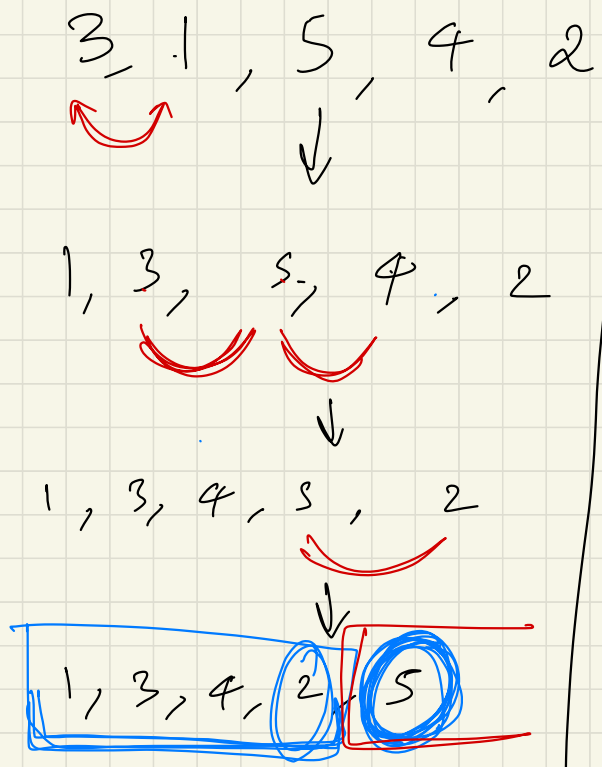
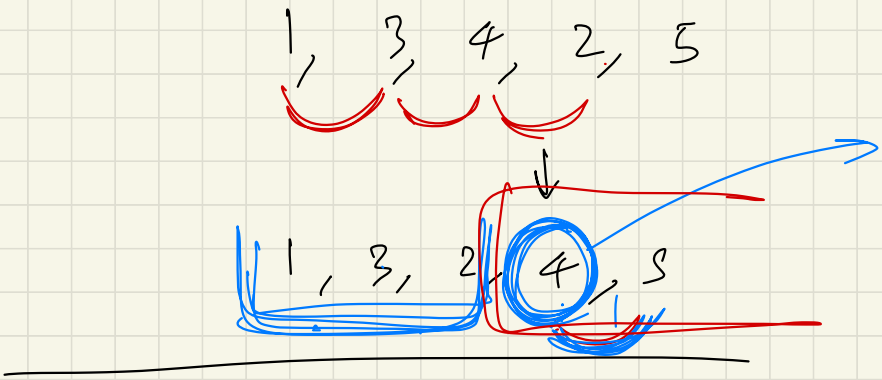
Bubble Sort

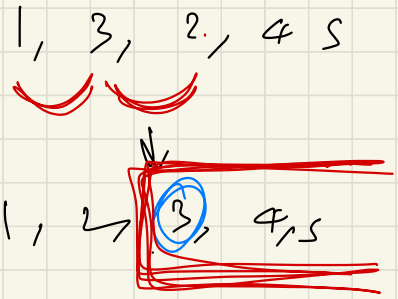


Why ?

