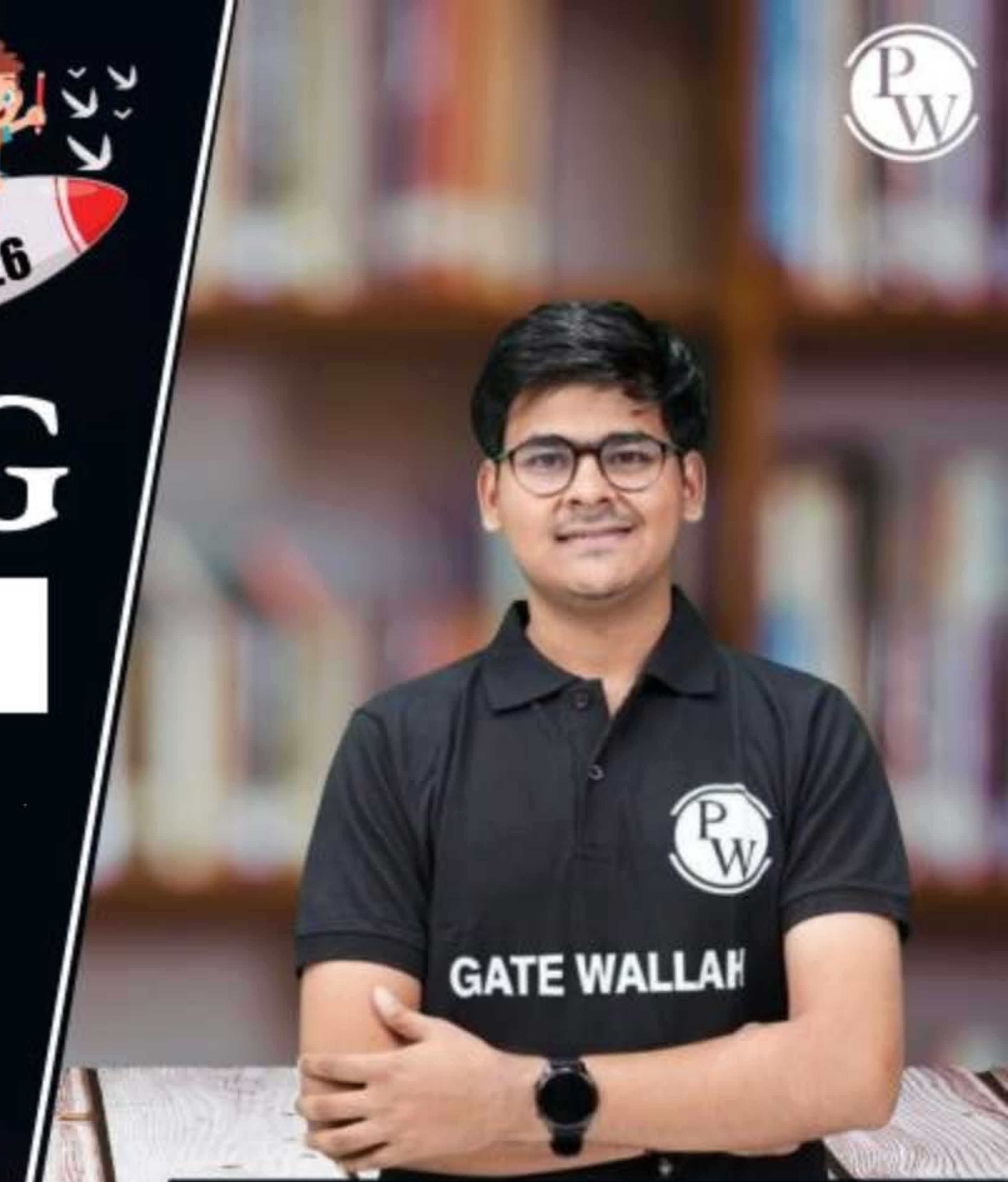
CS & IT ENGINEERING

Algorithms

Divide & Conquer

Lecture No.- 04



By- Aditya Jain sir

Topics to be Covered

Topic

Topic

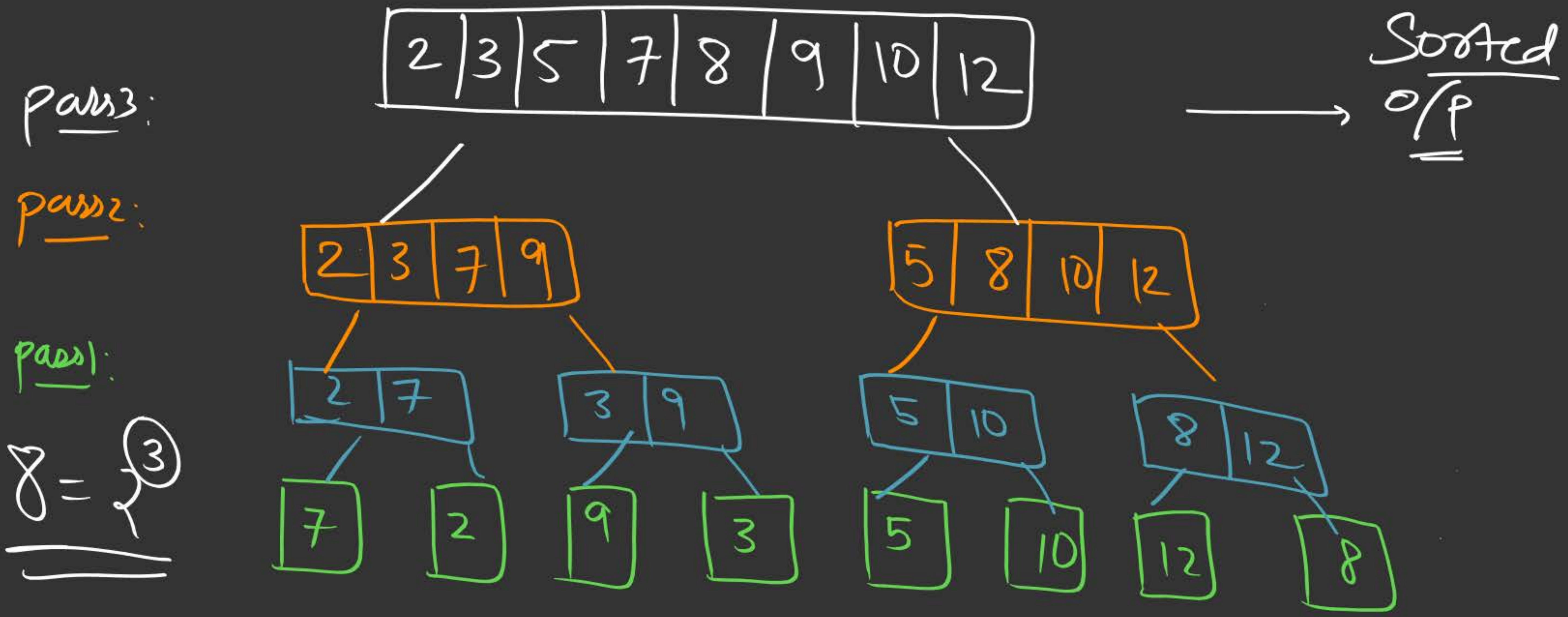
Partition Algo

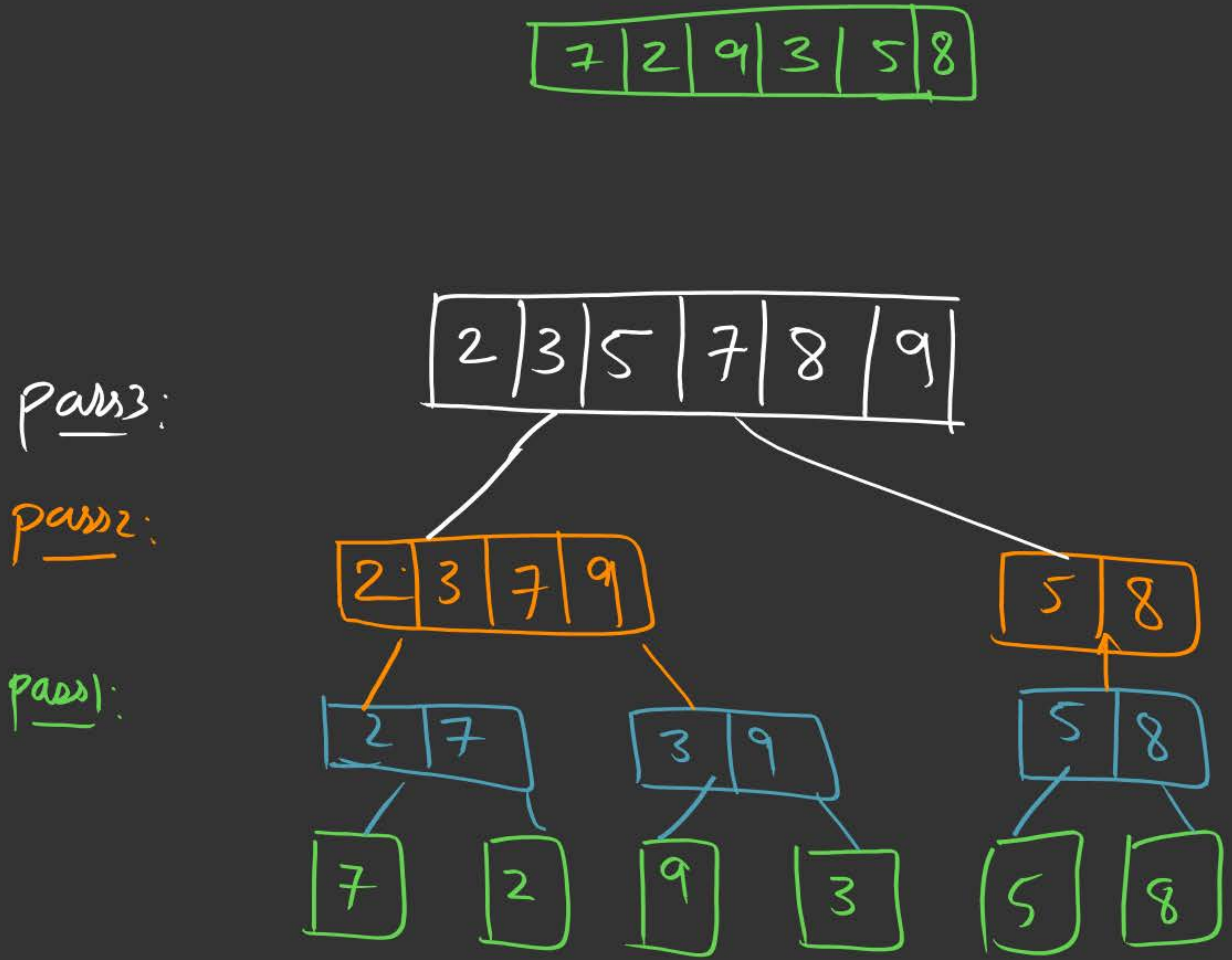
Quick Sort

Bottom-Up Merge Sort

7	2	9	3	5	10	12	8

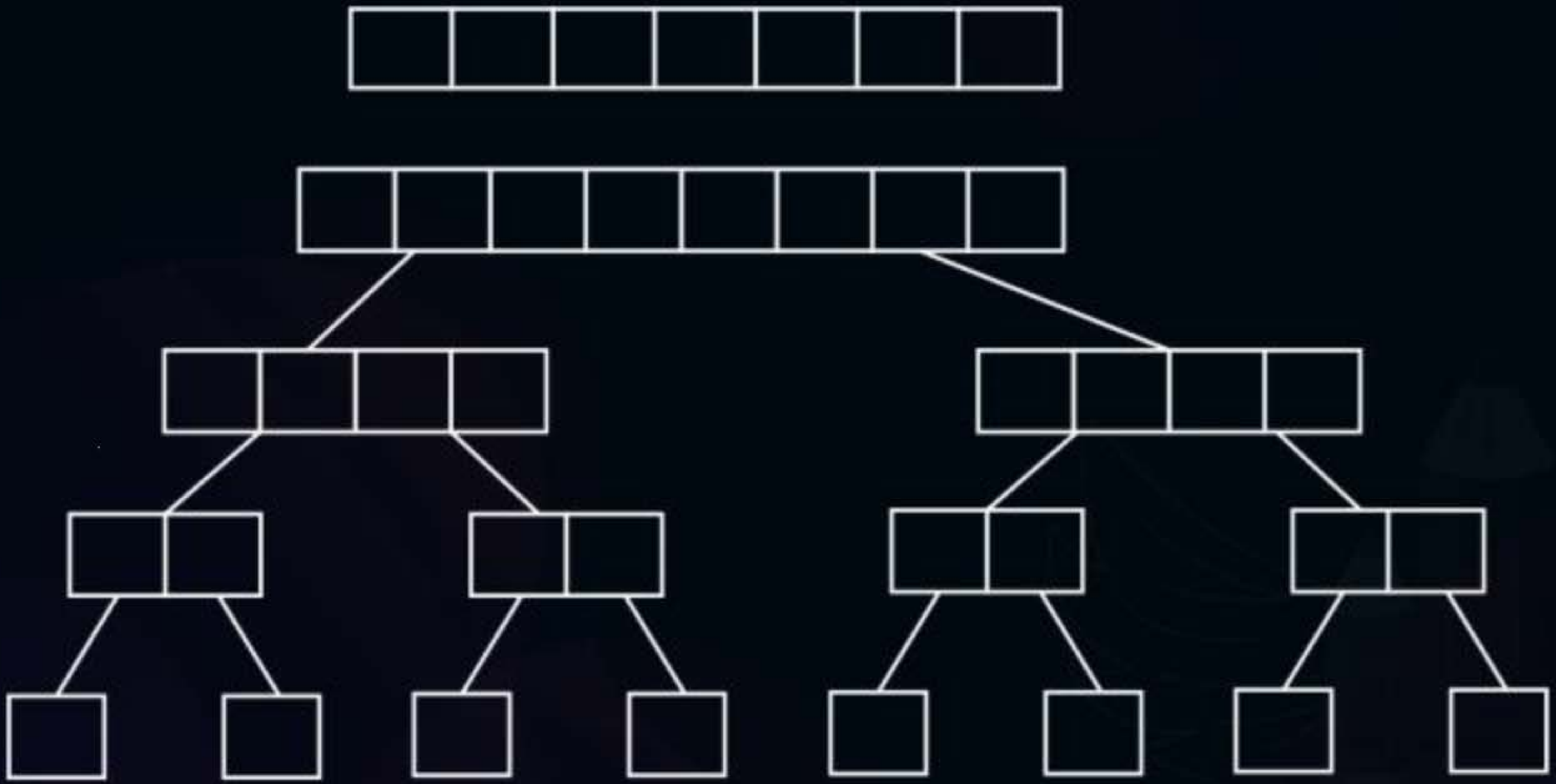
$\boxed{2^N}$





$O(n\log_2 n)$

Topic : 2 – Way Merge Sort (Bottom up merge Sort)



Topic : Divide and Conquer

#Q. If one uses straight two-way merge sort algorithm to sort the following elements in ascending order:

20, 47, 15, 8, 9, 4, 40, 30, 12, 17

then the order of these elements after second pass of the algorithm is:

