

Bubble Sort in DSA

◆ Introduction

Bubble Sort is the simplest sorting algorithm.

- It works by **repeatedly swapping adjacent elements** if they are in the wrong order.
- After each pass, the **largest element “bubbles up”** to its correct position at the end of the array.
- This process continues until the array is sorted.

It is called **Bubble Sort** because larger elements move (or "bubble") to the end step by step.

◆ Working of Bubble Sort

Example: Sort [5, 3, 8, 4, 2] in ascending order

Pass 1: Compare and swap adjacent elements

- (5,3) → swap → [3, 5, 8, 4, 2]
- (5,8) → no swap → [3, 5, 8, 4, 2]
- (8,4) → swap → [3, 5, 4, 8, 2]
- (8,2) → swap → [3, 5, 4, 2, 8]

Largest element (8) moved to last position.

Pass 2: Repeat on first n-1 elements

- $[3, 5, 4, 2, 8] \rightarrow [3, 4, 5, 2, 8] \rightarrow [3, 4, 2, 5, 8] \rightarrow [3, 2, 4, 5, 8]$

Second largest element (5) is now at correct position.

Pass 3:

- $[3, 2, 4, 5, 8] \rightarrow [2, 3, 4, 5, 8]$

Pass 4: Already sorted, no swaps.

Final Sorted Array = $[2, 3, 4, 5, 8]$

◆ Algorithm (Steps)

1. Start from the first element.
2. Compare the current element with the next element.
3. If the current element is greater than the next, swap them.
4. Continue for all elements (1st pass).
5. Repeat passes until no swaps are needed → array is sorted.

www.tpcglobal.in

◆ Pseudocode

```
procedure bubbleSort(arr, n):  
    for i = 0 to n-1:  
        swapped = false  
        for j = 0 to n-i-2:  
            if arr[j] > arr[j+1]:  
                swap(arr[j], arr[j+1])  
                swapped = true  
        if swapped == false:  
            break
```

The **swapped** flag optimizes the algorithm by **stopping early** if the array becomes sorted before completing all passes.

◆ Time Complexity

- **Best Case:** $O(n)$ → When array is already sorted (with swapped flag optimization).
- **Worst Case:** $O(n^2)$ → When array is sorted in reverse order.
- **Average Case:** $O(n^2)$ → Random order input.

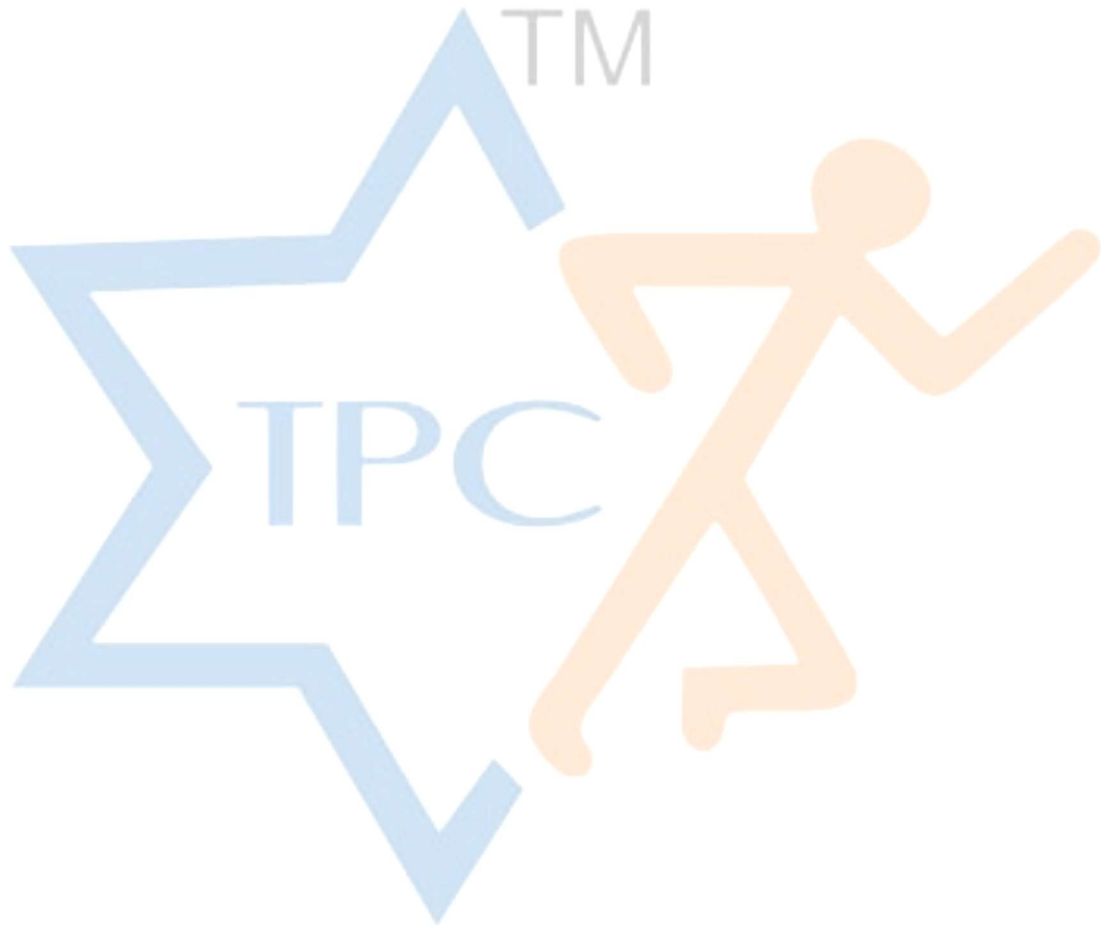
◆ Space Complexity

- $O(1)$ → In-place sorting (no extra memory required).

◆ Characteristics

- **In-place Sorting:** Requires no extra space.

- **Stable Sorting:** Preserves the order of duplicate elements.
- **Not efficient** for large datasets (better sorting algorithms exist like Merge Sort, Quick Sort).



www.tpcglobal.in