GATE

CRASH COURSE

CS & IT

Algorithms

Sorting Algorithms

Lecture 03

By - Aditya sir

Topics to be Covered

1. Sorting Algorithms
2. Imp terminologies
3.
4.

About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored working professions in field of Data Science and Analytics
11. Have been mentoring GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

Topic : Introduction to Sorting Algorithms

A sorting algorithm Syllabus:-

1. Bubble sort
2. Selection sort
3. Insertion sort
4. Radix sort
5. Merge sort
6. Quick sort
7. Heap sort

Divide & Conquer

Topic : Introduction to Sorting Algorithms

For every Algorithm

- Example
- Code
- Internal working
- Complexity Analysis
  - Time complexity
  - Space complexity
- Properties
- Practice Questions + PYQs

Topic : Introduction to Sorting Algorithms

Sorting:-

A process or ordering/ arranging the given element in a particular sequence / order as per the given criteria.

Topic : Introduction to Sorting Algorithms

Eg:- [50, 79, 3005, 452, 297]

1. Ascending order : [50, 79, 297, 452, 3005]
2. Descending order : [3005, 452, 297, 79, 50]

values ✓

[50, 79, 3005, 452, 297]

③ Arrange in ascending order of their last digit (rightmost)

Criteria

50	79	3005	452	297
↓ 0	↓ 9	↓ 5	↓ 2	↓ 7
①	⑤	③	②	④

Final O/P: [50, 452, 3005, 297, 79]

Sorting Algo

① Comparison based

non-Comparison based

→ Radix Sort

Topic : Introduction to Sorting Algorithms

- **Comparison based sorting**
  - Elements are compared with each other to achieve sorting

E.g. Bubble

Selection

Insertion

Merge

Quick

Heap

Topic : Introduction to Sorting Algorithms

- Non-Comparison based sorting
  - Sorting is achieved without any comparison among the elements.

  E.g. Radix Sort

Topic : Introduction to Sorting Algorithms

- In place Sorting Algorithms
  Algorithm whose auxiliary/Addition space requirement → Space Complexity
  - at max O(1) → Excluding Recursion
  - At mx O(logn) → For Recursion stack

E.g. Merge Sort → Not in place

Space 1) Temp array → $O(n)$
2) Recursion Stack → $O(\log_2 n)$

$O(n + \log_2 n)$
$= O(n)$

Topic : Introduction to Sorting Algorithms

- **Stable Sorting Algorithm:**

  Simple:-

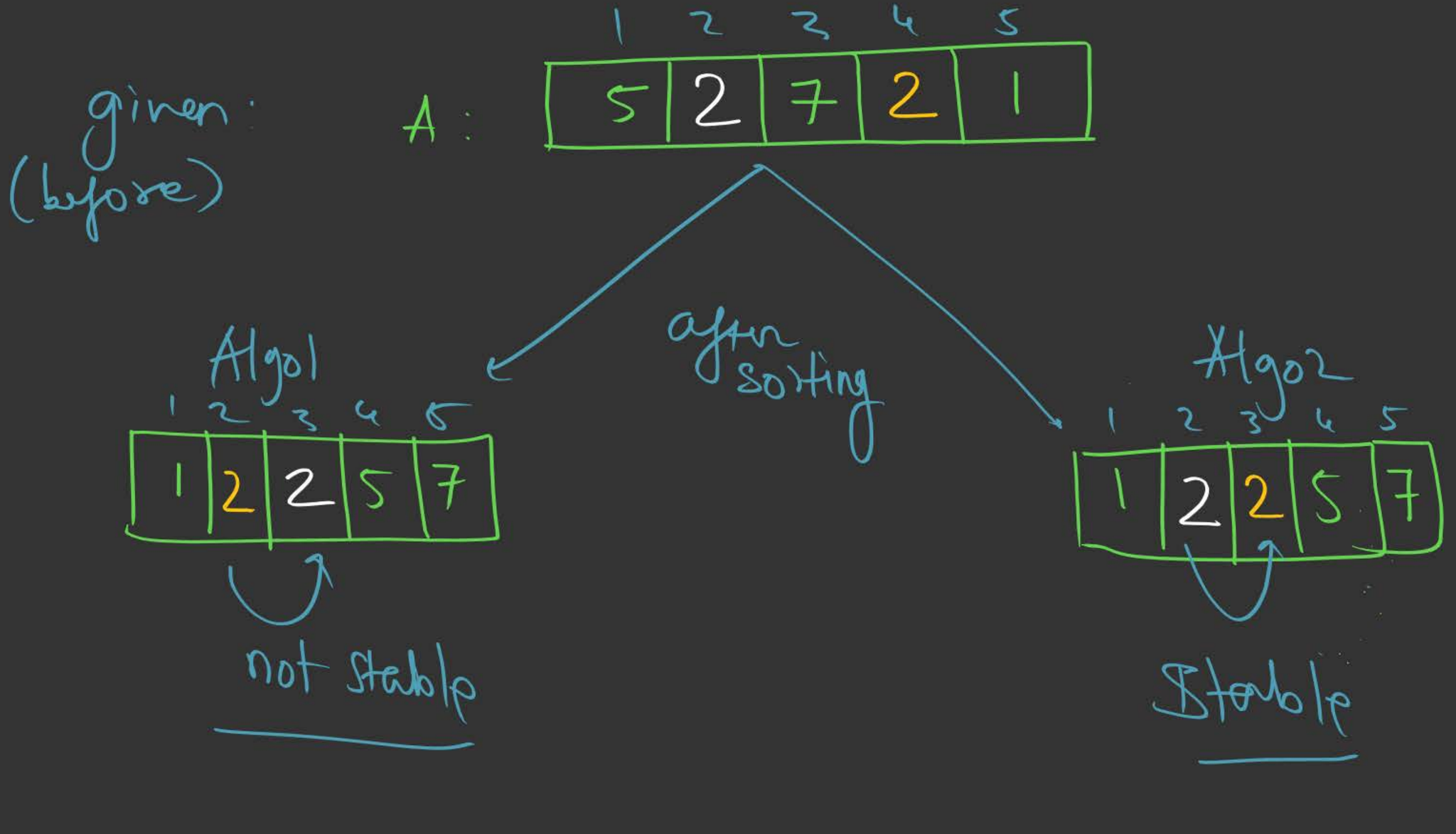
  - If relative position of non- distinct elements is maintained before and after sorting.

