12-09 Introduction to Sorting

12 September 2022 20:12

Sorting:- arranging things in a particular order.

asc / desc

arr : [1, 2, 3, 4, 5, 6]

[[l1, b1], [l2, b2], [l3, b3]]

1. based on length.
2. based on breadth
3. Perimeter.
4. Area.

* Array of Strings]

1. length.
2. lexicographical order.

Sorting algorithms:-

Basic → ① Easy to understand
② unoptimised.

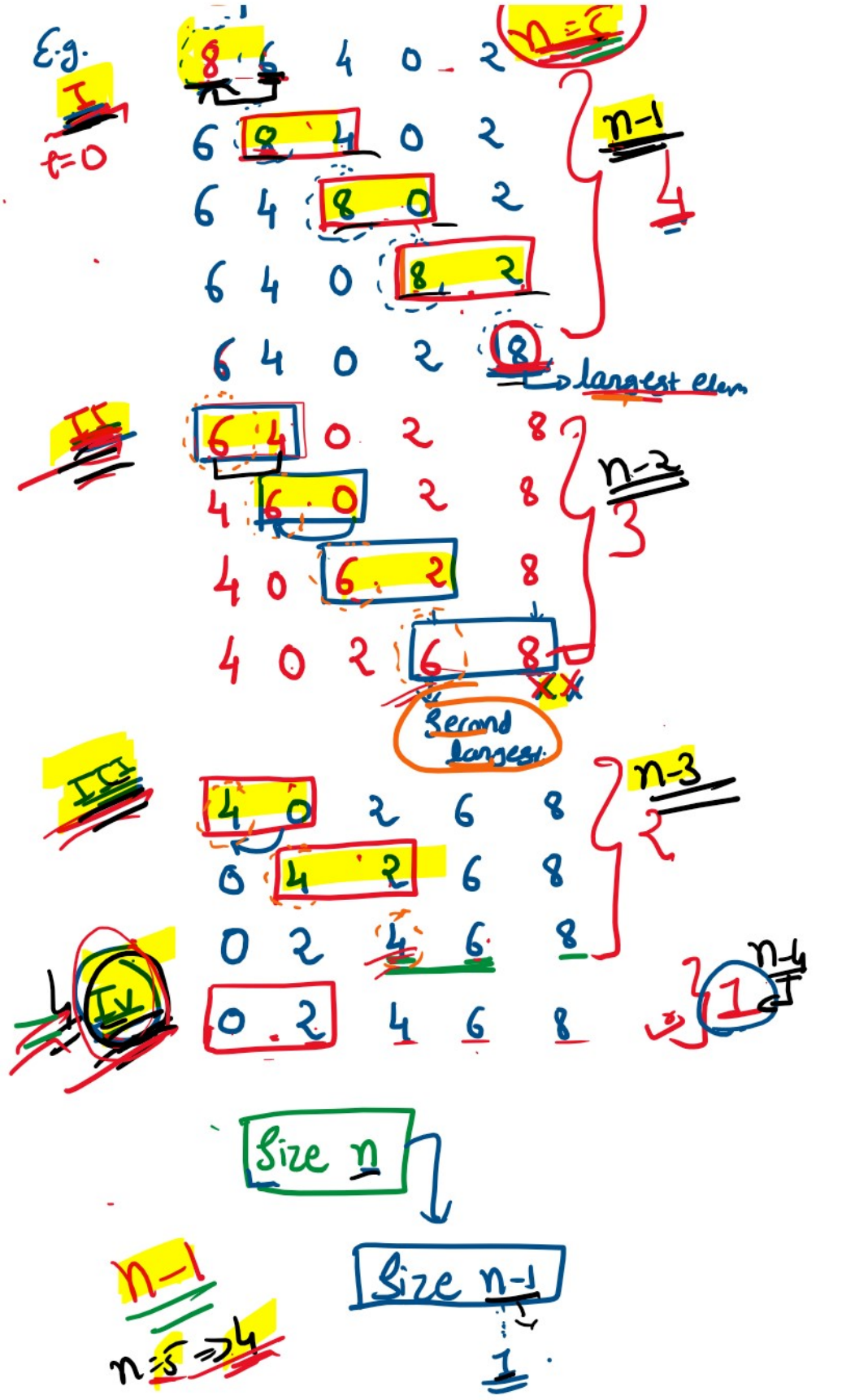
Advanced → ① advanced compared to basic
② faster & optimised compared to basic.

