PW SKILLS

Sorting

Today’s checklist

1. Sorting
2. Bubble sort Algorithm
3. Time complexity and space complexity
4. Bubble sort optimization
5. Stable and unstable sort
6. Practice Questions
7. Selection sort Algorithm
8. Time complexity and space complexity
9. Insertion sort Algorithm
10. Time complexity and space complexity
11. Stability of both
12. Programming questions

What is sorting?

PW SKILLS

arr = { 10, 1, 2, 18, 4, 5 }

↓ sort

{ 1, 2, 4, 5, 10, 18 }

Sorted Order
↓
Increasing
↓
Ascending
↓
Non-decreasing

Any given array is said to be sorted if `arr[i] <= arr[i+1]` for every i

where i = 0, 1, 2 .. n-2

Bubble sort algorithm

Q// Check if given array is sorted array.

1) arr = {1, 4, 7, 8, 10, 12} → Sorted

2) arr = {1, 4, 7, 6, 10, 12} → Unsorted

if (arr[i] > arr[i+1]) → unsorted.

Bubble sort algorithm

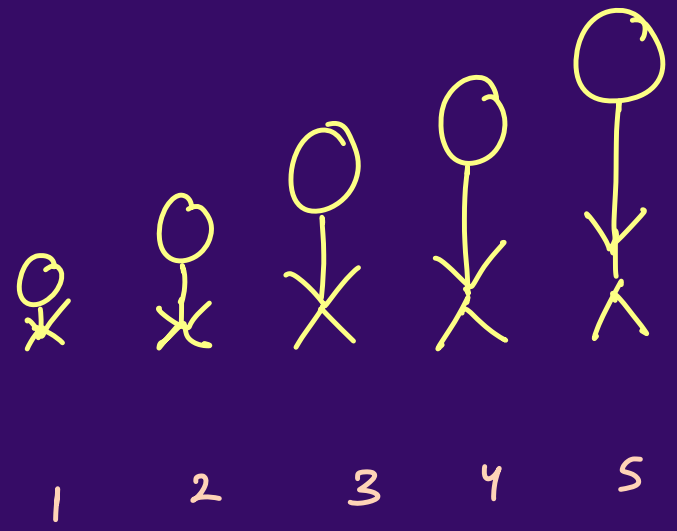
arr = { 7, 1, 2, 8, -4 }

↓ sort

{ -4, 1, 2, 7, 8 }

Bubble sort algorithm

arr = { 3, 1, 2, 5, 4 }

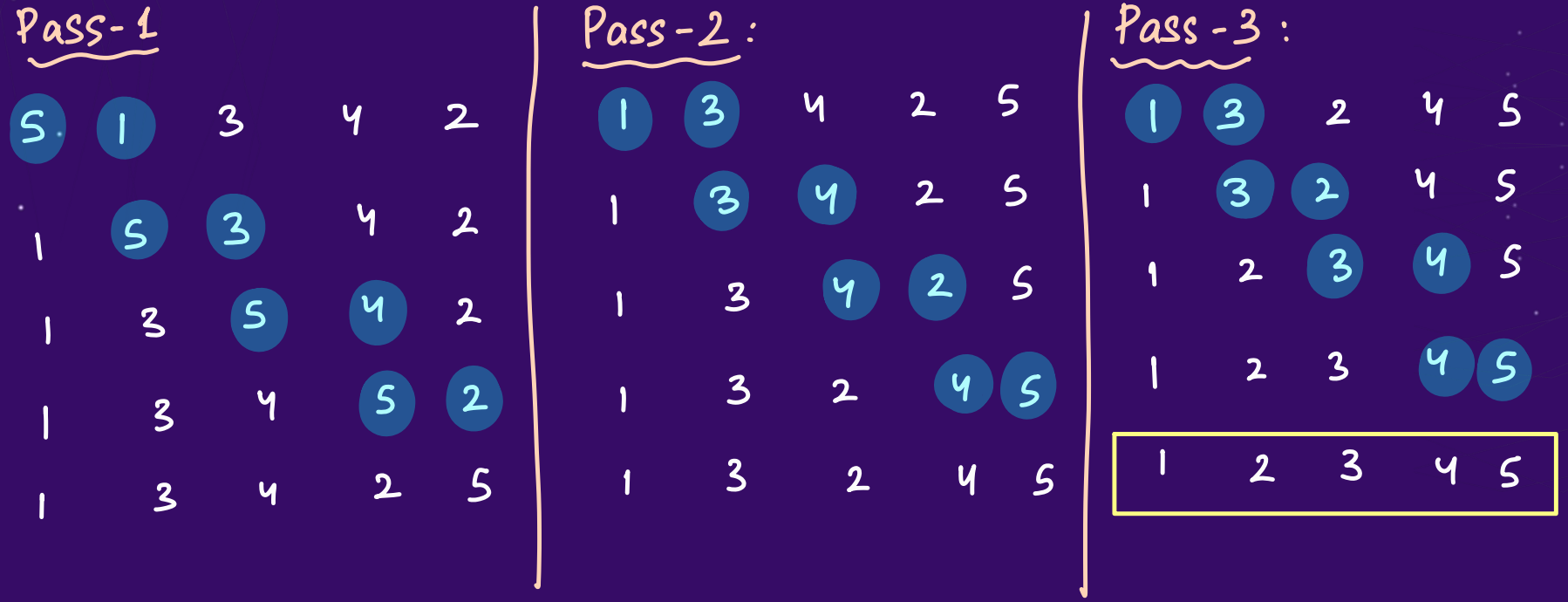


technique:

Swapping every 2 adjacent elements if arr[i] > arr[i+1]

Bubble sort algorithm

arr = { 5, 1, 3, 4, 2 }



Bubble sort algorithm

Worst case : arr = {10, 4, 1, 0, -2} n=5

Pass:1

10 4 1 0 -2

4 10 1 0 -2

4 1 10 0 -2

4 1 0 10 -2

4 1 0 -2 10

Pass:2

4 1 0 -2 10

1 4 0 -2 10

1 0 4 -2 10

1 0 -2 4 10