Topic : Introduction to Sorting Algorithms

- **Formal Definition**

If A [i] = A [j]

In given input: A [i] is before A[j] , (i<j) then the sorting algo is stable iff A[i] is also before A[j] in the final sorted output as well (after sorting)


given:
(before)
A:
1 2 3 4 5
5 2 7 2 1
after sorting
Algo1
1 2 3 4 5
1 2 2 5 7
not stable
Algo2
1 2 3 4 5
1 2 2 5 7
Stable

Topic : Introduction to Sorting Algorithms

- Inversion in an Array

Formal Definition

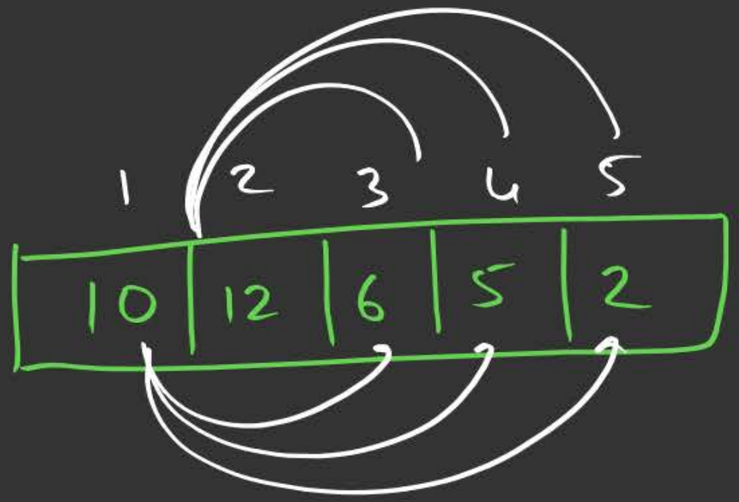
- A pair of indices (i,j) is considered to be an inversion of the given array A [1….n] if i<j and A [i] > A [j]

Note:- i & j need not be adjacent indices (not mandatory)

Eg-

Total number of inversion pairs in A = ?

eg:



(1,3) (1,4) (1,5) ⟶ (3)

(2,3) (2,4) (2,5) ⟶ (3)

(3,4) (3,5) ⟶ (2)

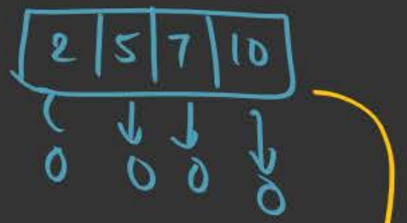
(4,5) ⟶ (1)

Total

3 + 3 + 2 + 1

= (9)

① min inv:

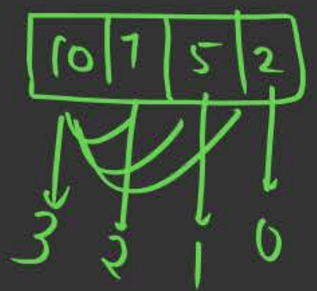


min = 0

asc

② max inv

desc



⇒ 3+2+1+0 = 6

in general

(n-1)+(n-2) - - - -1+0

$$max = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

Topic : Introduction to Sorting Algorithms

Summary

A [n]

$$0 \leq \text{number of inversion} \leq \frac{n(n-1)}{2}$$

Min
(Increase)

Max
(Decrease)

Topic : Introduction to Sorting Algorithms

- Time complexity of any Comparison- based sorting Algorithm
  - Depends upon:
    1. Number of element comparison
    2. number of swaps (number of inversion)

Note: By default → Ascending order of values

Topic : Introduction to Sorting Algorithms

1. Bubble Sort:

(AJ Sir's terminology)

1. Basic Algorithm → Algorithm 1
2. Better Algorithm → Algorithm 2
3. Optimized Bubble sort → Algorithm 3

Logic/Idea: In every i^{th} pass/ iteration, place the ith max element at its correct position.

