SDE-PROBLEMS

# L18. K-th Permutation Sequence | Leetcode

To exit full screen, press Esc

Day9 (Recursion):

1. **Combination sum-1**
   https://www.youtube.com/watch?v=OyZFFqQtu98&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=49

2. **Combination sum-2**
   https://www.youtube.com/watch?v=G1fRTGRxXU8&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=50

3. **Palindrome Partitioning**
   https://www.youtube.com/watch?v=WBgsABoCIE0&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=51

4. **Subset Sums**
   https://www.youtube.com/watch?v=rYkfBRtMJr8&list=PLgUwDviBIf0p4ozDR_kJJkONnb1wdx2Ma&index=52

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

6. **K-th permutation Sequence**

Day10: (Backtracking)

1. N queens Problem
2. Sudoko
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)

0:03 / 24:40 • Intro ›

## 60. Permutation Sequence

Hard 2081 360 Add to List Share

The set `[1, 2, 3, ..., n]` contains a total of `n!` unique permutations.

By listing and labeling all of the permutations in order, we get the following sequence for `n = 3`:

1. `"123"`
2. `"132"`
3. `"213"`
4. `"231"`
5. `"312"`
6. `"321"`

Given `n` and `k`, return the $k^{th}$ permutation sequence.

**Example 1:**

```
Input: n = 3, k = 3
Output: "213"
```

**Example 2:**

```
Input: n = 4, k = 9
Output: "2314"
```

**Example 3:**

```
Input: n = 3, k = 1
Output: "123"
```

TUF

n = 4 , K = 17

TUF

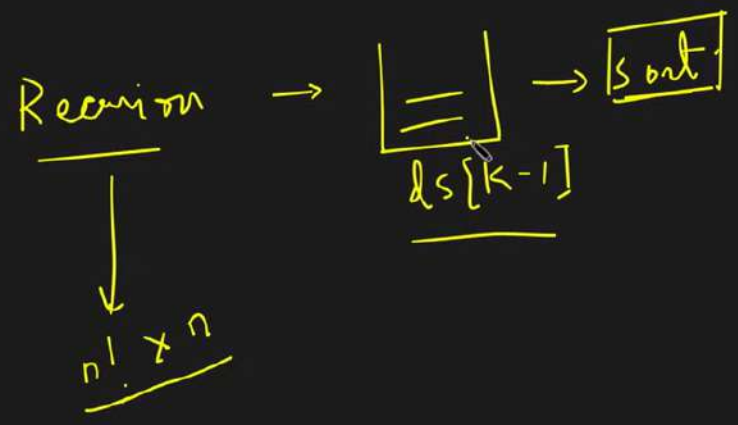
n = 4 , K = 17

Recursion → → sort.

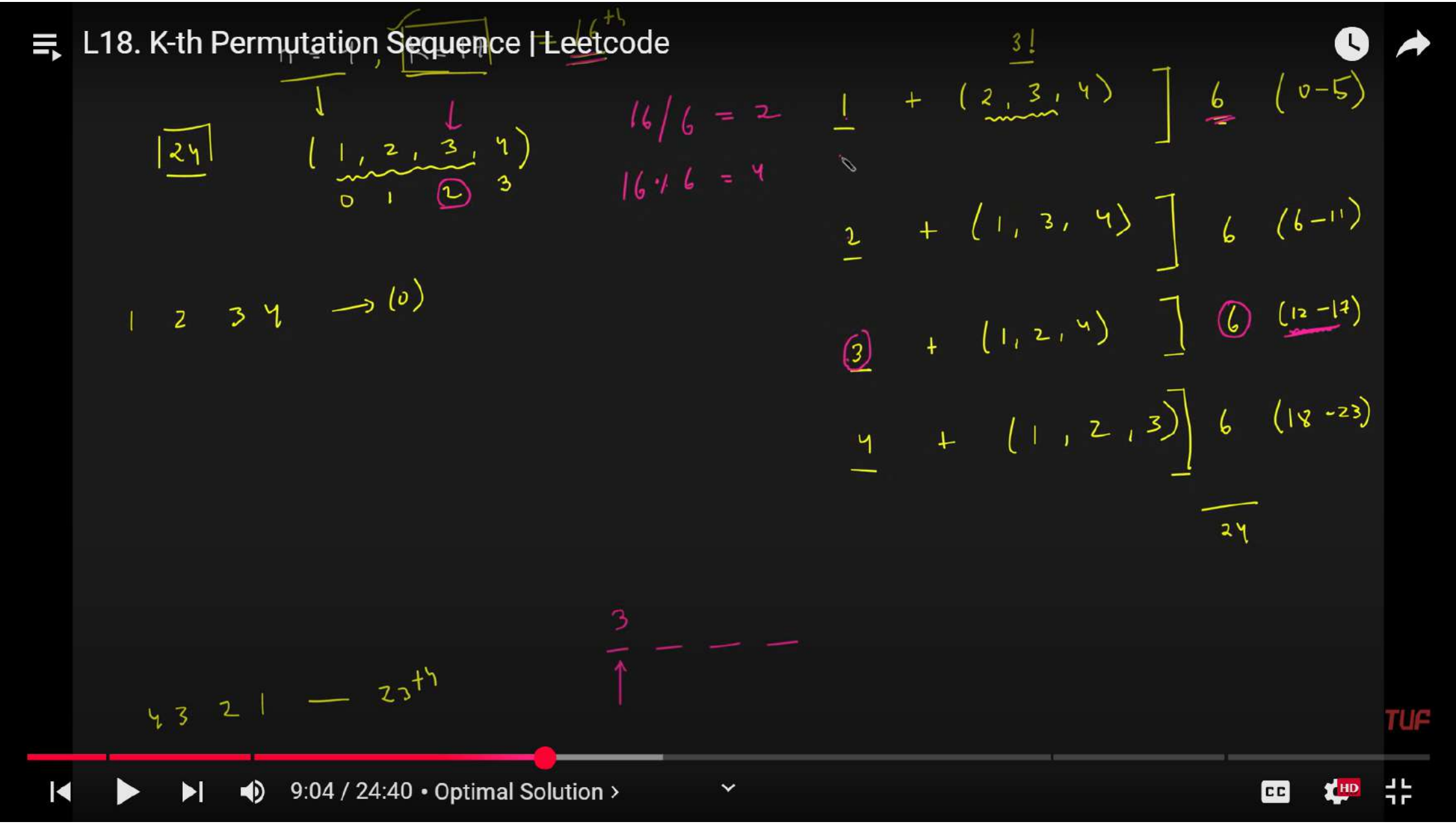
ds[K-1]

