

Bubble Sort

Bubble Sort is also called Sinking or Exchange Sort.

In every step, adjacent elements are compared and swapped.

The image shows handwritten notes on a dark background. On the left, the steps of Bubble Sort are illustrated with arrays and arrows indicating comparisons and swaps:

- Initial array: { 3, 1, 5, 4, 2 } (An orange arrow points from 3 to 1.)
- Step 1: { 1, 3, 5, 4, 2 } (A purple arrow points from 1 to 3, and an orange arrow points from 5 to 4.)
- Step 2: { 1, 3, 4, 5, 2 } (An orange arrow points from 5 to 4.)
- Step 3: { 1, 3, 4, 2, 5 } (A purple arrow points from 3 to 4, and an orange arrow points from 5 to 2.)
- Step 4: { 1, 3, 2, 4, 5 } (A purple arrow points from 3 to 2, and an orange arrow points from 5 to 4.)
- Step 5: { 1, 2, 3, 4, 5 } (A purple arrow points from 3 to 2.)

On the right, the complexity analysis is written:

- space complexity = $O(1)$
- extra space (like copying the array) is not required
- Time complexity :
- Best case = $O(n)$
- Worst case = $O(n^2)$ (arr is descending)
- number of comparisons
- worst case = $\frac{n(n-1)}{2} = \frac{n^2 - n}{2} = \frac{n^2}{2}$
- In Place Sorting

- Space Complexity = $O(1)$ [In-place Sorting Algorithm]
- Time Complexity :
 - Best Case : $O(n)$
 - Worst Case : $O(n^2)$
- In-place Sorting:
 - A Sorting Algorithm which uses constant Space Complexity i.e. zero Auxiliary Space
- Stable Sorting:
 - A Sorting Algorithm without changing indexes of equal elements while sorting.

```
package com.inclass;

/*
    Bubble sort is also known as Exchange or Sinking Sort.
    Bubble sort is an inplace sorting algorithm
*/
```

```
import java.util.Arrays;

public class BubbleSort {

    public static void main(String[] args) {
        int[] arr = {5, 4, 3, 2, 1};
        bubbleSort(arr);
        System.out.println(Arrays.toString(arr));
    }

    static void bubbleSort(int[] arr) {
        int count = arr.length - 1;
        int check = 0;
        while (count > 0) {
            boolean sorted = false;
            for (int i = 0; i < count; i++) {
                check++;
                if (arr[i] > arr[i + 1]) {
                    int temp = arr[i];
                    arr[i] = arr[i + 1];
                    arr[i + 1] = temp;
                    sorted = true;
                }
            }
            if (!sorted) {
                break;
            }
            count--;
        }
        System.out.println(check);
    }
}
```