# My trick

* 1. learned use of dictionary – key count
     + duplicate character in String
     + Unique character in string
  2. Learned % and / operator use
     + Integer reverse
     + Add integer string
  3. Search - Binary Search on Array – Sorted Array
  4. Sort – Merge Sort >> A stable sorting algorithm
     + Overview of Merge sort
       - Divide and concur approach
       - Recursive
       - O(n) Space complexity
       - O(nlongn) time complexity
     + Implementation
       - Return if nArra <2
       - Find mid of nArra
       - Make left and right array and divide the main array into it
       - Recursive for left and right array **-2 recursive**
       - **Merge call - 3** 
         1. Three while loop need – I , j k =0;

First while will have if else two put right or left array in main array

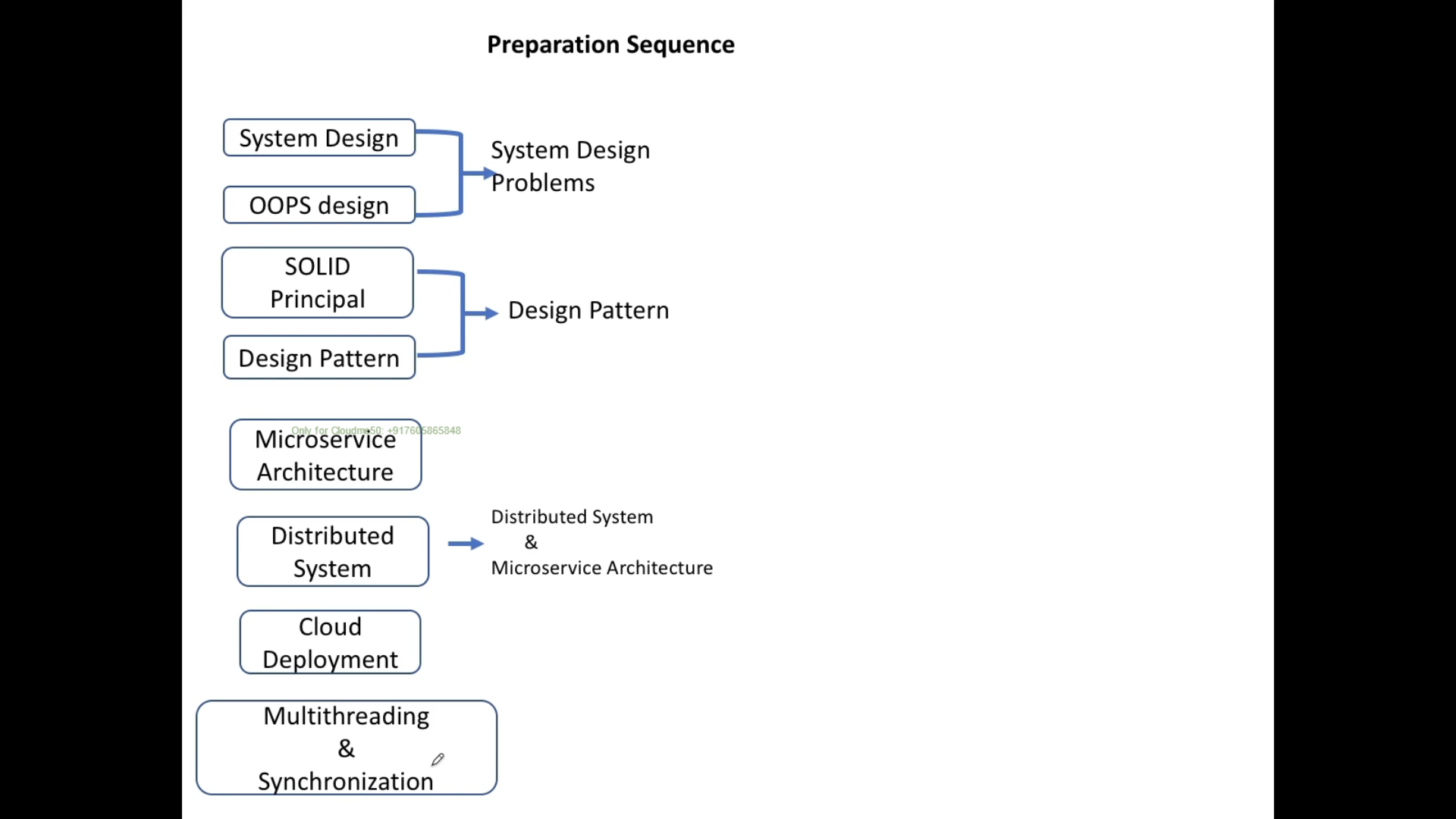
Second while loop to put extra left array in main loop

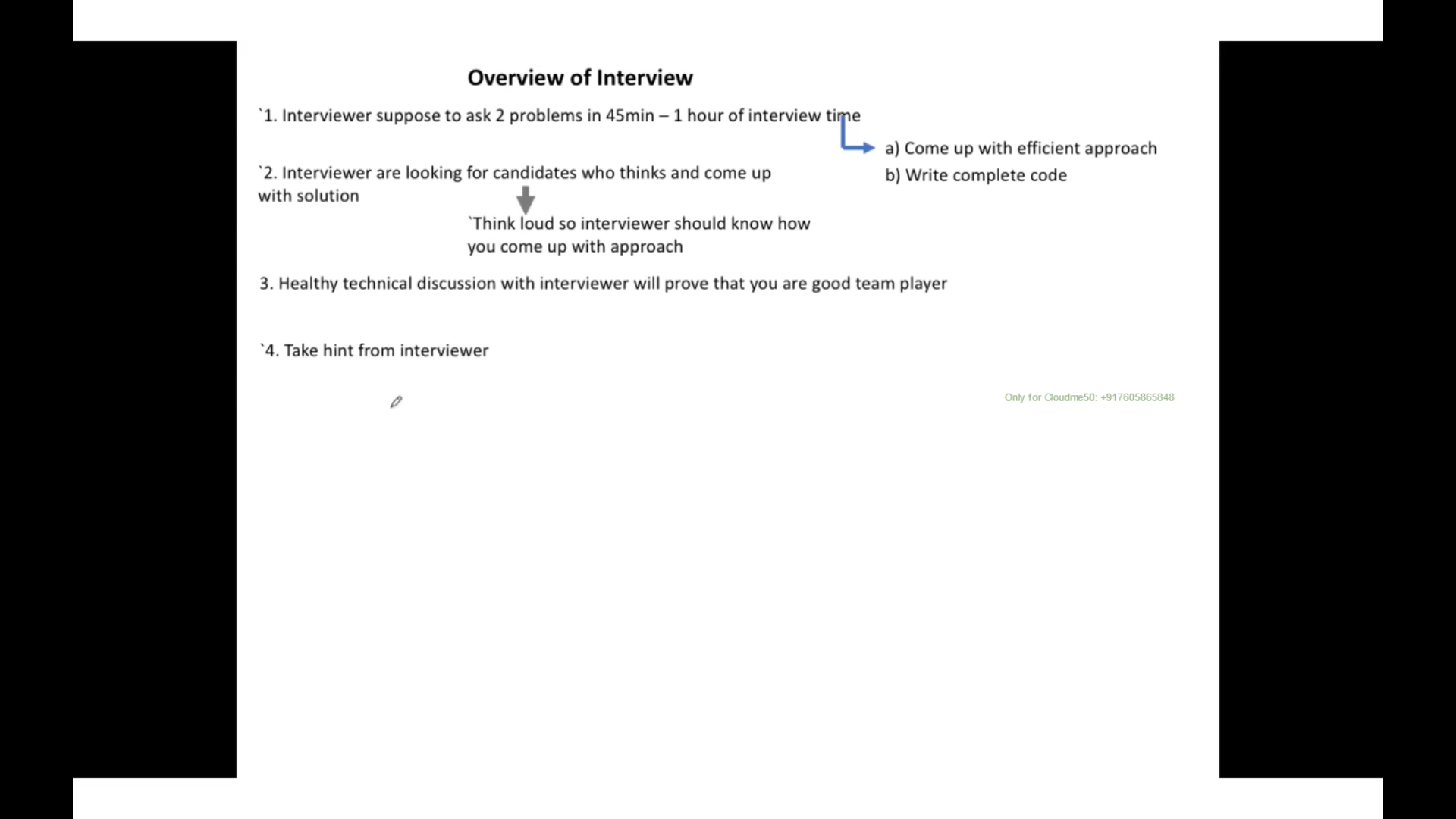
Third while loop to put extra right array in main loop

* 1. Recursion
     + Help in Binary Tree , DP, Graph , backtracking and many topics root is Recursion
     + Permutation of string
     + Knight walk problem in backtracking

