PW SKILLS

C++
SELECTION & INSERTION SORT

Lecture-22

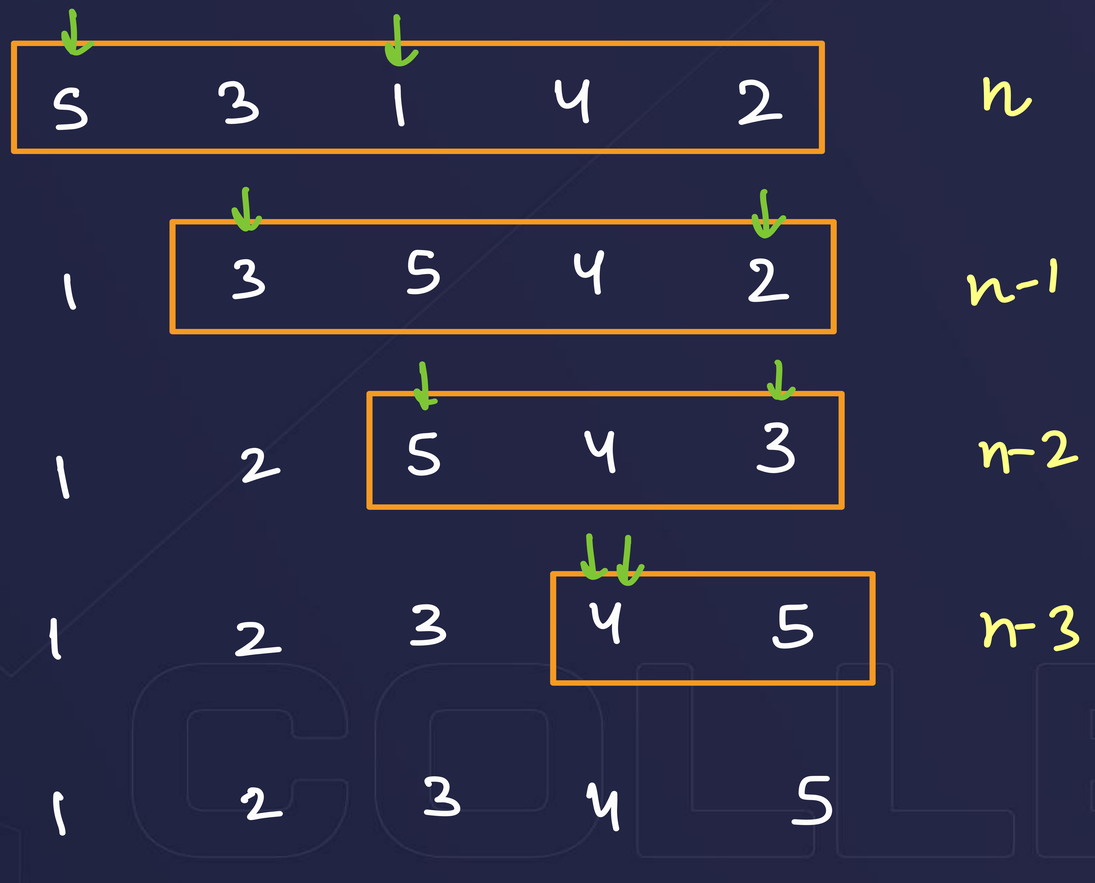
Raghav Garg

Today’s checklist

1) Sorting
2) Selection sort Algorithm
3) Time complexity and space complexity
4) Insertion sort Algorithm
5) Time complexity and space complexity
6) Stability of both