Pass:3

1 0 -2 4 10

0 1 -2 4 10

0 -2 1 4 10

Pass:4

0 -2 1 4 10

-2 0 1 4 10

Bubble sort algorithm

PW SKILLS

n = 5 size array

the maximum passes will always be 4 (n-1)

Example

Time and Space complexity

PW SKILLS

```
// Bubble Sort - 1
for(int x=1;x<=n-1;x++){ // n-1 passes
    for(int i=0;i<n-1;i++){
        if(arr[i]>arr[i+1]){
            int temp = arr[i];
            arr[i] = arr[i+1];
            arr[i+1] = temp;
        }
    }
}
```

S.C. = O(n) → Given array of 'n' size

Auxiliary Space → O(1)

T.C. = $O(n^2-2n+1)$

$\simeq O(n^2)$

no of ops = $(n-1)^2 = n^2-2n+1$

Time and Space complexity

```
// Bubble Sort - 2
for(int x=0;x<n-1;x++){ // n-1 passes
    for(int i=0;i<n-1-x;i++){
        if(arr[i]>arr[i+1]){
            int temp = arr[i];
            arr[i] = arr[i+1];
            arr[i+1] = temp;
        }
    }
}
```

T.n.o. =

x = 0 → n-1 times

1 → n-2 times

2

⋮

n-2 → 1 time

$\Rightarrow \underbrace{n-1 + n-2 + n-3 \cdots 1}_{n-1}$

$\Rightarrow \quad 1+2+3 \cdots n-1 = \boxed{\frac{n(n-1)}{2}} = \frac{n^2}{2} - \frac{n}{2}$ [T.C. = $O(n^2)$]

Can we optimize it further ?

PW SKILLS

↓

Yes. We can

For arrays like → arr = {3, 1, 2, 5, 4} → This can be sorted in one pass only

1) So, after every pass, we can check if that array is sorted or not

Can we optimize it further ?