# Learning Sequence



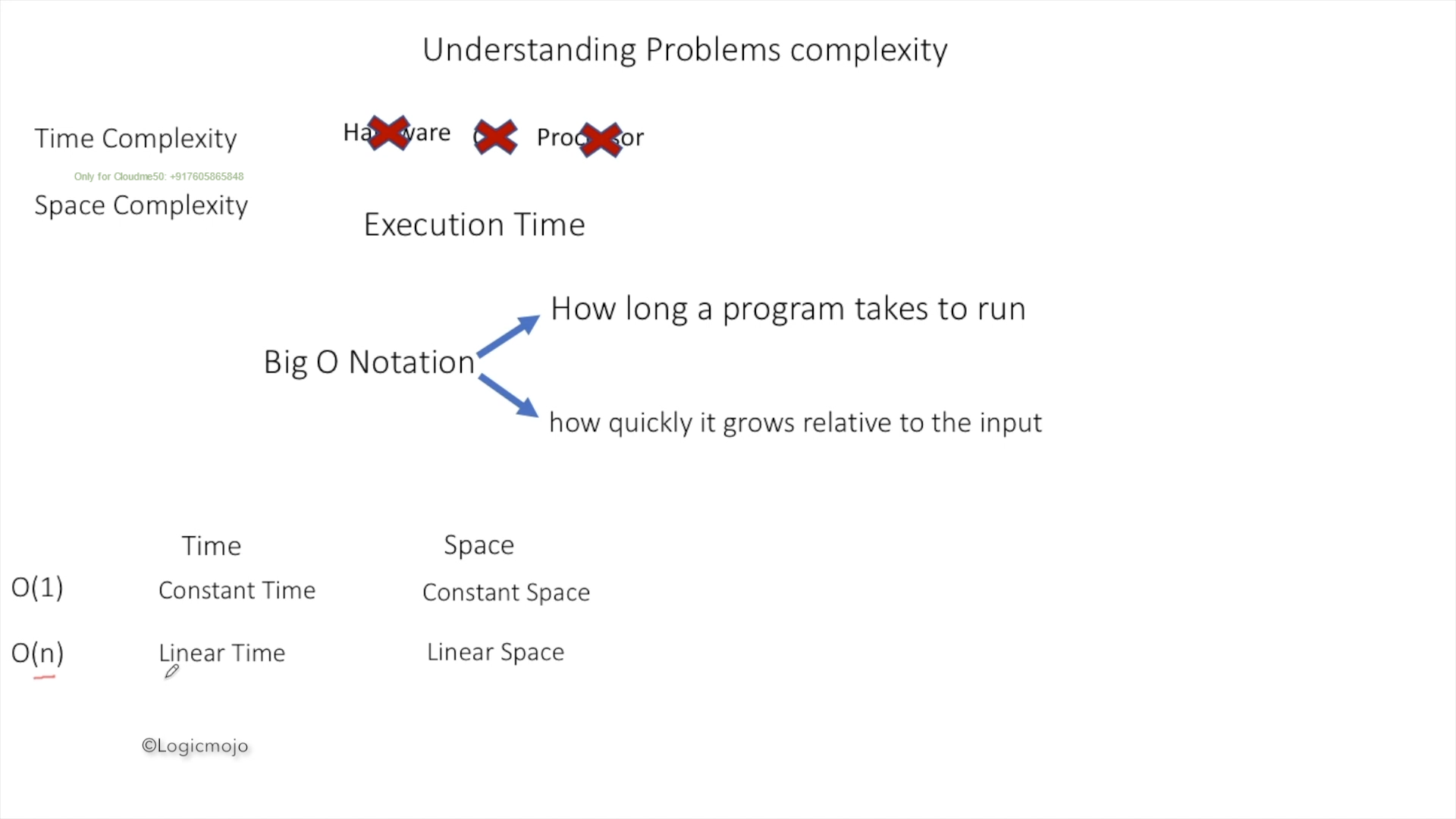


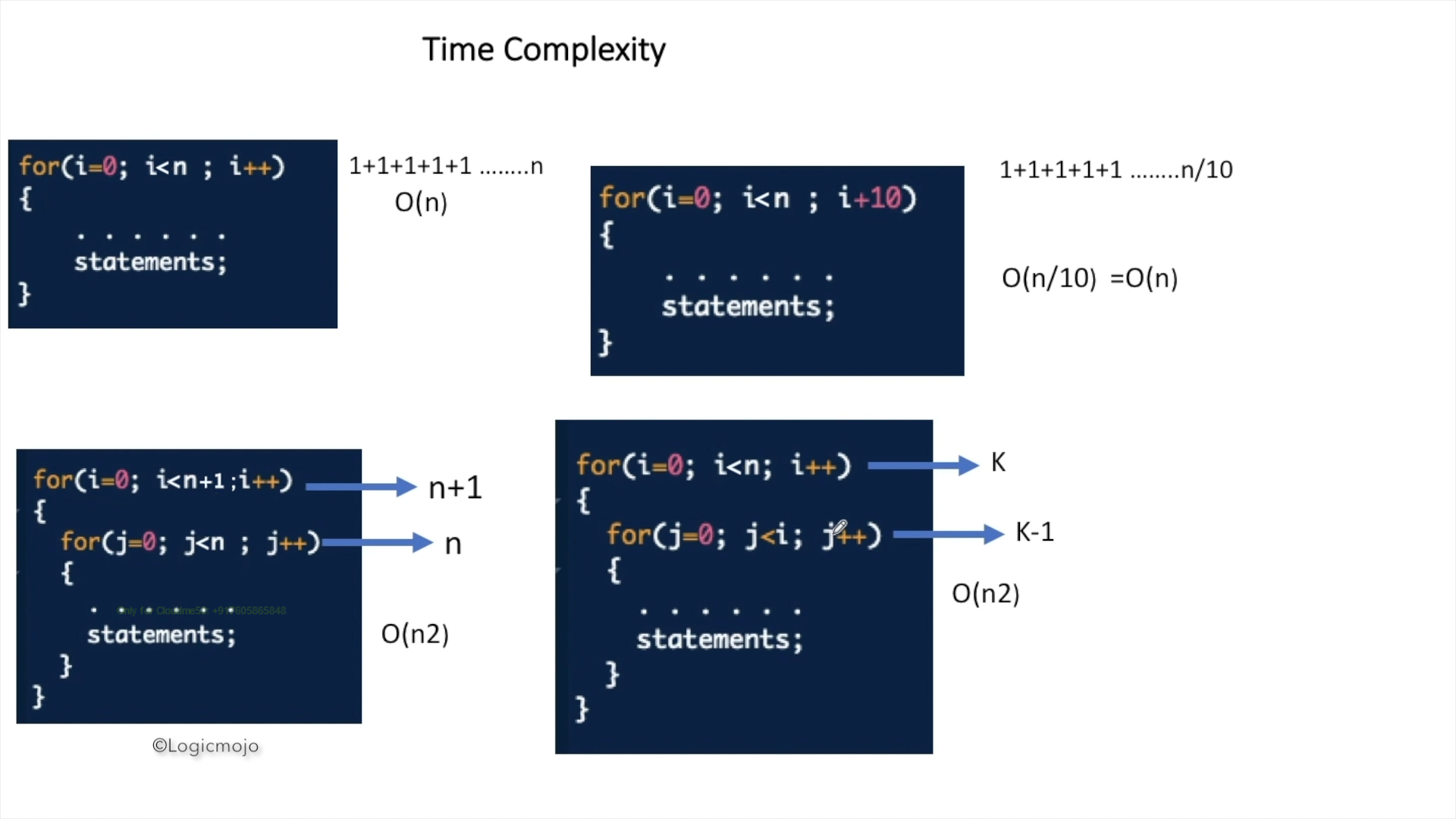


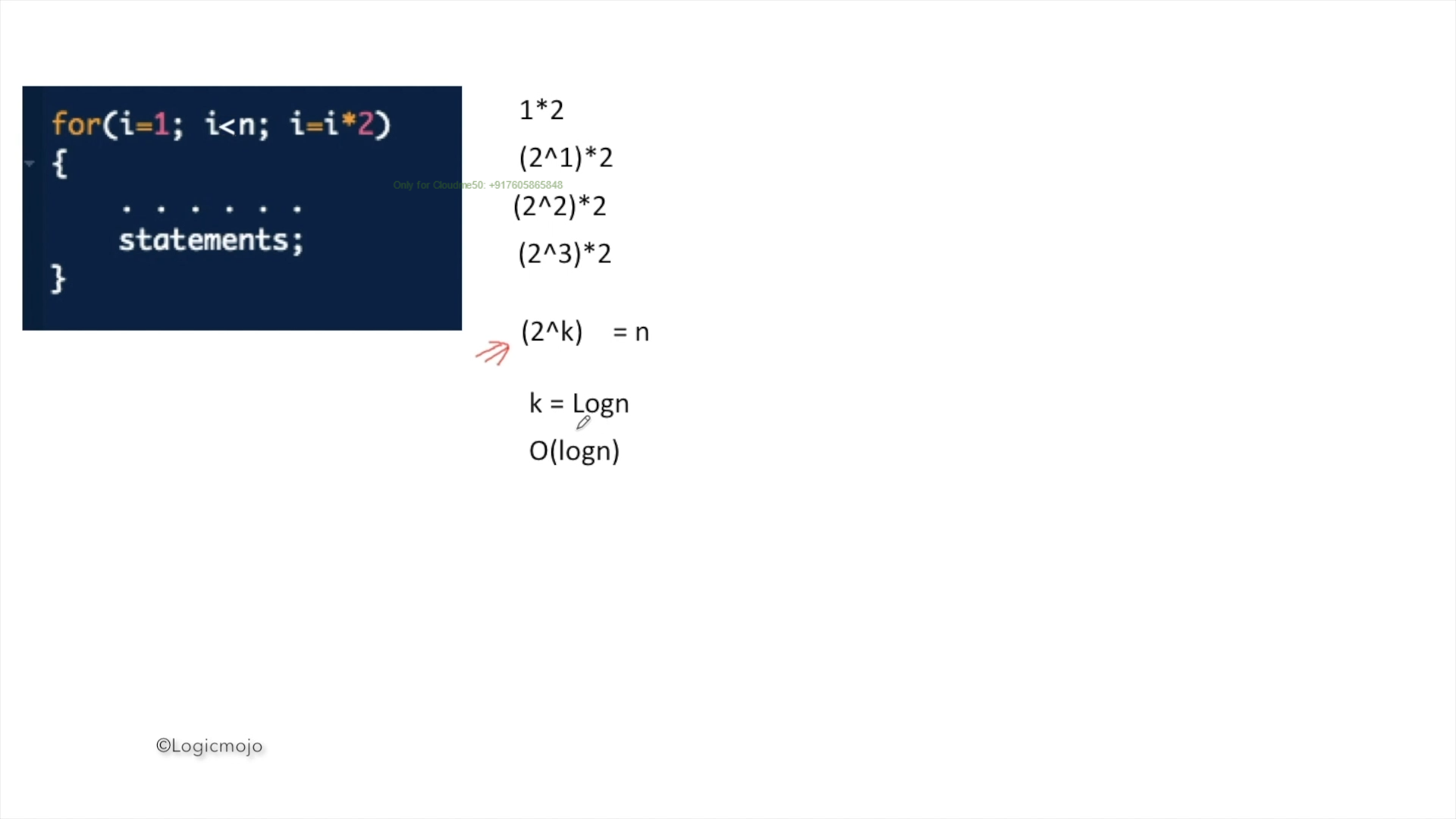
# **Space and Time Complexity of Algorithms by Using Big O Notation**

