Bubble Sort

09 September 2024 15:20

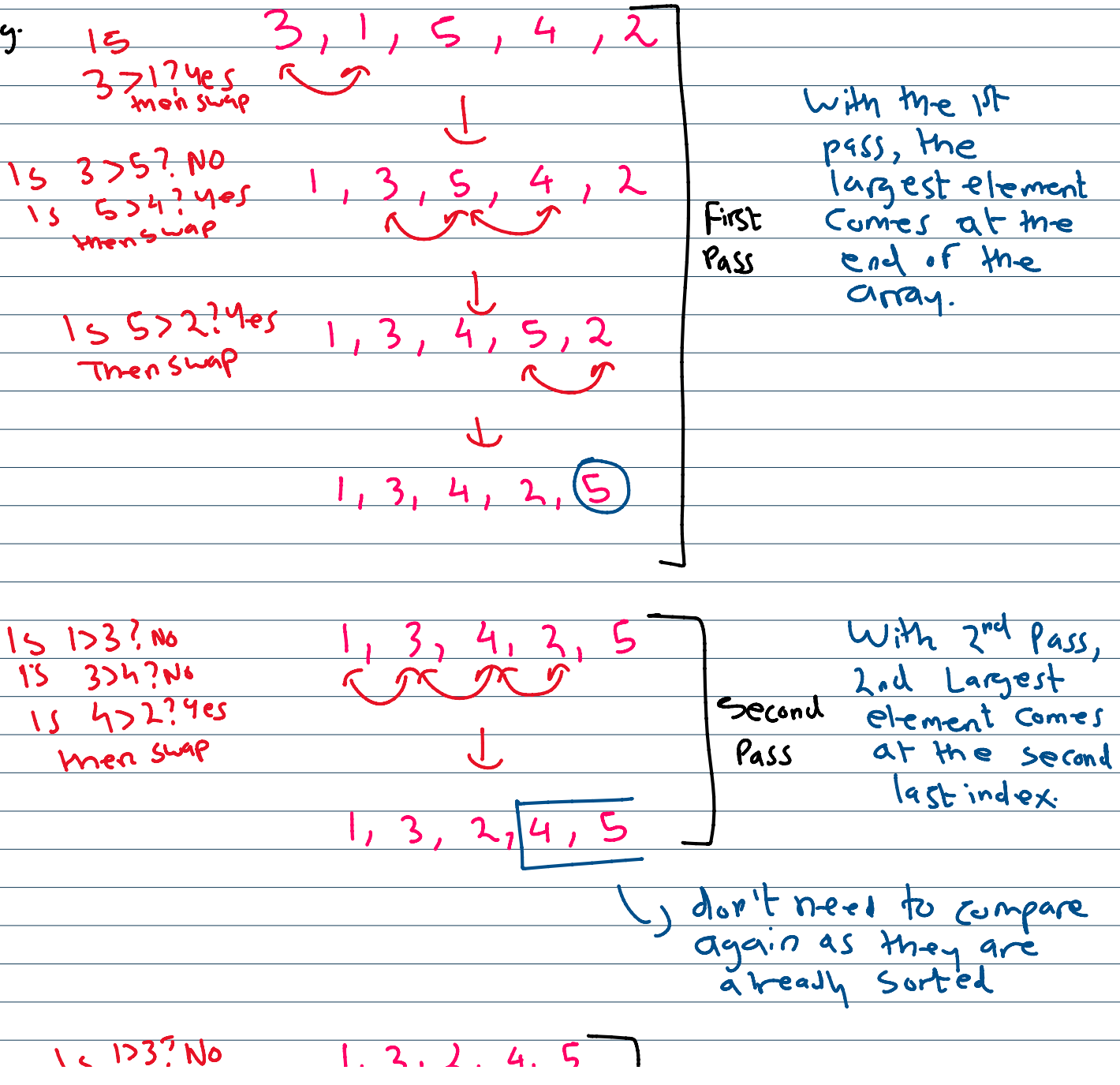
Sorting

-> Process of arranging items systematically.

Bubble Sort

-> It is the simplest sorting algo. It works by repeatedly comparing 2 adjacent elements & swapping them if they are in the wrong order.

Eg.



Is 1>3? No
3>2? yes
Then swap

1, 3, 2, 4, 5

Third Pass

1, 2, 3, 4, 5

Sorted Array!!