TUF

n = 4 , K = 17




TUF


L18. K-th Permutation Sequence | Leetcode
24
(1, 2, 3, 4)
0 1 2 3
16/6 = 2
16%6 = 4
1 2 3 4 → (0)
4 3 2 1 — 23th
3!
1 + (2, 3, 4) ] 6 (0-5)
2 + (1, 3, 4) ] 6 (6-11)
3 + (1, 2, 4) ] 6 (12-17)
4 + (1, 2, 3) ] 6 (18-23)
24
9:04 / 24:40 • Optimal Solution ›

n = 4, K = 17 = 16th

24

(1, 2, 3, 4)
0 1 2 3

16/6 = 2

3!

1 + (2, 3, 4) ] 6 (0-5)

2 + (1, 3, 4) ] 6 (6-11)

3 + (1, 2, 4) ] 6 (12-17)

4 + (1, 2, 3) ] 6 (18-23)

24

1 2 3 4 → (0)

4 3 2 1 — 23th

3 — — —
↑

TUF

n = 4 , K = 17 = 16th

24

(1, 2, 3, 4)
0 1 2 3

16/6 = 2

3!

1 + (2, 3, 4) ] 6 (0-5)

2 + (1, 3, 4) ] 6 (6-11)

3 + (1, 2, 4) ] 6 (12-17)

4 + (1, 2, 3) ] 6 (18-23)

24

1 2 3 4 → (0)

4 3 2 1 — 23th

3 — — —
↑
0

TUF

n = 4 , K = 17 = 16th

24

( 1 , 2 , 3 , 4 )
0 1 2 3

16 / 6 = 2
16 % 6 = (4)

{ 1 , 2 , 4 } , K = 4

4 3 2 1 — 23th

3 — — —
↑

3!

1 + ( 2 , 3 , 4 ) ] 6 (0-5)

2 + ( 1 , 3 , 4 ) ] 6 (6-11)

(3) + ( 1 , 2 , 4 ) ] (6) (12-17)

4 + ( 1 , 2 , 3 ) ] 6 (18-23)

24

TUF

n = 4 , K = 17 = 16th

24

( 1 , 2 , 3 , 4 )
0 1 2 3

{ 1 , 2 , 4 } , K = 4/2 = 2

3! = 6

① 1 { 2 4 } ] 2 (0-1)

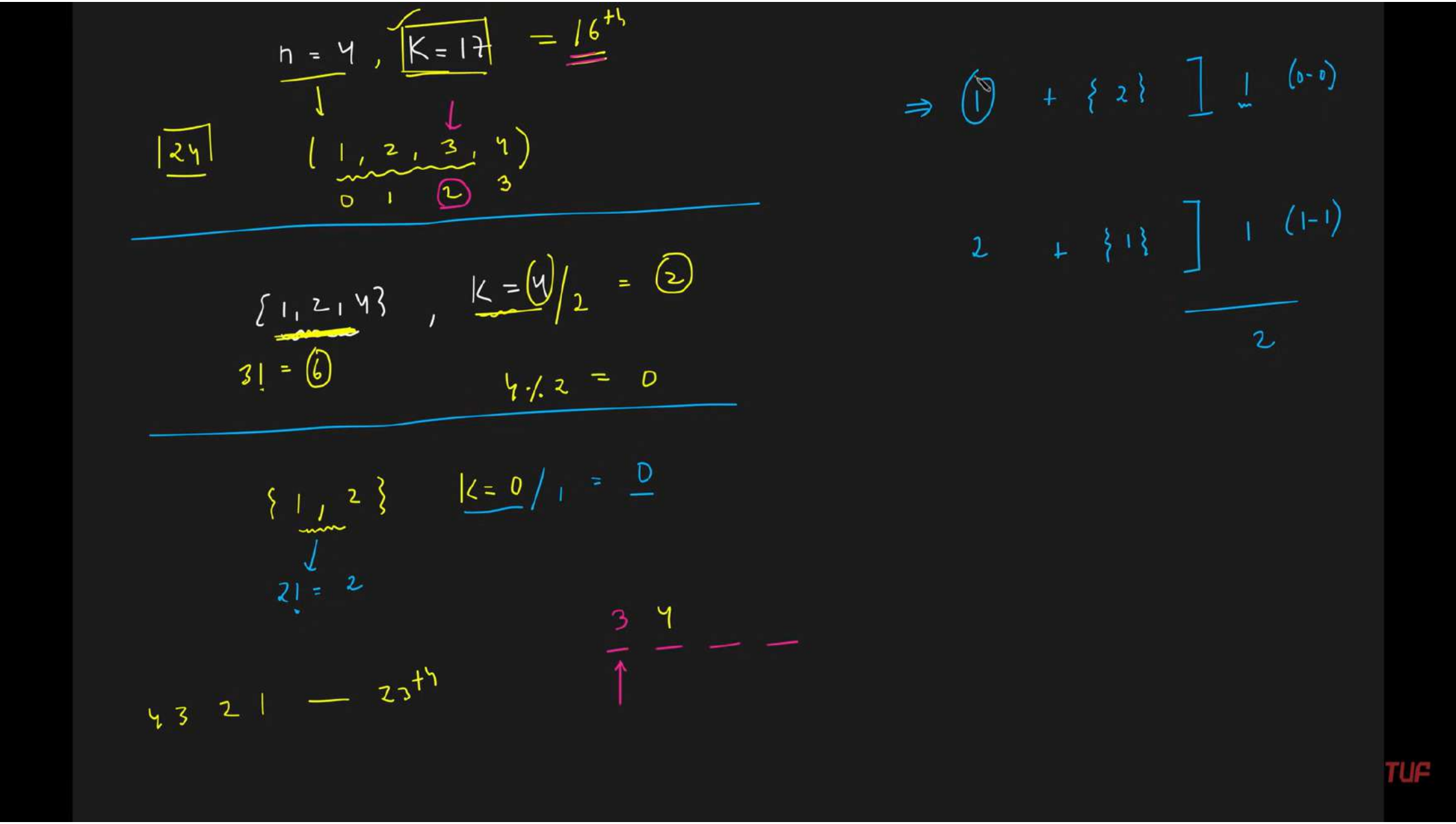
② 2 { 1 4 } ] 2 (2-3)

