Sorting basics

Tuesday, 11 October 2022 9:10 AM

4+ yer — Senior SDE 2 ⇒ Paypal, chennai
↳ Selling
↳ Amazon
↳ ADP

CET_Bhubaneswar ↓ 2018 Btech ⇓ Gold Medalist
+ GATE 2018 ⇓ 485 AIR in General

★ Teaching ✓
↳ Sde, → NS., → Boxes — 2 years

- Full stack, D.S. Algo
- System Design < LLD, HLD
- Complete social fundamentals.

8 days ✓ — Dry Run ⇓ VS Code (Dry Run)

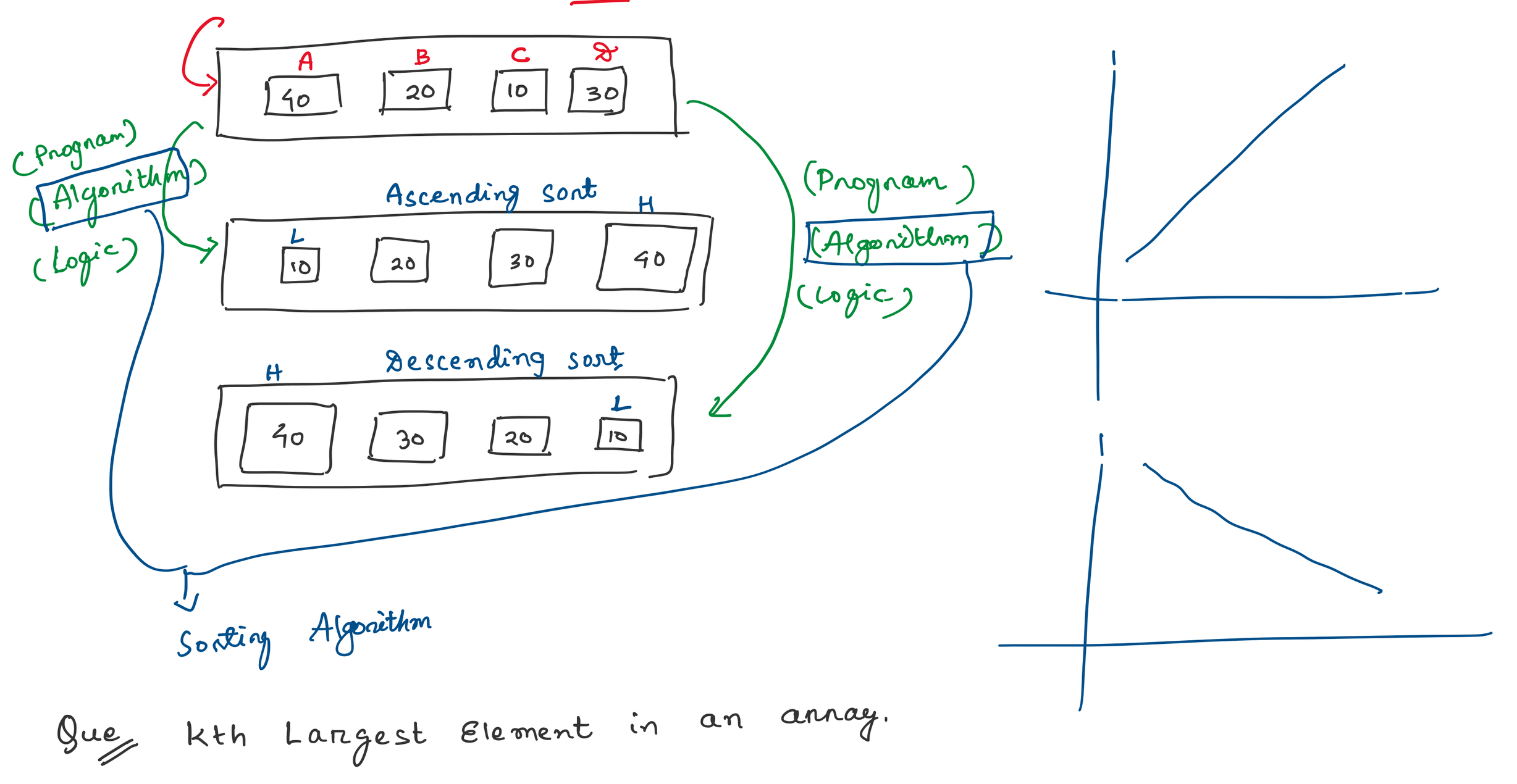
- +1 → Understood
- -1 → Not understood.