## **Time Complexity**

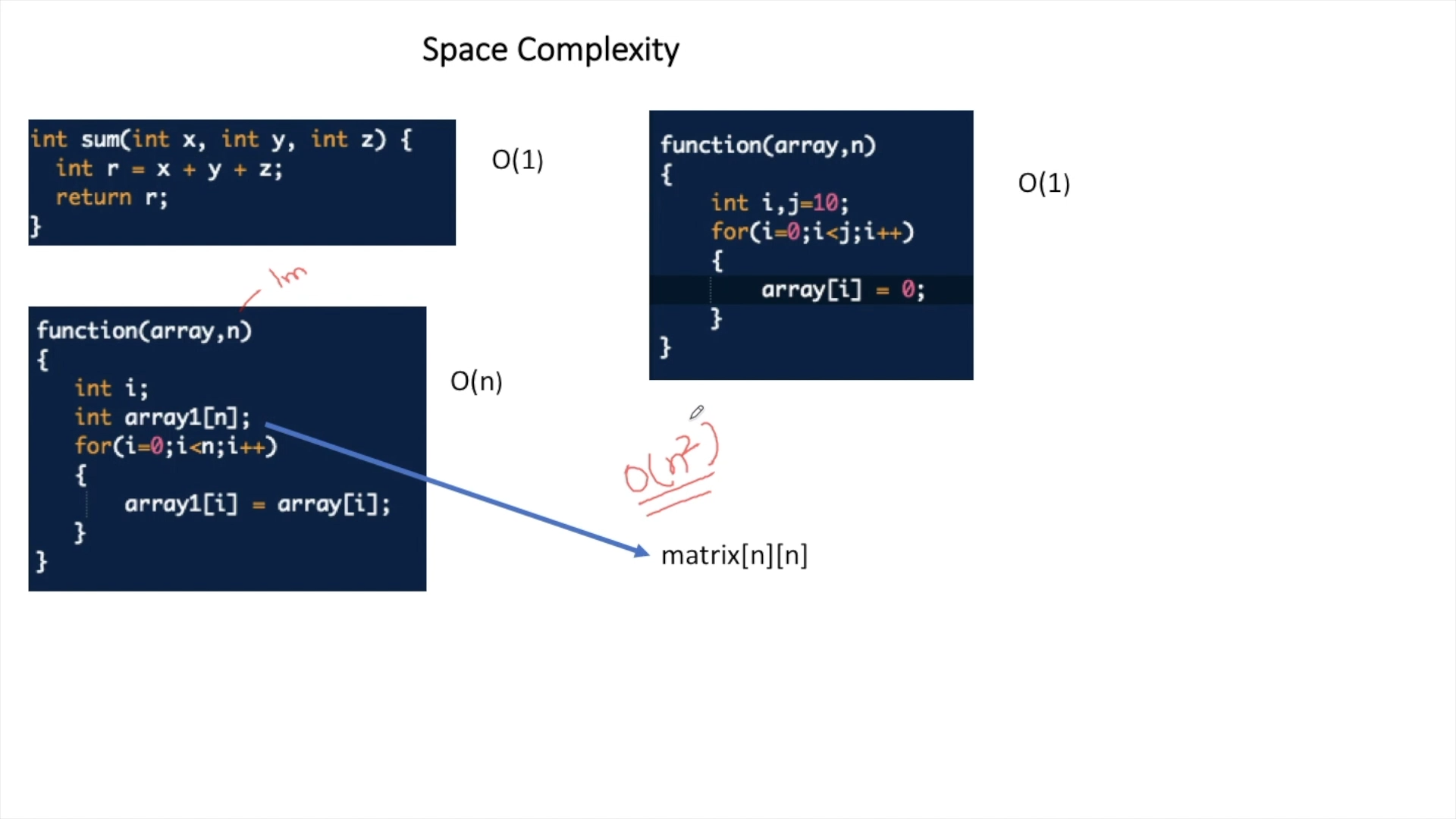
* + - One for loop – O(n)
    - Two nested for loop – O(n2)
    - Three nested for loop – O(n3)