Topic : Introduction to Sorting Algorithms

- Given: A

1	2	3	4	5	6
75	55	15	10	35	5

pass 1: 55 15 10 35 5 75

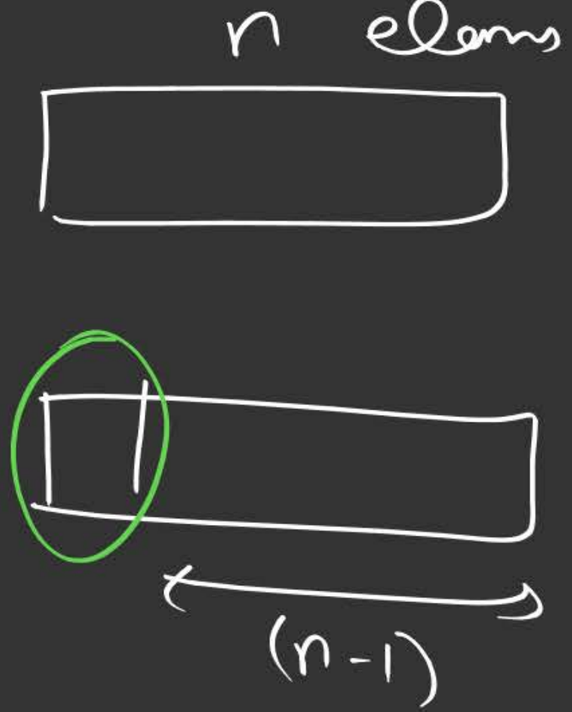
pass 2: 15 10 35 5 55 75

pass 3: 10 15 5 35 55 75

pass 4: 5 10 15 35 55 75

pass 5: 5 10 15 35 55 75

Obs



max (n-1) passes

Topic : Introduction to Sorting Algorithms

- Bubble Sort → Algorithm code:-

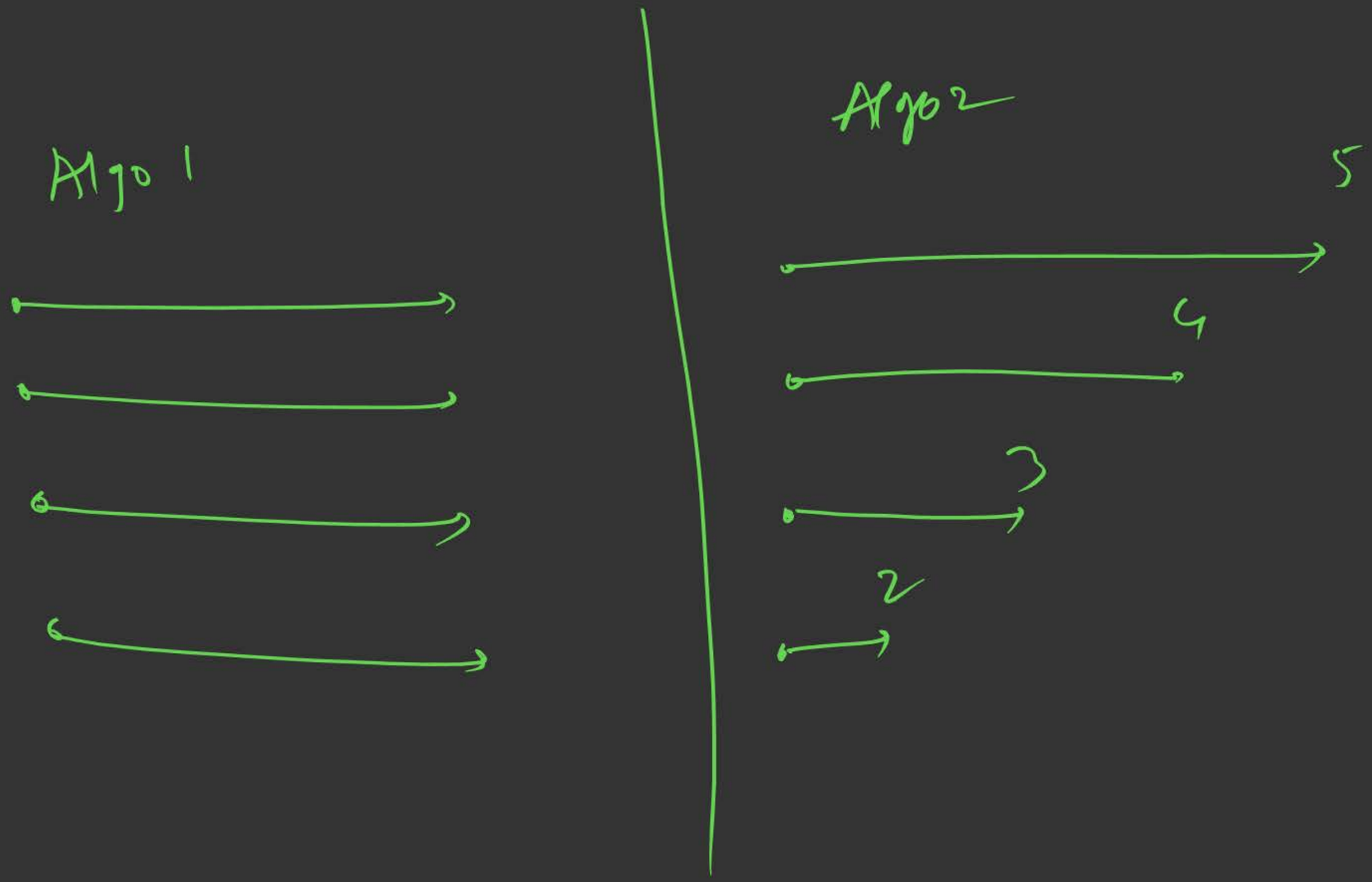
```
Algorithm Bubble sort (A,n)
ε
    For (pass = 1; pass < = (n-1) ; pass + +)
ε
        For(j = 1; j ≤ (n-1) ; j ++)
ε
            If (A [j] > A [j + 1])
ε
                Swap (A [j] , A [j + 1])
}
ε   ε ε ε ε
```