1. Bubble Sort
2. Selection Sort
3. Insertion Sort.

1. Merge Sort
2. Quick Sort
3. Count Sort
4. Bucket Sort
5. Radix Sort

① Bubble Sort:-

E.g. 8 6 4 0 2


E.g.
I
t=0
8 6 4 0 2
n=5
n-1
4
6 8 4 0 2
6 4 8 0 2
6 4 0 8 2
6 4 0 2 8
largest elem
II
6 4 0 2 8
n-2
3
4 6 0 2 8
4 0 6 2 8
4 0 2 6 8
XX
Second largest
III
4 0 2 6 8
n-3
2
0 4 2 6 8
0 2 4 6 8
IV
0 2 4 6 8
n-4
1
Size n
Size n-1
1
n-1
n=5 ⇒ 4
(n-1) + (n-2) + - - - - ⇒ n+n+n+n+ - -
n-1 terms
n-1
n

n-1 terms

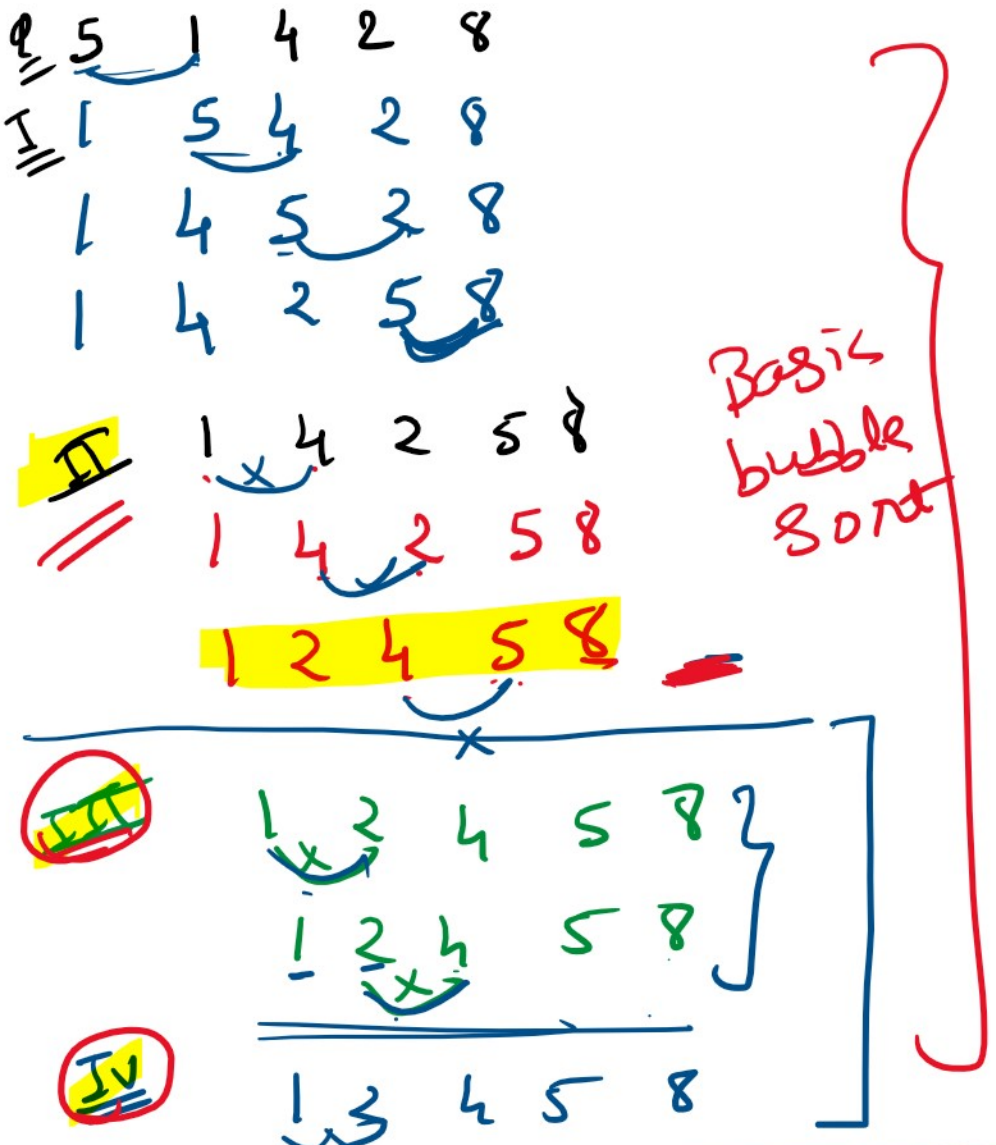
TC (n^2)

$n-1$

$n(n-1)$

n^2

$n^2 - n$



(1) Iterate array n-1 times.