PW SKILLS

```
// Bubble Sort Optimised
for(int x=0;x<n-1;x++){ // n-1 passes
    boolean flag = true;
    for(int i=0;i<n-1-x;i++) {
        if (arr[i] > arr[i + 1]) {
            int temp = arr[i];
            arr[i] = arr[i + 1];
            arr[i + 1] = temp;
            flag = false;
        }
    }
    if(flag==true) break;
}
```

Stable and Unstable sort

PW SKILLS

↓

Bubble Sort


```
if(arr[i]>arr[i+1])
  swap
```

1 3 3* 4 5 → Stable Sort

1 3* 3 4 5 → Unstable Sort

Ques:

worst case

Q1 : How much maximum swaps are needed to sort array of length 6 ?

n

Ex : arr = 6 5 4 3 2 1

5 6 4 3 2 1

5 4 6 3 2 1

5 4 3 6 2 1

5 4 3 2 6 1

5 4 3 2 1 6

$5 + 4 + 3 + 2 + 1 = 15$ swaps

$n-1 + n-2 + n-3 \dots 1$

$= \frac{n(n-1)}{2}$ max swaps

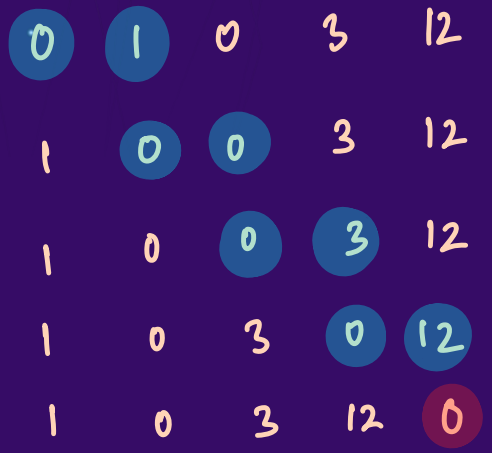
Pass: 1 → 5 swaps

Ques:

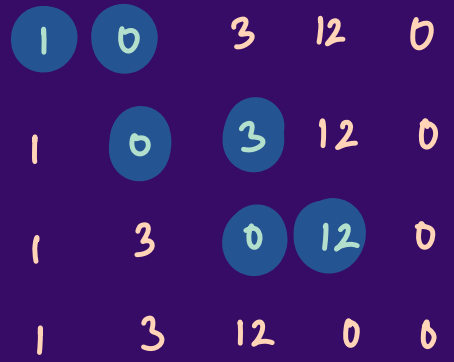
Q2: Push zeroes to end while maintaining the relative order of other elements.

arr = {0 1 0 3 12}

Pass - 1



Pass- 2



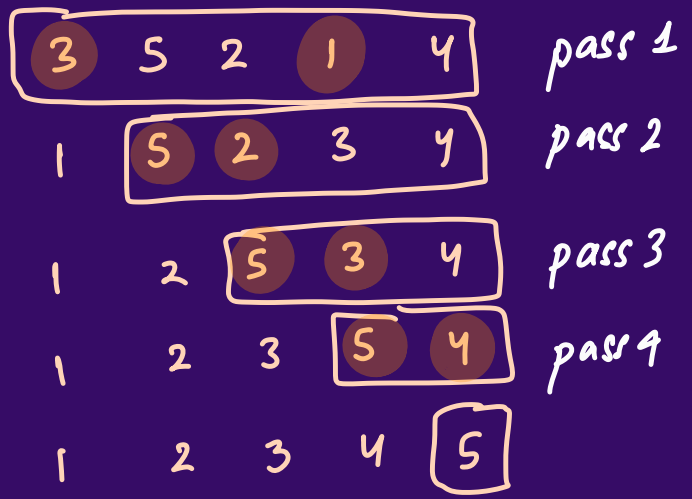
[leetcode 283]

