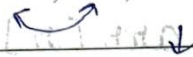


Bubble Sort

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

1, 3, 2, 4, 5

1, 2, 3, 4, 5

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

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 $j < \text{length}-i$
 1, 3, 4, 2, 5 $\leq \text{length}-i-1$
 ↓ j
 1, 3, 2, 4, 5

For $i=2$
 1, 3, 2, 4, 5
 j will
 check here

Space Complexity = $O(1)$ // constant // no extra space require
 a.k.a inplace sorting algorithm i.e. copying the array etc. not required

Time Complexity: Best case $\rightarrow O(N) \Rightarrow$ Sorted
 Worst case $\rightarrow O(N^2) \Rightarrow$ Sorted in opposite
 No. of comparisons

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

Best Case →

$i = 0$

↓

$\overset{0}{1}, \overset{0}{2}, \overset{0}{3}, \overset{0}{4}, \overset{0}{5}$

i th pass

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 →

$\overset{0}{5}, \overset{1}{4}, \overset{2}{3}, \overset{3}{2}, \overset{4}{1}$

$4, 5, \overset{0}{3}, \overset{1}{2}, \overset{2}{1}$

$4, 3, 5, \overset{0}{2}, \overset{1}{1}$

$4, 3, 2, 5, \overset{0}{1}$

$4, 3, 2, 1, 5 \Rightarrow (N-1)$ swaps

1 → 2nd pass

$4, 3, 2, 1, 5 \rightarrow$ Already sorted

$\overset{0}{3}, \overset{1}{2}, \overset{2}{4}, \overset{3}{1}$

$N-2$ times

$3, 2, 4, 1$

$3, 2, 1, 4$

1
 $2 \rightarrow 3^{\text{rd}} \text{ pass}$

3, 2, 1, 4, 5

$N-3$ times

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

1
 $3 \rightarrow 4^{\text{th}} \text{ pass}$

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

$N-4$ times

Total comparisons \rightarrow

$$\begin{aligned} & (N-1) + (N-2) + (N-3) + (N-4) + \dots \\ &= 4N - (1+2+3+4) \\ &= 4N - \left(\frac{N \times (N+1)}{2} \right) \\ &= 4N - \left(\frac{N^2 + N}{2} \right) \Rightarrow \frac{8N^2 - N^2 - N}{2} \\ &= O\left(\frac{7N^2 - N^2}{2}\right) = O(N^2) \end{aligned}$$

This is the stable sorting algorithm.

Stable \rightarrow

(10) (20) (20) (30) (10)
 $\downarrow \text{Sort}$
 (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 \rightarrow

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