CS & IT ENGINEERING

Algorithms

Sorting Algorithms

Lecture No.- 02

By- Aditya sir

Recap of Previous Lecture

Topic

Topic

Terminologies

Bubble Sort

Topics to be Covered

- Topic
- Topic — Bubble Sort
- Topic — Selection Sort
- Insertion Sort

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

Telegram Link for Aditya Jain sir:

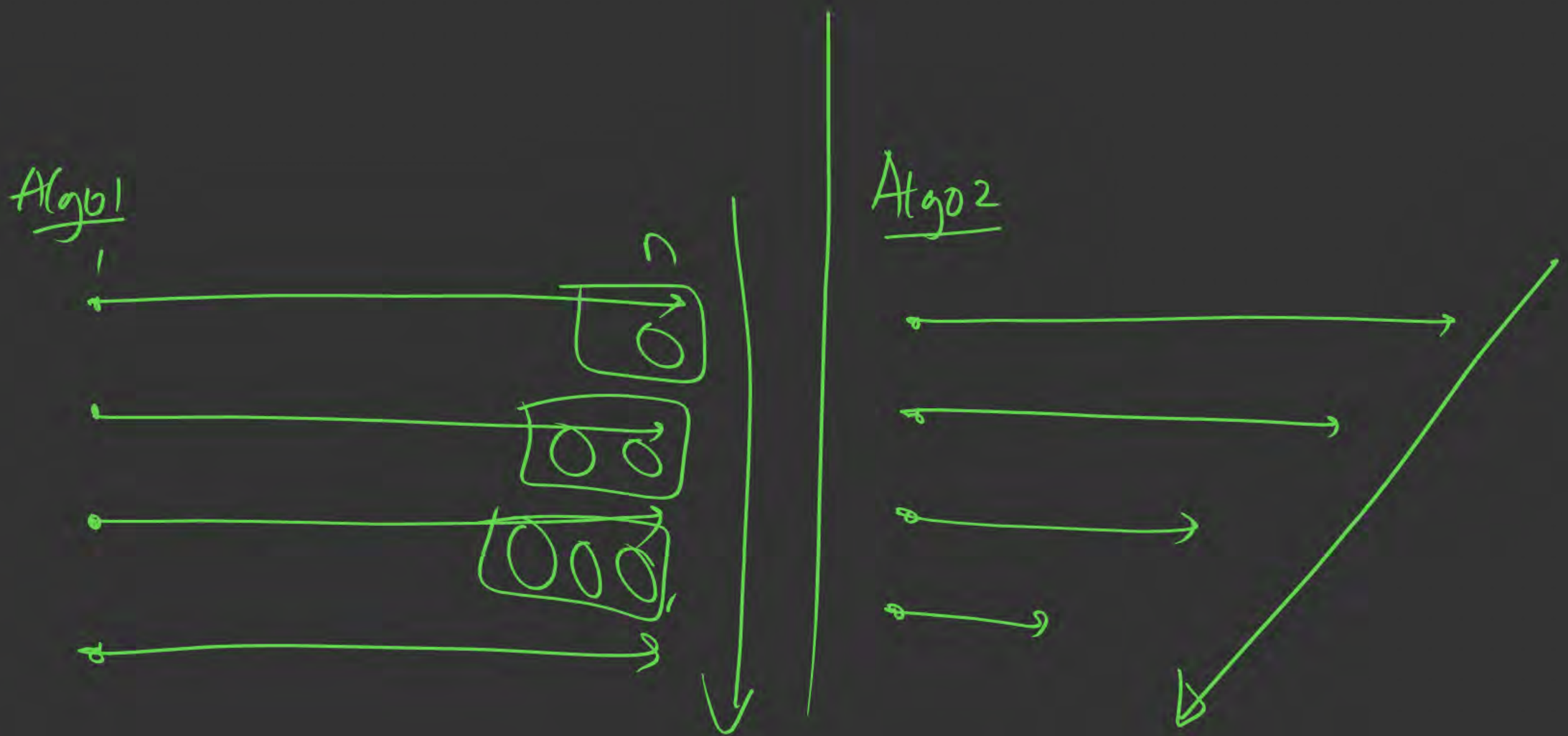
https://t.me/AdityaSir_PW

Topic : Sorting Algorithms

Algo 1: 1-based indexing

```
Algo BubbleSort(A, n)
{
    for (pass = 1; pass <= (n–1); pass++)
    {
        for (j = 1; j <= (n–1); j++)
        {
            if (A[j] > A[j+1])
            {
                swap(A[j], A[j+1])
            }
        }
    }
}
```

$\theta(n^2)$


Algo1
Algo2

Topic : Sorting Algorithms

Algo 2: Better version of Bubble sort (Algo 1)

```
Algo BubbleSort(A, n)
{
    for (pass = 1; pass <= (n–1); pass++)
    {
        for (j = 1; j <= (n–pass); j++)
        {
            if (A[j] > A[j+1])
            {
                swap(A[j], A[j+1])
            }
        }
    }
}
```

Topic : Sorting Algorithms

Time Complexity analysis of Algo1 & Algo 2:

Algo 1:

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

$\Theta(n^2)$

Topic : Sorting Algorithms

Time Complexity analysis of Algo1 & Algo 2:

Algo 2:

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

$(n-1)+(n-2)\ldots+1$

$\Theta(n^2)$

Topic : Sorting Algorithms

We need to reduce the unnecessary comparisons in Best Case.