⇒ (2) 4 { 1 2 } ] 2 (4-5)

6

4 3 2 1 — 23th

3 4 — — —
↑


n = 4 , K = 17 = 16th
24
( 1 , 2 , 3 , 4 )
0 1 2 3
{ 1 , 2 , 4 } , K = 4/2 = 2
3! = 6
4 % 2 = 0
{ 1 , 2 }
K = 0/1 = 0
2! = 2
4 3 2 1 — 23rd
3 4 _ _ _
⇒ 1 + { 2 } ] ! (0-0)
2 + { 1 } ] 1 (1-1)
2

n = 4 , K = 17 = 16th

24

(1, 2, 3, 4)
0 1 2 3

{1, 2, 4} , K = (4)/2 = (2)

3! = (6)

4 % 2 = 0

{1, 2} K = 0/1 = 0

K = 0 % 0 = 0

2! = 2

4 3 2 1 — 23rd

3 4 1 _ _ _

⇒ (1) + { 2 } ] ! (0-0)

2 + { 1 } ] 1 (1-1)

2

n = 4 , K = 17 = 16+1

24

( 1 , 2 , 3 , 4 )
0 1 2 3

{ 1 , 2 , 4 } , K = 4/2 = 2

3! = 6

4 % 2 = 0

{ 1 , 2 }  K = 0/1 = 0

K = 0 % 0 = 0

2! = 2

{ 2 }  K = 0

( 3 4 1 2 )

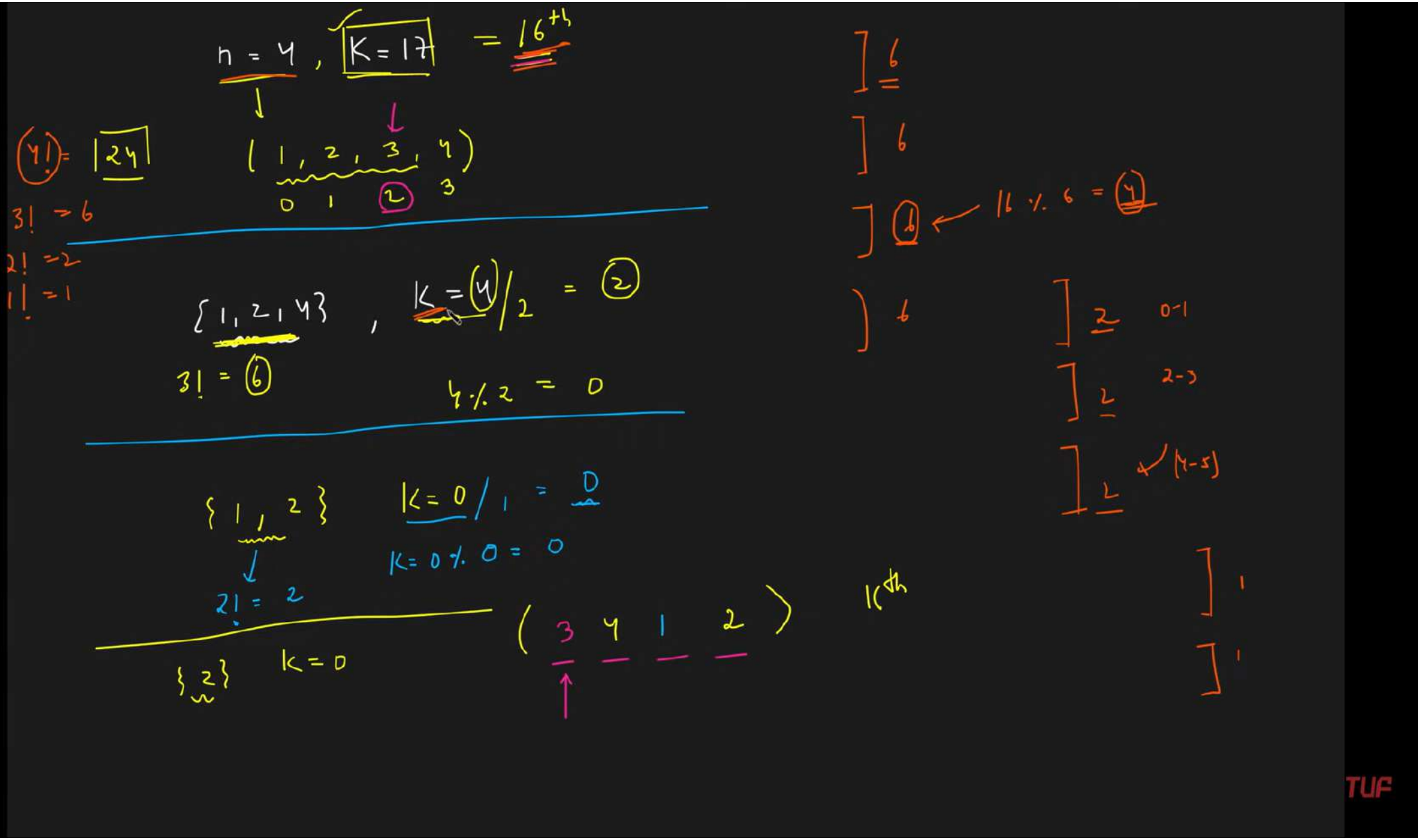
⇒ 1 + { 2 } ] ! (0-0)

