Bubble Sort

3, 1, 5, 4, 2

In Bubble sort at every step you are comparing adjacent elements.

3, 1, 5, 4, 2

1, 3, 5, 4, 2

1, 3, 4, 5, 2

1, 3, 4, 2, (5)

Why?
with the first pass through the array, the largest element will be at the end.

1, 3, 4, 2, 5

1, 3, 2, (4), 5

with pass two the 2nd largest element is at the 2nd from the last

1, 3, 2, 4, 5

1, 2, (3) 4, 5

Also known as sinking sort and exchange sort

i j

3, 1, 5, 4, 2

↓ j

1, 3, 5, 4, 2

↓ (j) → internal loop

1, 3, 4, 5, 2

↓ (j)

1, 3, 4, 2, 5

For i=1 // Second pass

1, 3, 4, 2] 5

↓ j

1, 3, 2, 4, 5

j < length - i

<= length - i - 1

For i=2

1, 3, 2] 4, 5

j will check here

Space Complexity = O(1) // constant // no extra space require i.e copying the array etc. not required

aka inplace sorting algorithm

Time Complexity: Best case → O(N) ⇒ Sorted

Worst case → $O(N^2)$ ⇒ Sorted in opposite

No of comparisons

As the size of array is growing, the no of comparisons is also growing.

Best case →

i = 0 ↓

ith pass

j j j j
1, 2, 3, 4, 5

Note → When j never swaps for a value of i, it means array is sorted, hence you can end the program

Best case comparisons → N-1
but constants are ignored

Worst case →

j
0 1 2 3 4
5, 4, 3, 2, 1

j
4, 5, 3, 2, 1

j
4, 3, 5, 2, 1

j
4, 3, 2, 5, 1

4, 3, 2, 1, 5 ⇒ (N-1) swaps

i → 2nd pass

j
4, 3, 2, 1, 5 → Already sorted

i
3, 2, 4, 1

j
3, 2, 1, 4

N-2 times

i = 2 → 3rd pass

3, 2, 1, 4, 5 (j)

N-3 times

2, 3, 1, 4, 5 (j)

2, 1, 3, 4, 5

i = 3 → 4th pass

2, 1, 3, 4, 5 (j)

1, 2, 3, 4, 5

N-4 times

Total comparisons →

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

$= 4N - (1+2+3+4)$

$= 4N - \left(\frac{N*(N+1)}{2}\right)$

$= 4N - \left(\frac{N^2+N}{2}\right) \Rightarrow \frac{8N^2 - N^2 - N}{2}$

$= O\left(\frac{7N - N^2}{2}\right) = O(N^2)$

This is the stable sorting algorithm.

Stable →

(10) (20) (20) (30) (10)

↓ Sort

(10) (10) (20) (20) (30)

In original array blue ball of 10 was before red ball of 10. And in the sorted one, this order is maintained. Hence stable.

Unstable sorted array →

(10) (10) (20) (20) (30)