Topic : Introduction to Sorting Algorithms

- Bubble Sort → Algorithm code:- (Algo2)

```
Algorithm Bubble sort (A,n)
ε
    For (pass = 1; pass < = (n-1) ; pass + +)
    ε
        For(j = 1; j ≤ (n-pass) ; j ++)
        ε
            If (A [j] > A [j + 1])
            ε
                Swap (A [j] , A [j + 1])
            }
ε       ε   ε   ε   ε
```

1st → n-1
2nd → n-2
3rd → n-3


Algo 1
Algo 2
5
4
3
2

Time Complexity of Proc Algo (Algo2)

Best Case:

I/p → Sorted

Comparisons

↓

$(n-1) + (n-2) + (n-3) \dots 1$

$= \frac{n(n-1)}{2}$

Swaps = 0

Algo2 : Best Case / Worst Case } $O(n^2)$

Topic : Sorting Algorithms

```
Algorithm Bubble sort (A,n)
{
    For (pass = 1; pass <= (n-1) ; pass + +)
    {
        Flag = 0
        For(j = 1; j ≤ (n-pass) ; j ++)
        {
            If (A [j] > A [j + 1])
            {
                Swap (A [j] , A [j + 1])
                Flag = 1
            }
        }
        If (flag == 0) break ;
        // means these are no increase
    }
}
```

(Algo3)

pass 1 : 3 Comp
0 swap

→ exit

→ Stop

TC of Algo3 (Bubble Sort)

	No. of Comp	No. of Swaps	overall TC
Best Case	$(n-1)$	0	$\Omega(n)$ ✓
Worst Case	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$	$O(n^2)$ ✓

Bubble Sort: 1) Space Complexity = O(1)

↓

(Inplace)

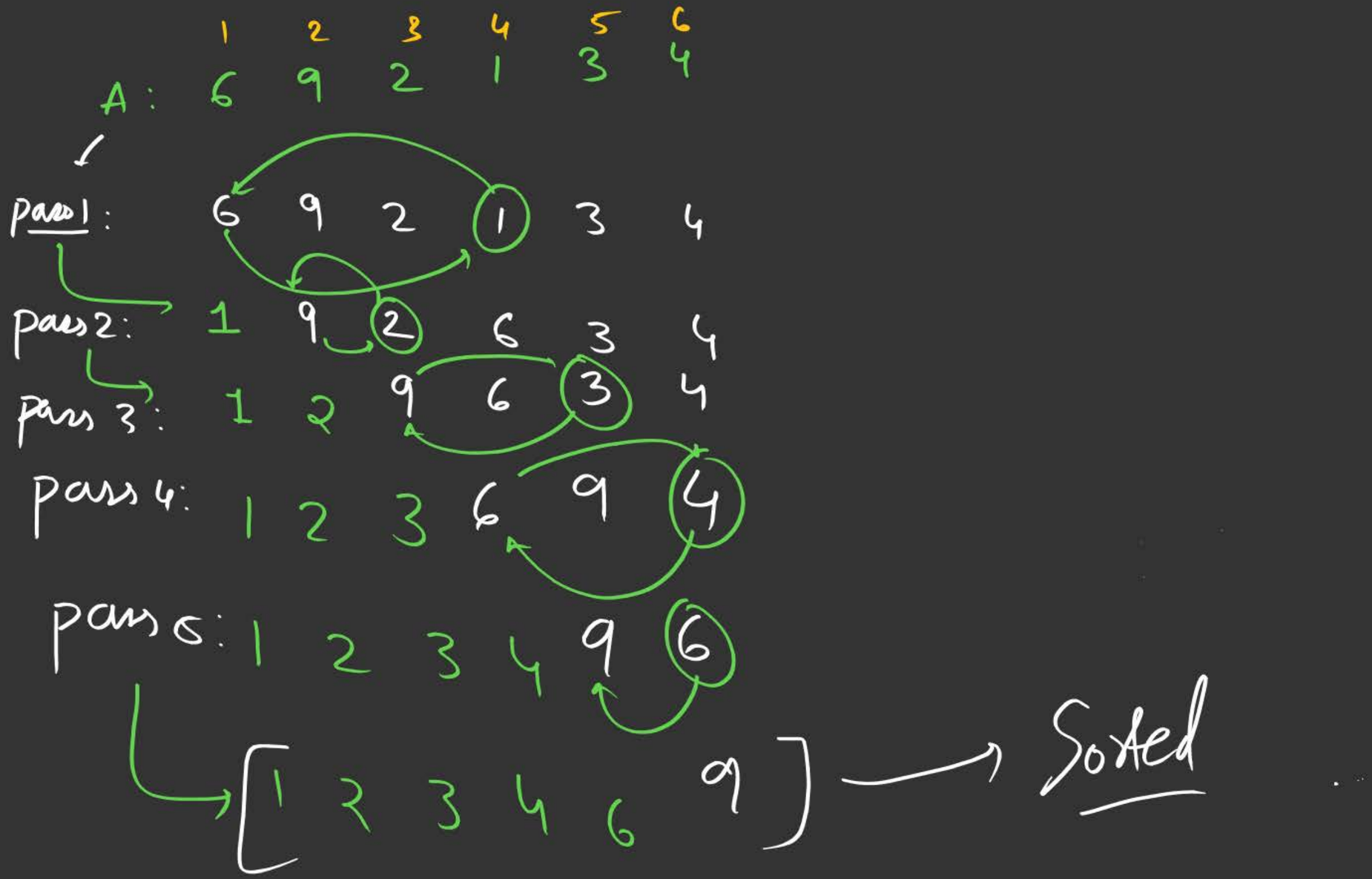
2) Stable

Selection Sort:

Idea:

1. In i^{th} pass, find the position of the 'i' the smallest elements.
2. Swap it with the elements that is present in the i^{th} index.
3. Keep repeating until the array is sorted.