A 8, 9, 15, 20, 47, 4, 12, 17, 30, 40

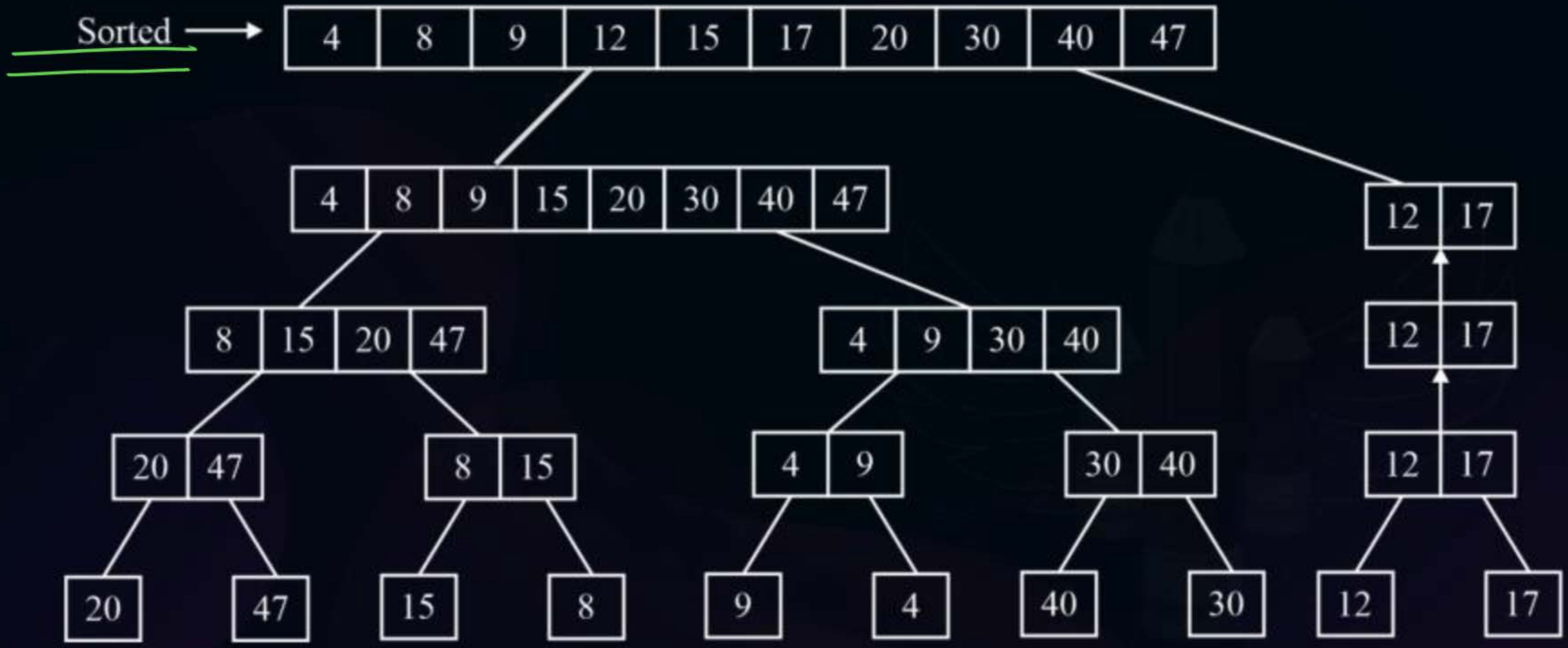
[✓ B] 8, 15, 20, 47, 4, 9, 30, 40, 12, 17 ✓

C 15, 20, 47, 4, 8, 9, 12, 30, 40, 17

D 4, 8, 9, 15, 20, 47, 12, 17, 30, 40

Topic : Divide and Conquer

[20, 47, 15, 8, 9, 4, 40, 30, 12, 17]



Sorted →

4 8 9 12 15 17 20 30 40 47

4 8 9 15 20 30 40 47

12 17

8 15 20 47

4 9 30 40

12 17

20 47

8 15

4 9

30 40

12 17

20

47

15

8

9

4

40

30

12

17

④ Quick Sort

↓

Ⓓ & C

Partition

X

Topic : Quick Sort

Quick Sort is based on Divide and Conquer Technique

Partitioning Algorithm / Process

Topic : Partitioning Process:- (Divide)

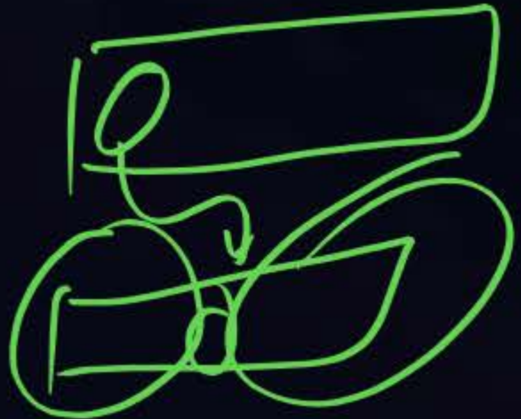
Default :- Usually, we take the first element as pivot.

Steps:- V.V. imp

(1) Find the correct place of Pivot element in the final sorted List.

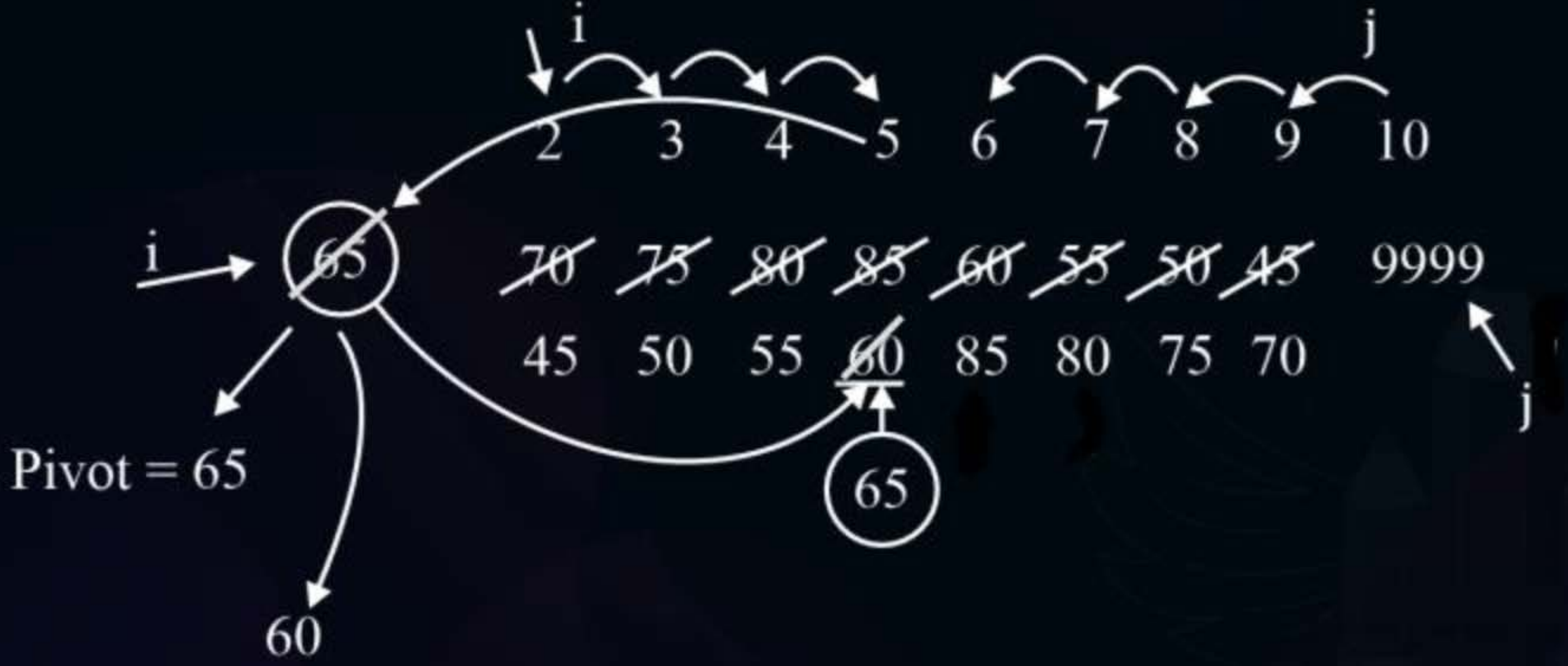
(2) After the pivot is placed at its correct position, all the elements that are lesser than pivot, will be on its left. (after Partitioning)

(3) And, similarly, all the elements that are greater than pivot, will be on it's right.



