**Bubble Sort:**

🡪Also Known as sinking sort and exchange sort.

🡪In this algirtihm we traverse through the array and in each iteration we compare the adjacent two elements and if the left hand side element is greater than right hand side element we will swap the element.

🡪By the time we complete the first traversal the greatest element will be moved to the last index.

🡪Ex: [3,4,5,2,1]

[3,4,5,2,1]🡪compares 3 and 4;

[3,4,5,2,1]🡪compares 4 and 5;

[3,4,5,2,1]🡪compares 5 and 2 and as 5 is greater than 2 both will be swapped. 🡪 [3,4,2,5,1]

[3,4,2,5,1]🡪compares 5 and 1 and as 5 is greater than 1 both will be swapped. 🡪[3,4,2,1,5]

🡪As we can observe the greates element has reached the last index by the first traversal.

🡪So in next traversal we will iterate till [n-2] index only.

🡪Similarly after the second traversal the second greatest element would reach the second last index.

🡪 And in next traversal we need not iterate through the last two elements as they are already sorted.

🡪This process continues until the total array is sorted.

🡪when there is no swapping occurred through the entire traversal that means the array is sorted and that is the end of the program.

**Time complexity:**

Best Case: When the array is sorted in ascending order:o(n)

Worst Case:When the array is sorted in descending order:o(n2);

(N-1)+(N-2)+(N-3)+(N-4)+…

4N-(1+2+3+4+..)

4N-(N\*(N-1)/2)

(8N-N2+N)/2

(9N-N2)/2

(N-N2)🡪Constants are ignored.

N2🡪Less dominating terms are removed.

**Space Complexity:**o(n)🡪Constant. That means no extra space is needed. That’s why these algorithms are also called inplace sorting algorithms.

**Algorithm:**

**package** SortingAlgo;

**import** java.util.Arrays;

**public** **class** BubbleSort {

**public** **static** **void** main(String[] args) {

**int**[] arr= {1,-3,4,-2,0,5};

*bubble*(arr);

System.***out***.println(Arrays.*toString*(arr));

}

**static** **void** bubble(**int**[] arr) {

**boolean** swapped;

//run the loop n-1 times

**for**(**int** i=0;i<arr.length;i++) {

swapped=**false**;

**for**(**int** j=1;j<arr.length-i-1;j++) {

**if**(arr[j]<arr[j-1]) {

**int** temp=arr[j];

arr[j]=arr[j-1];

arr[j-1]=temp;

swapped=**true**;

}

}

**if**(!swapped) {

**break**;

}

}

}

}