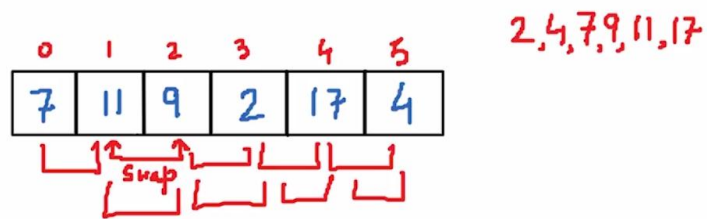
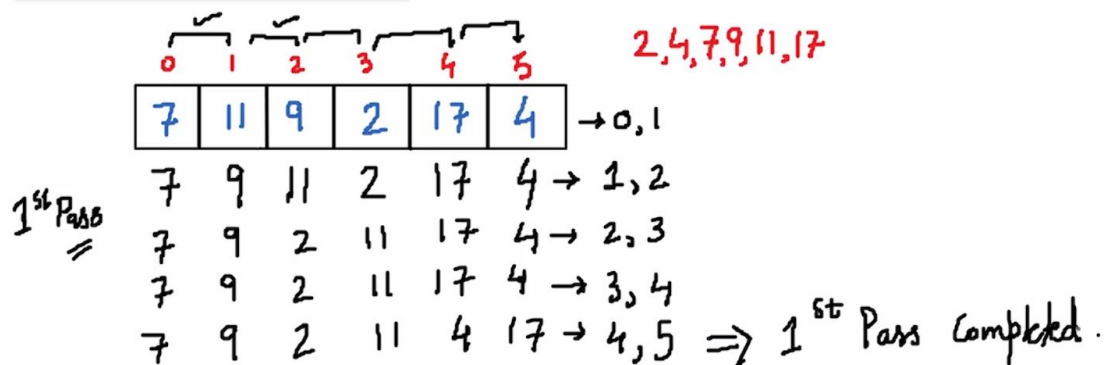


Bubble Sort Explained!



1st pass

Bubble Sort Explained!



By 1st pass, → largest element will be placed at last position. Since last element is sorted so, in next pass we only have to sort (size - 1) elements

2nd pass

Bubble Sort Explained!

0 1 2 3 4 5

7 11 9 2 17 4 → 0, 1

1st Pass

7 9 11 2 17 4 → 1, 2

7 9 2 11 17 4 → 2, 3

7 9 2 11 17 4 → 3, 4

7 9 2 11 4 17 → 4, 5 ⇒ 1st Pass Completed.

2nd Pass

7 9 2 11 4 17 → 0, 1

7 2 9 11 4 17 → 1, 2

7 2 9 11 4 17 → 2, 3

7 2 9 4 11 17 → 3, 4

2, 4, 7, 9, 11, 17

here last 2 elements are sorted so, in next pass we only have to do sorting for (size - 2) elements

3rd pass

1st Pass

7 9 2 11 4 17 → 3, 4

7 9 2 11 4 17 → 4, 5 ⇒ 1st Pass Completed.

2nd Pass

7 9 2 11 4 17 → 0, 1

7 2 9 11 4 17 → 1, 2

7 2 9 11 4 17 → 2, 3

7 2 9 4 11 17 → 3, 4

3rd Pass

2 7 9 4 11 17 → 0, 1

2 7 9 4 11 17 → 1, 2

2 7 4 9 11 17 → 2, 3

and similarly for others

2nd pass → 7 1 2 11 4 17 → 0, 1
 7 2 9 11 4 17 → 1, 2
 7 2 9 11 4 17 → 2, 3
 7 2 9 4 11 17 → 3, 4

3rd pass → 2 7 9 4 11 17 → 0, 1
 2 7 9 4 11 17 → 1, 2
 2 7 4 9 11 17 → 2, 3

4th pass → 2 4 7 9 11 17
 gives

5th pass → 2 4 7 9 11 17
 gives

Notes: for size of array = n

1. No. of passes required = $n - 1$
2. Total no. of comparison / No. of possible swaps = $(n - 1) + (n - 2) + (n - 3) + \dots + 2 + 1 = n * \frac{n-1}{2} = O(n^2) \rightarrow$ time complexity
3. No. of comparison / No. of possible swaps in 1st pass = $n - 1$
4. No. of comparison / No. of possible swaps in 2nd pass = $n - 2$
 ... Similarly, for other passes ...

2nd pass → 7 1 2 11 4 17 → 0, 1
 7 2 9 11 4 17 → 1, 2
 7 2 9 11 4 17 → 2, 3
 7 2 9 4 11 17 → 3, 4

3rd pass → 2 7 9 4 11 17 → 0, 1
 2 7 9 4 11 17 → 1, 2
 2 7 4 9 11 17 → 2, 3

4th pass → 2 4 7 9 11 17
 gives

5th pass → 2 4 7 9 11 17
 gives

→ Sorted Array = ☺

Bubble sort is

1. stable algorithm
2. not a recursive algorithm
3. by default it is not adaptive but it can be made adaptive (in 1st pass if there is no successful swaps)

Total Number Of Comparisons

$$1 + 2 + 3 + 4 + \dots + (n-1) = \frac{n(n-1)}{2} = O(n^2)$$

→ order is same as the order in input array ⇒ Hence stable.

It is k/a Bubble sort because lighter element move to left side and heavier element move to right side similar to stone thrown in water

For already sorted array → time complexity = $O(n)$