((n-1) passes)


1 2 3 4 5 6
A: 6 9 2 1 3 4
pass 1: 6 9 2 1 3 4
pass 2: 1 9 2 6 3 4
pass 3: 1 2 9 6 3 4
pass 4: 1 2 3 6 9 4
pass 5: 1 2 3 4 9 6
[1 2 3 4 6 9]
Sorted

Topic : Selection Sort

(1-based indexing)

```
Algorithm SelectionSort (A,n)
ε
    For (pass = 1; pass <= (n-1) ; pass ++)        (n-1) passes
        ε  Min-ind = pass
            For(j = pass+1; j ≤ (n-1) ; j ++) //initialize
                ε
                    If (A [j] < A [min-ind])
                    ε
                        [min-ind] = j
ε       ε   ε   ε
    Swap (A [pass] ,A [min-ind])  →placing pass^th min element at its correct
ε                                  position
```

Imp:

1) Always (n-1) passes

2) 1 swap in every pass

3) Total swaps = 1 * (n-1)

= (n-1) swaps

Topic : Sorting Algorithms

- Time complexity Analysis (Selection Sort)

	No. of ~~complexity~~ comparisons	Number of Swaps	Overall TC
Best Case	$\frac{n(n-1)}{2}$	(n-1)	$\Omega(n^2)$
Worst Case	$\frac{n(n-1)}{2}$	(n-1)	$O(n^2)$

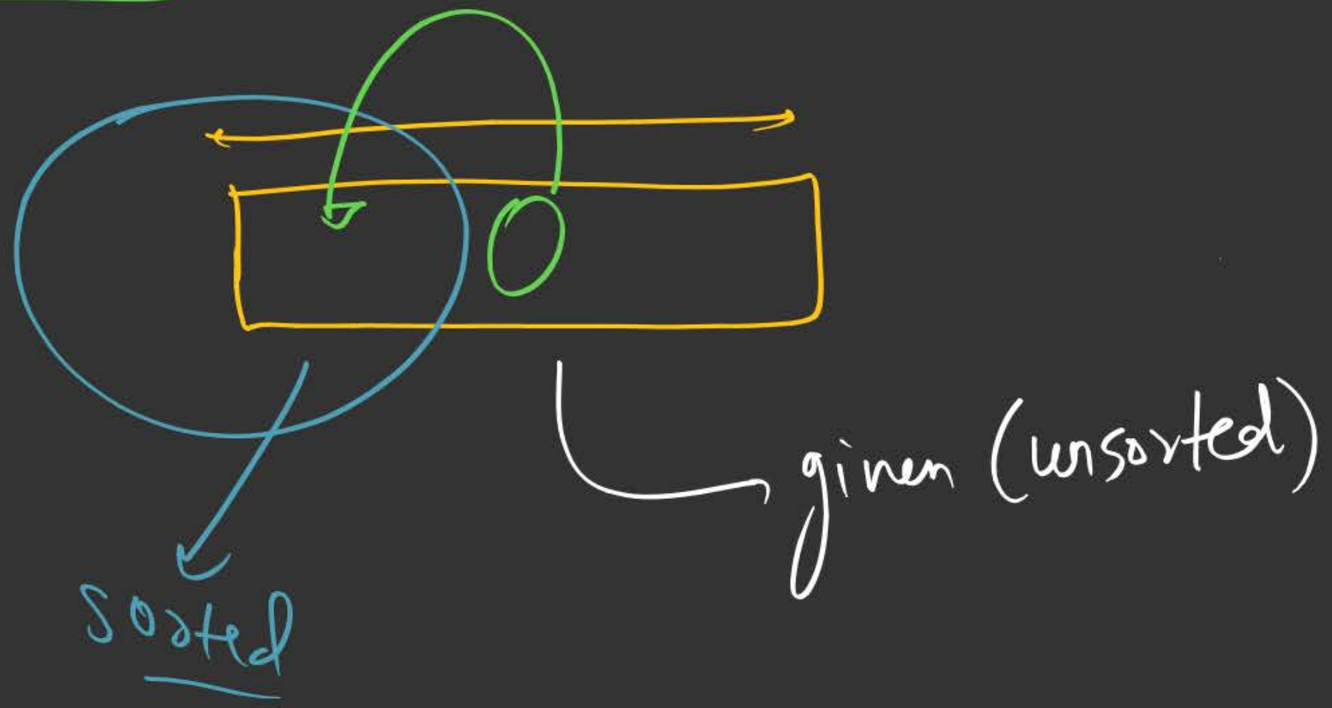
$\rightarrow \underline{\underline{\Theta(n^2)}}$

Topic : Sorting Algorithms

- Imp. Observations:- (Selection Sort)
  1. Space Complexity → O(1) → (in place)
  2. Always takes (n–1) passes (input size : A [n])
  3. Every pass → Exactly 1 Swap
  4. Total Swaps = (n–1) *1 = (n–1) Swaps always
  5. Selection Sort takes the min number of Swaps among all comparison based sorting algorithm in worst case: (n–1) Swaps →O(n).
  6. Unstable and inplace