- → Attention
- → Response

- Fast ×
- Slow ×

Sorting

⇒ Arranging
⇒ Arrange data in a specific Order — Increasing / Decreasing



Que kth Largest Element in an array.

A = [3, 6, 5, 1] (indices 0 1 2 3) K = 3 → 3rd Largest element

- 1st Largest ⇒ 6
- 2nd Largest ⇒ 5
- 3rd Largest ⇒ 3
- 4th Largest ⇒ 1

n = A.length

Sorting → A = [1, 3, 5, 6] (indices 0 1 2 3) — Magic function

return A[n-K]

- K=2 ; A[4-2] = A[2] = 5
- K=3 ; A[4-3] = A[1] = 3

Sorting Algorithm

Comparison based

→ Comparison is required b/w each elements.

[10, 40, 30, 20, 50, 60]

→ T.C ↑↑
S.C ↓↓

Non-Comparison based

[1, 0, 1, 0, 1, 1, 1, 0, 0, 0]

count0 = ? x
count1 = ? y — Range (0-1)

[x 0's , y 1's]

A = ['A', 'B', 'A', 'C', 'A', 'D', 'E', 'A']

[0-26] ⇒ (A-Z) Range

[A's, B's, C's ...]

[0, 1, 2, 4, 1, 2, 9, 8, 1, 8, 2]
↳ non comparison
Range (0...9)

X ... < 0 < ... Y

→ TC ↓↓
SC ↑↑

Comparison based	Non-Comparison based
→ Bubble sort	→ Counting sort
→ Selection sort	→ Bucket sort
→ Insertion Sort	→ Radix sort
→ Merge Sort	
→ Quick sort	
→ Heap Sort	