Topic : Sorting Algorithms

Issue with Algo1 and Algo2:

A:	10	20	25	40

Swaps → 0

Topic : Sorting Algorithms

Optimized Bubble Sort:

- If there are no swaps in any of the pass, then the array has got sorted already.

Topic : Sorting Algorithms

Algo 3: Optimized Bubble sort

```
Algo BubbleSort(A, n)
{
    for (pass = 1; pass <= (n–1); pass++)
    {
        for (j = 1; j <= (n–pass); j++)
        {
            if (A[j] > A[j+1])
            {
                swap(A[j], A[j+1])
                flag = 1                // inversion was present
            }
        }
        if (flag == 0) { break; } // means there are not any inversions
    }
}
```

flag = 0

SC: O(1)

inplace and stable

Topic : Sorting Algorithms

Time Complexity analysis of Algo3 Bubble Sort:

Best Case: 1 pass, 0 swaps, (n–1) comparisons

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

$\Omega(n)$

$O(n^2)$

eg:- A : [100, 85, 10, 25, 45, 50]

Algo1 | Algo2 | Algo3 ✓

Topic : Sorting Algorithms

Code Walkthrough (Algo2 & Algo3):

100	85	10	25	45	50

100	85	10	25	45	50

Algo2: 5 passes

Pass1: 85 10 25 45 50 100

Pass2: 10 25 45 50 85 100

Pass3: 10 25 45 50 85 100

Pass4: 10 25 45 50 85 100

Pass5: 10 25 45 50 85 100

Algo3: 3 passes

Pass1: 85 10 25 45 50 100

Pass2: 10 25 45 50 85 100 (Flag=1)

Pass3: 10 25 45 50 85 100 (Flag=0)

Topic : Sorting Algorithms

Selection Sort:

Steps:

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

Topic : Sorting Algorithms

Code Walkthrough:

A :

60	90	15	10	30	45

Pass1: 10 | 90 15 60 30 45

Pass2: 10 15 | 90 60 30 45

Pass3: 10 15 30 | 60 90 45

Pass4: 10 15 30 45 | 90 60

Pass5: 10 15 30 45 60 | 90

Topic : Sorting Algorithms

Important Observation:

After 1^{st} pass → 1^{st} smallest will be placed at 1^{st} position

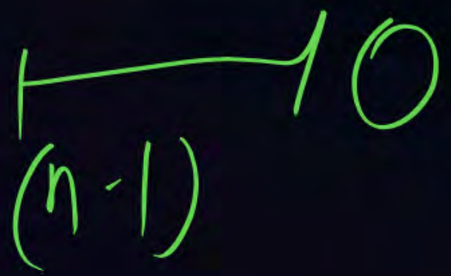
After 2^{nd} pass → 2^{nd} smallest will be placed at 2^{nd} position

.

.

.

After $(n-1)^{th}$ pass → $(n-1)^{th}$ smallest will be placed at $(n-1)^{th}$ position



Topic : Sorting Algorithms

1 based indexing

```
Algo SelectionSort(A, n) {
    for (pass = 1; pass <= (n–1); pass++) {
        min-ind = pass                          // initialize
        for (j = pass+1; j <= n; j++) {
            if (A[j] < A[min-ind]) {
                min-ind = j                     // index of min element found so far
            }
        }
        swap(A[pass], A[min-ind])               // place pass^th min at its correct position
    }
}
```

SC: O(1)

HO

Topic : Sorting Algorithms

Time Complexity analysis of Selection Sort:

	No. of Comparisons	No. of Swaps (~~inversions~~)
Best case	$O(n^2)$	$(n-1)$
Worst Case	$O(n^2)$	$(n-1)$