③ Insertion Sort:

Idea:

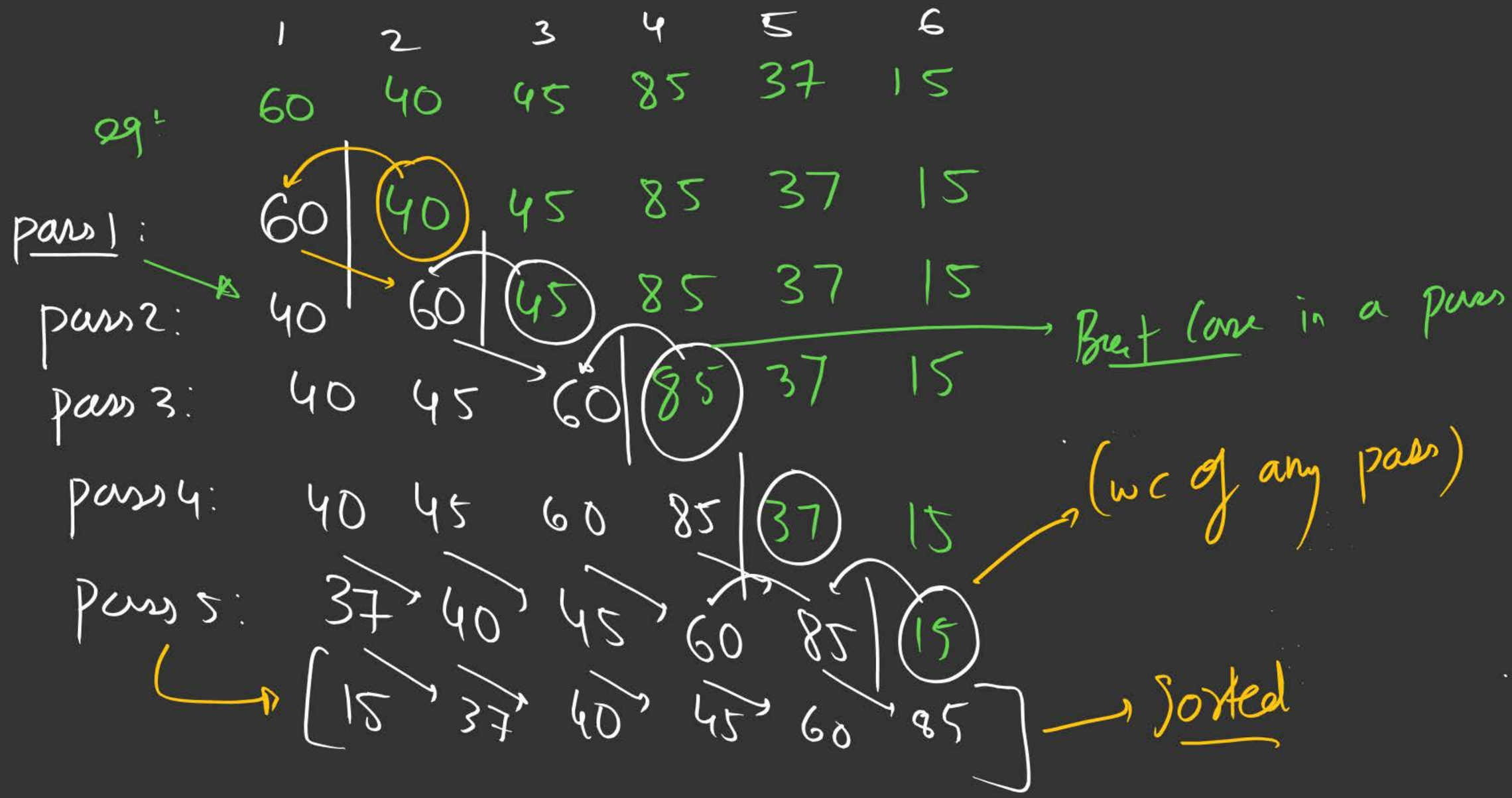


Topic : Insertion sort code

```
Algo InsertionSort(A, n)
{
    For(j = 2; j ≤ n; j++)
    {
        key = A[j]
        i = j – 1
        while (i > 0 and A[i] > key)
        {
            A[i + 1] = A[i]
            i--
        }
    } A[i +1] = key
}
```

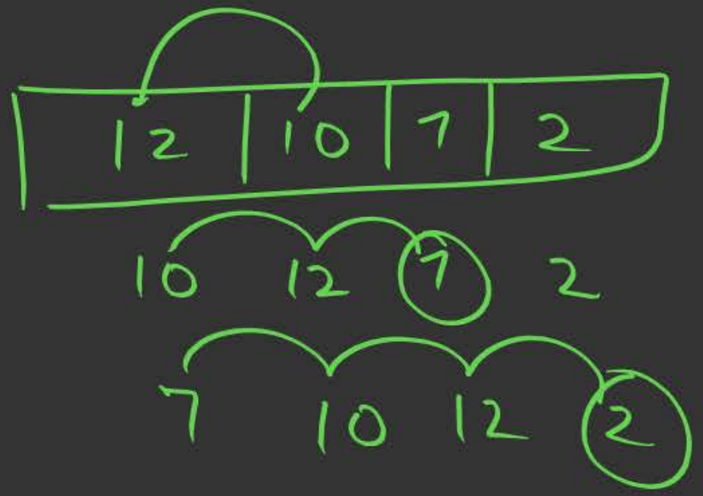
Dry Run of code

eq:- A = [80 20 40 90 30 60]