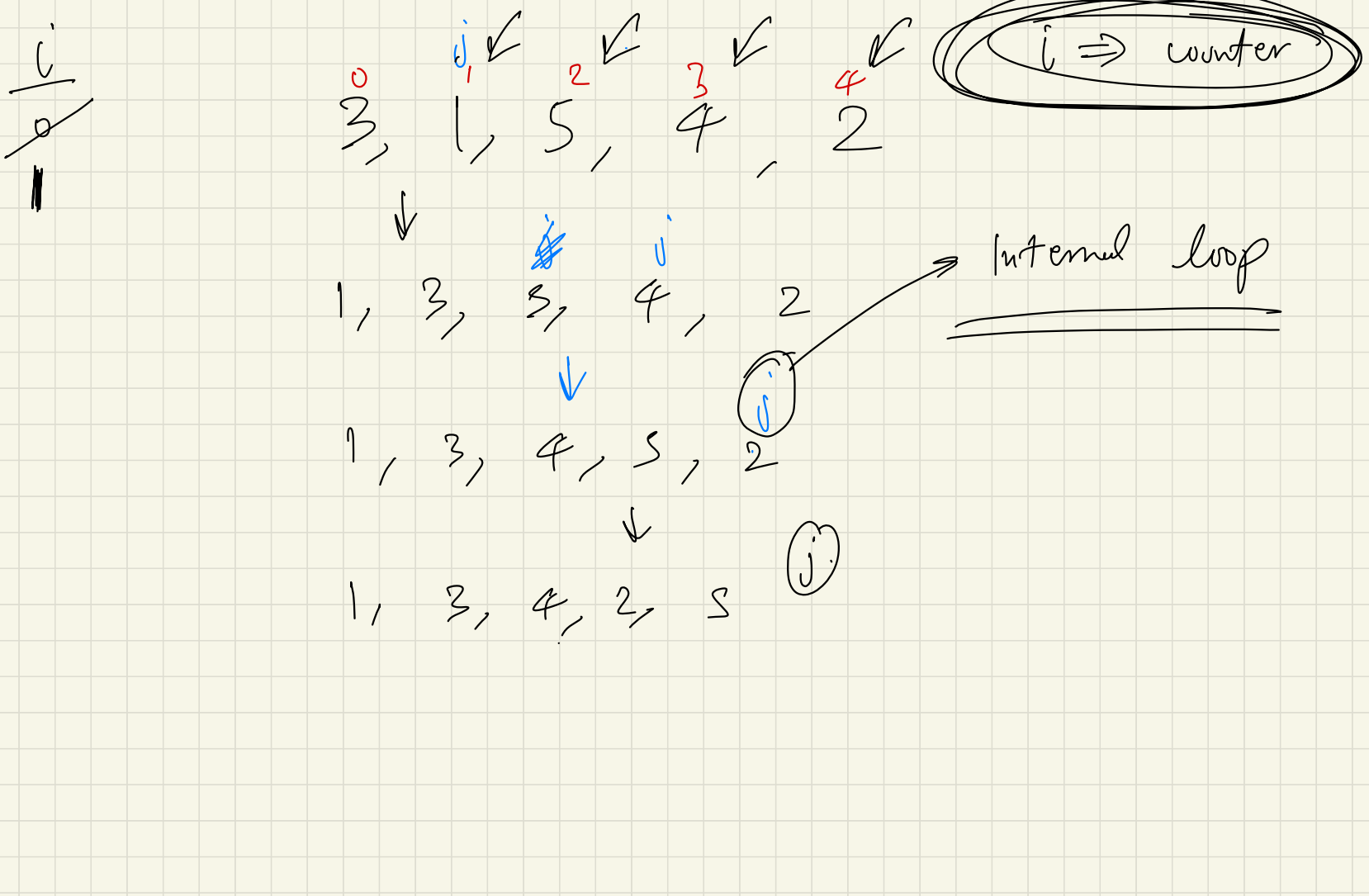
With the first pass through the array, the largest element come to the end.



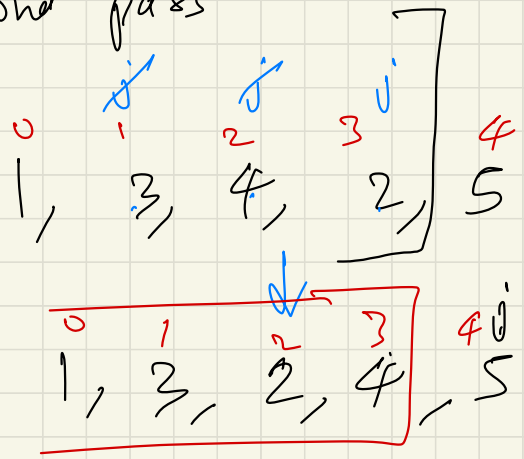
with pass no. 2
2nd largest element
is at the 2nd
from last index.



Aka sinking sort /
exchange sort


i
0
1
i
0
1
2
3
4
3, 1, 5, 4, 2
i
j
1, 3, 3, 4, 2
i
j
1, 3, 4, 5, 2
j
1, 3, 4, 2, 5
i ⇒ counter
Internal loop

for i=1 // second pass



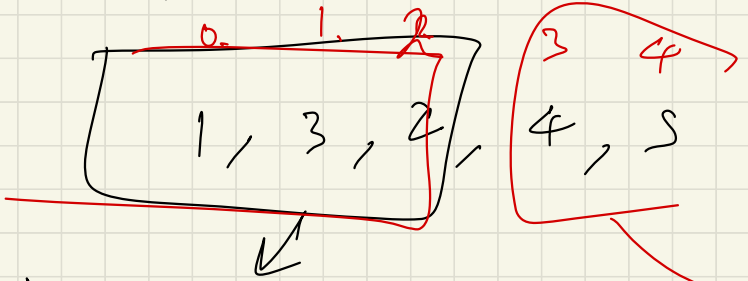
< length - i

= 5 - 1 = 4

or

<= len - i - 1

for i=2 // third pass



len - i = 5 - 2

= 3

i=3

1, 2, 3, 4, 5

j will only check this because → This is already sorted

Space complexity = O(1) // constant // No extra space required i.e. copying the array, etc. not required.

aka inplace sorting algorithms.

