Bubble Sort

Bubble Sort is one of the simplest sorting algorithms, and also one of the slowest if the data size grows large.
It repeatedly steps through the list, compares adjacent elements, and swaps them if they’re in the wrong order.
Like bubbles rising to the top, the largest elements “bubble up” to the end of the list after each pass.

Think of it like this:

> You’re arranging books by height on a shelf.
> You start from the left, compare each pair, and swap if the right one’s shorter.
> After one full sweep, the tallest book is at the far right, then you repeat for the rest.

🔹 Algorithmic Steps (in pseudocode / algorithm format)

```
Algorithm: BubbleSort(A, n)

Input: Array A of n elements

Output: Sorted array A in ascending order


1. for i ← 0 to n-2 do

2.     swapped ← false

3.     for j ← 0 to n-2-i do

4.         if A[j] > A[j+1] then

5.             swap A[j] and A[j+1]

6.             swapped ← true

7.     end for

8.     if swapped = false then

9.         break     // No swaps means array is already sorted

10. end for

11. return A
```

Key point:
The inner loop handles pairwise comparison and swapping.
The outer loop runs n-1 times, but it can end early if no swaps happen in a pass; a simple optimization.

Implementation in **C++**

```
#include <iostream>

using namespace std;

void bubbleSort(int arr[], int n) {

    for (int i = 0; i < n - 1; i++) {

        bool swapped = false;

        for (int j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

                // swap elements

                int temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

                swapped = true;

            }

        }

        // If no swaps in this pass, array is sorted

        if (!swapped)

            break;

    }

}

// Just call this over an array and its size “n” in main() function
```

Time Complexity

Case	Comparisons	Swaps	Time
Best (already sorted)	n-1	0	O(n)
Average	$\sim n^2/2$	$\sim n^2/4$	$O(n^2)$
Worst (reverse order)	$\sim n^2/2$	$\sim n^2/2$	$O(n^2)$

Space complexity: O(1), sorting is done in place.
Stable? ✅ Yes, equal elements keep their order.
Adaptive? ✅ With the swapped flag optimization, yes.