index], nums[i]);
            recurPermute(index+1, nums, ans);
            swap(nums[index], nums[i]);
        }

    }
public:
    vector<vector<int>> permute(vector<int>& nums) {
        vector<vector<int>> ans;
        recurPermute(0, nums, ans);
        return ans;
    }
};
```