Time complexity :

Best case : O(N) ⇒ Sorted

Worst case : $O(N^2)$ = Sorted in opposite

No. of comparisons.

We 10,000 x4 2500.

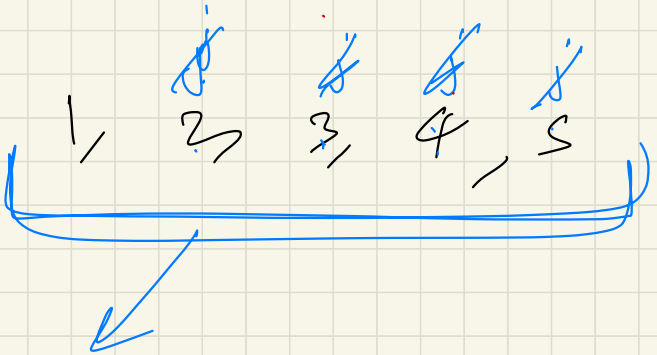
BC 100 x2 50

✡ As the size of array is growing, the no. of comparisons is also growing.

Best Case:

i = 0

i^th pass



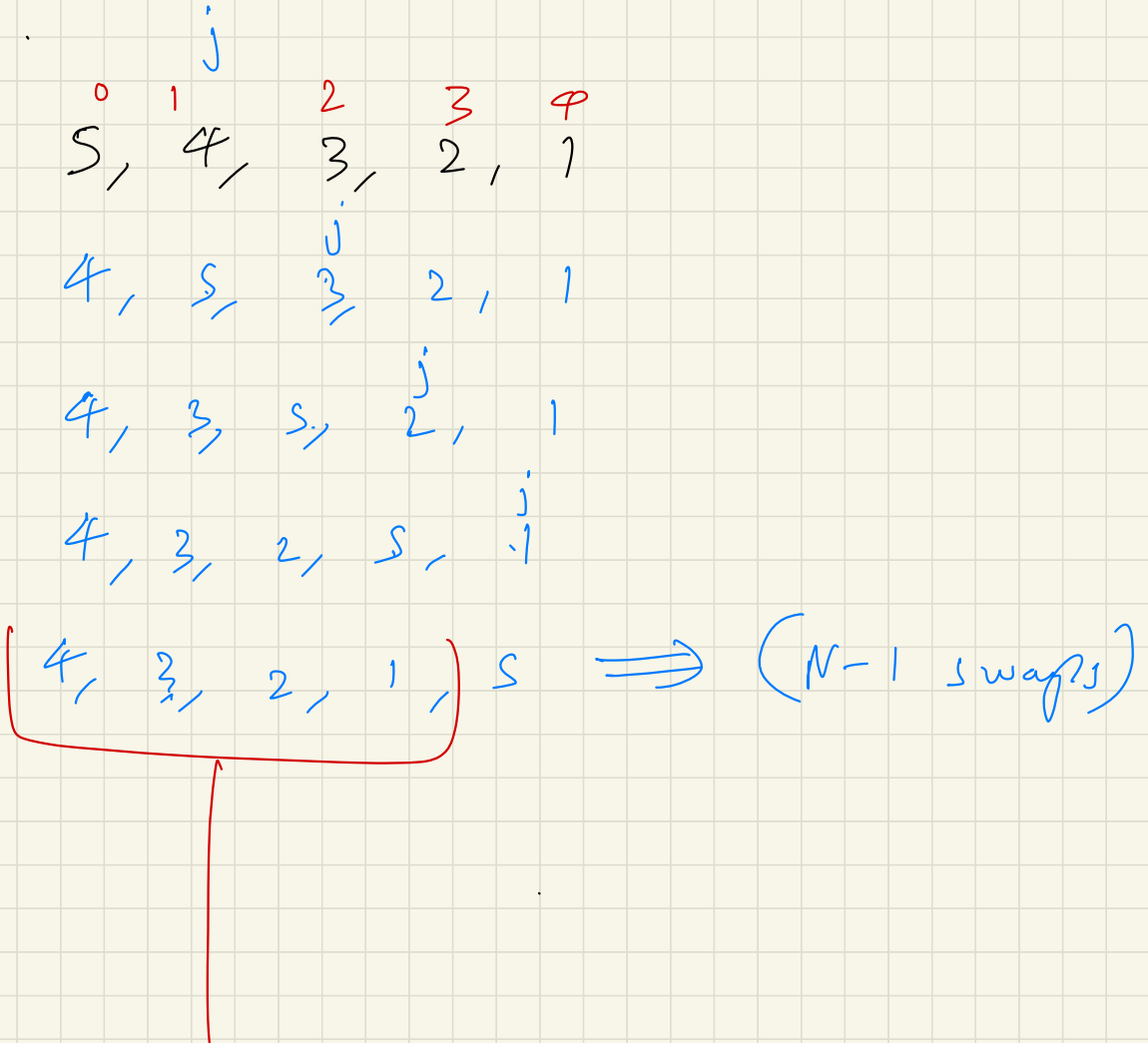
Once.