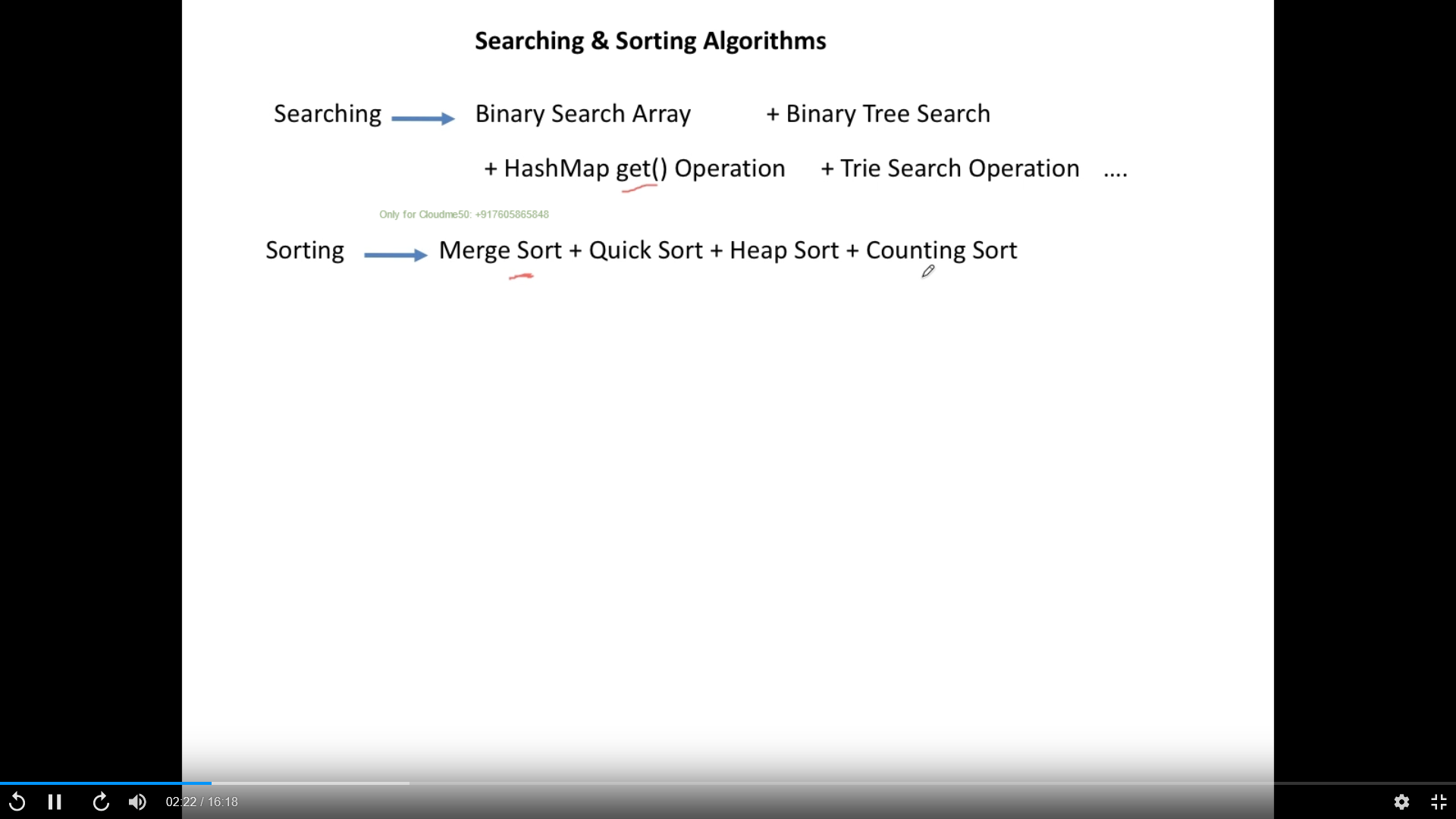


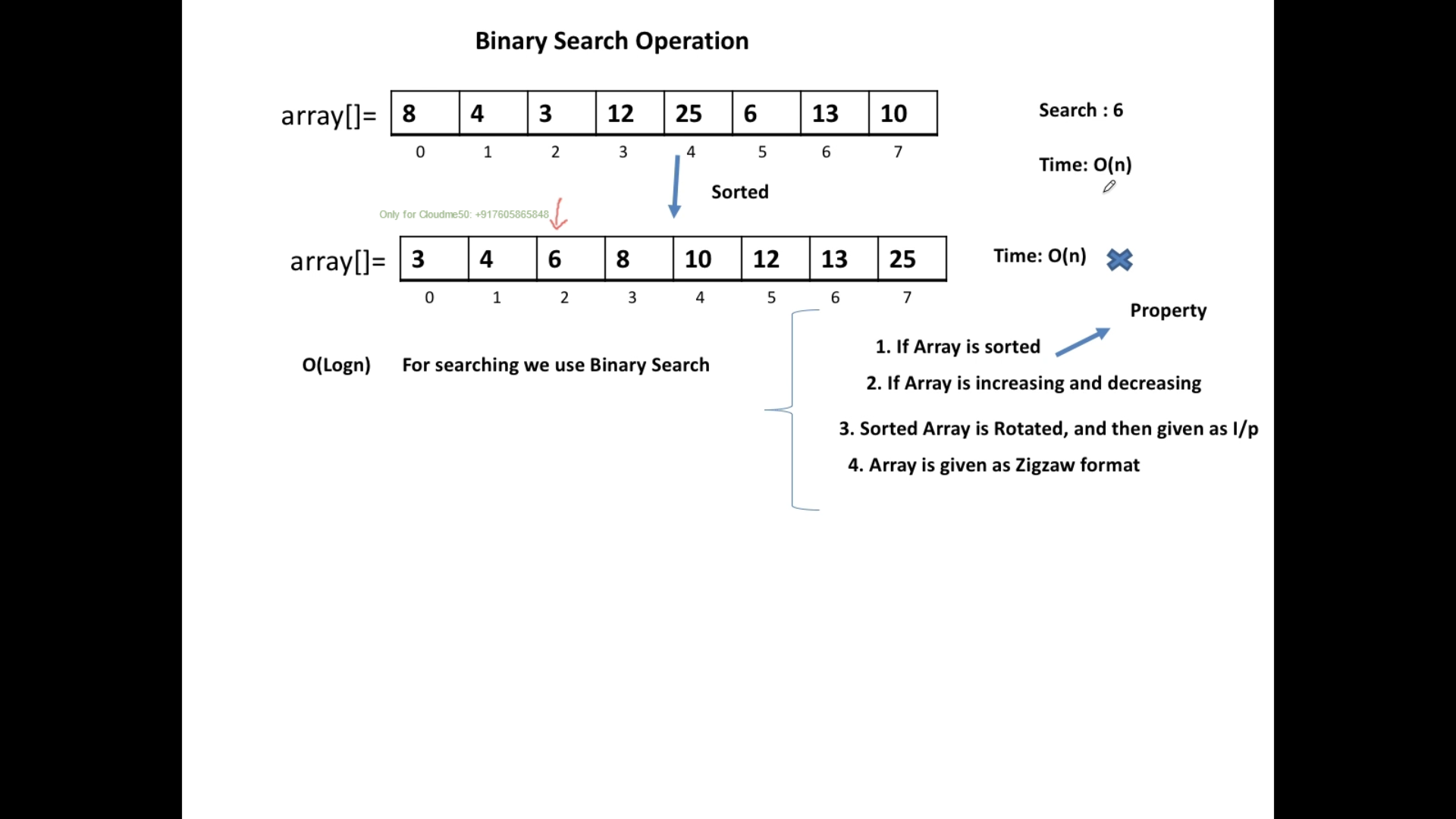
## **2.2 Space Complexity**

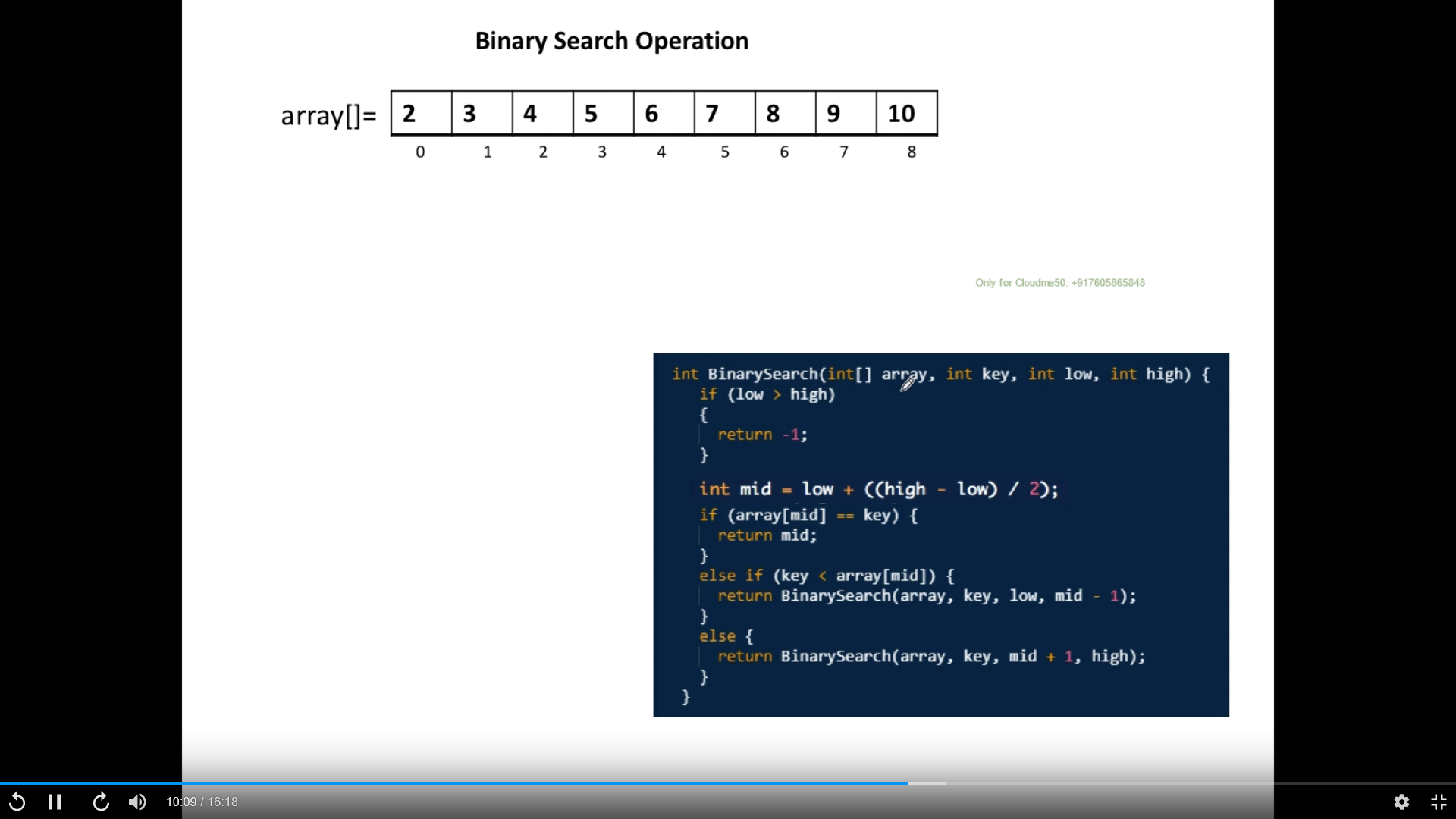