Selection Sort Algorithm

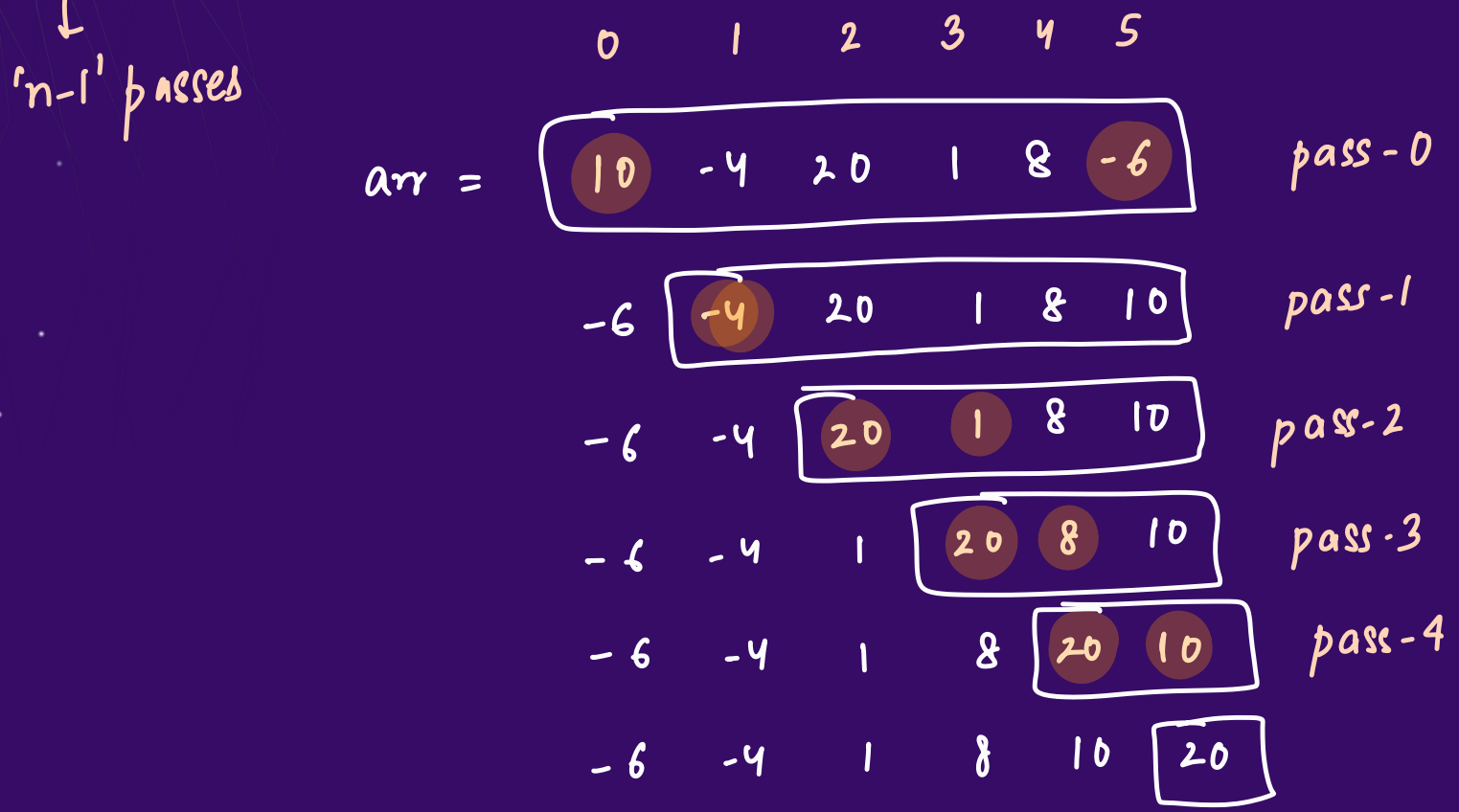
↓

In every pass, we find the K^{th} smallest element & put it in its right place

arr =



Selection Sort Algorithm



Selection sort Code and dry run

```java
// Selection Sort
for(int i=0;i<n-1;i++){
    int min = Integer.MAX_VALUE;
    int mindx = -1;
    for(int j=i;j<n;j++){
        if(arr[j]<min){
            min = arr[j];
            mindx = j;
        }
    }
    swap(arr,i,mindx);
}
```

```java
public static void swap(int[] arr, int i, int j){
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

Time and Space complexity

PW SKILLS