(2) for loop.

$n-1$

$i=0$?

(2) for loop.

1 → n-1 swaps ← i=0
2 → n-3 swaps ← i=1
3 → n-3 swaps ← i=3

```
for ( )
{
    =}
}
```

if (compare with adj elem)
    swap ()

n=5

n-1-i

i=0 → 4 → n-1 → 5-1-0
i=1 → 3 → n-2 → 5-1-1
i=2 → 2 → n-3 → 5-1-2

Q Modify Bubble Sort algorithm to detect the kth largest / smallest element in the array

k=3 [1 2 5 6 3]

kth Smallest
1st → 1
2nd → 2
3rd → 3
4th → 5

kth largest
1st → 6
2nd → 5
3rd → 3

4th → 5

8 6 4 0 2 (i)

6 8 4 0 2

6 4 8 0 2

6 4 0 8 3

6 4 0 2 8

largest elem in the right

4 n-1

n
n-1

Worst ⇒ $O(n^2)$

Avg ⇒ $O(n^2)$

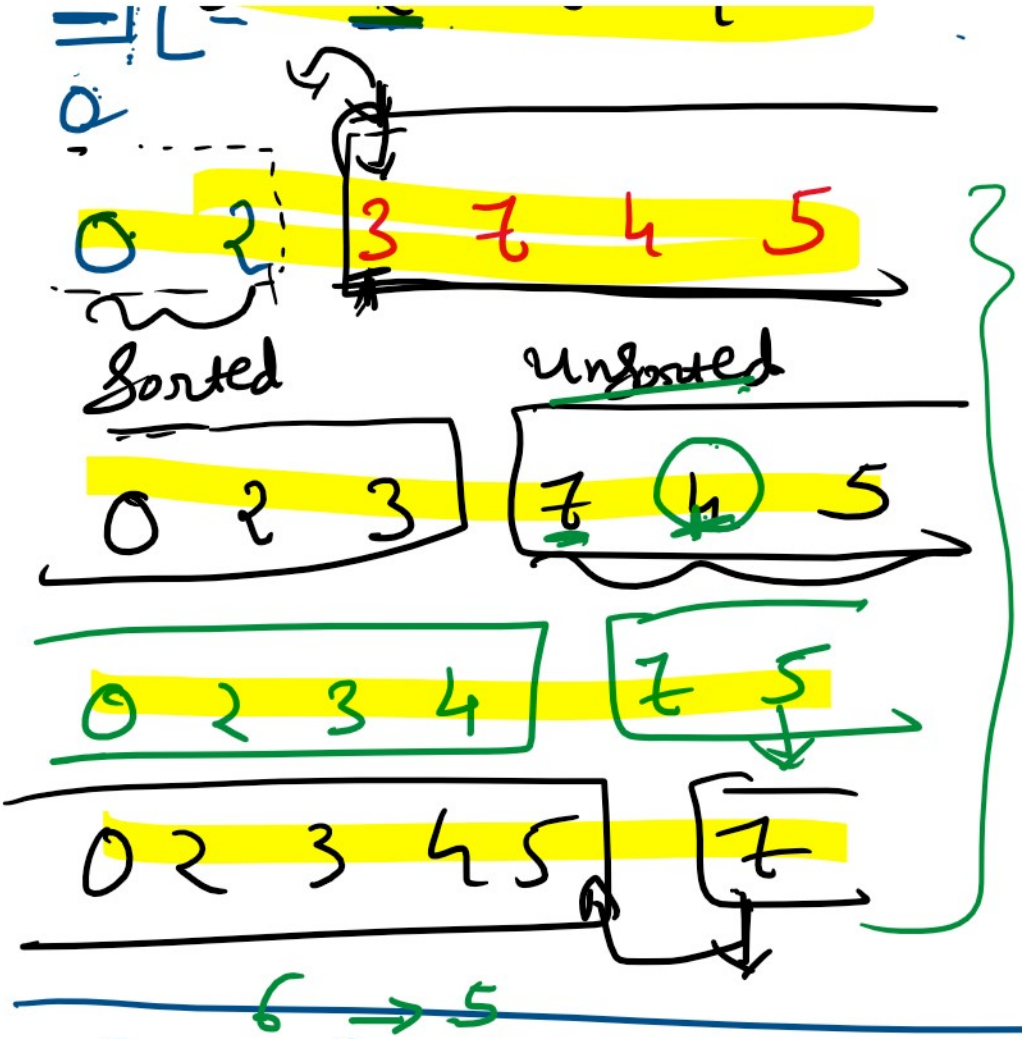
Best → $O(n^2)$. [Basic Bubble Sort]