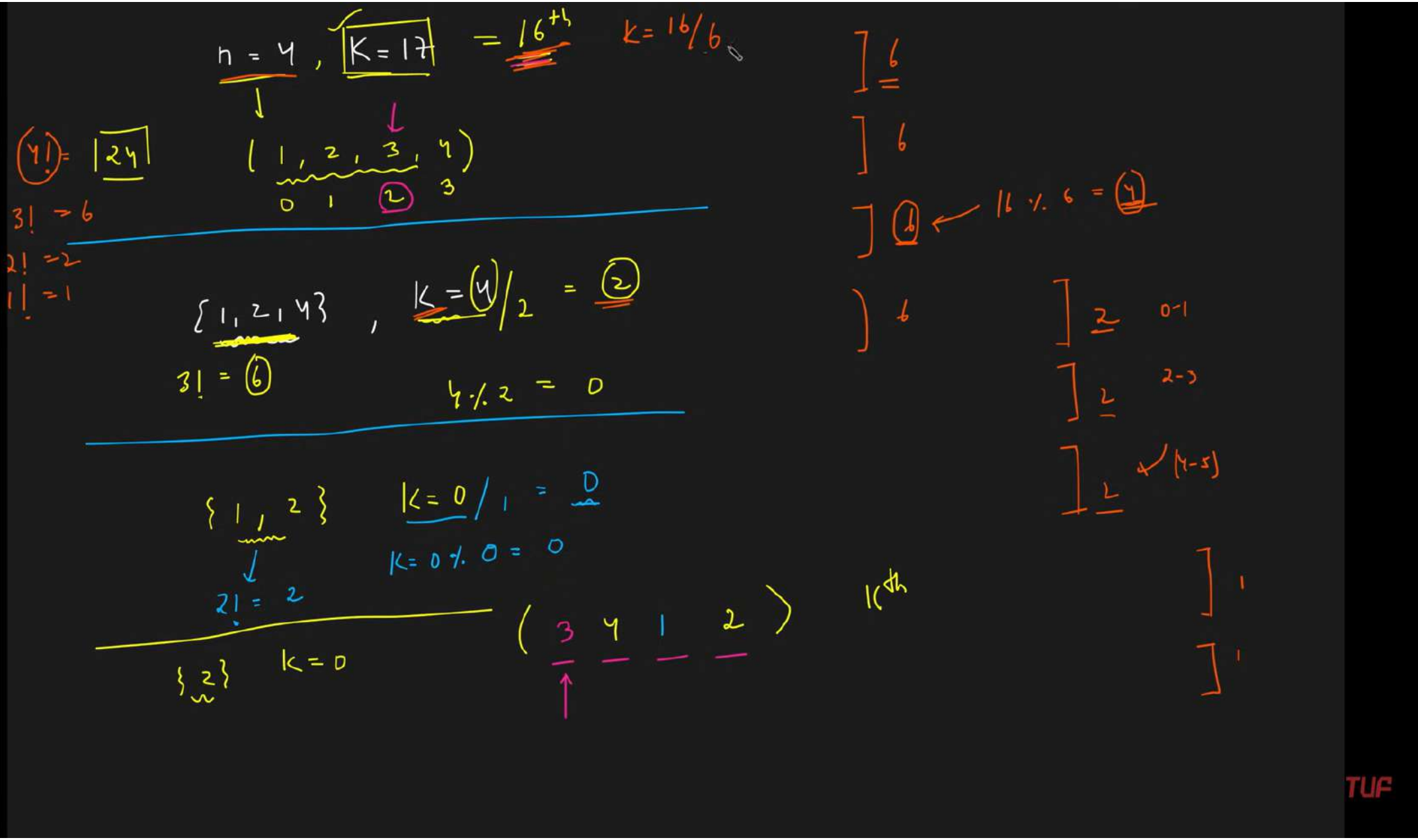
2 + { 1 } ] 1 (1-1)