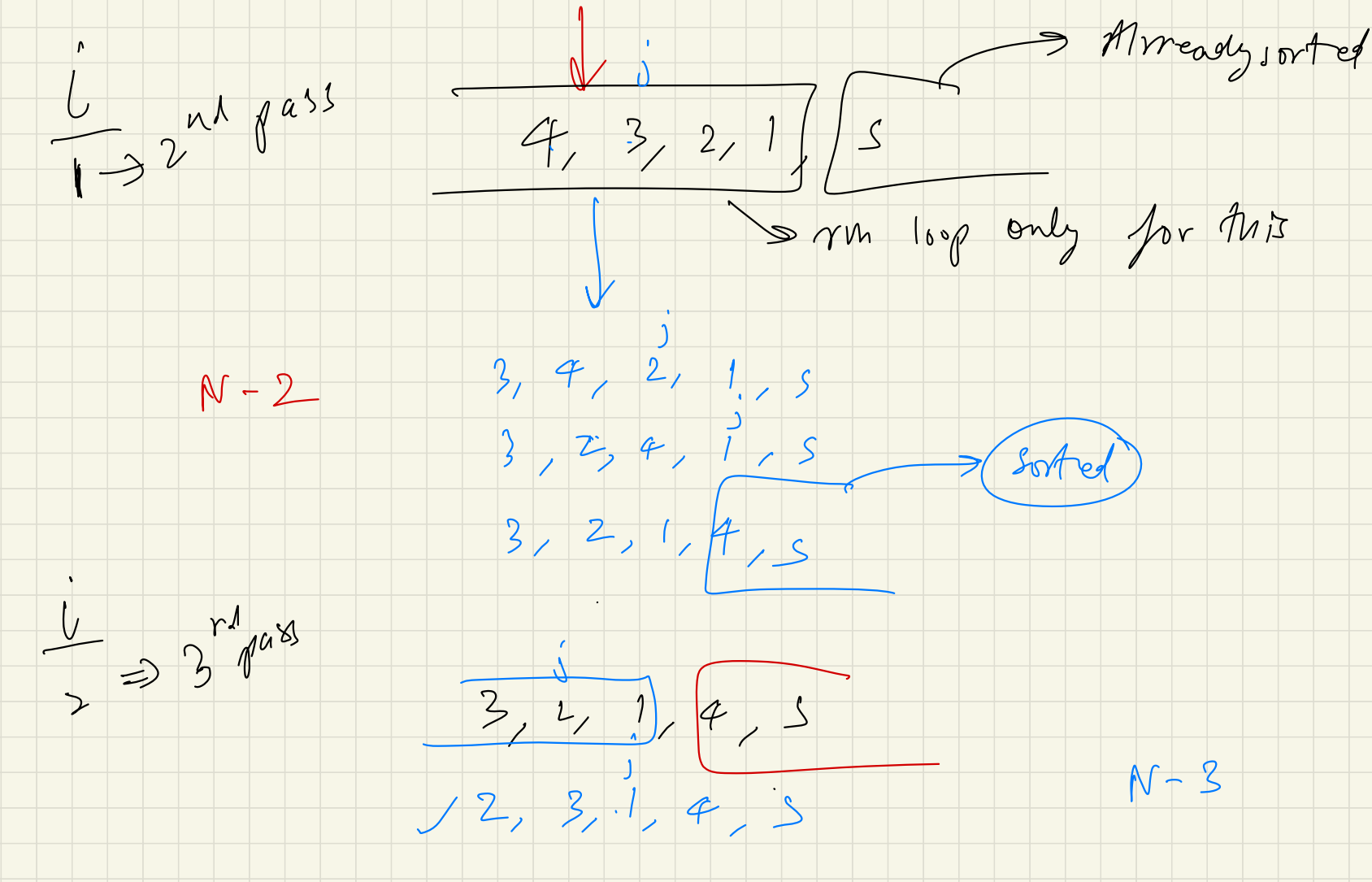
NOTE: When j never swaps for a value of i, it means array is sorted, hence you can end the program