1 2 3 4 5 6
60 40 45 85 37 15
pass 1:
pass 2:
pass 3:
pass 4:
pass 5:
Best Case in a pass
(wc of any pass)
Sorted

Total (n-1) passes

Worst Case



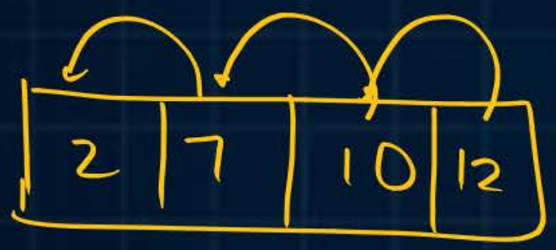
$$\text{Comp} = \frac{n(n-1)}{2}$$

$$\text{Interchanges} = \frac{n(n-1)}{2}$$

Topic : Time Complexity Analysis:

(i) Best Complexity:-

When I/P is already sorted in ascending order.



$\Rightarrow$ $(n-1)$ Comparison

$\rightarrow$ 0 interchange

Topic : Insertion sort:

(1) Always takes (n – 1) passes.

(2) Time complexity:

BC → $\Omega(n)$

WL = $O(n^2)$

} $O(n^2)$

(3) Space complexity → $O(1)$, inplace

(4) Stable Algo

Topic : Insertion sort:

(Bonus)

If the input list is pre-sorted, then it takes time of O(n + d),

Where n = no. of elements

d = no. of inversions

Note:- we use this while solving questions when the actual input sequence order is not known but only info about no. of insertion is mentioned.

Topic : Radix Sort

(Non-Comparison based sorting)

(1) No. of passes

= (no. of digits in the max element of the given input array)

(2) It makes are of buckets / bins concept.

Eg:-

I/P: A: [64 723, 99, 83, 545, 65, 333, 7, 4, 124]

3 passes

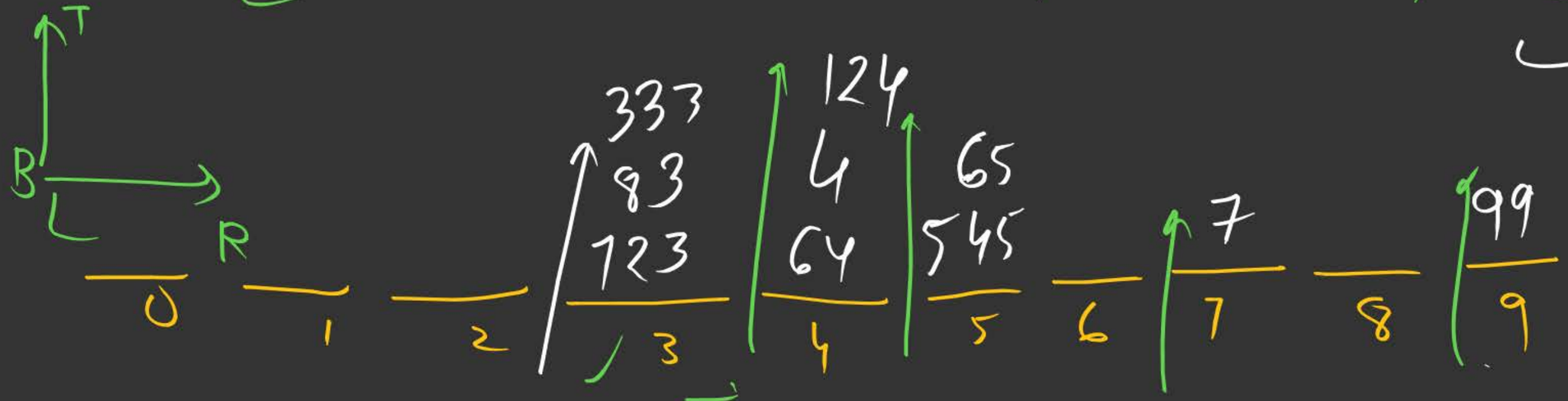
i/p: [64, 723, 99, 83, 545, 65, 333, 7, 4, 124]

4 3 9 3 5 5 3 7 4 4

pass 1 → unit place

pass 1 o/p: [723, 83, 333, 64, 4, 124, 545, 65, 7, 99]