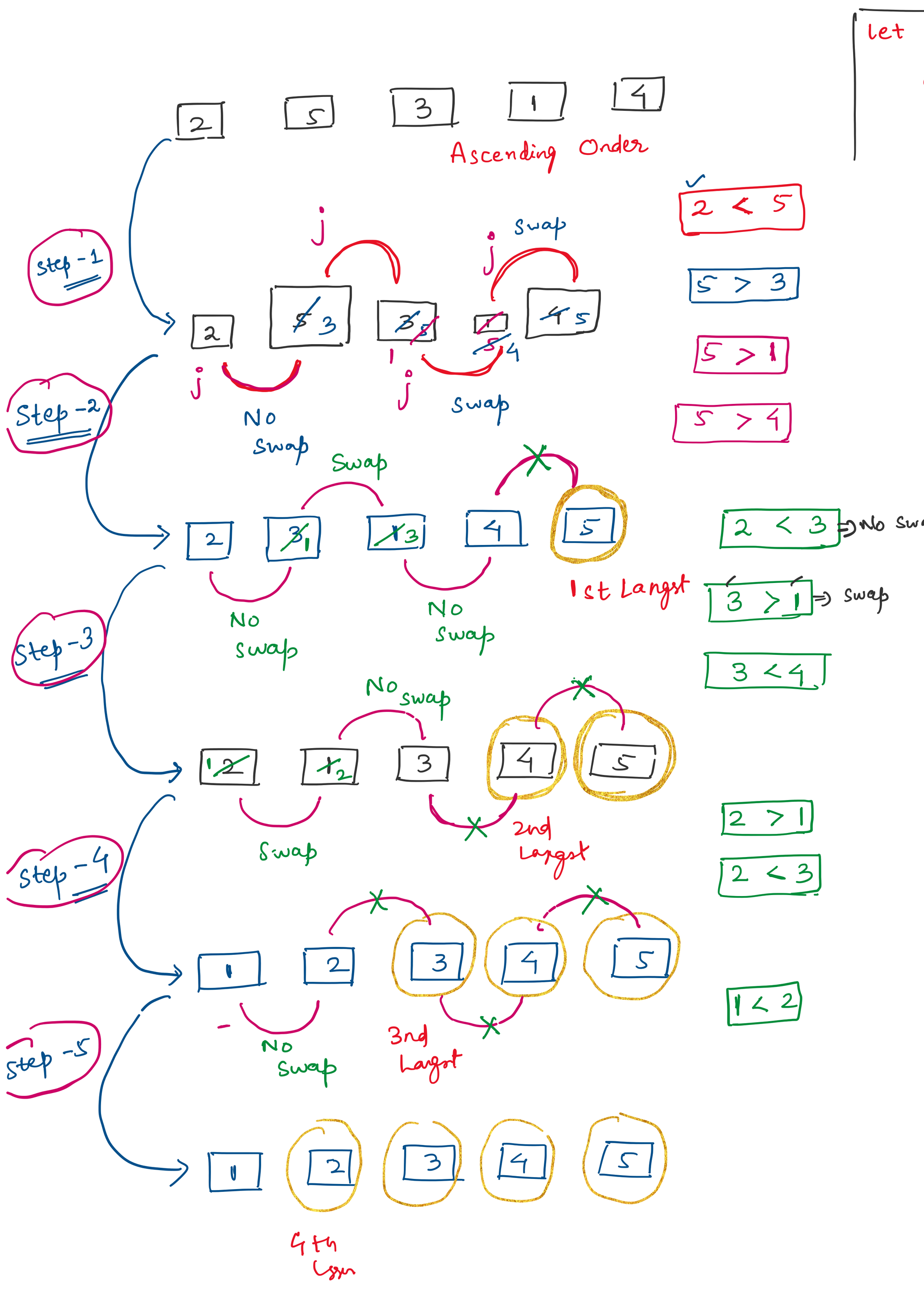
Bubble Sort

→ Compare b/w 2 adjacent elements & swap them until they are in sorted Order.

Swapping

```
let temp = A;
A = B;
B = temp;
```

A: ~~10~~ 20   B: ~~20~~ 10



i=1 — j = 0, 1, 2, — n-1-i = 5-1-1 = ③

indices 0 1 2 3 4

i=0 — j = 0, 1, 2, ③ — 5-1-0 = ④ — j < 4

```
function bubbleSort( A , n) {
  for ( let i=0; i<n-1; i++) {
    let flag = false;
    for( let j=0 ; j < n-i-1 ; j++) {
      if ( A[j] > A[j+1]) {
        flag = true;
        let temp = A[j];
        A[j] = A[j+1];
        A[j+1] = temp;
      }
    }
    if( flag == false) { break; }
  }
}
```

O(n) (outer loop), O(n) (inner loop), O(1) (body)

T.C = O(n * n+1)
T.C. = O(n²)

# of comparisons

- i=0 ; ⇒ j = 0...n-2 = n times
- i=1 ; j = 0... n-3 ⇒ (n-1) times.
- i=2 ; j = 0... n-4 ⇒ (n-2) times
- + ⋮ 1 times.

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

$= O(n^2 + n)$ — n^2 → Dominant

$= O(n^2)$

S.C = O(1) ⇒ No extra space

Worst Case:-

Given Array ⇒ sorted in descending order.
↳ we have to sort it in Ascending Order

[5, 4, 3, 2, 1]

1 iteration ⇒ (n-1) swaps
2nd ⇒ (n-2) swaps

∴ T.C = O(n²)
S.C = O(1)

[1, 2, 3, 5, 4, 7, 6] (indices 0–6) — NS NS NS S NS S

[1, 2, 3, 4, 5, 6, 7] — NS NS NS NS NS

Best Case :-

Sorted in Ascending Order
↳ we have to sort it in Ascending order

⇒ [1, 2, 3, 4, 5] — NS NS NS NS

[1, 2, 3, 4, 5] — NS NS NS

[1, 2, 3, 4, 5] — NS NS

Steps are required ? ×

3rd ✓ NS
4th ×
5th ×
6th ×

∴ T.C = O(n)
S.C = O(1)

Average Case :- Random Order / Arrangement.

T.C = O(n²)
S.C = O(1)