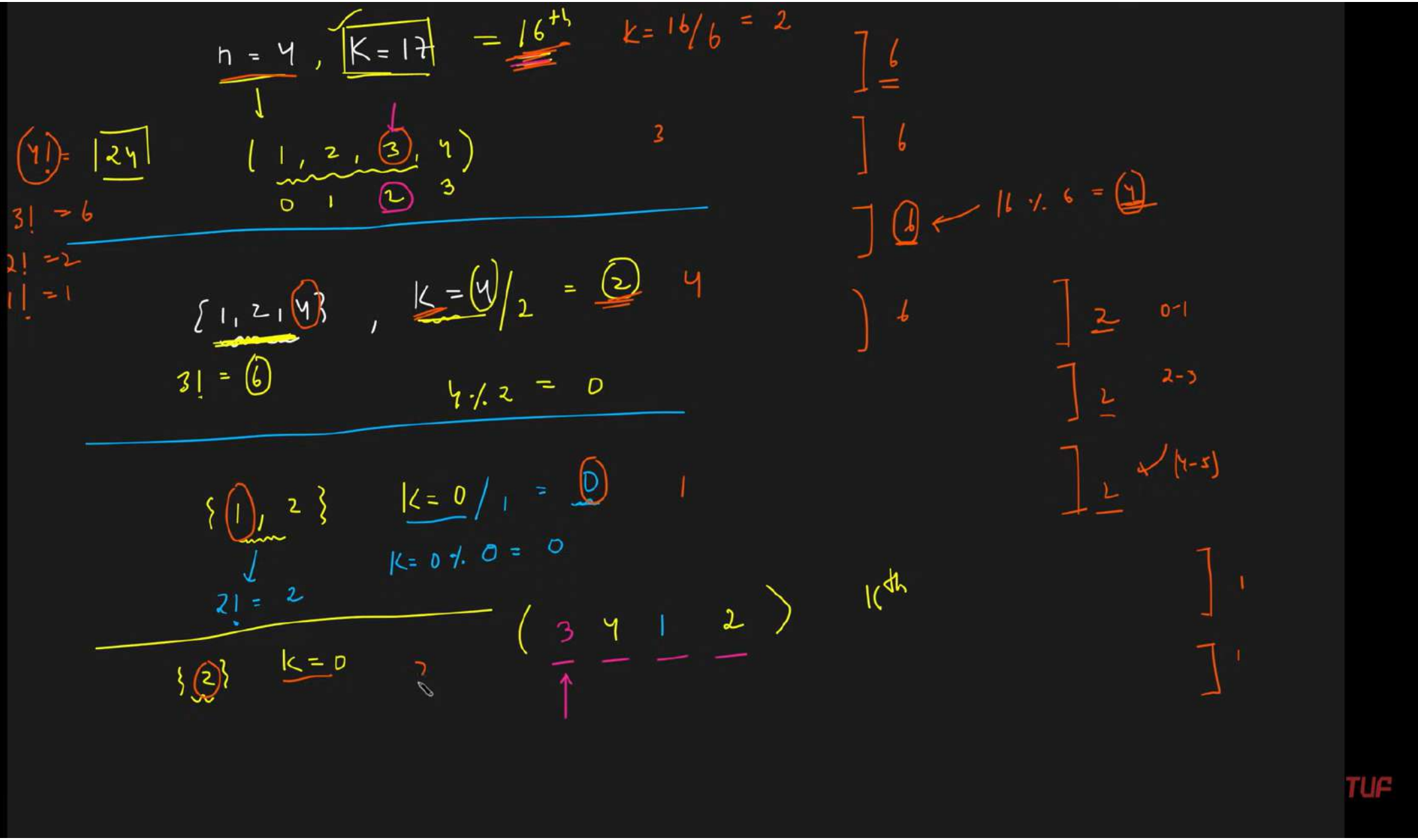
2

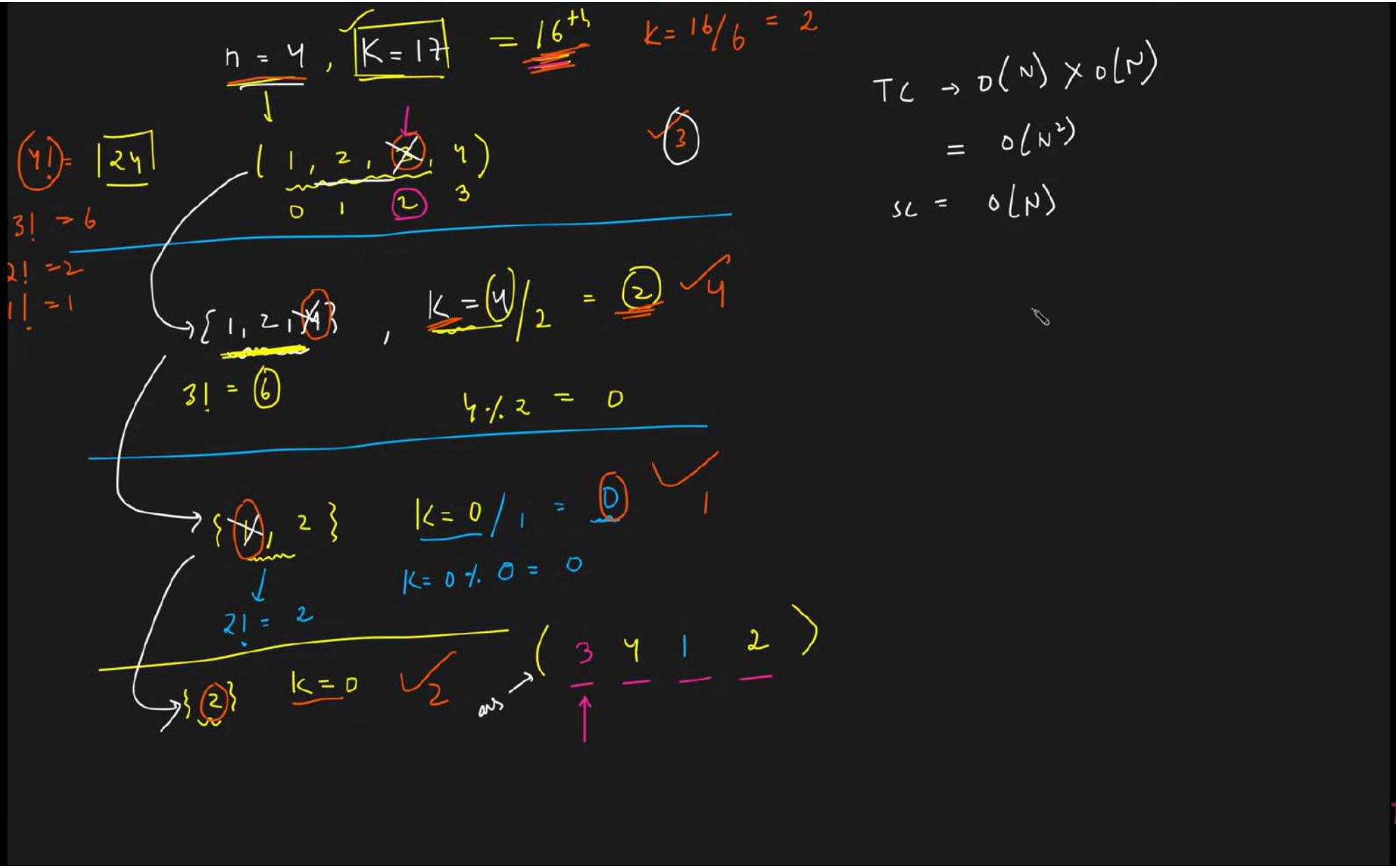
2 + { __ }

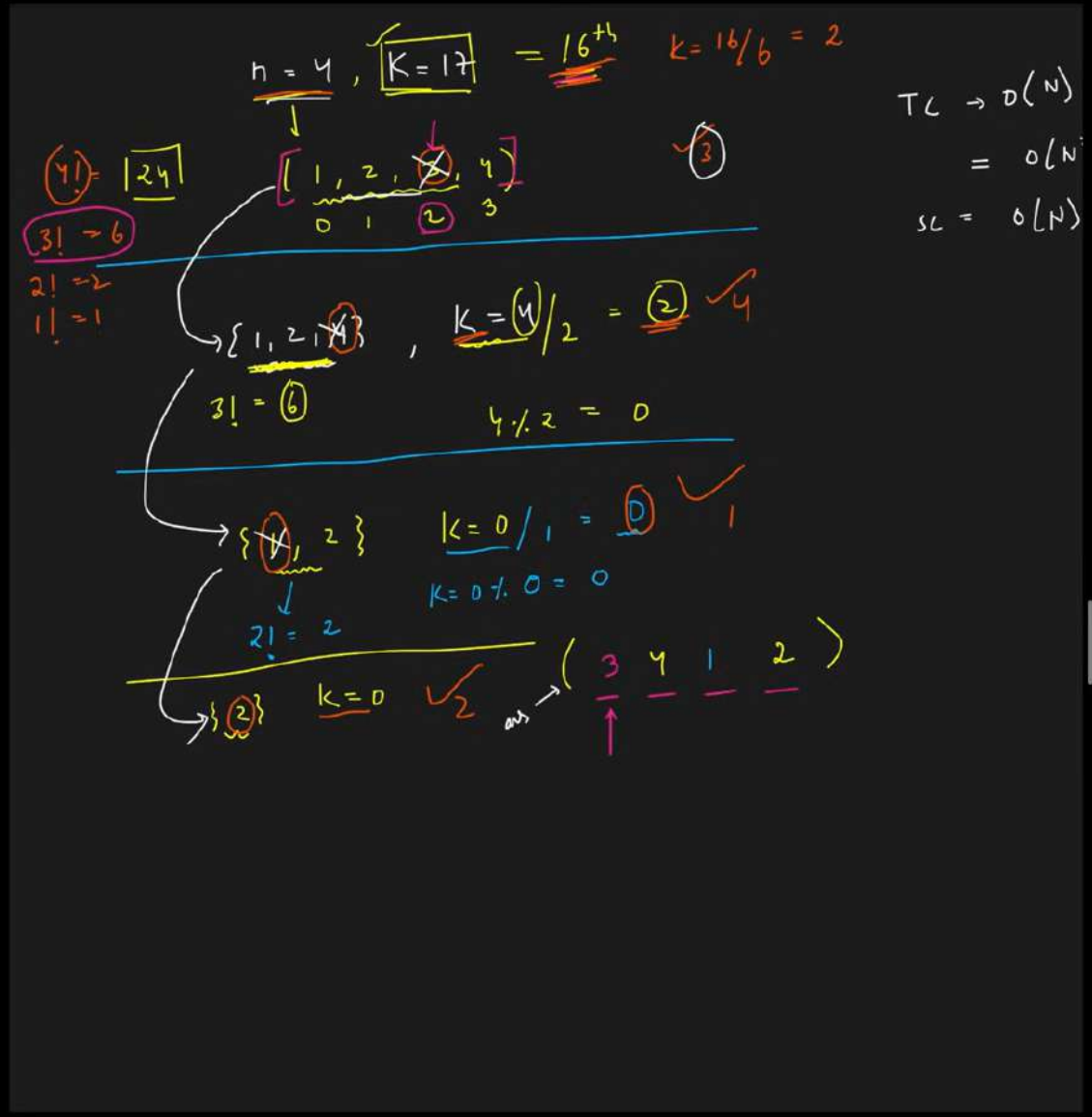
TUF


n = 4 , K = 17 = 16th
(4!) = 24
(1, 2, 3, 4)
0 1 2 3
3! = 6
2! = 2
1! = 1
{1, 2, 4} , K = (4)/2 = (2)
3! = (6)
4 % 2 = 0
{1, 2} K = 0/1 = 0
K = 0 % 0 = 0
2! = 2
{2} K = 0
(3 4 1 2)
16th
] 6
] 6
] (1) ← 16 % 6 = (4)
) 6
] 2 0-1
] 2 2-3
] 2 (4-5)
] 1
] 1


n = 4 , K = 17 = 16th
k = 16/6
(4!) = 24
(1, 2, 3, 4)
0 1 2 3
3! = 6
2! = 2
1! = 1
{1, 2, 4} , k = 4/2 = 2
3! = 6
4 % 2 = 0
{1, 2}
k = 0/1 = 0
k = 0 % 0 = 0
2! = 2
{2}