# Search and Sorting

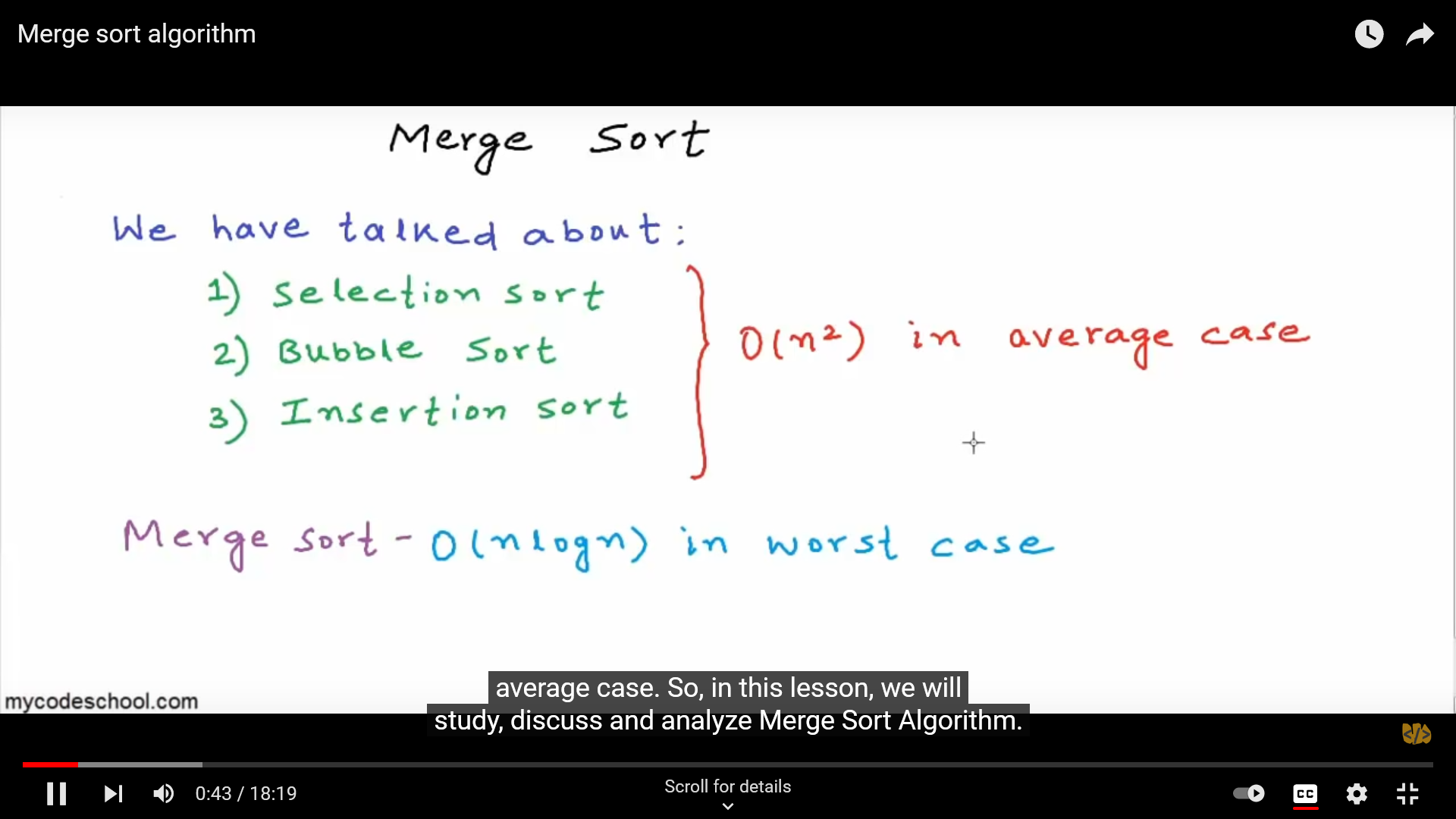
* Mostly Merge sort used 97 % following by Heap sort and Counting sort
* Binary search has O(logN) time

