Sorting – Bubble Sort

1. Introduction

Bubble Sort is one of the **simplest sorting algorithms** used to arrange elements of an array in **ascending or descending order**.

It works by **repeatedly comparing adjacent elements** and swapping them if they are in the wrong order.

Because of its simplicity, Bubble Sort is mainly used for **learning and understanding sorting concepts**, rather than for performance-critical applications.

---

2. What is Bubble Sort?

Bubble Sort gets its name because:

- Larger elements **"bubble up" to the end** of the array after each pass
- Smaller elements move toward the beginning

After each iteration, the **largest unsorted element** is placed in its correct position.

---

3. Why Do We Learn Bubble Sort?

Bubble Sort is important because:

- It is easy to understand and implement
- Helps beginners learn comparison-based sorting
- Builds foundation for advanced sorting algorithms
- Commonly asked in basic DSA exams and interviews

---

4. Basic Idea Behind Bubble Sort

The algorithm works as follows:

- Compare the first two elements

- Swap them if they are in the wrong order
- Move to the next pair
- Repeat this process for the entire array
- Perform multiple passes until the array is sorted

---

5. Logic for Bubble Sort (Plain English)

1. Start from the first element of the array
2. Compare the current element with the next element
3. If the current element is greater, swap them
4. Move to the next pair of elements
5. Repeat this process for the entire array
6. After each pass, the largest element moves to the end
7. Continue passes until no swaps are required

---

6. Visualization of Bubble Sort

Consider the array:

```
[5, 3, 8, 4, 2]
```

Pass 1:

```
[3, 5, 8, 4, 2]
[3, 5, 4, 8, 2]
[3, 5, 4, 2, 8] → Largest element placed at end
```

Pass 2:

```
[3, 4, 5, 2, 8]
[3, 4, 2, 5, 8]
```

Pass 3:

```
[3, 2, 4, 5, 8]
```

Final sorted array:

```
[2, 3, 4, 5, 8]
```

---

7. Optimization in Bubble Sort

An **optimized Bubble Sort** stops early if:

- No swaps occur in a full pass

This means the array is already sorted.

This optimization improves performance for nearly sorted arrays.

---

8. 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)

  (Sorting is done in place)

---

9. Advantages of Bubble Sort

- Simple and easy to implement
- Good for small datasets
- In-place sorting algorithm
- Stable sorting algorithm

---

10. Limitations of Bubble Sort

- Very slow for large datasets
- High time complexity
- Not suitable for real-world applications
- Mostly used for educational purposes

---

11. Real-World Applications

- Teaching sorting concepts
- Small datasets where simplicity is preferred
- Debugging sorting logic
- Introductory algorithm courses

---

12. Comparison with Other Sorting Algorithms

Algorithm	Time Complexity	Efficiency
Bubble Sort	$O(n^2)$	Low
Selection Sort	$O(n^2)$	Low
Insertion Sort	$O(n^2)$	Medium
Merge Sort	O(n log n)	High

---

13. Summary

- Bubble Sort compares adjacent elements
- Largest element moves to the end in each pass
- Easy to understand but inefficient
- Time complexity is $O(n^2)$
- Mainly used for learning purposes
