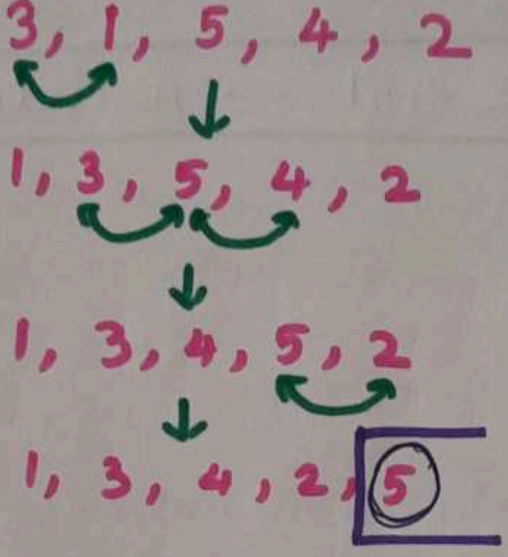
Bubble Sort

Sorting: It is a process of arranging items systematically.

Bubble Sort: It is the simplest algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.

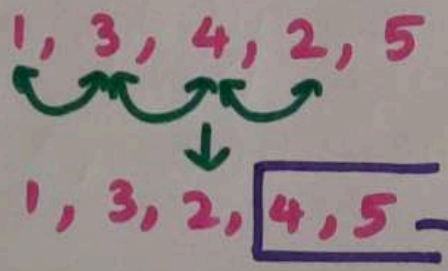
Example:

First pass:



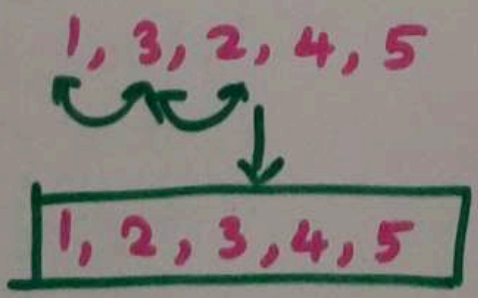
with first pass through the entire array, the largest element (here, 5) came to the end

Second pass:



with second pass second largest element comes at second from last index.

Third pass:

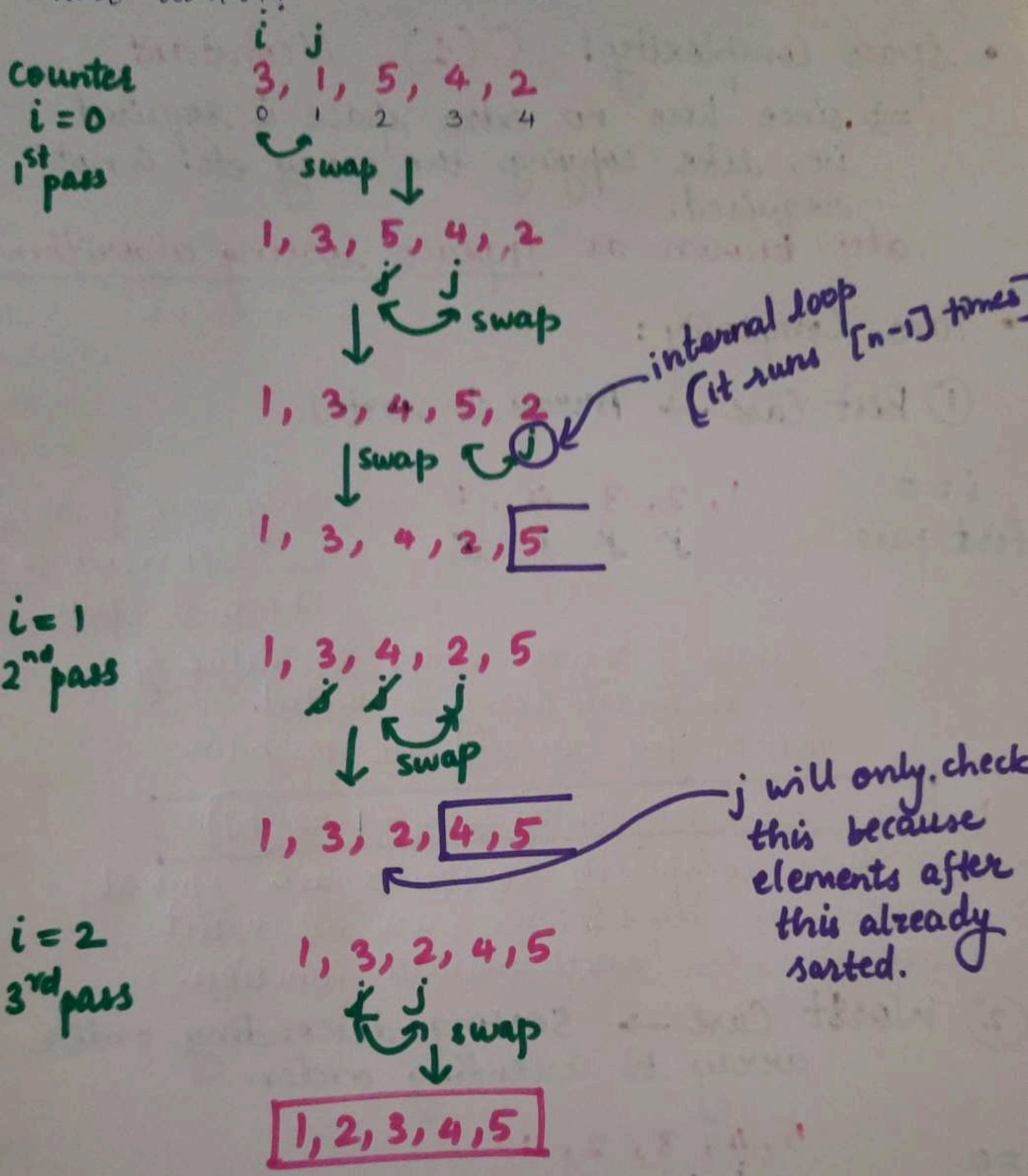


sorted!!

don't need to compare again & again since it is already sorted.

~~Sorth~~ Bubble Sort is also known as Sinking Sort or Exchange Sort.

Let's understand more deeply how actually bubble sort works!!



Complexity :

- Space Complexity: O(1) // constant
  ⇒ since here no extra space is required. i.e., like copying the array etc. is not required.
  also known as inplace sorting algorithm

- Time Complexity :

① Best Case → Array is sorted.

i=0 first pass

1, 2, 3, 4, 5 → only once it ran we don't need to check it again.

NOTE: when j never swaps for value of i, it means array is sorted. Hence, you can end the program.

☆ Best Case comparisons = N − 1 ⇒ (N)

since in time complexity constants are ignored, we don't want exact time, we just want relationship i.e., mathematical function.

② Worst Case → Sorting descending order array to ascending order.

i=0 first pass

5, 4, 3, 2, 1

4, 5, 3, 2, 1

4, 3, 5, 2, 1

4, 3, 2, 5, 1

4, 3, 2, 1, 5 ⟹ (N−1) swaps

i = 1
second pass

4, 3, 2, 1, 5 ← already sorted
j
run loop only for this.

3, 4, 2, 1, 5
j

3, 2, 4, 1, 5
j

3, 2, 4, 1, 5 ⇒ (N-2) swaps

i = 2
third pass

3, 2, 1, 4, 5
j

2, 3, 1, 4, 5
j

2, 1, 3, 4, 5 ⇒ (N-3) swaps

i = 3
fourth pass

2, 1, 3, 4, 5
j

1, 2, 3, 4, 5 ⇒ (N-4) swaps

total comparisons = $N-1+N-2+N-3+N-4$

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

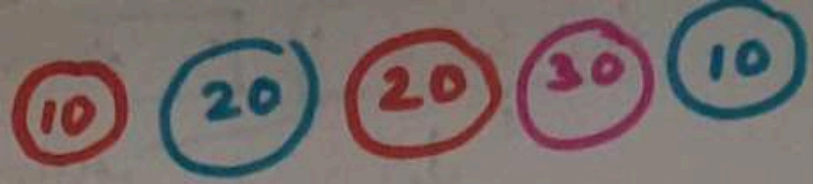
$= 4N - \left[\frac{N \times (N+1)}{2}\right]$

$= 4N - \frac{N^2+N}{2}$

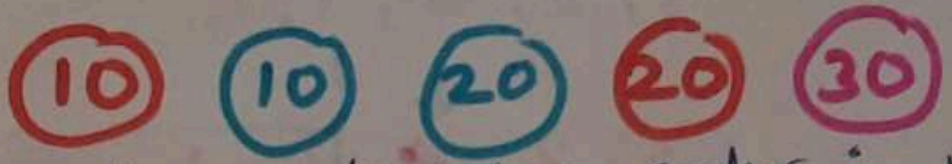
$= O\left(\frac{7N - N^2}{2}\right)$

total comparisons = $O(N^2)$ → In time complexity, constant & less dominating terms are ignored.

Stable Sorting Algo :

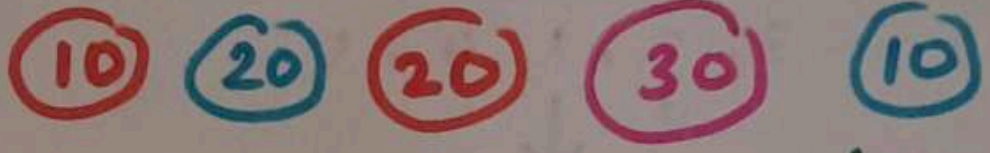


↓ after sorting



⇒ Here after sorting the order is maintained.
⇒ Order is same when value is same

Unstable Sorting Algo :



↓ after sorting