k = 0
(3 4 1 2)
16th
16 % 6 = 4
0-1
2-3
(4-5)


n = 4 , K = 17 = 16th
K = 16/6 = 2
(4!) = 24
3! = 6
2! = 2
1! = 1
( 1 , 2 , 3 , 4 )
0 1 2 3
3
{ 1 , 2 , 4 }
K = 4/2 = 2
3! = 6
4 % 2 = 0
4
{ 1 , 2 }
K = 0/1 = 0
1
2! = 2
K = 0 % 0 = 0
{ 2 }
K = 0
( 3 4 1 2 )
16th
] 6
] 6
] 0 ← 16 % 6 = 4
) 6
] 2 0-1
] 2 2-3
] 2 4-5
] 1
] 1


n = 4 , K = 17 = 16th
k = 16/6 = 2
(4!) = 24
3! = 6
2! = 2
1! = 1
( 1 , 2 , 3 , 4 )
0 1 2 3
3
{ 1 , 2 , 4 } , K = 4/2 = 2
3! = 6
4 % 2 = 0
{ 1 , 2 }
K = 0/1 = 0
K = 0 % 0 = 0
2! = 2
{ 2 }
K = 0
ans → ( 3 4 1 2 )
TC → O(N) × O(N)
= O(N²)
SC = O(N)

