CS & IT ENGINEERING

Algorithms

Sorting Algorithms

Lecture No.- 03

By- Aditya sir

Recap of Previous Lecture

- Topic
- Topic

Bubble

Selection

Topics to be Covered

- Topic
- Topic
- Topic

Insertion Sort
Radix Sort
PYQs + Practice

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 12,000+ students & working professions in field of Data Science and Analytics
11. Have been mentoring & teaching 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.

Telegram Link for Aditya Jain sir:

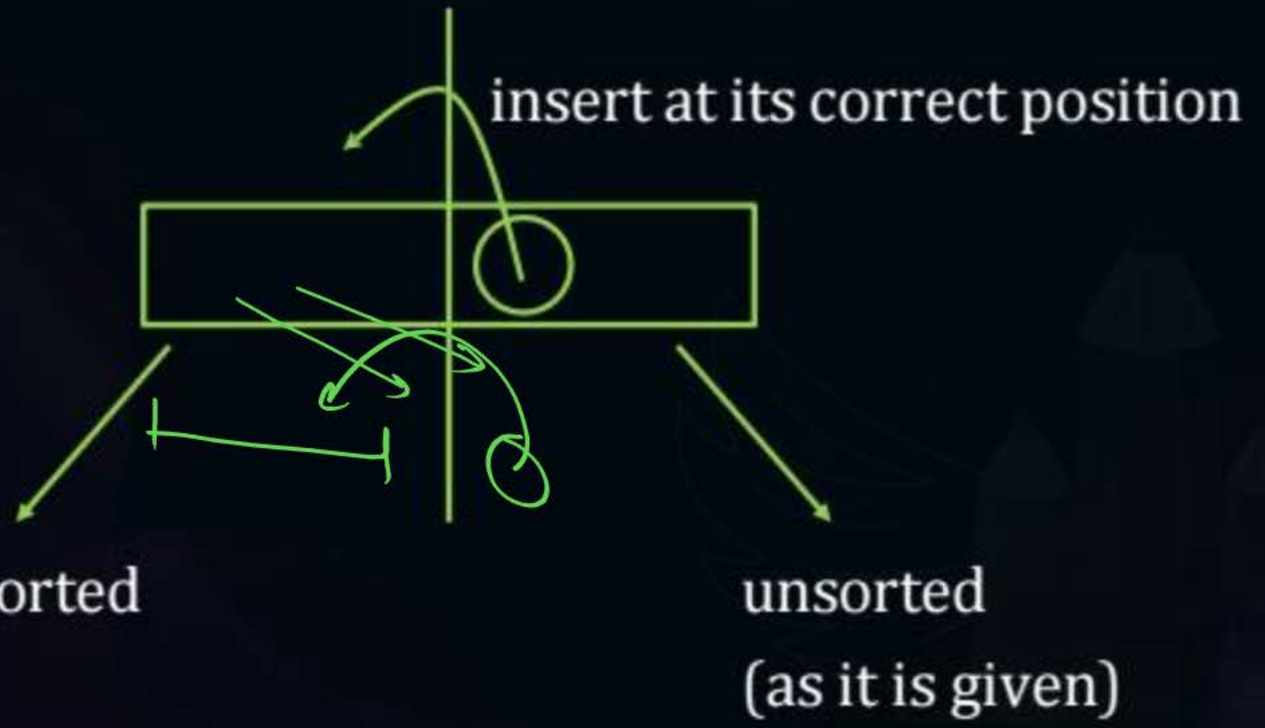
https://t.me/AdityaSir_PW

Topic : Sorting Algorithms

Insertion Sort:

Topic : Sorting Algorithms

Logic:



Topic : Sorting Algorithms

Code Walkthrough:

