Bubble Sort

Bubble Sort: Complete Guide for Job Interviews (with Java Code & Questions)

What is Bubble Sort?

Bubble Sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The process is repeated until the list is sorted. It gets its name because the largest elements "bubble up" to the top.

Algorithm Explanation

1. Start from the first element of the array.
2. Compare the current element with the next one.
3. If the current element is greater than the next one, swap them.
4. Move to the next pair of elements and repeat the process.
5. Repeat the above steps for every element in the array.
6. Reduce the number of elements to check after each pass, as the largest elements have already reached their correct position.

Dry Run Example

Input: {64, 34, 25, 12, 22, 11, 90}

Pass 1:

- Compare 64 & 34 → Swap → {34, 64, 25, 12, 22, 11, 90}
- Compare 64 & 25 → Swap → {34, 25, 64, 12, 22, 11, 90}
- Compare 64 & 12 → Swap → {34, 25, 12, 64, 22, 11, 90}
- Compare 64 & 22 → Swap → {34, 25, 12, 22, 64, 11, 90}
- Compare 64 & 11 → Swap → {34, 25, 12, 22, 11, 64, 90}
- Compare 64 & 90 → No Swap
- **Largest element (90) is now in its correct position.**

Pass 2:

- Repeat process, ignoring last sorted element (90).

Final Sorted Array: {11, 12, 22, 25, 34, 64, 90}

Time & Space Complexity

- **Worst-case time complexity:** $O(n^2)$ (when the array is sorted in reverse order)
- **Best-case time complexity:** $O(n)$ (if the array is already sorted)
- **Average-case time complexity:** $O(n^2)$
- **Space Complexity:** $O(1)$ (in-place sorting, no extra space used)

Common Interview Questions on Bubble Sort

1. Explain the working of Bubble Sort.

Answer: Bubble Sort repeatedly compares adjacent elements and swaps them if they are in the wrong order. The largest elements bubble up to the rightmost position, and this process continues until the entire array is sorted.

2. What is the time complexity of Bubble Sort?

Answer:

- Worst-case: $O(n^2)$
- Best-case: $O(n)$ (optimized version)
- Average-case: $O(n^2)$

3. Is Bubble Sort a stable sorting algorithm?

Answer: Yes, Bubble Sort is **stable** because it does not change the relative order of equal elements.

4. Is Bubble Sort an in-place sorting algorithm?

Answer: Yes, Bubble Sort is **in-place** since it does not require additional storage for sorting.

5. How can you optimize Bubble Sort?

Answer: By adding a **swapped flag**, we can stop the algorithm early if no swaps occur in a full pass, reducing the best-case time complexity to $O(n)$.

6. When should you use Bubble Sort?

Answer: Bubble Sort is rarely used in practice because of its poor time complexity. However, it can be used:

1. For small datasets.
2. When the dataset is already nearly sorted.
3. In educational scenarios to teach sorting concepts.

7. What are the disadvantages of Bubble Sort?

Answer:

1. **Inefficient** for large datasets (O(n^2) complexity).
2. **Slow** compared to advanced sorting algorithms like QuickSort and MergeSort.
3. **Unnecessary comparisons** even when the array is already sorted (if not optimized).

8. How does Bubble Sort compare to other sorting algorithms?

Sorting Algorithm	Time Complexity (Worst)	Time Complexity (Best)	Space Complexity	Stable?	In-Place?
Bubble Sort	O(n^2)	O(n) (Optimized)	O(1)	Yes	Yes
Selection Sort	O(n^2)	O(n^2)	O(1)	No	Yes
Insertion Sort	O(n^2)	O(n)	O(1)	Yes	Yes
Merge Sort	O(n log n)	O(n log n)	O(n)	Yes	No
Quick Sort	O(n^2)	O(n log n)	O(log n)	No	Yes

Conclusion

Bubble Sort is an easy-to-understand sorting algorithm but is inefficient for large datasets. While it's useful for learning purposes, real-world applications typically use more efficient algorithms like Merge Sort, Quick Sort, or Heap Sort.

For your **job interview**, be prepared to: Explain the **working** of Bubble Sort.
- Discuss its **time & space complexity**.

- Compare it with other sorting algorithms.
- Explain **why it is not used** in real-world scenarios.
- Write an **optimized** version in **Java**.