Given:-

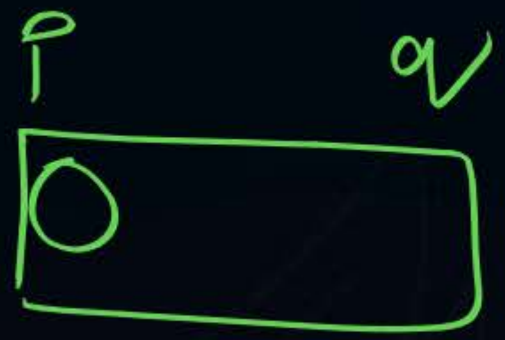
Array : 65, 70, 75, 80, 85, 60, 55, 50, 45



	i	j	
	1	10	
→	2	9	→ Swap a[2] ⇔ a[9], (70 ⇔ 45)
→	3	8	→ Swap a[3] ⇔ a[8], (75 ⇔ 50)
→	4	7	→ Swap a[4] ⇔ a[7], (80 ⇔ 55)
→	5	6	→ Swap a[5] ⇔ a[6], (85 ⇔ 60)
→	6	5	→ (i ≤ j) → False → Stop

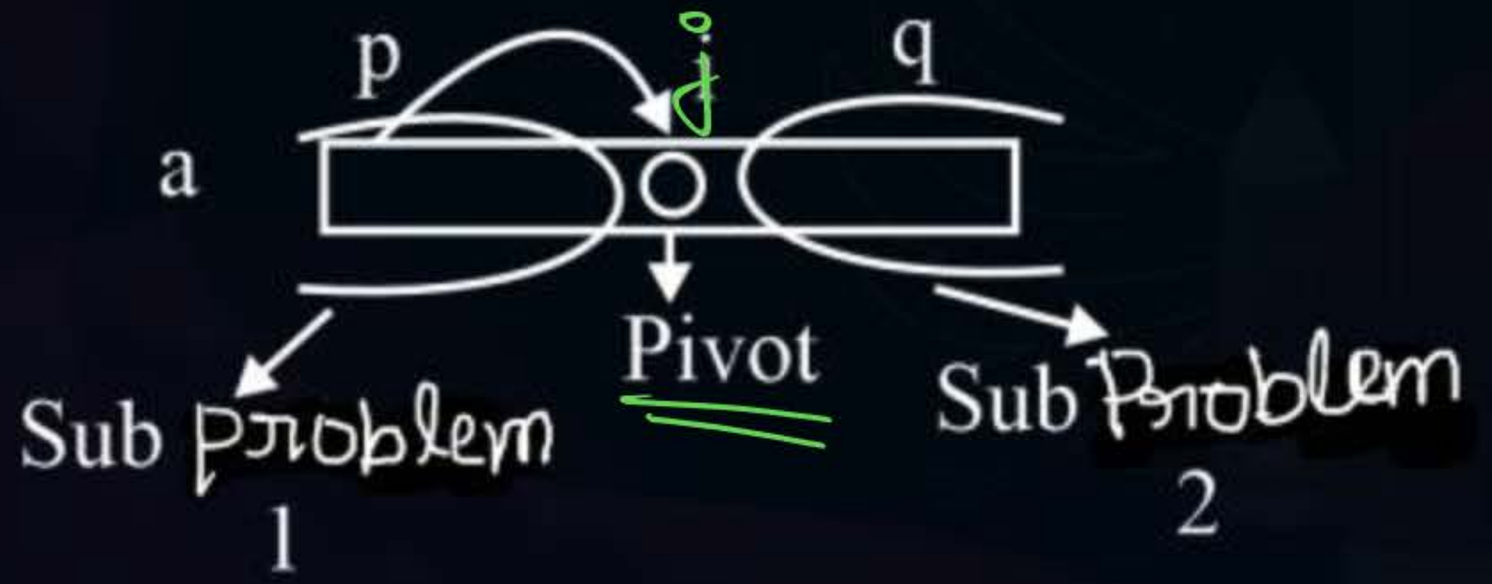
Topic : Divide and Conquer

```
Algorithm QuickSort(p, q)
// Sorts the elements a[p],..., a[q] which reside in the global
// array a [1: n] into ascending order: a[n + 1] is considered to
    // be defined and must be ≥ all the elements in [l: n].
    {
            if (p < q) then // If there are more than one element
            {
                    // divide P into two subproblems.
                    j := Partition(a, p, q + 1);
```



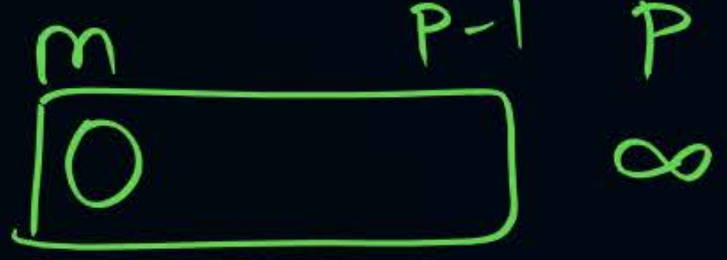
Topic : Divide and Conquer

```
                // Solve the subproblems.
                QuickSort(p, j – 1);
                QuickSort(j + 1,q);
                //There is no need for combining solutions.
          }
    }
```



Topic : Divide and Conquer

```
Algorithm Partition(a,m,p)
// Within a[m],a[m + 1],....., a[p – 1] the elements are
// rearranged in such a manner that if initially t = a[m],
    // then after completion a[q] = t for some q between m
    // and p – 1, a[k] ≤ t for m ≤ k < q, and a[k] > t
    // for q< k < p. q Is returned. Set a[p] = ∞.
    {
            v : = a[m]; i := m; j := p;
            do:
            {
```



$m \le q \le p-1$

Topic : Divide and Conquer

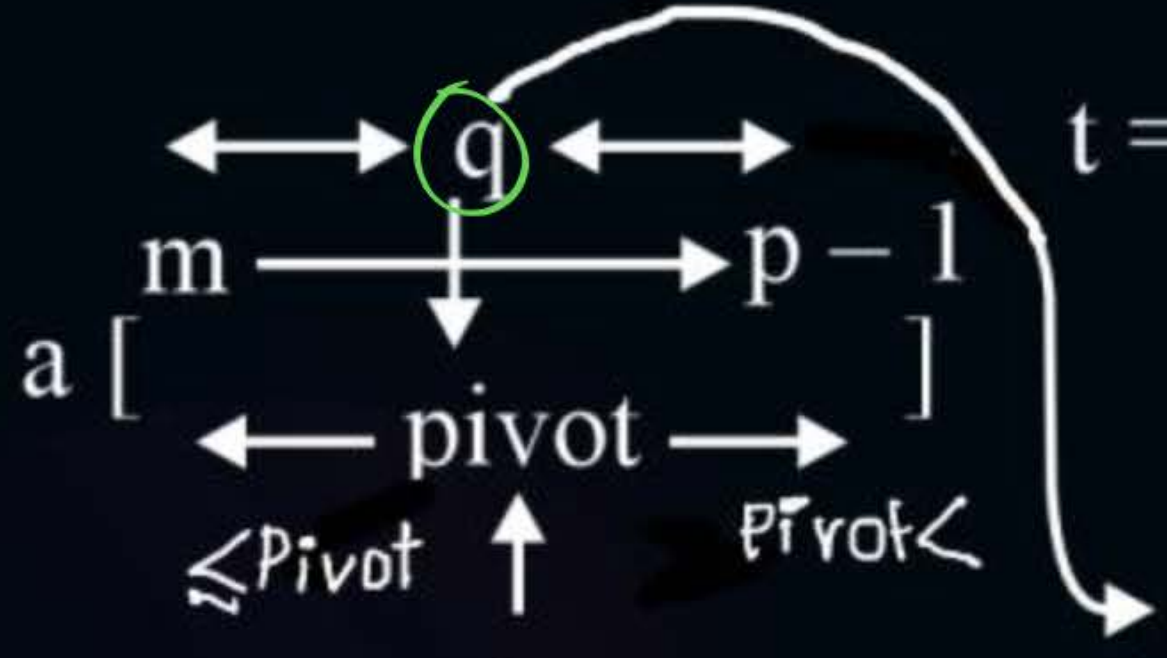
```
            do:
            i: = i + 1;
            while (a[i] ≤ pivot);
            do:
            J: = J- 1;
            while (a[j] ≥ pivot);
            if (i < j) then Interchanged (a, i, j);
        } while (i < j);
        a[m] := a[j]; a[J] := v; return j;
    }
```

TC : O(n)

Topic : Divide and Conquer

```
Algorithm Interchanged(a, i, j)
    // Exchange a[i] with a[j].
    {
            p := a[i];
            a[i] := a[j]; a[j] := p;
    }
```

