Sorting – Insertion Sort

1. Introduction

Insertion Sort is a simple and intuitive sorting algorithm that works the way we **sort playing cards in our hands**.

It builds the final sorted array **one element at a time** by taking each element and inserting it into its correct position in the already sorted part.

Insertion Sort is especially efficient for **small datasets and nearly sorted arrays**.

2. What is Insertion Sort?

In Insertion Sort:

- The array is divided into:
  - **Sorted part** (initially contains the first element)
  - **Unsorted part**
- Elements from the unsorted part are picked one by one
- Each picked element is placed at the correct position in the sorted part

3. Why Do We Learn Insertion Sort?

Insertion Sort is important because:

- It is easy to understand and implement
- Efficient for small and partially sorted arrays
- Helps understand shifting and insertion concepts
- Commonly used in real systems for small inputs

4. Basic Idea Behind Insertion Sort

The algorithm works by:

- Picking the next element from the unsorted part
- Comparing it with elements in the sorted part
- Shifting larger elements one position to the right
- Inserting the element at its correct position

---

5. Logic for Insertion Sort (Plain English)

1. Assume the first element is already sorted
2. Pick the next element from the array
3. Compare it with elements in the sorted part
4. Shift all elements greater than it one position to the right
5. Insert the element at its correct position
6. Move to the next element
7. Repeat until the entire array is sorted

---

6. Visualization of Insertion Sort

Consider the array:

```
[8, 3, 5, 2]
```

Step-by-step process:

Initial:

```
[8 | 3, 5, 2]
```

Insert 3:

```
[3, 8 | 5, 2]
```

Insert 5:

```
[3, 5, 8 | 2]
```

Insert 2:

```
[2, 3, 5, 8]
```

Final sorted array:

```
[2, 3, 5, 8]
```

7. Time and Space Complexity

Case	Time Complexity
Best Case (Already Sorted)	$O(n)$
Average Case	$O(n^2)$
Worst Case (Reverse Sorted)	$O(n^2)$

- **Space Complexity:** $O(1)$
  (In-place sorting algorithm)

8. Advantages of Insertion Sort

- Simple and easy to implement
- Efficient for small datasets
- Performs well on nearly sorted arrays
- Stable sorting algorithm
- In-place (no extra memory required)

9. Limitations of Insertion Sort

- Inefficient for large datasets

- Time complexity becomes high for random data
- Slower compared to advanced algorithms like Merge Sort

10. Real-World Applications

- Sorting small lists
- Used internally in hybrid sorting algorithms
- Sorting data as it arrives (online sorting)
- Educational and academic purposes

11. Comparison with Other Sorting Algorithms

Algorithm	**Best Case**	**Worst Case**	**Stability**
Bubble Sort	$O(n)$	$O(n^2)$	Stable
Selection Sort	$O(n^2)$	$O(n^2)$	Not Stable
Insertion Sort	$O(n)$	$O(n^2)$	Stable

12. Summary

- Insertion Sort inserts elements into their correct position
- Builds sorted array gradually
- Efficient for small or nearly sorted arrays
- Time complexity is $O(n^2)$ in worst case
- Commonly used for educational purposes