```java
// Selection Sort
for(int i=0;i<n-1;i++){
    int min = Integer.MAX_VALUE;
    int mindx = -1;
    for(int j=i;j<n;j++){
        if(arr[j]<min){
            min = arr[j];
            mindx = j;
        }
    }
    swap(arr,i,mindx);
}
```

$i = 0 \rightarrow j = 0, 1, 2, 3 \ldots n-1 \rightarrow n$ times

$1 \rightarrow j = 1, 2, 3 \ldots n-1 \rightarrow n-1$ times

$2 \rightarrow j = 2, 3, \ldots n-1 \rightarrow n-2$ times

$\vdots$

$n-2$

No. of operations: $n + n-1 + n-2 + n-3 \ldots 1$

$$= \frac{n(n+1)}{2} = \frac{n^2}{2} + \frac{n}{2}$$

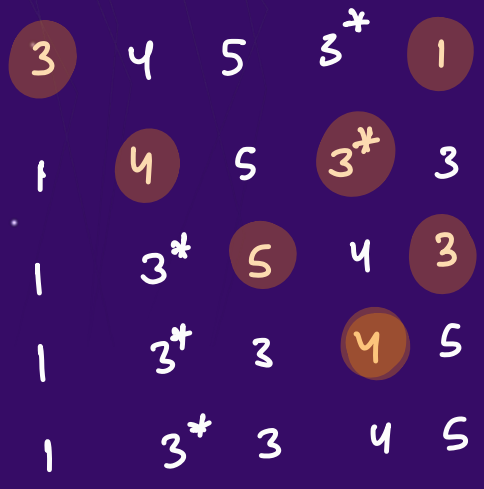
$T.C. = O(n^2)$

Auxillary Space $= O(1)$

Stable and Unstable sort

PW SKILLS

↳ Selection Sort



→ Unstable sort

→ Stable Sort