Topic : Explanation Of Code Description



$t = a[m]$
pivot

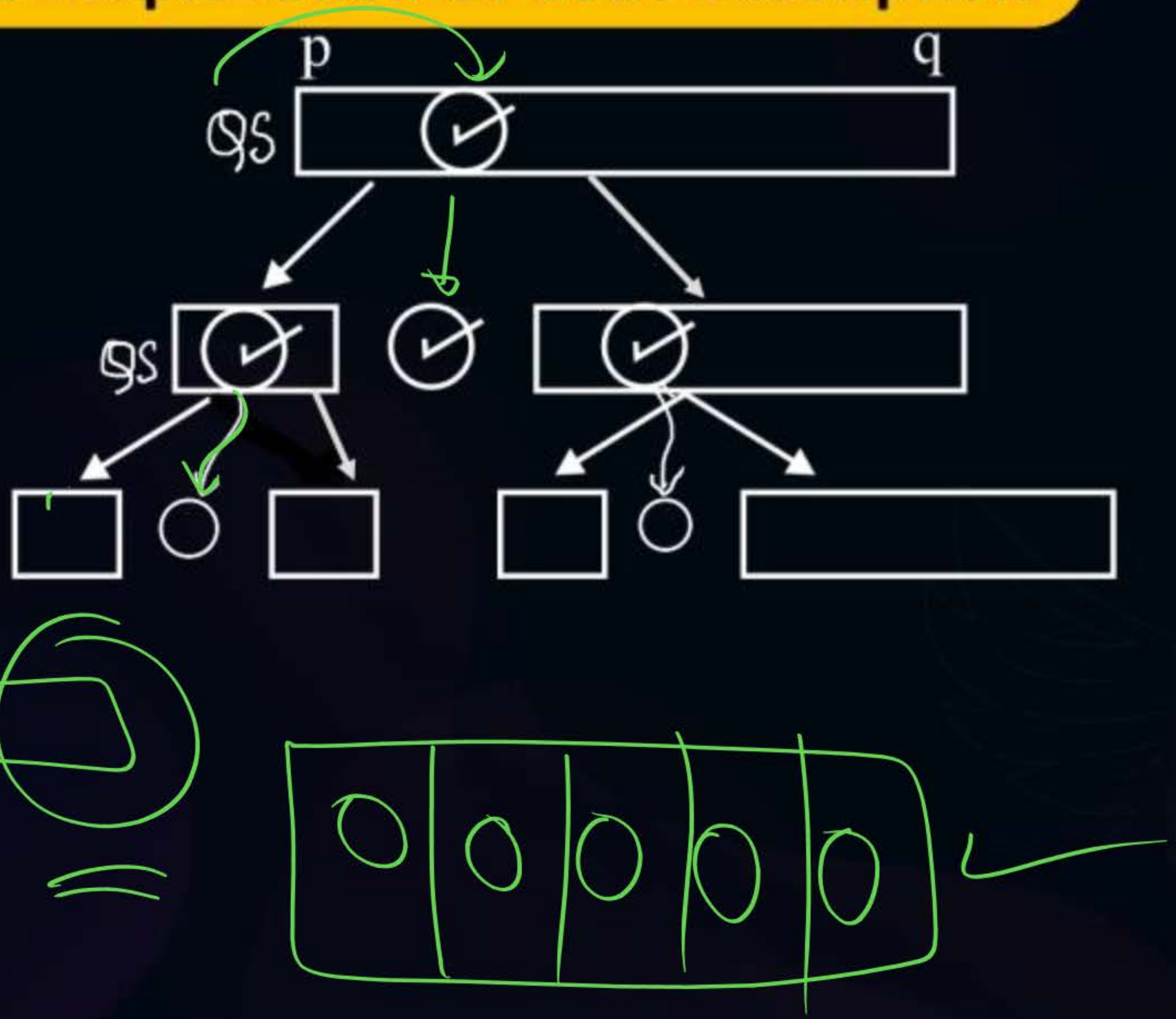
q= Correct position of pivot element

q $q < k < p$

$m \leq k < q$ $q < k < p$

$a[k] < t$ $a[k] > t$

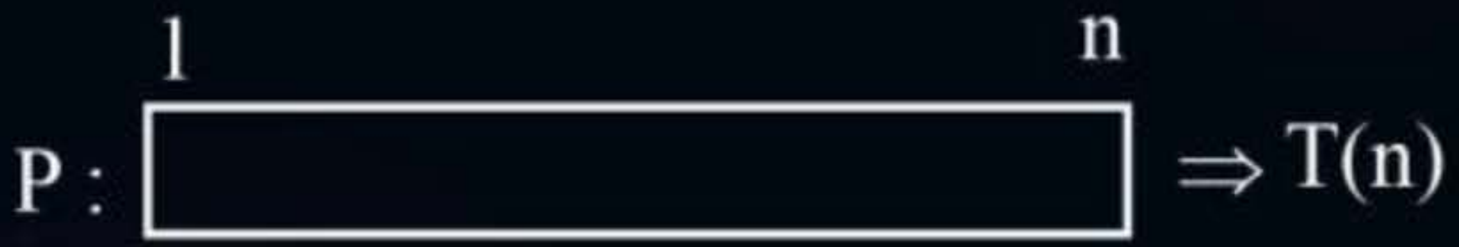
Topic : Explanation Of Code Description



Topic : Divide and Conquer

Performance:- Complexity analysis of ~~Quiche~~ Quick Sort

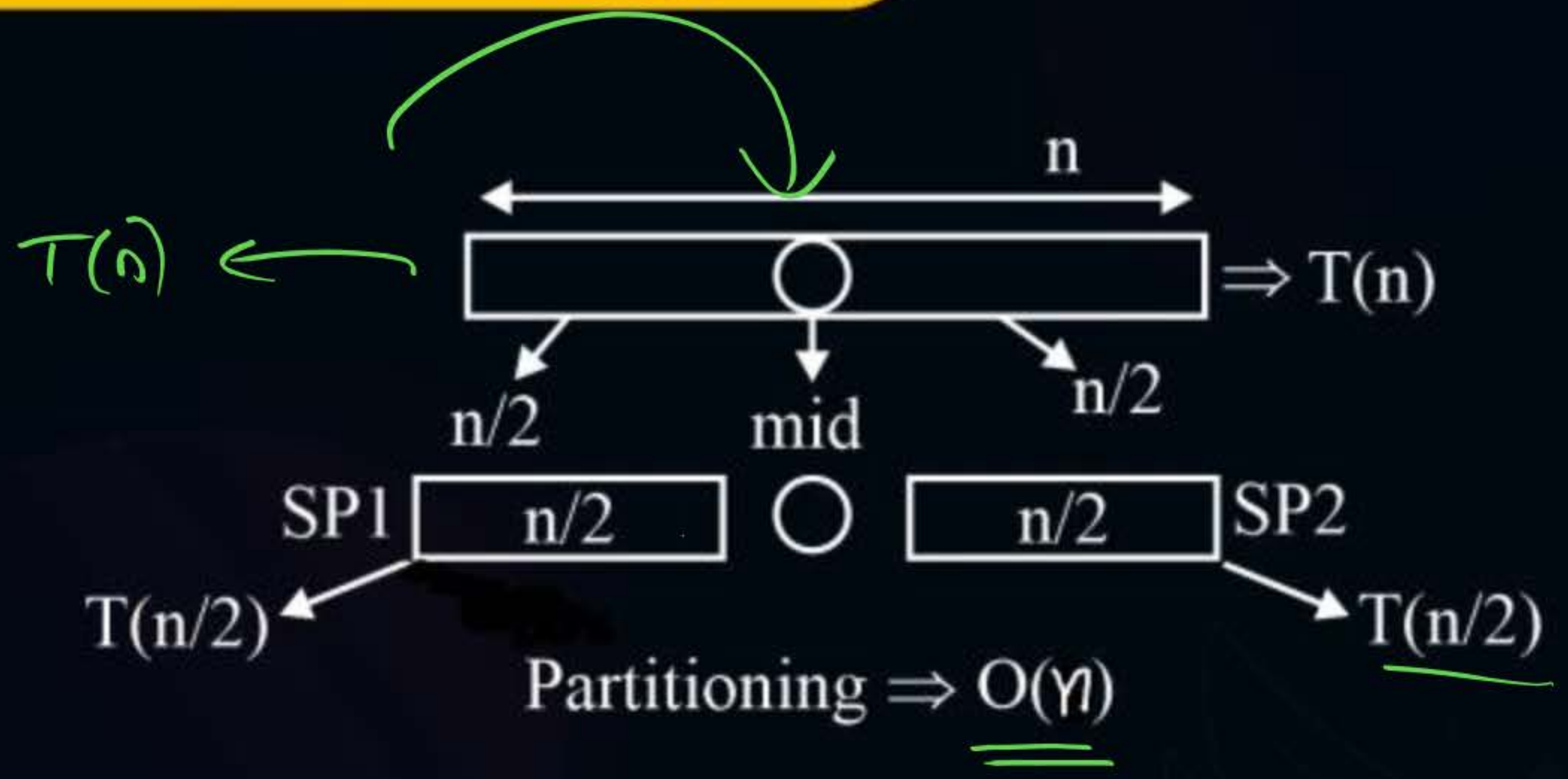
Problem :



Case 1: After partitioning, pivot always gets placed at the middle position, there by dividing the array into 2 nearly almost equal parts.