Bubble Sort also known as: Sinking Sort OR Exchange Sort

In Depth Working of Bubble Sort

i = 0
1st pass

Is j>j-1? No
Then swap

i j
0 1 2 3 4
3, 1, 5, 4, 2

Is j>j-1? yes
Is j>j-1? No
Then swap

i j
1, 3, 5, 4, 2

Is j>j-1? No
Then swap

1, 3, 4, 5, 2

j → 'j' is an internal loop which runs n-1 times

1, 3, 4, 2, 5

'i' here only acts as a counter which keeps on increasing

i = 1
2nd pass

Is j>j-1? yes
Is j>j-1? yes
Is j>j-1? No
Then swap

i j
1, 3, 4, 2, 5

0 1 2 3 4
1, 3, 2, 4, 5

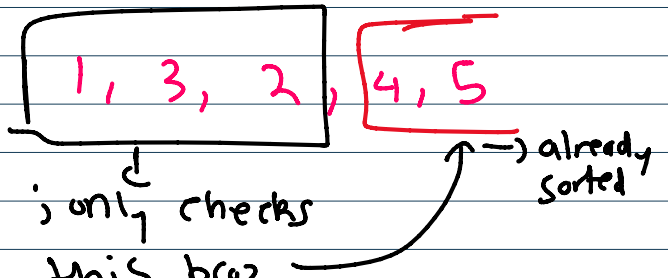
It doesn't make a sense to get 'j' here bcoz its already sorted

its already sorted

Thus 'j' basically runs tills <= length - i - 1

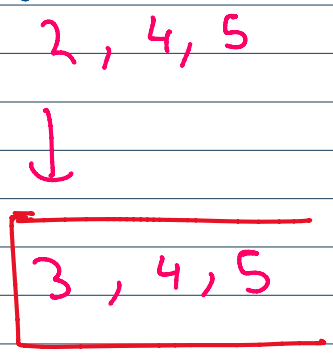
in 2nd pass => 5 - 1 - 1
=> (3) -> So 'j' only runs till 3rd index

i = 2
3rd pass



j only runs till
<= length - i - 1
= 5 - 2 - 1
= (2)
only till 2nd index

Is j > j-1? Yes
Is j > j-1? No
then swap



Complexity

Space Complexity: O(1) // constant

-> No extra space is needed here eg copying of array etc.
-> Also known as Inplace Sorting Algorithms.

Time Complexity: Best Case : O(N) => Sorted array

Worst Case : O(N²) => Sorted in descending order

-> As the size of array grows, no. of Comparisions also grow.

Best Case -> Array is Sorted

Best Case → Array is Sorted

i = 0
first pass

j j j j
1, 2, 3, 4, 5 → Run only once Don't check it again

Note: When 'j' never swaps for the value of 'i', it means that array is sorted. Thus, you can end the program.

Best Case Comparsions : N − 1 ≈ N

We ignore constants in T.C bcoz we need to find a relationship i.e mathematical equation

Best Case: O(N)