```
if (arr[i] < min){
    min = arr[i]
    minIx = i
3
```

Time and Space complexity

PW SKILLS

Selection Sort

Cannot be optimised

Unstable Sort

No. of swaps are less

↓

n-1 swaps

Bubble Sort

Can be optimised

Stable Sort

No. of swaps are more

↓

$\frac{n(n-1)}{2}$ max swaps

If cost of swapping is something to consider then selection sort is better.

Best case T.C. is $O(n^2)$

Best case T.C. is $O(n)$

Time and Space complexity

PW SKILLS

Time Complexity

	B.S	S.S
Avg. Case	$O(n^2)$	$O(n^2)$
Worst Case	$O(n^2)$	$O(n^2)$
Best Case	$O(n)$	$O(n^2)$

Homework:

PW SKILLS

Sort a given array in decreasing order using bubble sort

arr = {3, 1, 2, 5, 4} → {5, 4, 3, 2, 1}

Hint: After every pass the smallest element will be at end.

if(arr[i] < arr[i+1]) swap

Homework:

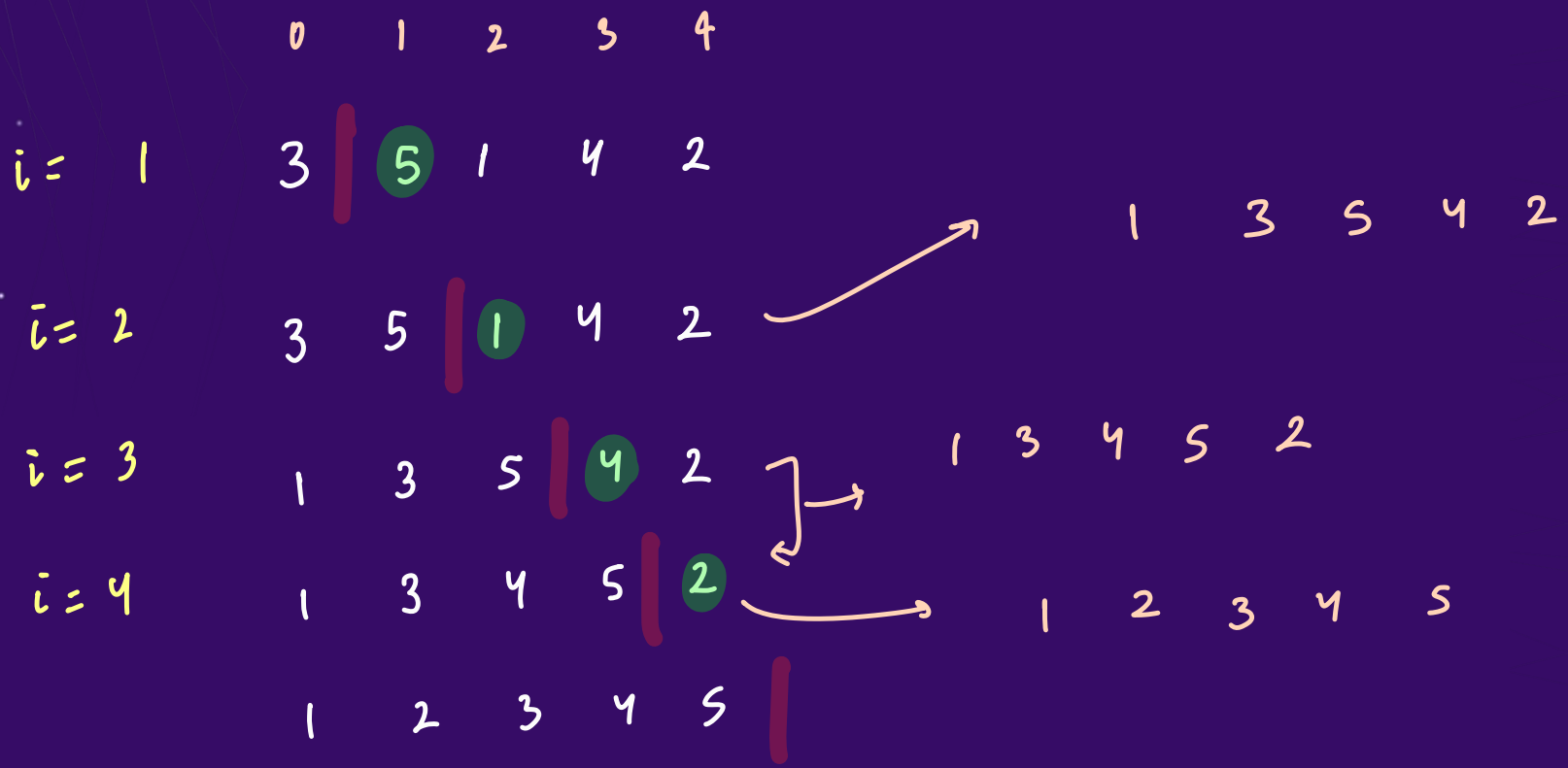
Sort a given Array in increasing order using selection sort, but in each pass, put the kth maximum element at the right position.

Insertion Sort Algorithm

Example: you have playing cards numbered from 1 to 10 in random order. you have to sort them.

1 2 3 4 5 6 7 8 9 10

Insertion Sort Algorithm



Insertion Sort Algorithm

PW SKILLS

n-1 passes → no. of times outer loop will run.

i = 0 ; i < n-1 or i = 1 ; i < n

Inner loop → we start from

j = i ; j >= 0 → ∝

j = i ; j >= 1

arr[0] < arr[-1]

Error

if (arr[j] < arr[j-1])
    swap(arr[j], arr[j-1])

Insertion sort Code and dry run

PW SKILLS