1	2	3	4	5	6
85	22	49	97	86	15

Pass1: 85 | 22 49 97 86 15

Pass2: 22 85 | 49 97 86 15 ⟶ Pass1 O/P

Pass3: 22 49 85 | 97 86 15

Pass4: 22 49 85 97 | 86 15

Pass5: 22 49 85 86 97 | 15

(15 22 49 85 86 97) Sorted

Topic : Sorting Algorithms

Insertion Sort Code: 1 bonded indexing

```
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

Topic : Sorting Algorithms

Dry-Run of Previous Insertion Sort Code:

i/p : A = 80 |20 40 90 30 60

P1: j = 2;

key = A[j] = A[2] = 20

i = j – 1 = 2 – 1 = 1

while (i > 0 and A[i] > key)

{

A[i + 1] = A[i]

}

A[0 + 1] = 20

Topic : Sorting Algorithms

Dry-Run of Previous Insertion Sort Code:

A = 20 80 | 40 90 30 60

P2: j = 3;

key = A[3] = 40

i = j – 1 = 3 – 1 = 2

A[1 + 1] = 40

Topic : Sorting Algorithms

Dry-Run of Previous Insertion Sort Code:

A = 20 40 80 | 90 30 60

P3: j = 4;

key = A[4] = 90

i = j – 1 = 4 – 1 = 3

A[3 + 1] = key = 90

A[4] = 90

This is not a swap

Topic : Sorting Algorithms

Dry-Run of Previous Insertion Sort Code:

A = 20 40 80 90 | 30 60

P4: j = 5;

key = A[5] = key = 30

i = j – 1 = 5 – 1 = 4

A[1+1] = 30

Topic : Sorting Algorithms

Dry-Run of Previous Insertion Sort Code:

A = 20 30 40 80 90 | 60

P5: j = 6;

key = A[6] = key = 60

~~i = j - 1 = 5 - 1 = 3~~ i = 6 - 1 = 5

A[3+1] = 60

20 30 40 60 80 90 → Sorted

Topic : Sorting Algorithms

Time Complexity Analysis:

Worst Case:

i/p is sorted in opposite order

A = 80, 70, 65, 40, 10

Pass1 → 1 comp, 1 swap

Pass2 → 2 comp, 2 swap

Pass3 → 3 comp, 3 swap

.

.

Pass(n–1) → (n–1) comp, (n–1) swap

Total in all (n–1) passes

No. of comp

$$= 1 + 2 + 3 \ldots (n-1) = \frac{n(n-1)}{2}$$

No. of swaps

$$= 1 + 2 + 3 \ldots (n-1) = \frac{n(n-1)}{2}$$

Topic : Sorting Algorithms

Summary:

	No. of Comparisons	No. of Swaps (inversions)
Best case increasing order	$(n-1)$	0
Worst Case Descending order	$\frac{n(n-1)}{2}$	$\frac{n(n-1)}{2}$

$\Omega(n)$

$O(n^2)$

Space Complexity → $O(1)$ → Inplace

Topic : Sorting Algorithms

Important Points on Insertion Sort:

1. Always takes (n – 1) passes
2. Time complexity

   BC → $\Omega(n)$

   WC → $O(n^2)$
3. Space Complexity → $O(1)$ → Inplace
4. Also stable
5. If the input list is pre-sorted, then it takes time of $O(n + d)$,

   where, n = no. of elements