Worst Case

i = 0
1st pass

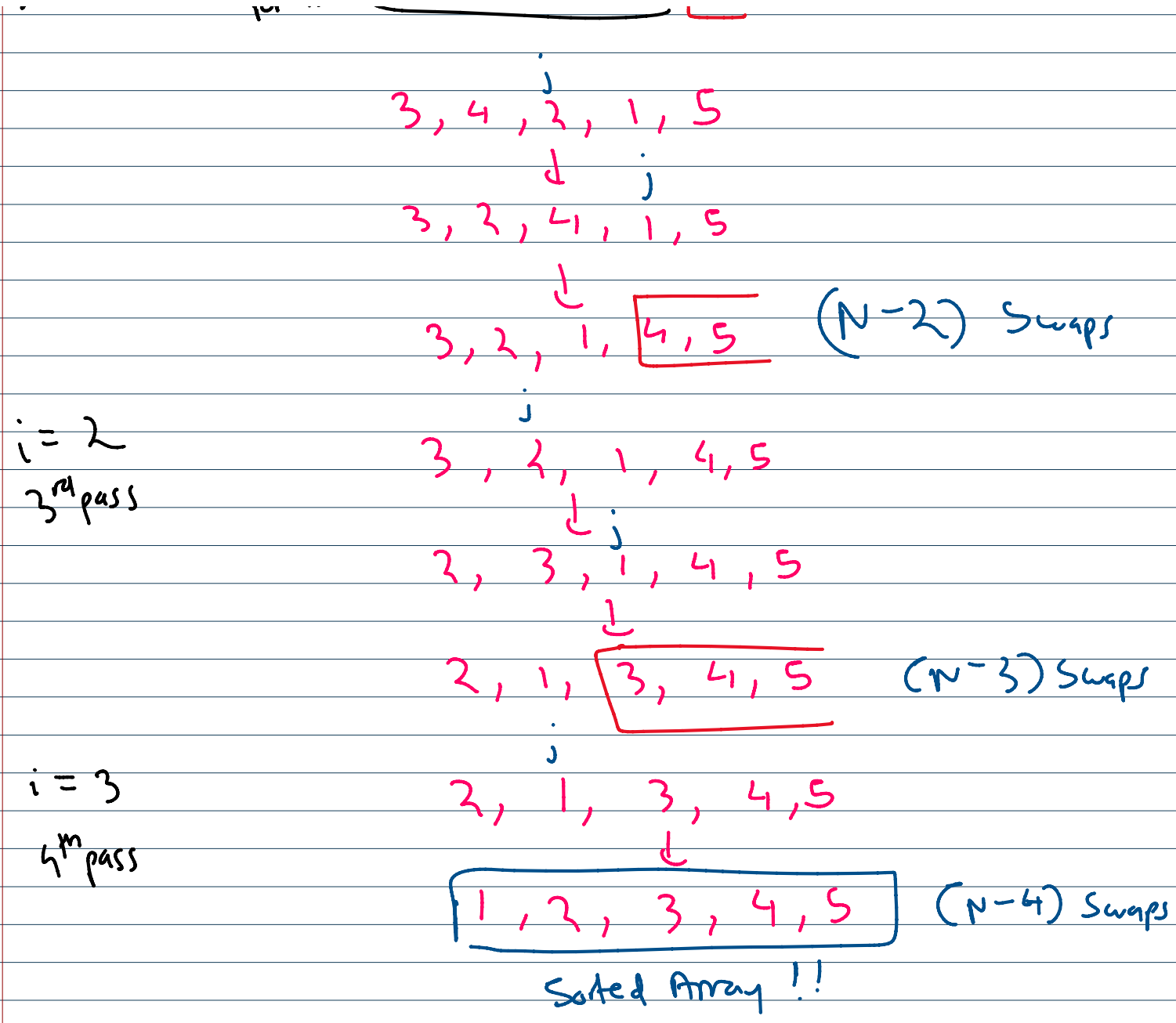
5, 4, 3, 2, 1
4, 5, 3, 2, 1
4, 3, 5, 2, 1
4, 3, 2, 5, 1
4, 3, 2, 1, 5

(N − 1) Swaps

i = 1
2nd Pass

Run loop only for this

4, 3, 2, 1, 5 → Sorted


3, 4, 3, 1, 5
3, 3, 4, 1, 5
3, 2, 1, 4, 5
(N-2) Swaps
i = 2
3rd pass
3, 2, 1, 4, 5
2, 3, 1, 4, 5
2, 1, 3, 4, 5
(N-3) Swaps
i = 3
4th pass
2, 1, 3, 4, 5
1, 2, 3, 4, 5
(N-4) Swaps
Sorted Array !!

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

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

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

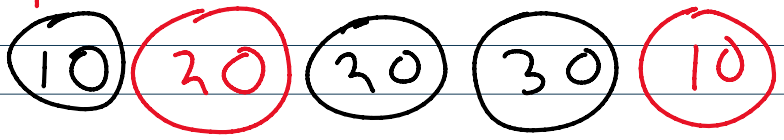
$= 4N - \frac{N^2+N}{2}$

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

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

Worst Case : $O(N^2)$

☆ Stability

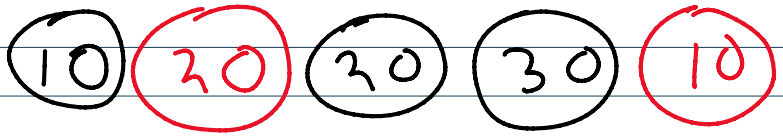


↓ After Sorting

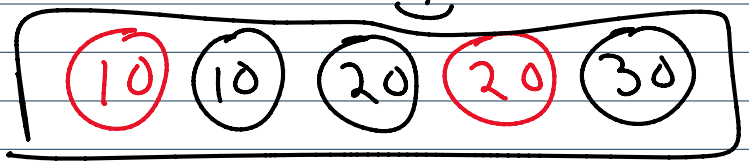
Before Sorting: Elements in White were before (in case of 10) & Elements of Red were before (in case of 20)

After Sorting: Same order is maintained

Thus, this is called Stable Sort



After Sorting



=> unstable since order isn't maintained