TC : $O(n \log n)$

Topic : Divide and Conquer



Case 1: Time Complexity:-

$$T(n) = \underbrace{T(n/2)}_{\text{left Sub-problem}} + \underbrace{T(n/2)}_{\text{right Subproblem}} + \overset{\text{partitiong}}{O(n)}$$

Topic : Divide and Conquer

$$T(n) = 2T(n/2) + O(n)$$

$$\boxed{T(n) \quad = 2T(n/2) + n}$$

$$T(n) \underline{\underline{= O(n\log_2 n)}} \text{ (Best case)}$$

Topic : Divide and Conquer

Case 2:- When after partitioning, the pivot gets placed always at the 1st position or the last position, there by reducing the size of the problem by just 1.

↓

When array is already sorted in ascending or descending order.

TC: $\underline{\underline{O(n^2)}}$

Topic : Divide and Conquer

Input :-

$$\overleftrightarrow{[\textcircled{5}, 10, 15, 20, 25]}^{n} \Rightarrow T(n)$$

$$\downarrow$$

$$\textcircled{5} \quad \underline{\overset{n-1}{[\textcircled{10}, 15, 20, 25]}} \Rightarrow \underline{\underline{T(n-1)}}$$

$$\downarrow$$

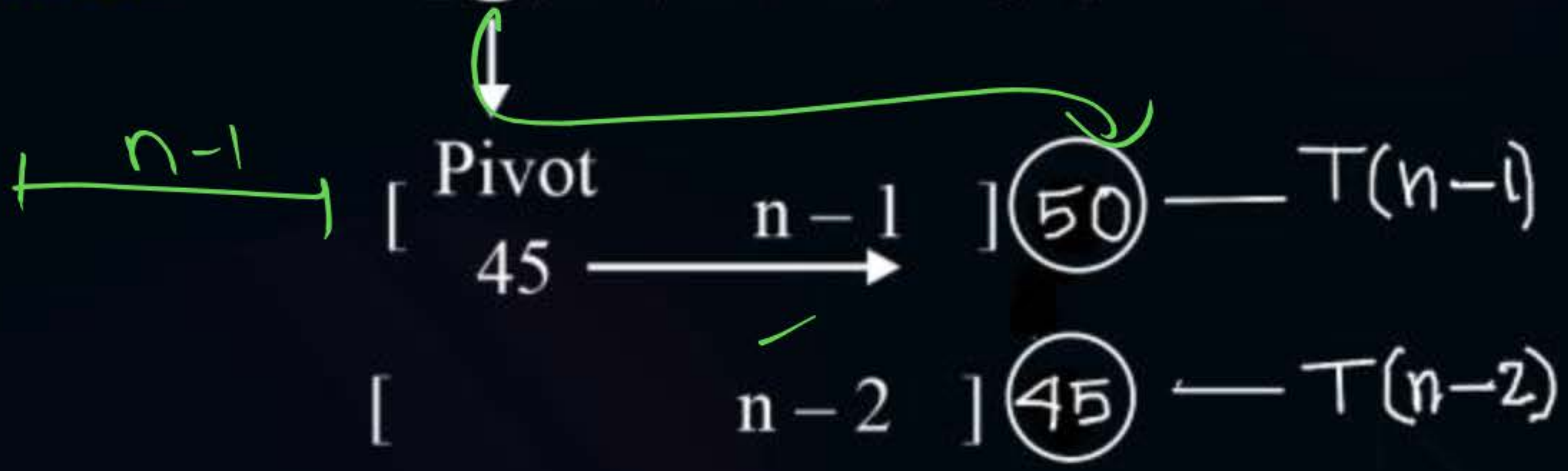
$$10\ [\qquad\qquad] \Rightarrow T(n-2)$$

$$\Rightarrow T(n-3)$$

Note :- This happens when the array is already sorted in ascending order.

Topic : Divide and Conquer

Input 2 :- $A = [50, 45, 40, 30, 21, 9]^n$



Note :- This happens when input array is already sorted in descending order.

Topic : Divide and Conquer

Time Complexity Recurrence for Case 2 :

Input: Input 2

$T(n) = \underbrace{T(n-1)}_{\text{Subproblem}} + O(n) \rightarrow \underline{\text{partitioning}}$

↓

Subproblem

$T(n) = T(n-1) + n$

$= T(n-2) + (n-1) + n$

$= T(n-3) + (n-2) + (n-1) + n$

$= T(n-k) + (n-(k-1)) + (n-(k-2)) + \ldots\ldots + (n-2) + (n-1) + n$

Topic : Divide and Conquer

General form

$$T(n) = T(n-k) + (n-(k-1)) + (n-(k-2)) + \dots\dots\dots + (n-2) + (n-1) + n$$

For Base Condition

$n - k = 1$

$k = (n-1)$

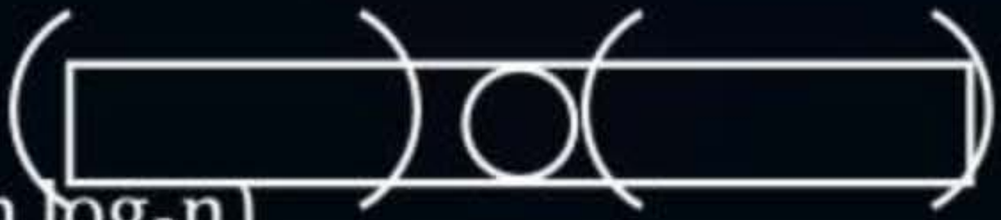
$$T(n) = T(1) + (n-(n-1)) + (n-(n-2)) + \dots\dots\dots + (n-2) + (n-1) + n$$

$$= T(1) + (1 + 2 + \dots\dots\dots + (n-2) + (n-1) + n)$$

$$= T(1) + \frac{n(n+1)}{2} = \boxed{O(n^2)}$$

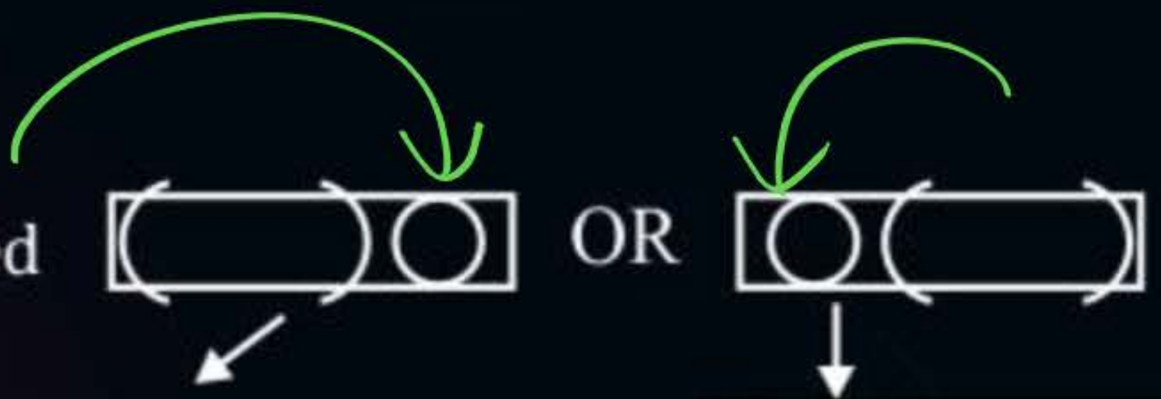
Topic : Time Complexity Sammary

(1) Best Case:- Input is unsorted



Avg Case → TC = O(n $\log_2$n)