* Selection Sort:-

unsorted array:- 8 6 2 4 0

4 3 2 7 0 5

0 | 3 2 7 4 5

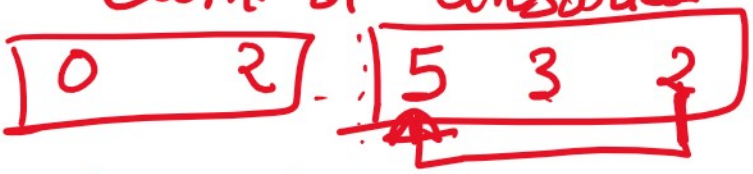


Everytime you are taking smallest elem from unsorted region ↓ last ind of sorted

Size → n ⇒ n-1 iterations

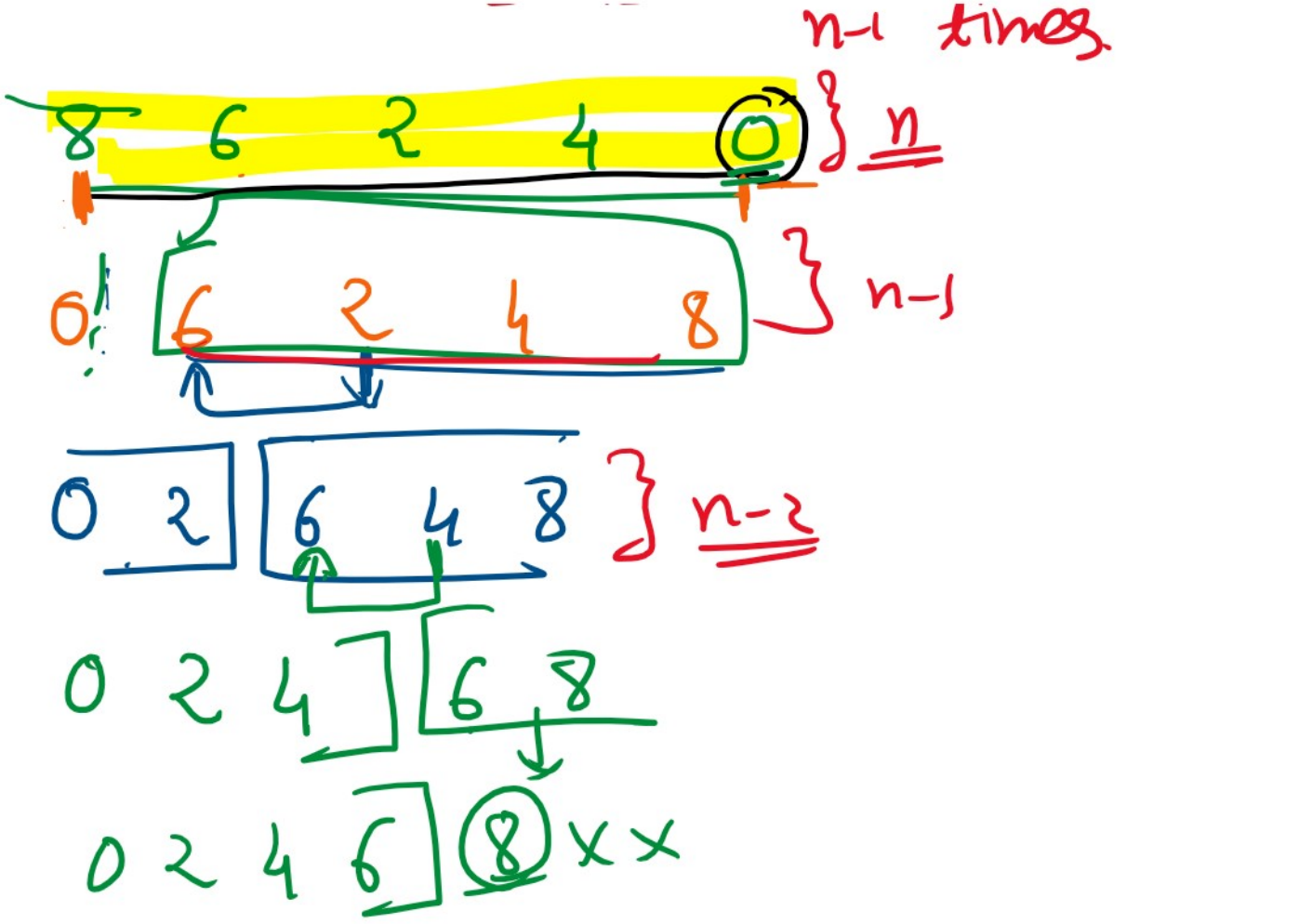
1) find out the min elem. from unsorted reg.