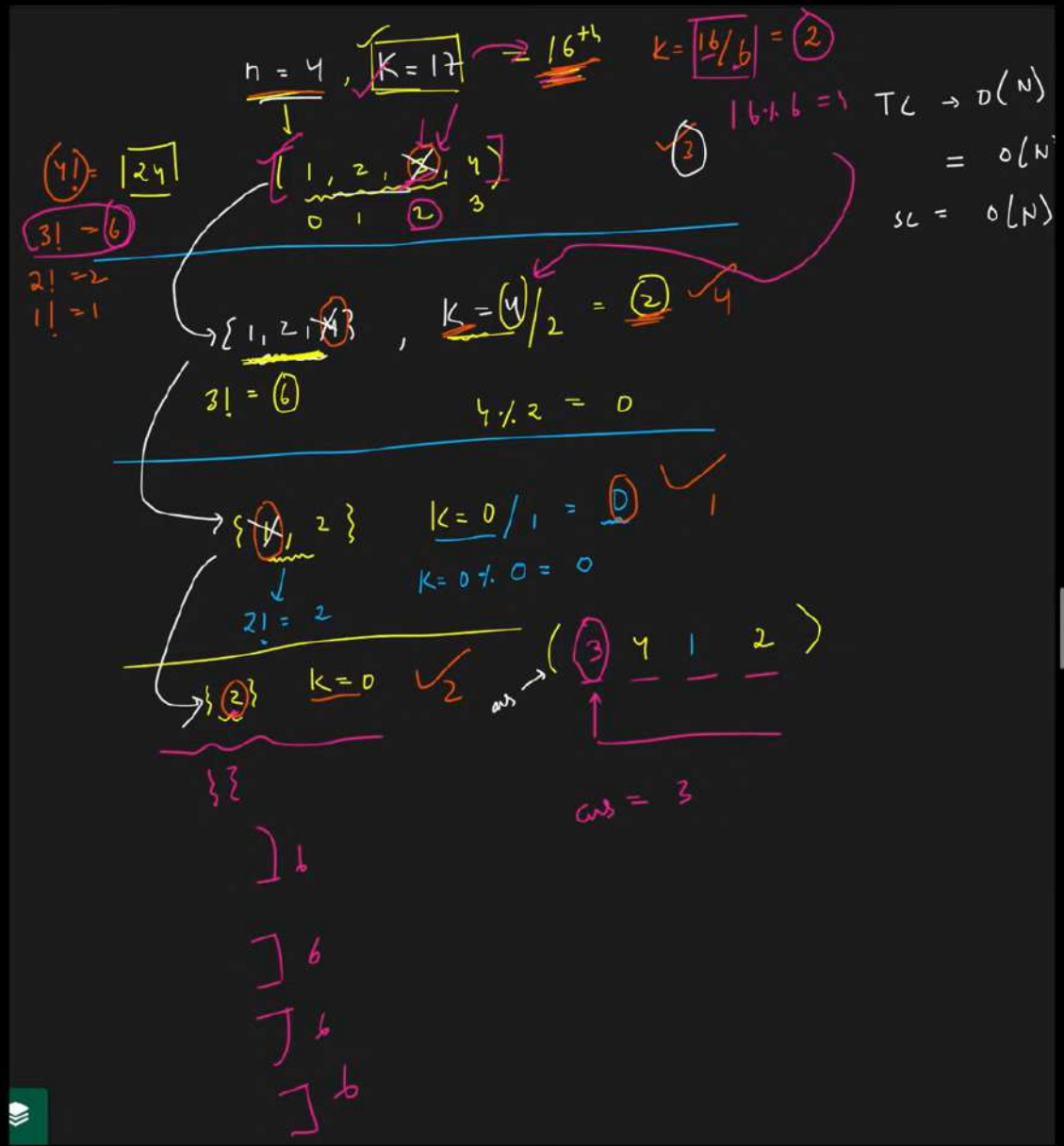


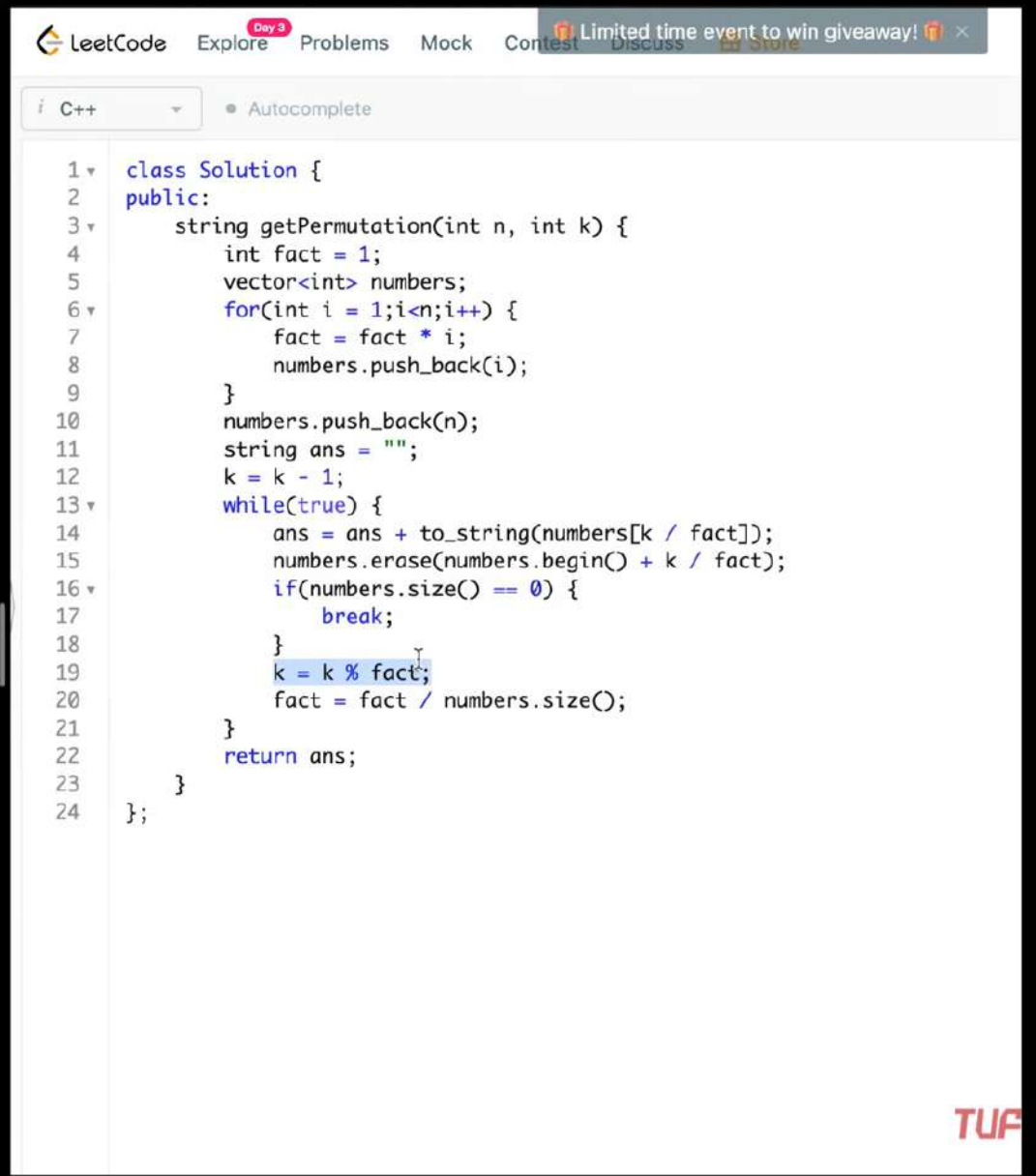

LeetCode Explore Day 3 Problems Mock Contest Discuss Store
Limited time event to win giveaway!

C++ Autocomplete

```cpp
class Solution {
public:
    string getPermutation(int n, int k) {
        int fact = 1;
        vector<int> numbers;
        for(int i = 1;i<n;i++) {
            fact = fact * i;
            numbers.push_back(i);
        }
        numbers.push_back(n);
        string ans = "";
        k = k - 1;
        while(true) {
            ans = ans + to_string(numbers[k / fact]);
            numbers.erase(numbers.begin() + k / fact);
            if(numbers.size() == 0) {
                break;
            }
            k = k % fact;
            fact = fact / numbers.size();
        }
        return ans;
    }
};
```

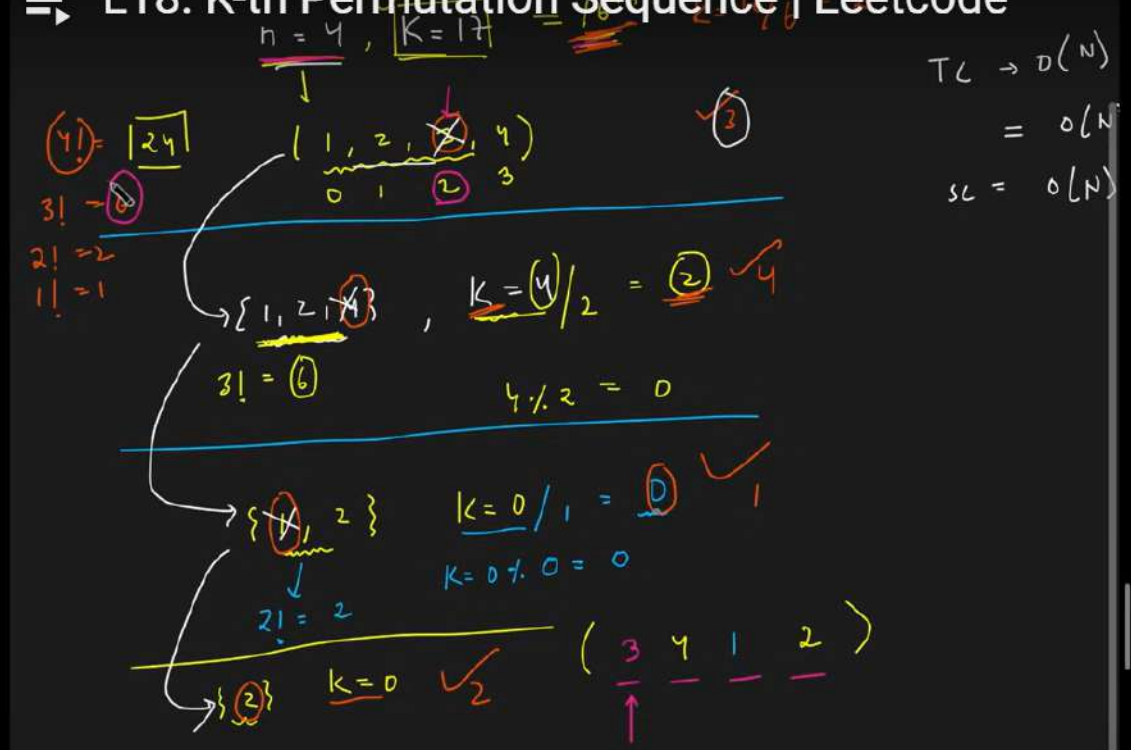
TUF




```cpp
class Solution {
public:
    string getPermutation(int n, int k) {
        int fact = 1;
        vector<int> numbers;
        for(int i = 1;i<n;i++) {
            fact = fact * i;
            numbers.push_back(i);
        }
        numbers.push_back(n);
        string ans = "";
        k = k - 1;
        while(true) {
            ans = ans + to_string(numbers[k / fact]);
            numbers.erase(numbers.begin() + k / fact);
            if(numbers.size() == 0) {
                break;
            }
            k = k % fact;
            fact = fact / numbers.size();
        }
        return ans;
    }
};
```

# L18. K-th Permutation Sequence | Leetcode




LeetCode Explore Problems Mock Contest Discuss
Limited time event to win giveaways

Java · Autocomplete

```java
class Solution {
    public String getPermutation(int n, int k) {
        int fact = 1;
        List<Integer> numbers = new ArrayList<>();
        for(int i = 1;i<n;i++) {
            fact = fact * i;
            numbers.add(i);
        }
        numbers.add(n);
        String ans = "";
        k = k - 1;
        while(true) {
            ans = ans + numbers.get(k / fact);
            numbers.remove(k / fact);
            if(numbers.size() == 0) {
                break;
            }

            k = k % fact;
            fact = fact / numbers.size();
        }
        return ans;
    }
}
```

TUF

21:32 / 24:40 • Java Solution ›