   d = no. of inversions

Topic : Sorting Algorithms

Note:

We use this while solving questions when the input sequence/order is not known and only no. of inversions are mentioned.

$TC = O(n + d)$

Topic : Sorting Algorithms

Radix Sort (Non-Comparison based)

- No. of passes = no. of digits in the max element of given input
- Makes use of Buckets/Bins

Topic : Sorting Algorithms

Example: Base = 10 (decimal)

i/p: A = 64, 723, 99, 83, 545, 65, 333, 7, 4, 124

Input : {L → R}

o/p : Buckets {L → R, Bottom to Top}

max (A) = 723

no. of digits (723) = 3 → means 3 passes

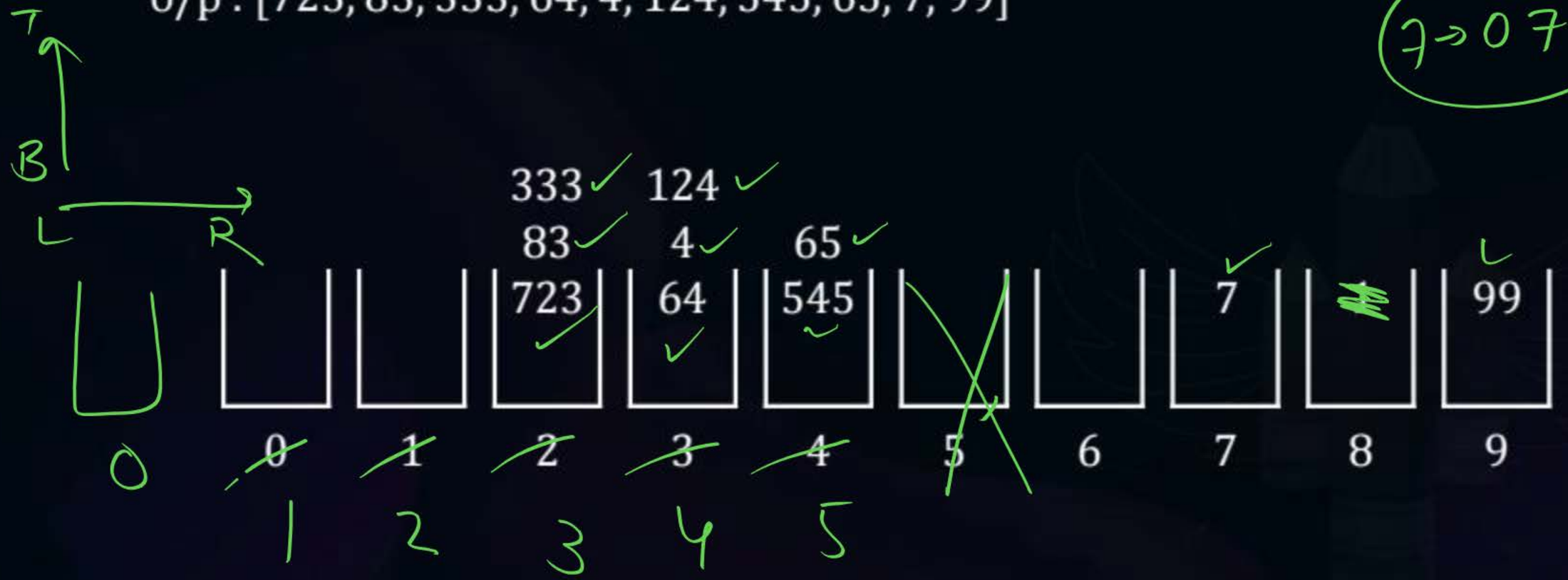
Topic : Sorting Algorithms

Pass1: Bucket as per unit place digit (Right most digit)

64, 723, 99, 83, 545, 65, 333, 7, 4, 124

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

7 → 07



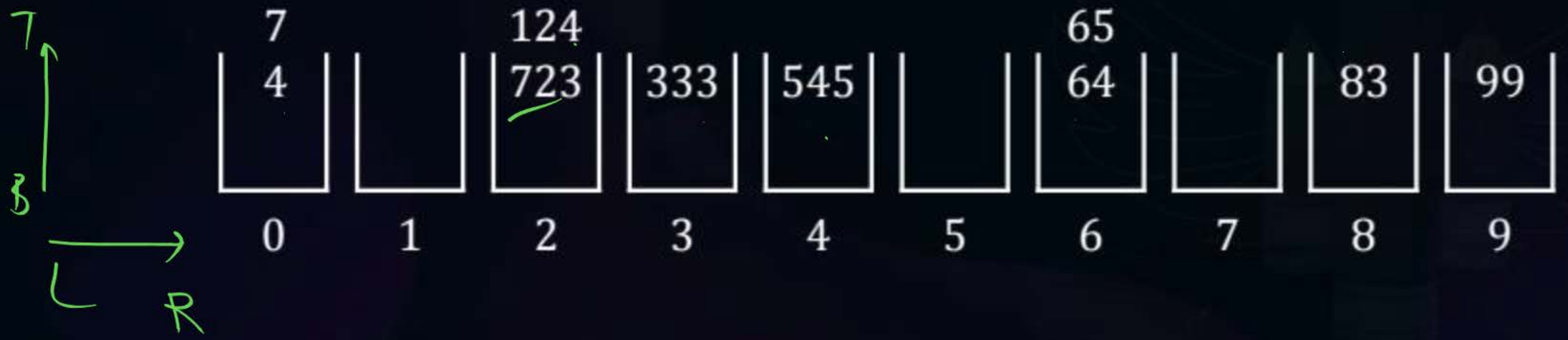
Topic : Sorting Algorithms

Pass2:

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

2 8 3 6 0 2 4 6 0 9

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



Topic : Sorting Algorithms

Pass3:

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

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

Final sorted o/p with total 0 comparisons.

Topic : Sorting Algorithms

Note:

Time Complexity: $O(d * (n+b)) \approx O(n * d)$

b = base of given input

d = no. of digits in max elements of given i/p array

n = no. of elements in given i/p array

Radix Sort
- → Not inplace
- → Stable

Topic : Bubble Sort

#Q. What is the o/p after 3rd pass of Radix Sort?

A = [7, 900, 352, 793, 4217, 529, 30, 5]

Topic : Bubble Sort

$(n^{k/2}, n^k]$

#Q. If we use Radix Sort to sort n integers in the range ~~$(n^{k12}, n^k]$~~, for some $k > 0$ which is independent of n, the time taken would be:

A $\Theta(n)$

B $\Theta(kn)$

C $\Theta(n \log n)$

D $\Theta(n^2)$

TC : $O(n * d)$

1024 $\longrightarrow$ $d = \lfloor \log_{10} n \rfloor + 1$

$$d = \log_{10}(n^k) = \lceil k * \log n \rceil$$

$$TC: O(n*d) = (n * k \log n) = \underline{O(n \log n)}$$

Topic : Bubble Sort

#Q. Give the correct matching for the following pairs :

List-I	List-II
A. $O(\log n)$	P. Selection Sort
B. $O(n)$	Q. Insertion sort
C. $O(n \log n)$	R. Binary search
D. $O(n^2)$	S. Merge sort

Codes:

A – R B – P C – Q D – S