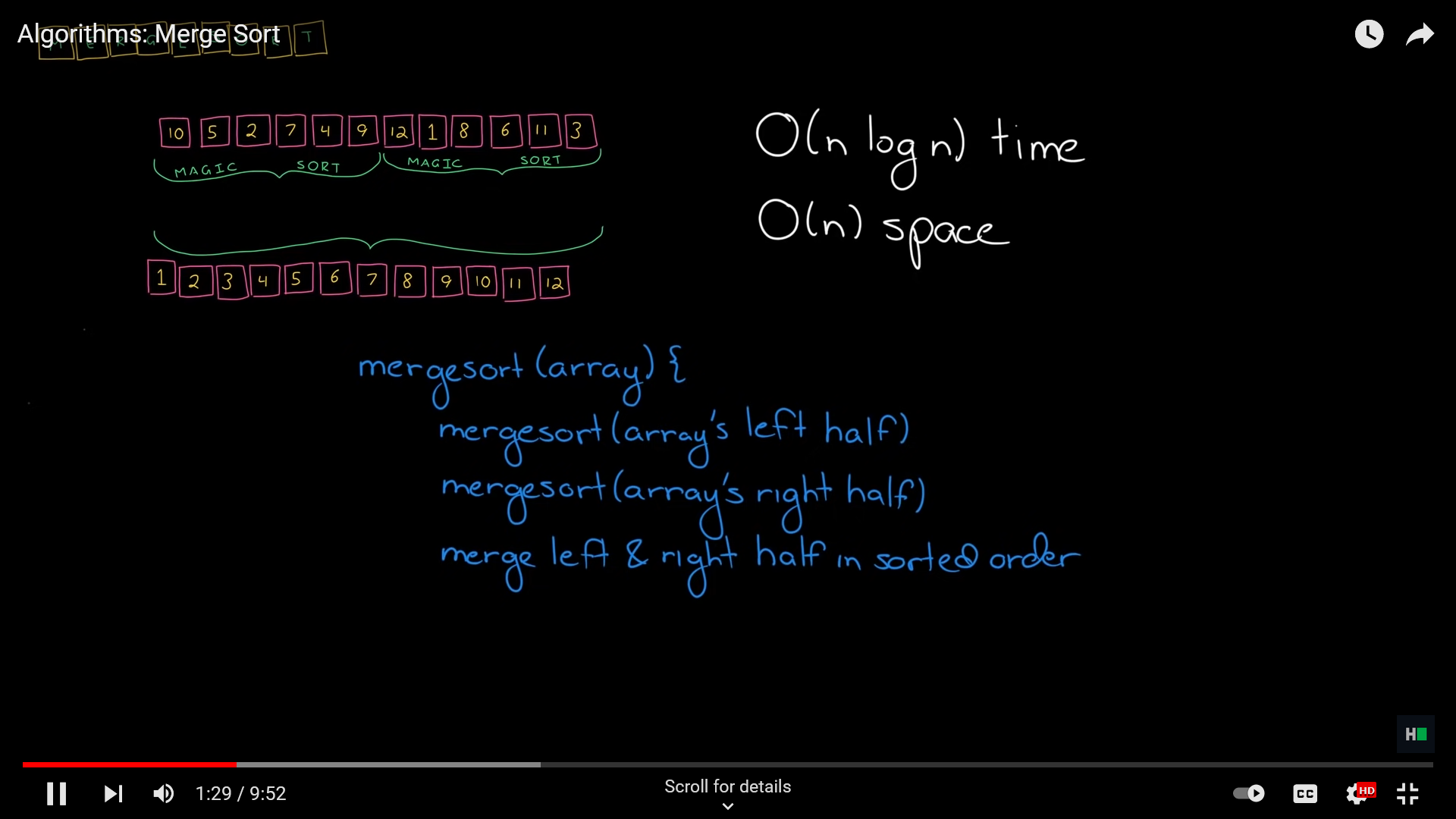


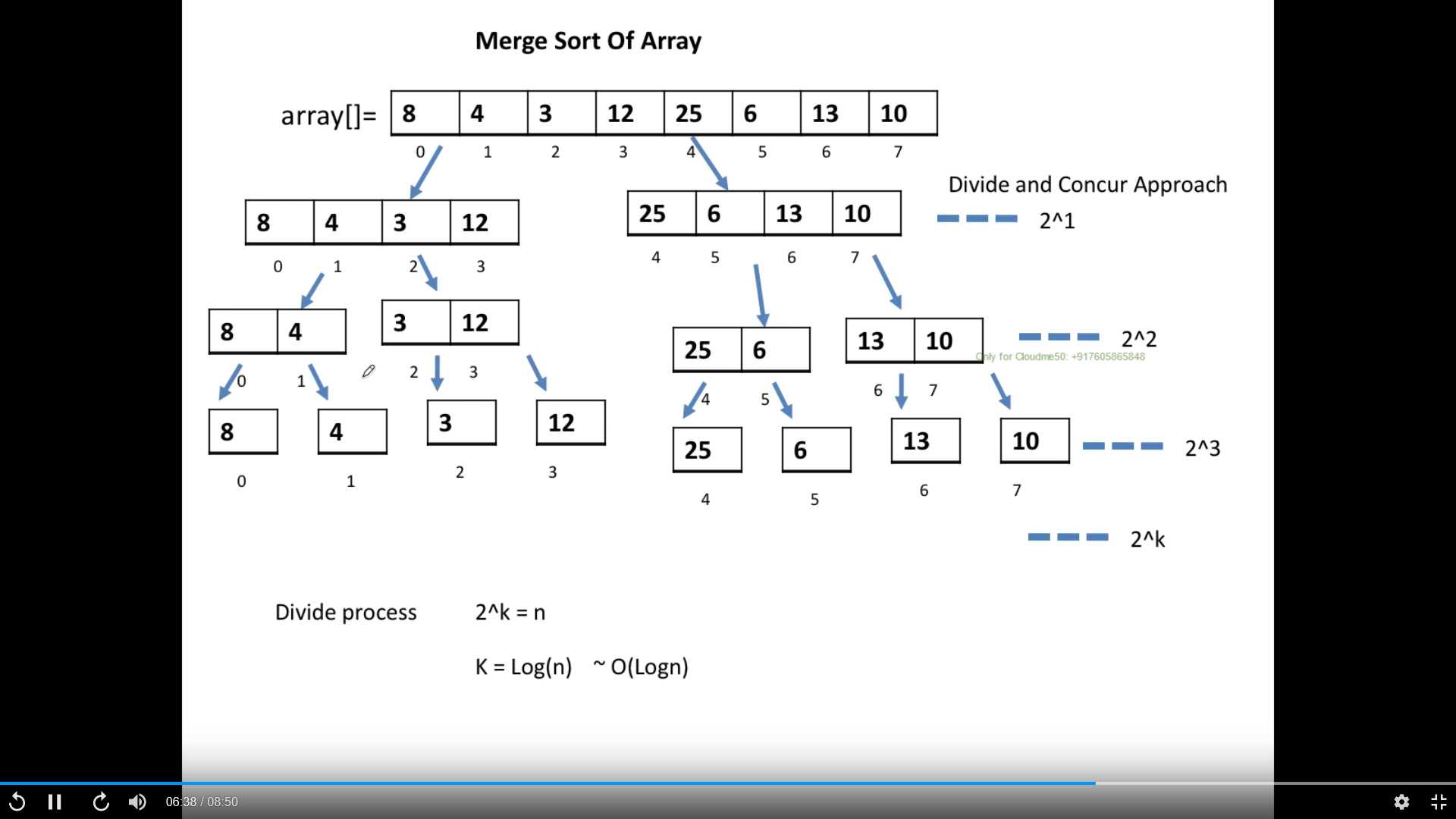


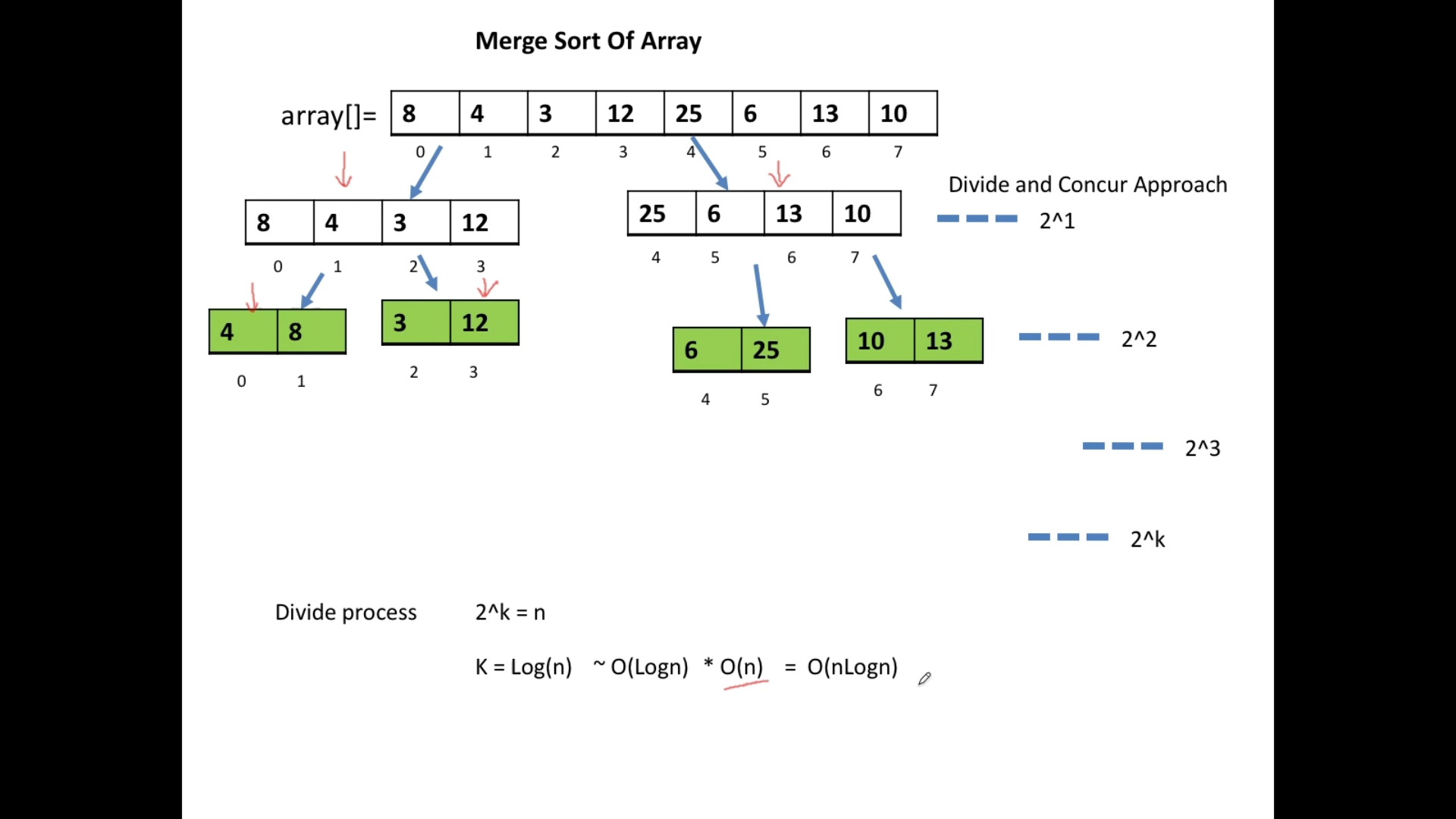


## Sorting Algorithm –

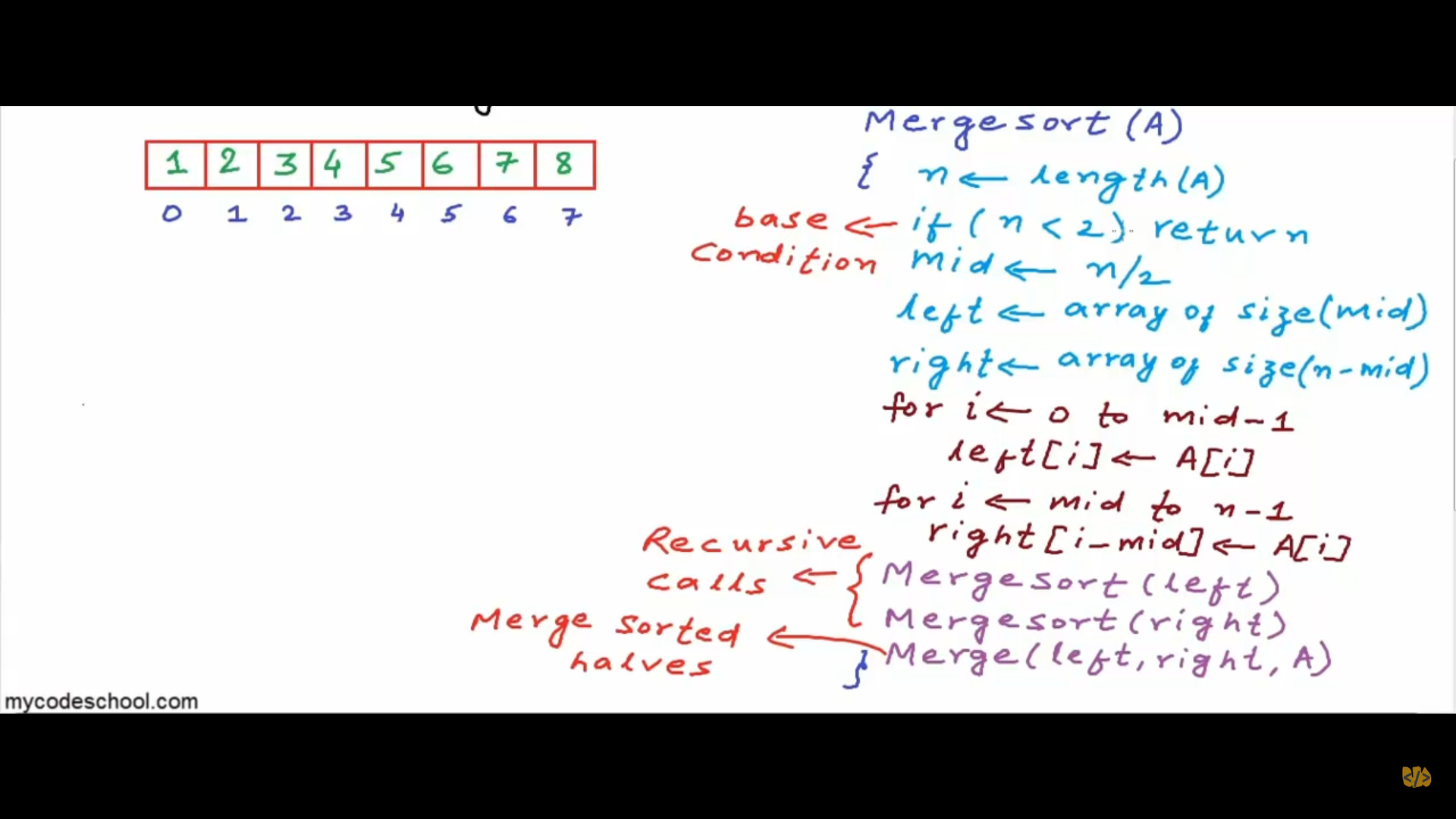
* merge sort used 98 %
  + Divide and concur approach
    - Keep diving an array until it has only one element
* 