Best case → $\Omega(n^2)$

Worst Case → $O(n^2)$

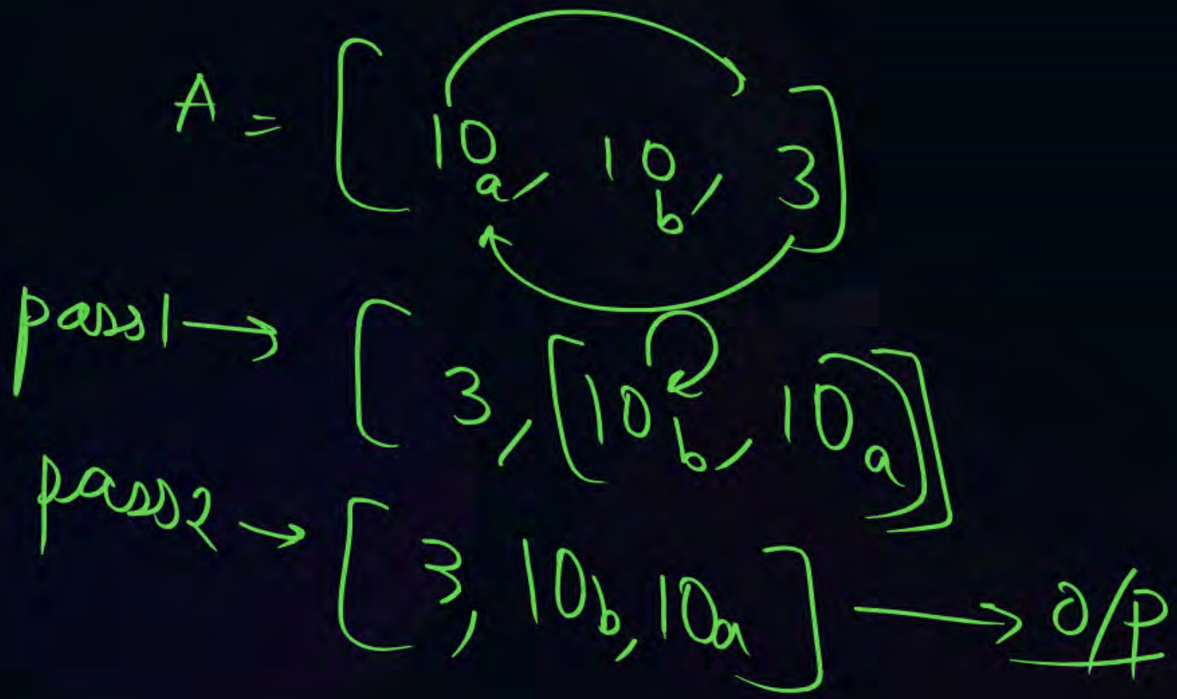
2 5 7 9

$\Theta(n^2)$

Topic : Sorting Algorithms

Space Complexity analysis of Selection Sort:

$O(1) \rightarrow$ Constant $\rightarrow$ {inplace}



Not Stable

Topic : Sorting Algorithms

Important Points/Observations:

1. ~~Stable~~ → Inplace
2. Always taken (n – 1) passes
3. Every pass, exactly 1 swap

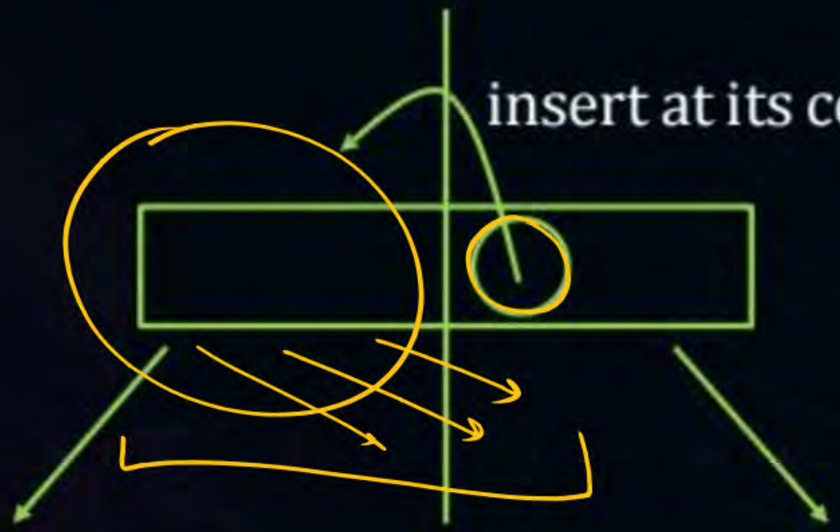
   Total = 1 * (n – 1) = (n – 1) swaps to get sorted
4. Among other algorithms, taken least swaps in worst case → (n – 1) → O(n)

5. Unstable

Topic : Sorting Algorithms

Insertion Sort:

- Insertion is achieved by swapping or updating the adjacent elements.



Topic : Sorting Algorithms

1-based indexing:

H.W

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

2 mins Summary

- Topic
- Topic
- Topic — Bubble Sort
- Topic — Selection Sort
- Topic

THANK - YOU