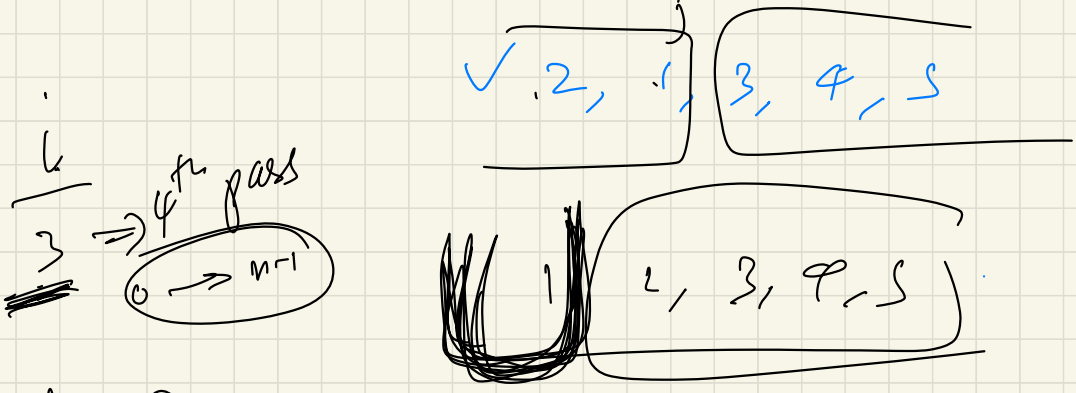
Best Case comparison = N-1 → In time complexity, constants are ignored.

Worst Case:

i
0




i
1 → 2nd pass
i
4, 3, 2, 1
5
already sorted
run loop only for this
N - 2
j
3, 4, 2, 1, 5
3, 2, 4, 1, 5
sorted
3, 2, 1, 4, 5
i
⇒ 3rd pass
i
3, 2, 1, 4, 5
j
2, 3, 1, 4, 5
N - 3



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

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

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

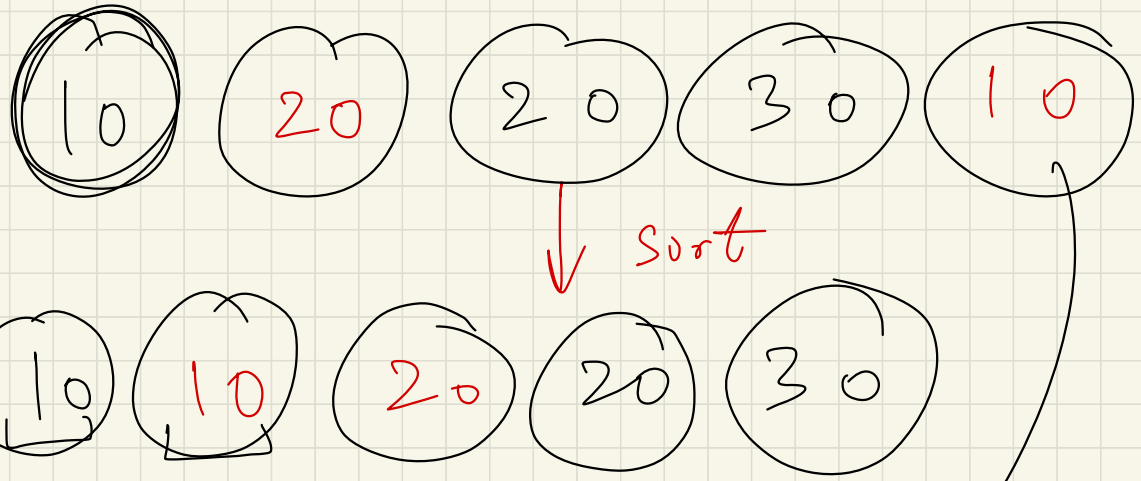
$= 4N - \left(\frac{N^2+N}{2}\right) \Rightarrow \frac{8N - N^2 - N}{2}$

$\frac{4 \times 5}{2} = 10$

$= O\left(\frac{N \cdot N^2}{N}\right) = O(N^2)$ Why?

Come back in Time complexity lecture.

stability



In original array, black ball of 10 was before red ball of 10. And in the sorted one, this order is maintained.