B – R B – P C – S D – Q

C – P B – R C – S D – Q

D – P B – S C – R D – Q

A-R, B-Q, C-S, D-P

A-P, B-R, C-S, D-Q

Topic : Bubble Sort

#Q. If all permutations are equally likely, what is the expected number of inversions in a randomly chosen permutation of 1.....n?.

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

B $\frac{n(n-1)}{4}$

C $\frac{n(n+1)}{4}$

D $2n[\log_2 n]$

n elem $\rightarrow$ ${}^{n}C_2$ pairs $= \frac{n(n-1)}{2}$

Prob(pair = 'inversion') $= 1/2$

Overall ans = no. of pairs * Prob

$= \frac{n(n-1)}{2} * \frac{1}{2} = \boxed{\frac{n(n-1)}{4}}$

Topic : Bubble Sort

#Q. What would be the worst case time complexity of the insertion sort algorithm, if the inputs are restricted to permutation of 1... n with at most n inversions ?

A $\Theta(n^2)$

B $\Theta(n \log n)$

C $\Theta(n^{1.5})$

D $\Theta(n)$

41.67%

TC : $O(n+d)$

$d = O(n)$

TC : $O(n+n) = O(n)$

Topic : Bubble Sort

#Q. Assume that the algorithm considered here sort the input sequence in ascending order. If the input is already in ascending order, which of the following are TRUE ?

I. Quick sort runs in $\Theta(n^2)$ time.

II. Bubble sort runs in $\Theta(n^2)$ time.

III. Merge sort runs in $\Theta(n)$ time.

IV. Insertion sort runs in $\Theta(n)$ time.

A. I and II only

B. I and III only

C. II and IV only

D. I and IV only

D

Topic : Bubble Sort

HW

#Q. Consider the following array :

23	32	45	69	72	73	89	97

Which algorithm out of the following options uses the least number of comparisons (among the array elements) to sort the above array in ascending order ?

A Quick sort using the last elements as pivot

B Selection sort

C Merge sort

D Insertion sort

Topic : Bubble Sort

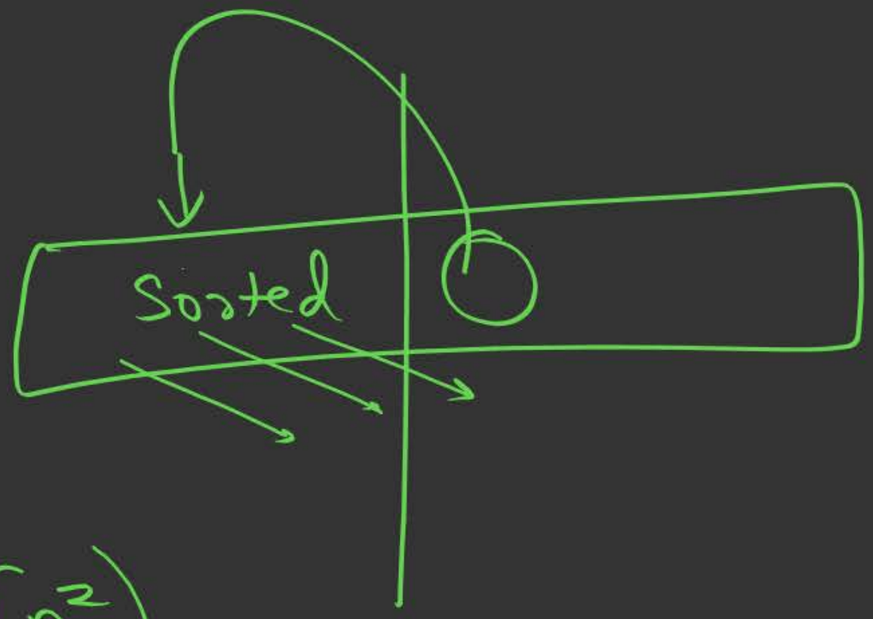
#Q. The usual $\Theta(n^2)$ implementation of Insertion Sort to sort an array uses linear search to identify the position where an element is to be inserted into the already sorted part of the array. If instead, we use binary search to identify the position, the worst case running time will.

A Remains $\Theta(n^2)$

B Become $\Theta(n (\log n)^2)$

C Become $\Theta(n (\log n)$

D Become $\Theta(n)$



w.c: $O(n^2)$

nlogn

2 mins Summary

- Topic
- Topic
- Topic
- Topic
- Topic

Insertion

Radix

THANK - YOU