```
// Insertion Sort
for(int i=1;i<n;i++){ // n-1 passes
    for(int j=i;j>=1;j--){
        if(arr[j]<arr[j-1])
            swap(arr,j,j-1);
        else break;
    }
}
```

Worst Case :

0	1	2	3	4
1	2	3	4	5

j-1 j

$i = 1$ to 4 $i++$

$j = i$ to 1 $j--$

No. of swaps = 1 + 2 + 3 + 4

or

$1 + 2 + 3 + \cdots n-1 = \frac{n(n-1)}{2}$ max swaps

Best Case :

0	1	2	3	1
1	2	3	4	5

j-1 j

No. of swaps = 0.

Time and Space complexity

PW SKILLS

Avg. Case = $O(n^2)$

Worst. Case = $O(n^2)$

Best Case = $O(n)$

Stability of Insertion ~~and Selection~~ Sort

Insertion Sort is a stable sort.

Bubble Sort

- O(n) T.C. in best case
- Stable
- $\frac{n(n-1)}{2}$ max swaps
- you have to optimise using extra variable (boolean)

Selection Sort

- $O(n^2)$ T.C. in best case
- Unstable
- n-1 swaps
- Cannot Optimise

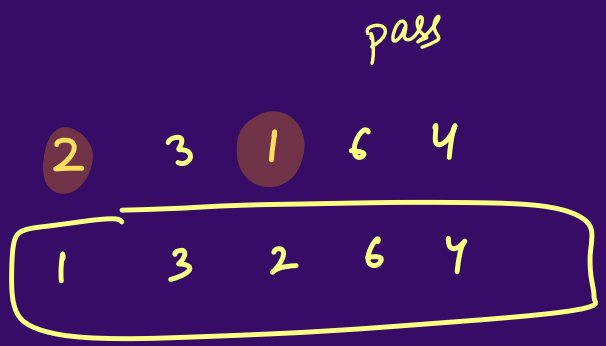
Insertion Sort

- O(n) T.C. in best case
- Stable
- $\frac{n(n-1)}{2}$ max swaps
- Always optimised

Ques:

Q4 : What will the array look like after the first iteration of selection sort [2,3,1,6,4]

a) [1,2,3,6,4]
b) [1,3,2,4,6]
c) [1,3,2,6,4]
d) [2,3,1,4,6]



Ques:

PW SKILLS

Q5 : Which sorting technique is used here?
A player is sorting a deck of cards numbered from 1 to 52. She first picks one card then picks the next card and puts it after the first card if if it is bigger or before the first card if it is smaller, then she picks another card and puts it into its proper position.

a) Bubble sort
b) Insertion sort
c) Selection sort
d) None of these

Ques:

Q6 : Which of the following is not a stable sorting algorithm?

a) Insertion sort
b) Selection sort
c) Bubble sort
d) None of these

Ques:

Q7: Majority Element

[LeetCode 169]

Ques:

PW SKILLS

Q8 : Given an array with N distinct elements, convert the given array to a form where all elements are in the range from 0 to N-1. The order of elements is the same, i.e., 0 is placed in the place of the smallest element, 1 is placed for the second smallest element, ... N-1 is placed for the largest element.

↳ all are +ve

n = 6
size of array

#Hint :

Selection Sort

	0	1	2	3	4	5
arr =	54	11	28	47	91	63
↳	3	0	1	2	5	4

SKILLS

'n' passes
↓
in every pass iterate over every +ve element

0	1	2	3	4	5
54	11	28	91	47	63
54	0	28	91	47	63
54	0	-1	91	47	63
54	0	-1	91	-2	63
-3	0	-1	91	-2	63
-3	0	-1	91	-2	-4
-3	0	-1	-5	-2	-4
3	0	1	5	2	4

x -1

int x = 0
↓
~~x++~~ x--

x = 0 -1 -2 -3 -4 -5

THANK YOU