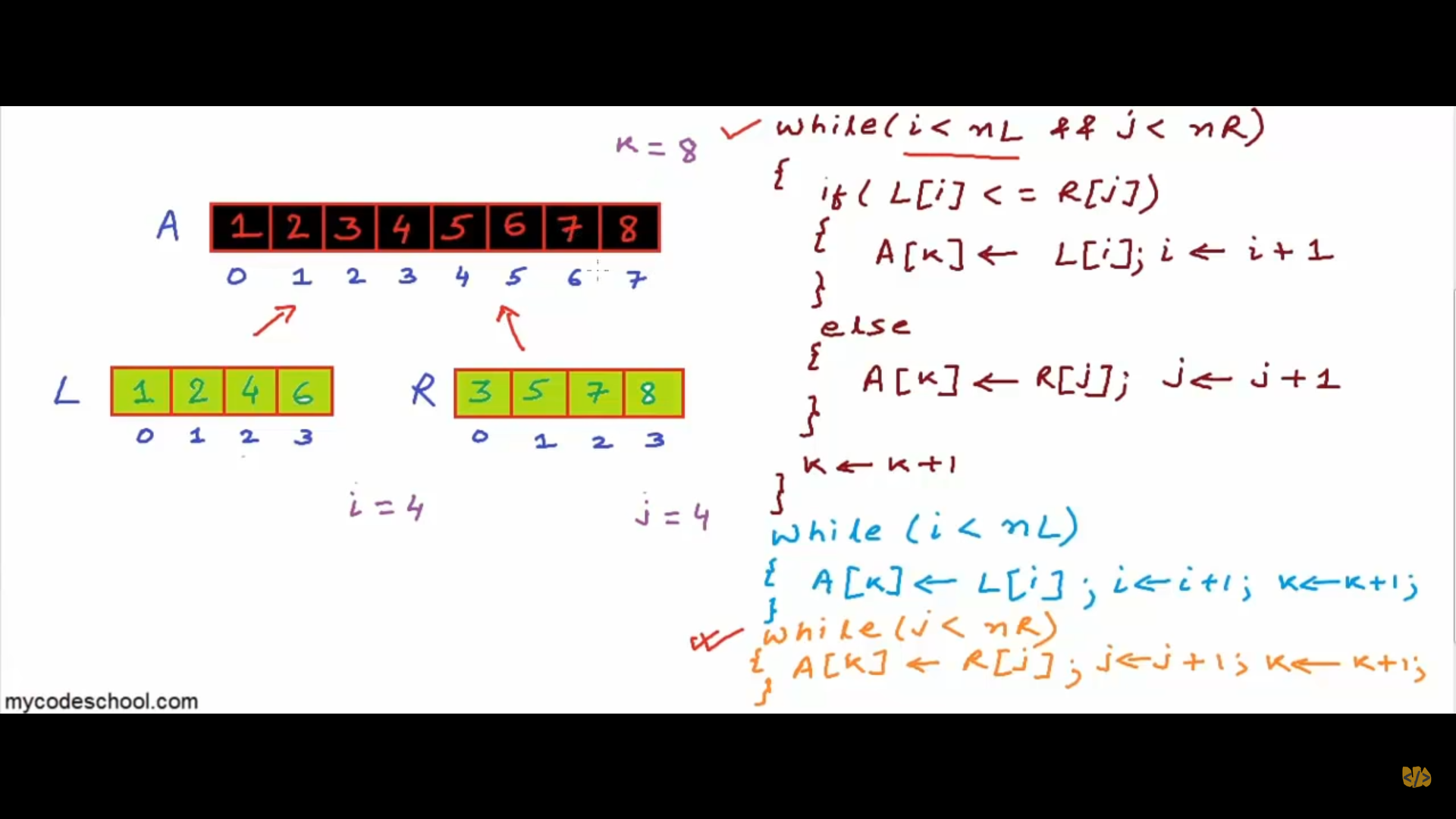






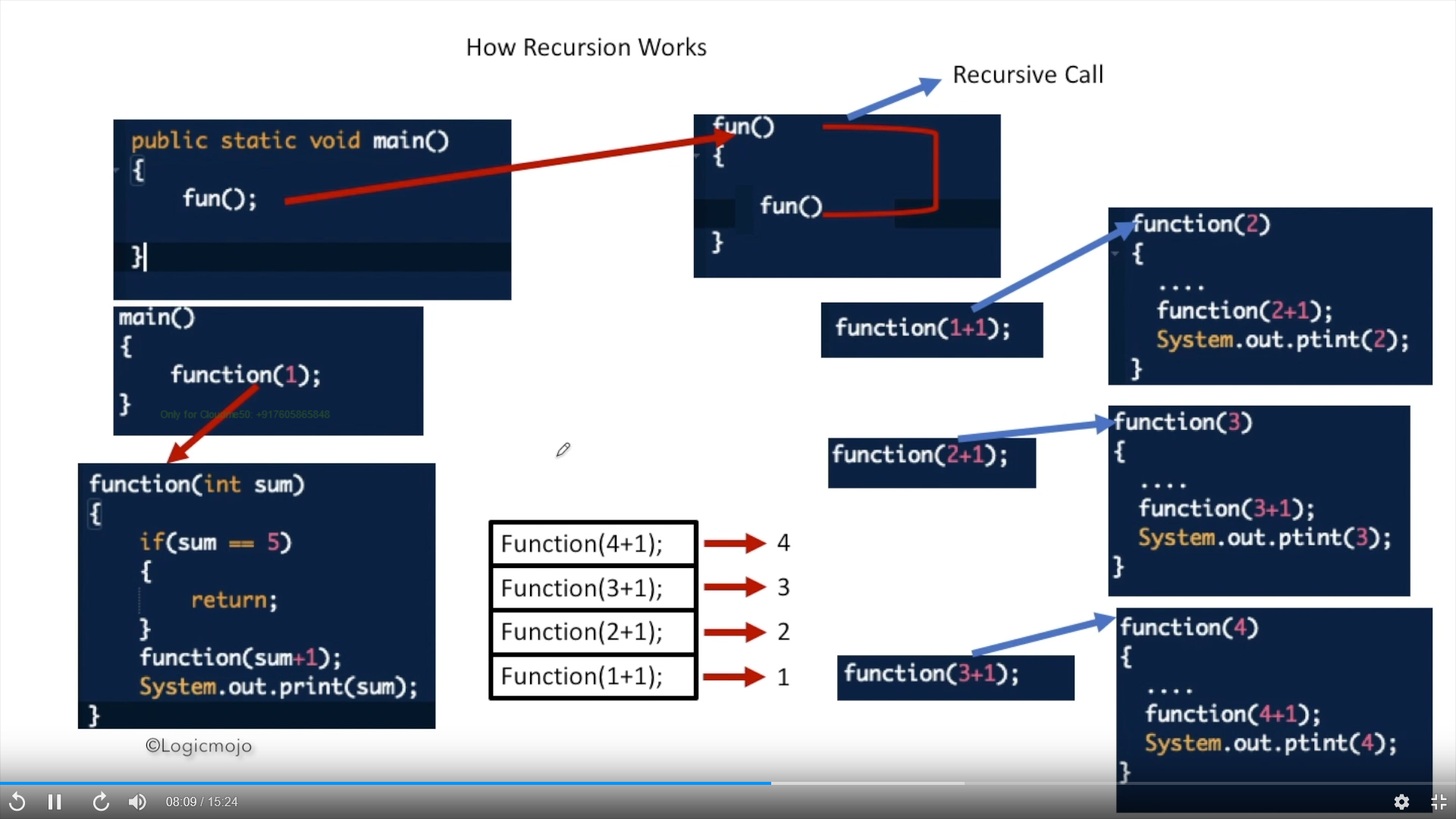
## Merge Sort solution from youtube

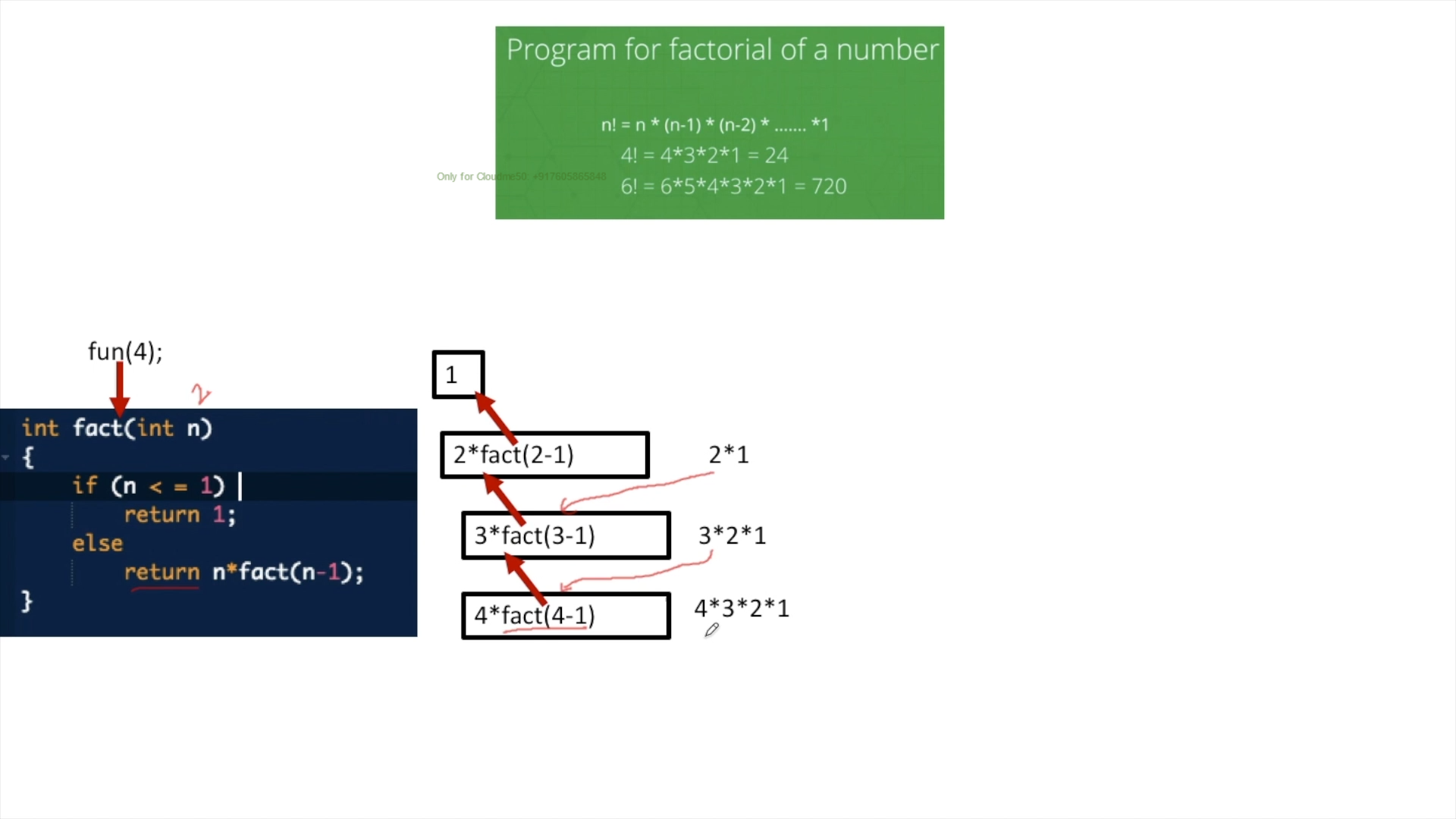


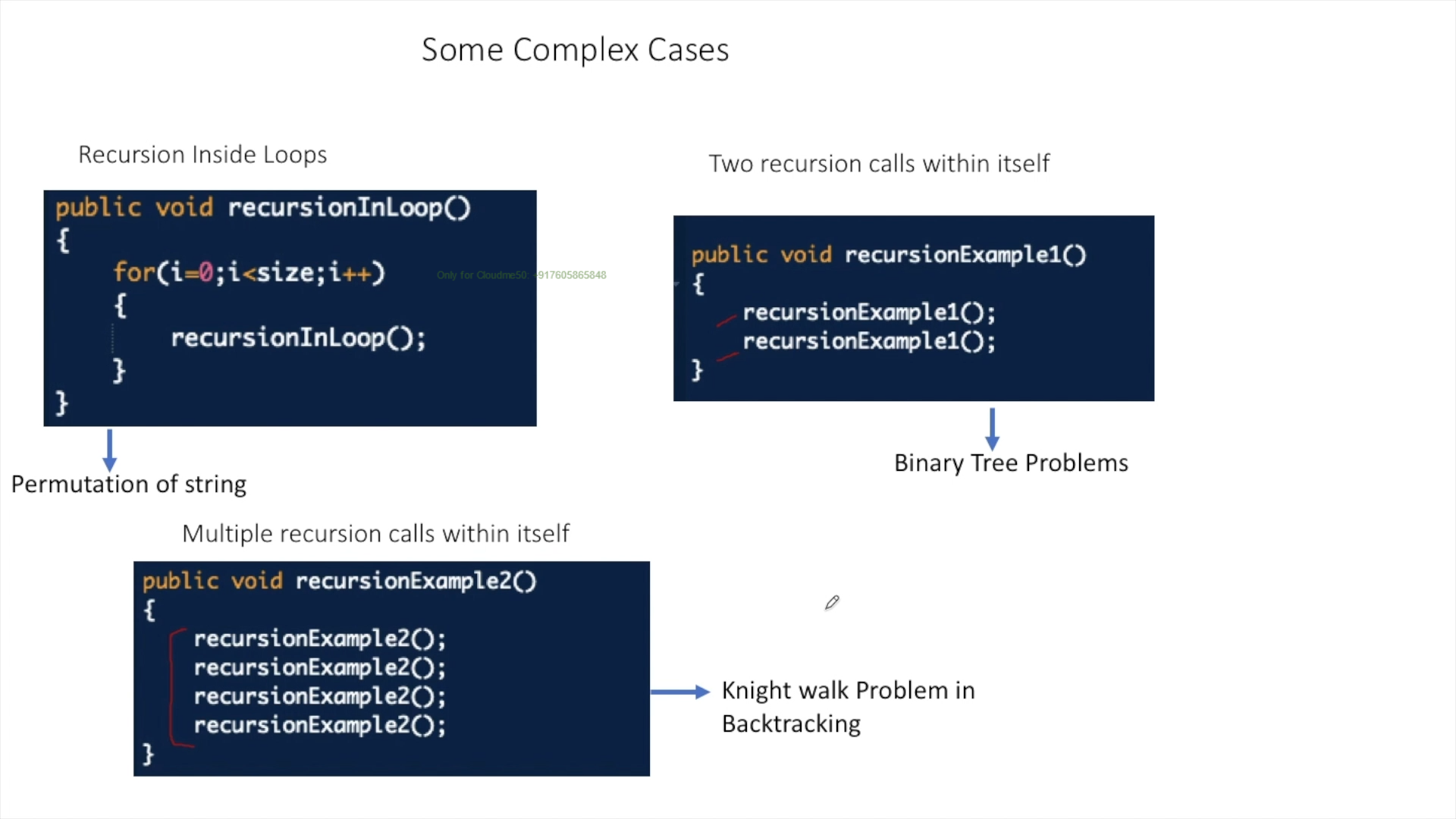


# Recursion

* **Main important** is that call get piled up as stack – LIFO – start executing from first
* Permutation of string
* Knight walk problem in backtracking







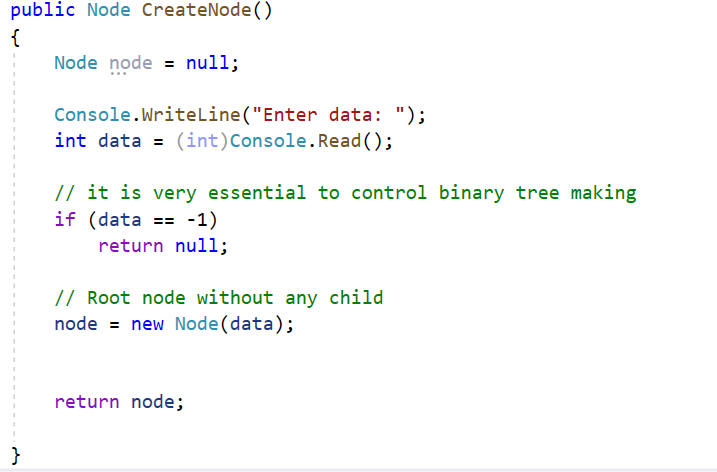
