Important Points

- Min time → $O(n)$ (already sorted)
- Max time → $O(n^2)$

1) Bubble Sort → (Adaptive) (Stable) (we made it by using flag)

- Compares consecutive pairs.

{8, 5, 7, 3, 2} 5 elements 4 pairs.

1st Pass

8	5	5	5	5
5	8	7	7	7
7	7	8	3	3
3	3	3	8	2
2	2	2	2	(8)

- Largest element sorted after first Pass ★★★
- n elements (n−1) comparisons.
- 4 swaps possible.

2nd Pass

5	5	5	5	
7	7	3	3	
3	3	7	2	
2	2	2	(7)	
8	8	8	(8)	

- 3 elements are sorted
- 3 comparisons, 3 swaps possible.

3rd Pass

5	3	3
3	5	2
2	2	(5)
7	7	(7)
8	8	(8)

- 3 elements sorted
- 2 comparisons, 2 swaps Possible.

4th Pass

3	(2)
2	(3)
5	(5)
7	(7)
8	(8)

- All elements sorted.
- 1 comparison, 1 swap Possible.

Effect ☆ just like stone (Heavy element) in water. It goes down & bubbles (lighter elements) come up. So Bubble Sort.

Total → 4 Passes N elements (N−1) Passes N=5

Comparisons → $1+2+3+\dots+(n-1) = \frac{n(n-1)}{2} \Rightarrow O(n^2)$ ✓ This is Time Complexity.

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

Important →
- To find largest no. → Perform 1st Pass.
- " " K largest → " K Passes.

Kyuki (n-1) passes honge Total

runs (passes)

-i Kyuki no. of comparisons 0 se to km hote Hai.

swaps

```
for(i=0 ; i<(n-1) ; i++) {
    swap = false;
    for(j=0 ; j<(n-1-i) ; j++) {
        if(a[j]>a[j+1]) {
            swap(a[j] ; a[j+1]);
            swap = true;
        }
    }
    if(swap == false)
        break;
}
```

If no swapping done then list is already sorted so break.