↳ D comp



4 → 04

7 → 07

I/P

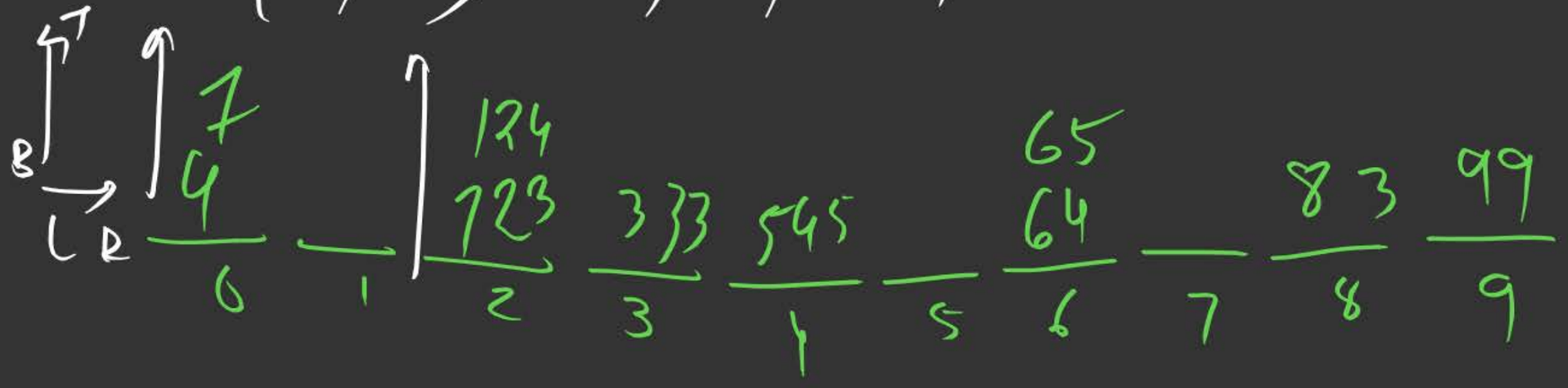
Pass 2 : [723, 83, 333, 64, 4, 124, 545, 65, 7, 99]

2 8 3 6 0 2 4 6 0 9

↳ Bucket as per 10's place digit

Pass 1

O/P : [4, 7, 723, 124, 333, 545, 64, 65, 83, 99] → 0 Comp



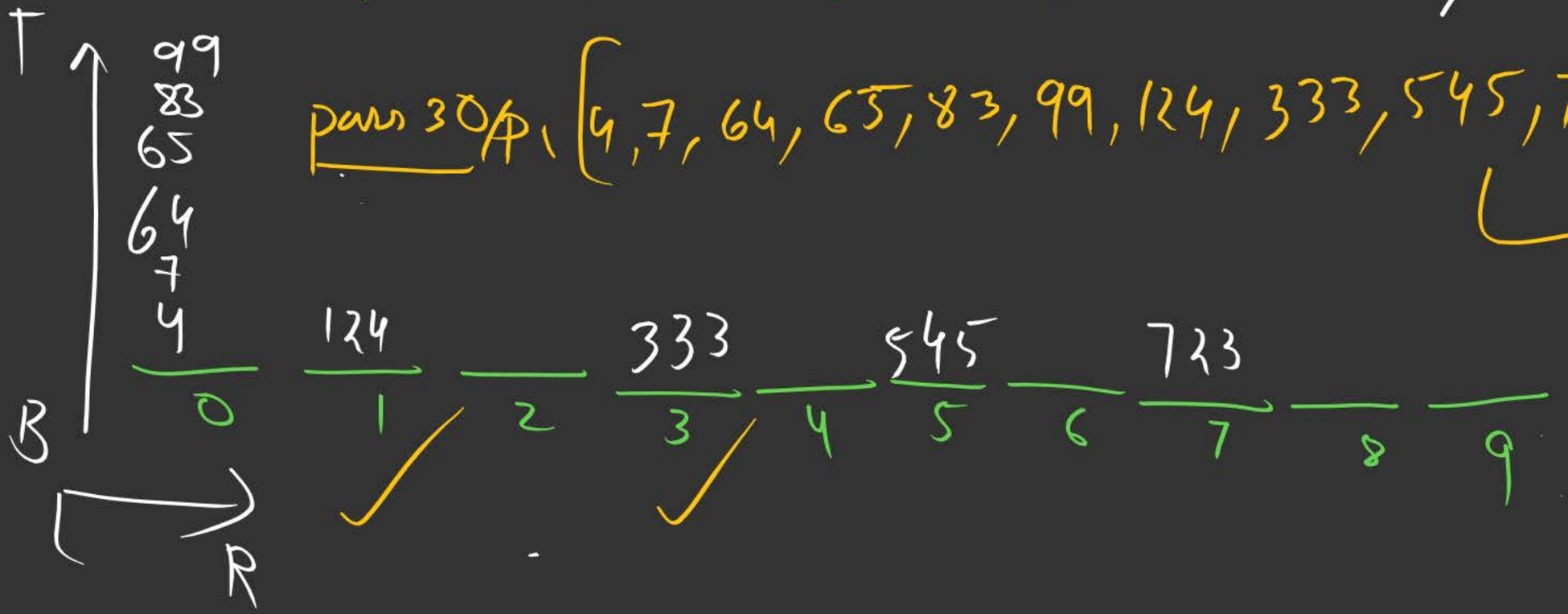
004 → 007 → 64 → 064

i/p pass 3 : [4, 7, 723, 124, 333, 545, 64, 65, 83, 99]

0 0 7 1 3 5 0 0 0 0

Bucket as per 100's place digit (3rd digit from right)

0 Comp



pass 3 o/p : [4, 7, 64, 65, 83, 99, 124, 333, 545, 723]

Sorted

Note:-

(1) Time Complexity:-

$O(d*(n + b))$

Radix sort $\approx O(n * d)$

b → base of the given input (eq: decimals, b = 10)

d → no. of digits in the max element of given i/p Arr

n → no. of elements in given array.

(2) Radix Sort → Not inplace

→ Stable

SC: O(n + k)

n → no. of element

k → largest element in array.

Summary

1) Terminologies

1) Stable
2) Inplace
3) Inversions.

2) Bubble Sort →

3) Selection Sort

4) Insertion Sort

5) Radix Sort.

"Aditya Jain Sir"

Thank

THANK

Keep Hustling!