2) Swap the min elem. with the left most elem. of unsorted region.

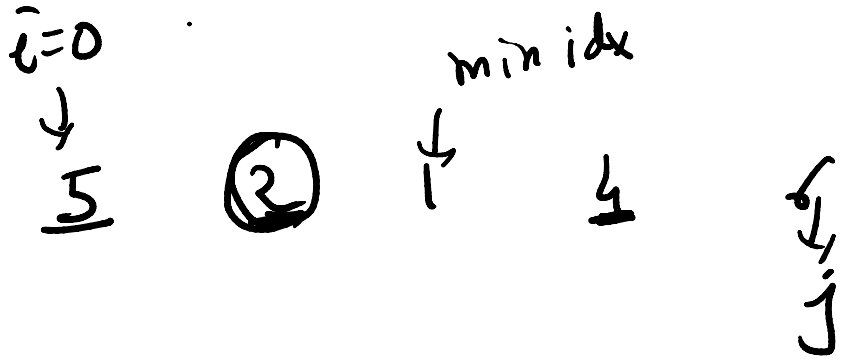


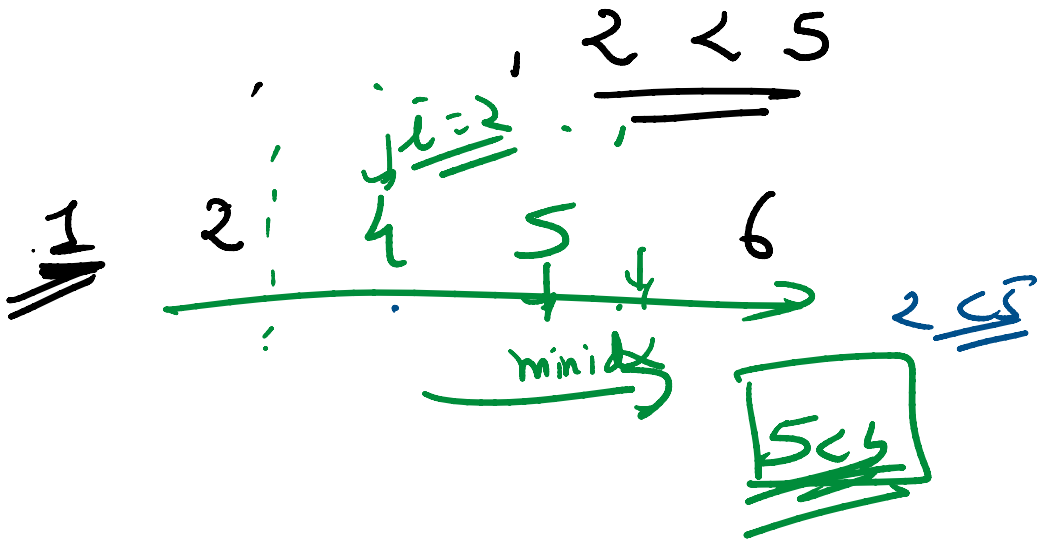
3) Then the size of the unsorted reg would be decreased by 1

4) The above steps will be done n-1 times




```
for(i=0; i<n-1; i++) {
    let minidx = i;
    for(j=i+1; j<n; j++) {
        if(arr[j] < arr[minidx]) {
            minidx = j;
        }
    }
    [arr[minidx], arr[i]] = [arr[i], arr[minidx]]
}
```





Iter => n

n + (n-1) + (n-2) + ...

n-1

$O(n^2)$

Bubble

Selection

S.C. O(1) O(1)

→ Such sorting algo, which takes a space complexity of O(1) } ⇒ inplace algorithm.