(2) Worst Case:- Input sorted



OR

array is in descending order

When array input is already sorted in ascending order

$$\boxed{TC = O(n^2)}$$

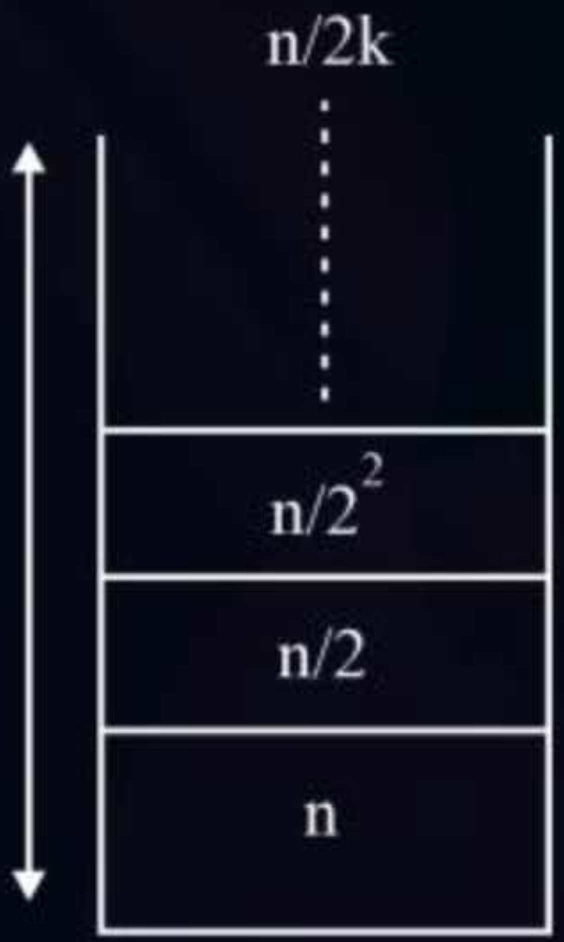
Topic : Space Complexity

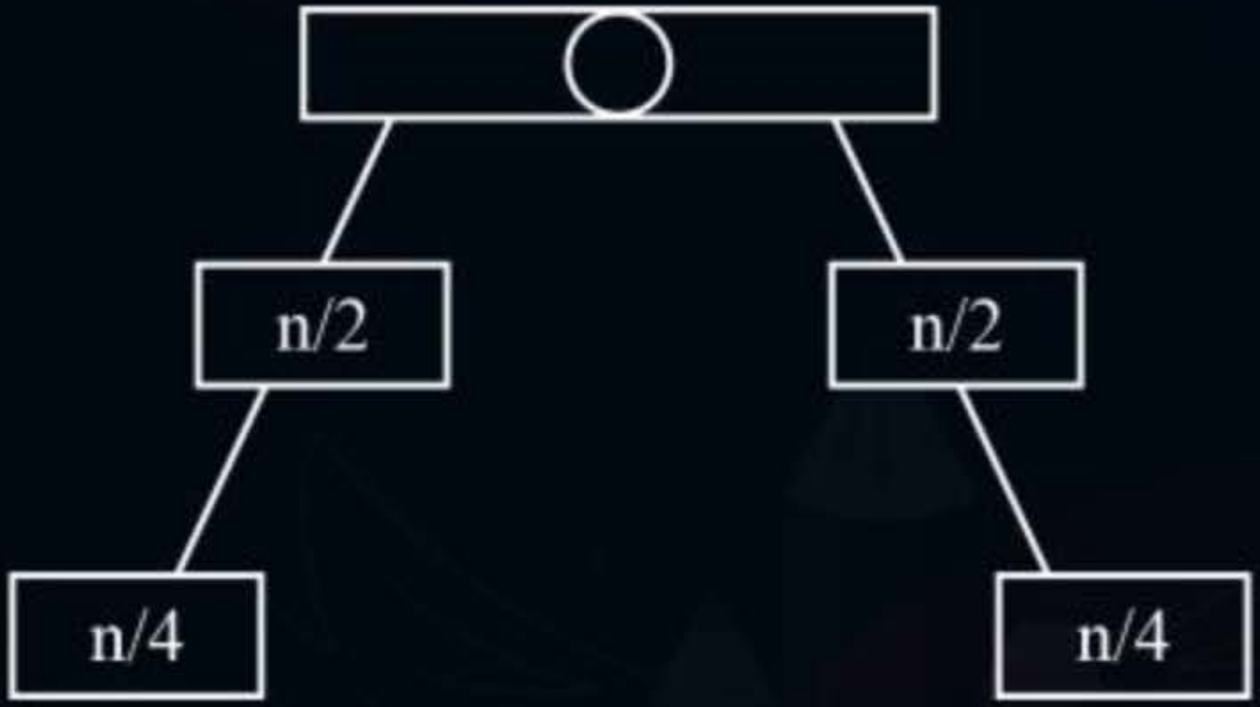
SC

Best Case :- $O(\log_2 n)$

Recursion stack

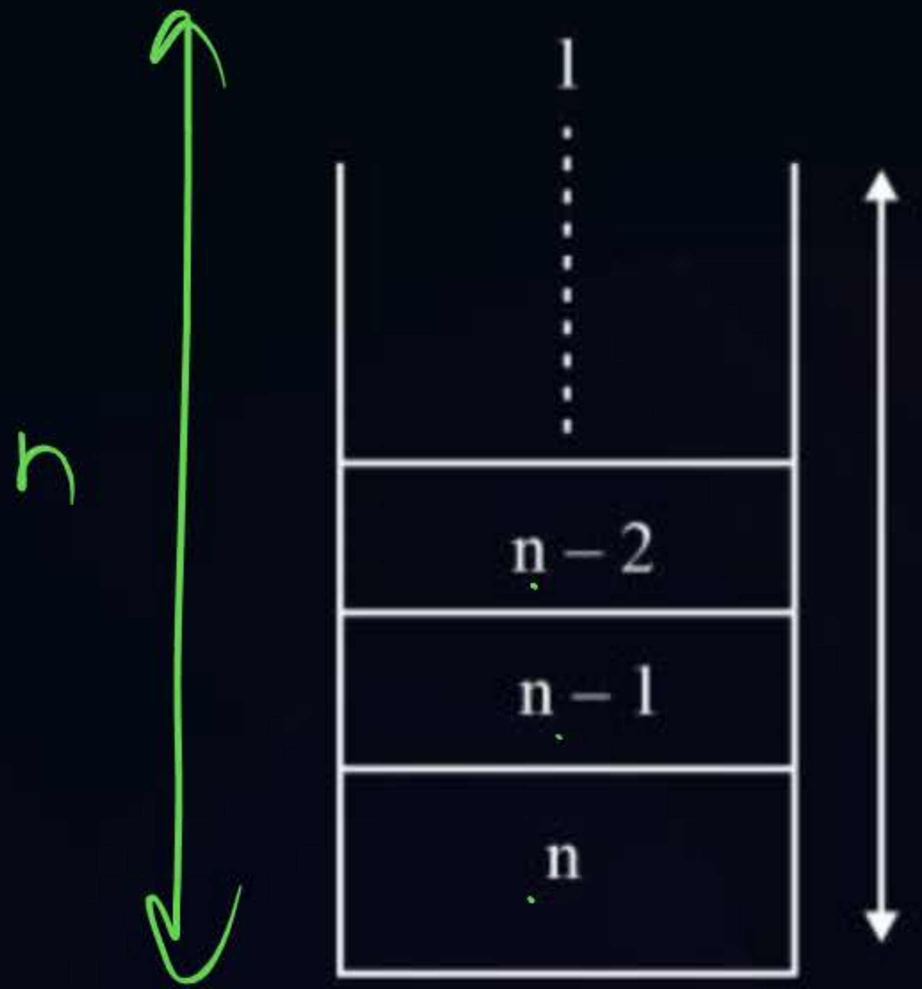
$\boxed{O(\log_2 n)}$



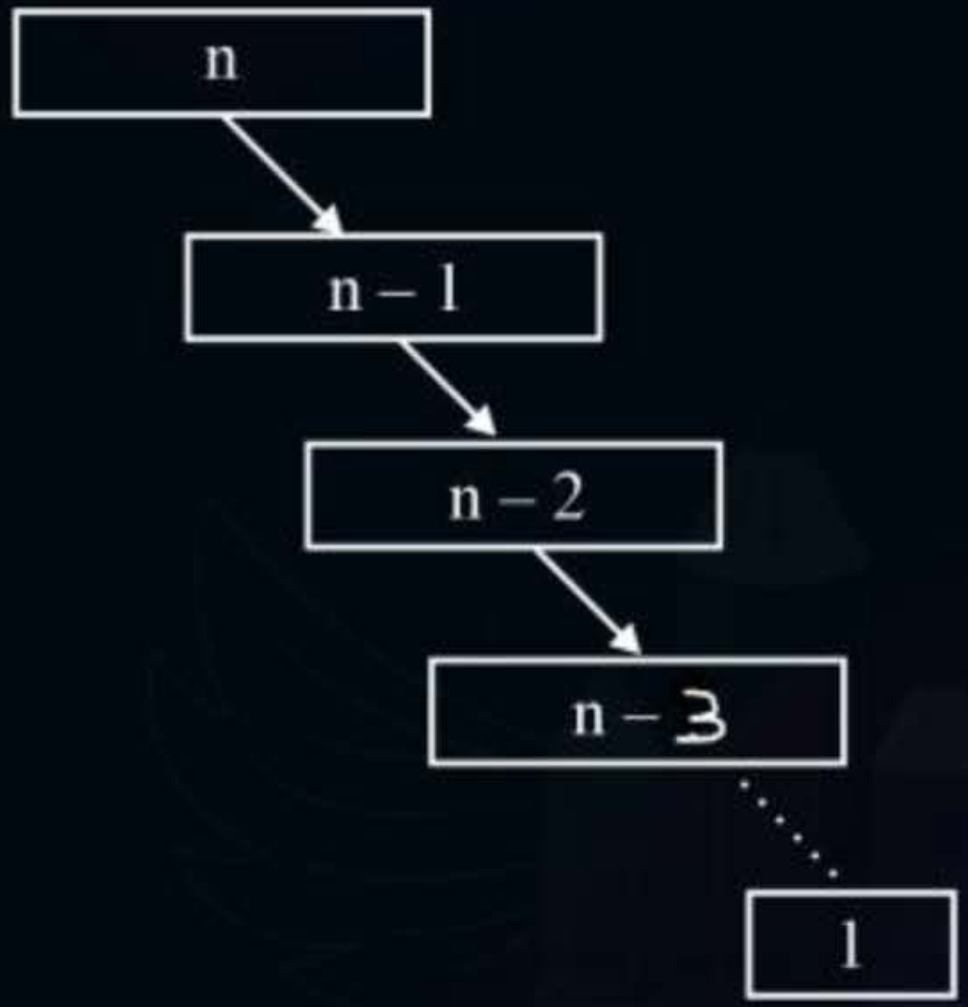


Topic : Worst Case

Recursion Stack



$\Rightarrow \boxed{O(n)}$ worst case Space Complaxity



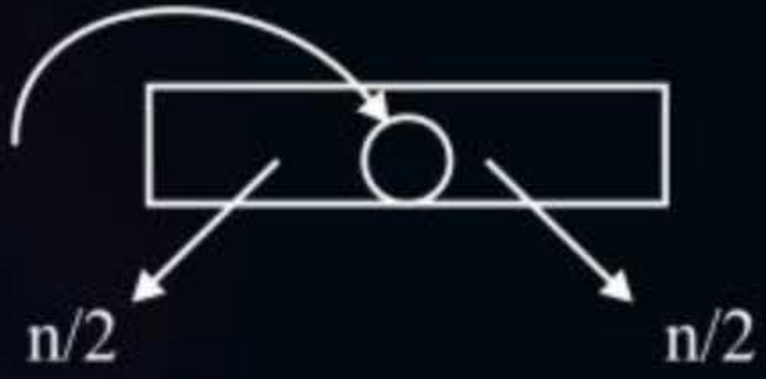
Topic : Divide & Conquer

Quick Sort :

1. **Time Complexity :**

(1) Best case : O(nlogn) : Unsorted input

(random)



(2) Worst case : $O(n^2)$: Sorted input

(asc/disc)

Topic : Divide & Conquer

Space Complexity :

1. Best Case : O(logn) → Recursion stack
2. Worst Case : O(n) → ~~Spaced~~ Recursion stack

Topic : Divide & Conquer

#Q. Let P be a quick sort program to sort numbers in ascending order. Let t_1 and t_2 be the time taken by the program for the inputs [1 2 3 4] and [5 4 3 2 1] respectively. Which of the following holds?

A $t_1 = t_2$

B $t_1 > t_2$

[✓ C] $t_1 < t_2$

D $t_1 = t_2 = = 5 \log 5$

$n_1 = 4$

$n_2 = 5$

$t_1 < t_2$

Topic : Divide & Conquer

70%

#Q. Let P be a Quick Sort Program to sort numbers in ascending order using the first element as pivot. Let t_1 and t_2 be the number of comparisons made by P for the inputs $\underbrace{\{1, 2, 3, 4, 5\}}_{WC}$ and $\underbrace{\{4, 1, 5, 3, 2\}}_{AC}$ respectively. Which one of the following holds.