# Binary Tree

## Lab-1

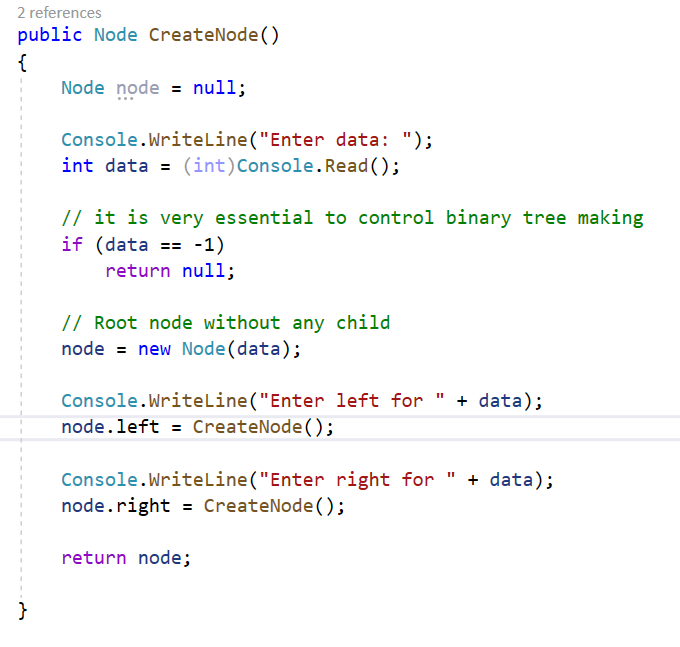
### Create a Node class



### Create a Binary Tree Parent node

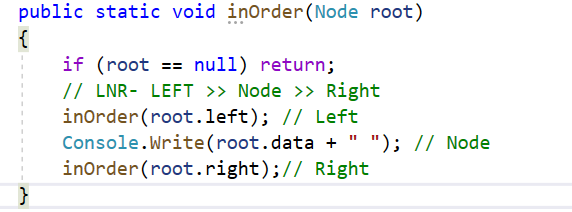


### Create Binary tree left and right node

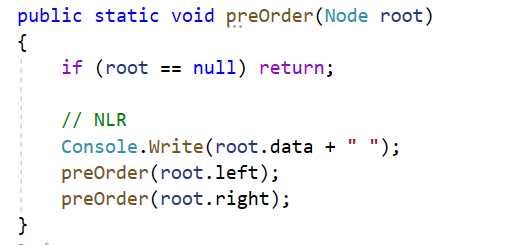


### Traverse the binary tree

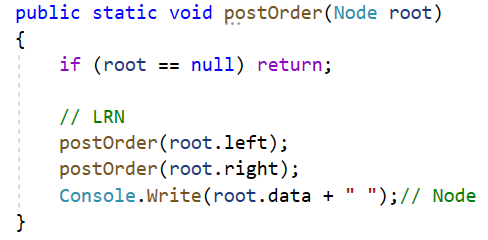
* + In order – **LNR**



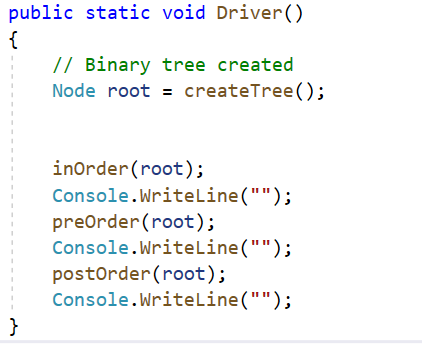
* + Pre- order – **NLR**



* + Post Order LRN



### Call tree and invoke it



* Final

