

Bubble Sort

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

i = 0	j	0	1	2	3	4	5	6	7
	0	5	3	1	9	8	2	4	7
	1	3	5	1	9	8	2	4	7
	2	3	1	5	9	8	2	4	7
	3	3	1	5	9	8	2	4	7
	4	3	1	5	8	9	2	4	7
	5	3	1	5	8	2	9	4	7
	6	3	1	5	8	2	4	9	7
i = 1	0	3	1	5	8	2	4	7	9
	1	1	3	5	8	2	4	7	
	2	1	3	5	8	2	4	7	
	3	1	3	5	8	2	4	7	
	4	1	3	5	2	8	4	7	
	5	1	3	5	2	4	8	7	
i = 2	0	1	3	5	2	4	7	8	
	1	1	3	5	2	4	7		
	2	1	3	5	2	4	7		
	3	1	3	2	5	4	7		
	4	1	3	2	4	5	7		
i = 3	0	1	3	2	4	5	7		
	1	1	3	2	4	5			
	2	1	2	3	4	5			
	3	1	2	3	4	5			
i = 4	0	1	2	3	4	5			
	1	1	2	3	4				
	2	1	2	3	4				
i = 5	0	1	2	3	4				
	1	1	2	3					
i = 6	0	1	2	3					
	1	2							

- *Worst and Average Case Time Complexity:* $O(n^2)$.

Worst case occurs when array is reverse sorted.

- *Best Case Time Complexity:* $O(n)$, Best case occurs when array is already sorted.
- *Auxiliary Space:* $O(1)$
- *Sorting In Place:* Yes
- *Stable:* Yes

Boundary Cases: Bubble sort takes minimum time (Order of n) when elements are already sorted.

Due to its simplicity, bubble sort is often used to introduce the concept of a sorting algorithm.

Concept

The above function always runs $O(n^2)$ time even if the array is sorted. It can be optimized by stopping the algorithm if inner loop didn't cause any swap.

```
static void bubbleSort(int arr[], int n) {
    int i, j, temp;
    boolean swapped;
    for (i = 0; i < n - 1; i++) {
        swapped = false;
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
                swapped = true;
            }
        }

        // IF no two elements were
        // swapped by inner loop, then break
        if (swapped == false)
            break;
    }
}
```