A $t_1 = 5$ $n_1 = 5$

B $t_1 < t_2$ $n_2 = 5$ $\boxed{t_2 < t_1}$

[✓ C] $t_1 > t_2$

D $t_1 = t_2$ ✗

Topic : Divide & Conquer

$\boxed{90^{+}\%}$

#Q. Quick Sort is run on two inputs shown below to sort in ascending order taking first element as pivot

i. 1, 2, 3 . . . . n

ii. n, n – 1, n – 2, . . . , 2, 1

Let C_1 and C_2 be the number of comparisons made for the inputs (i) and (ii) respectively. Then,

A $C_1 > C_2$

B $C_1 < C_2$

[✓ C] $C_1 = C_2$

D We can't say anything for arbitrary n

$$\left.\begin{array}{l} n_1 = n \\ n_2 = n \end{array}\right\} \boxed{C_1 = C_2}$$

Topic : Divide & Conquer

#Q. An array of 25 distinct elements is to be sorted using quicksort. Assume that the pivot element is chosen uniformly at random. The probability that the pivot element gets placed in the worst possible location in the first round of partitioning (rounded off to 2 decimal places) is ________.

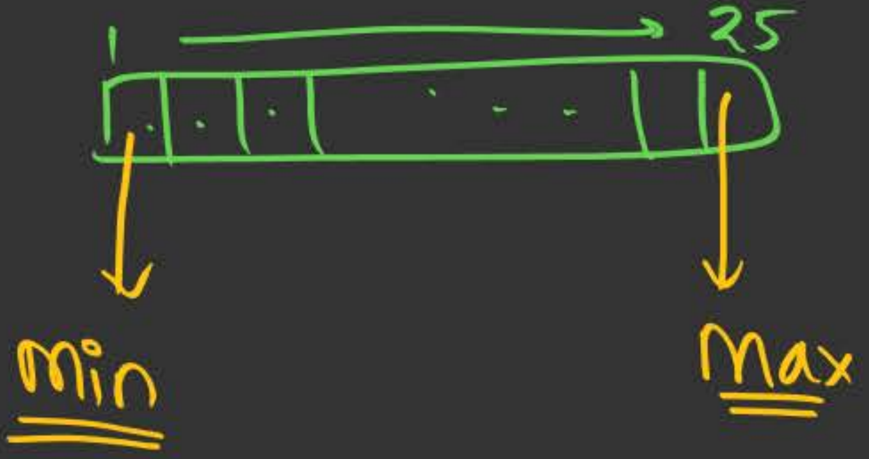
$\boxed{\text{val} * 100}$

50%

$= 0.08 \times 100$

$= \boxed{8}$ ✓

wc



$\frac{1}{25}$

$$P(\text{pivot} = \text{min or max}) = \frac{1}{25} + \frac{1}{25} = \frac{2}{25} = 0.08$$

$$\boxed{P(\text{min/max}) = \frac{2}{25}}$$

Topic : Divide & Conquer

#Q. Assume that merge sort takes 30 sec to sort 64 elements in worst case. What is the approximate number of elements that can be sorted in the Worst case using merge sort using 6 minutes?

HW

THANK - YOU