Selection Sort Algorithm

Array size → n



Steps-

- Orange Array me se min ele ko first ele ke sath swap
- 'n-1' total swaps

Selection sort Code and dry run

n=4

```
// selection sort
for(int i=0;i<n-1;i++){
    int min = INT_MAX;
    int mindx = -1;
    // minimum element calculation in orange box
    for(int j=i;j<n;j++){
        if(arr[j]<min){
            min = arr[j];
            mindx = j;
        }
    }
    swap(arr[i],arr[mindx]);
}
```

i<3
i=0, 1, 2

arr

0	1	2	3
4	-2	9	6
-2	4	9	6
-2	4	9	6
-2	4	6	9

i = 0 1 2 3

min = ~~INT_MAX~~ 4 -2 ~~INT_MAX~~ 4 ~~INT_MAX~~ 9 6

mindx = ~~-1~~ 0 1 ~~1~~ 1 ~~1~~ 2 3

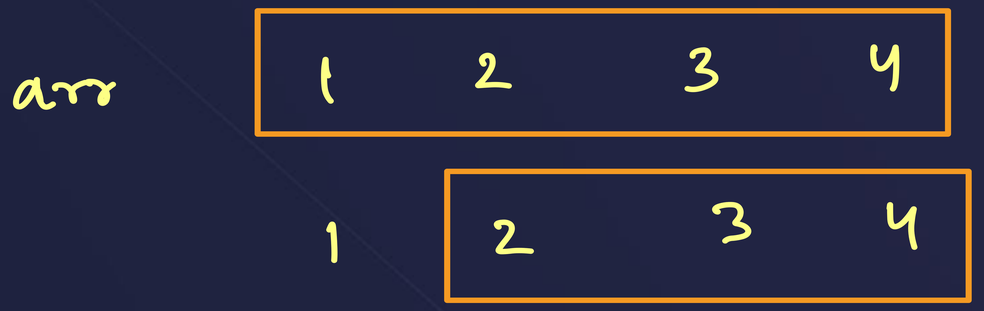
Time and Space complexity

Time Complexity	
Best Case	$O(n^2)$
Avg. Case	$O(n^2)$
Worst Case	$O(n^2)$

Space Complexity

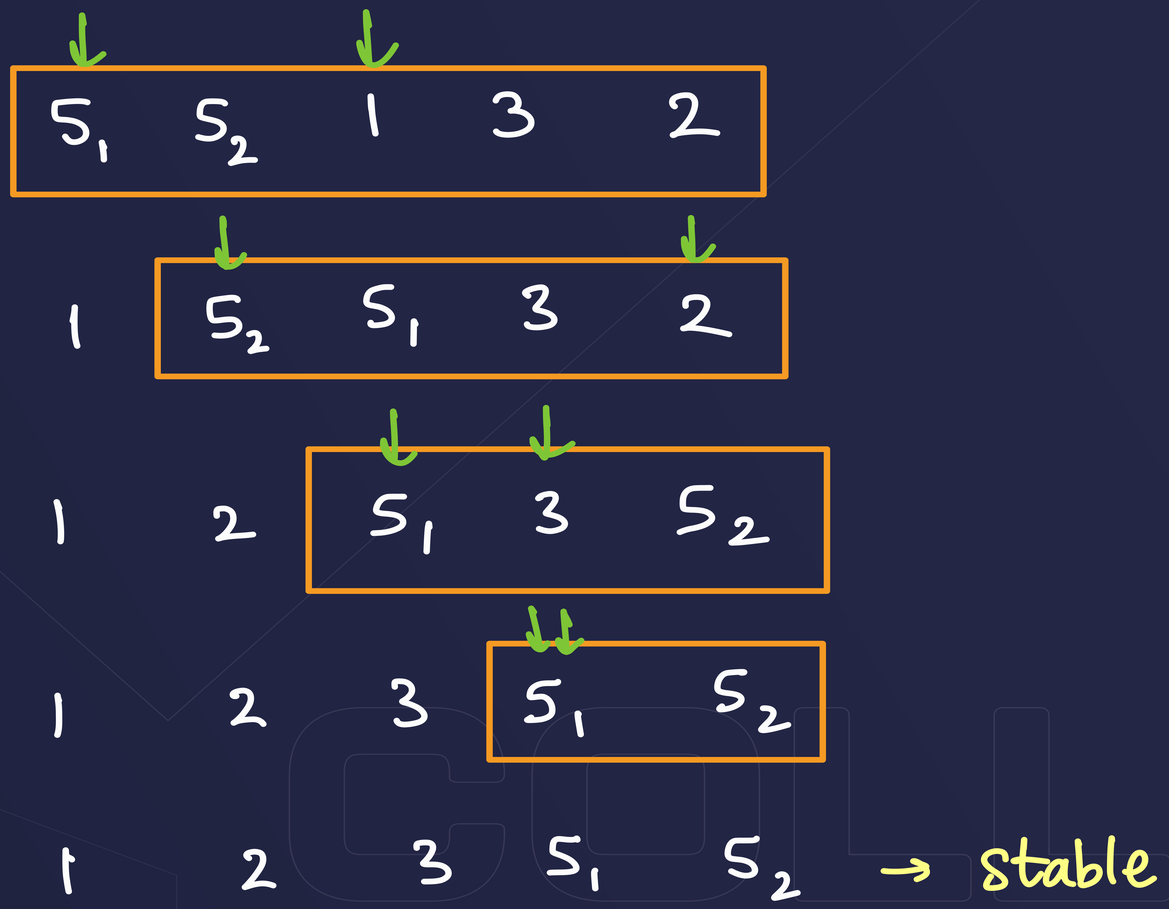
$O(1)$

Time and Space complexity

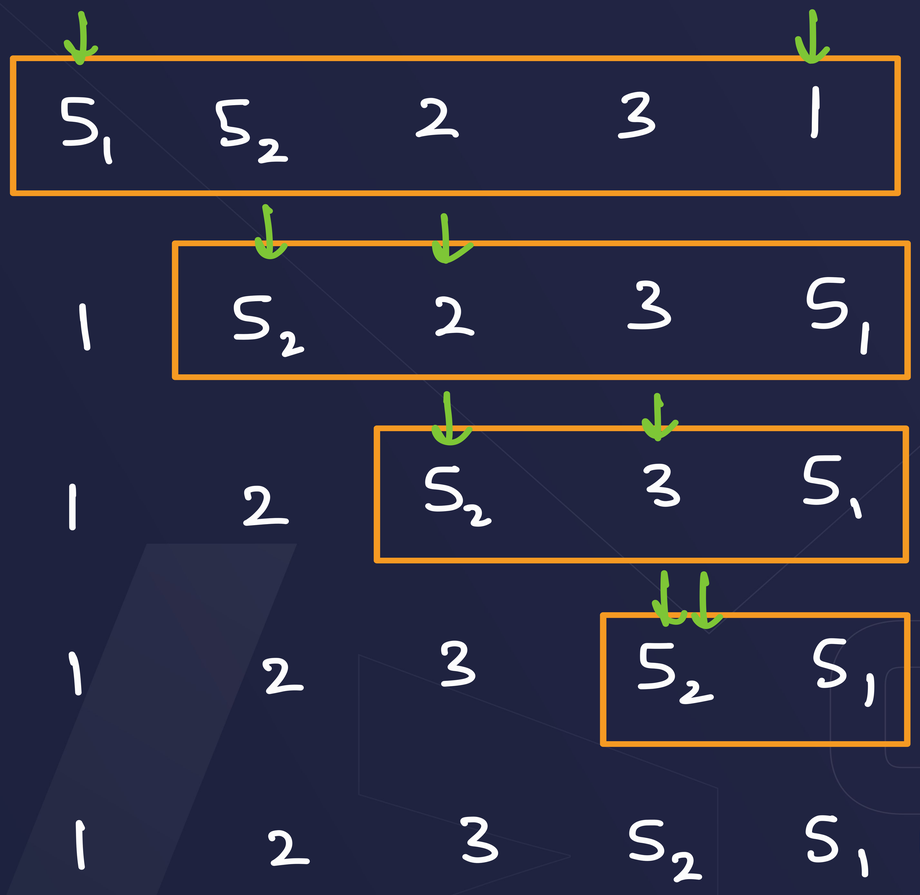


Stability of Selection Sort

Selection Sort Algorithm



Selection Sort Algorithm



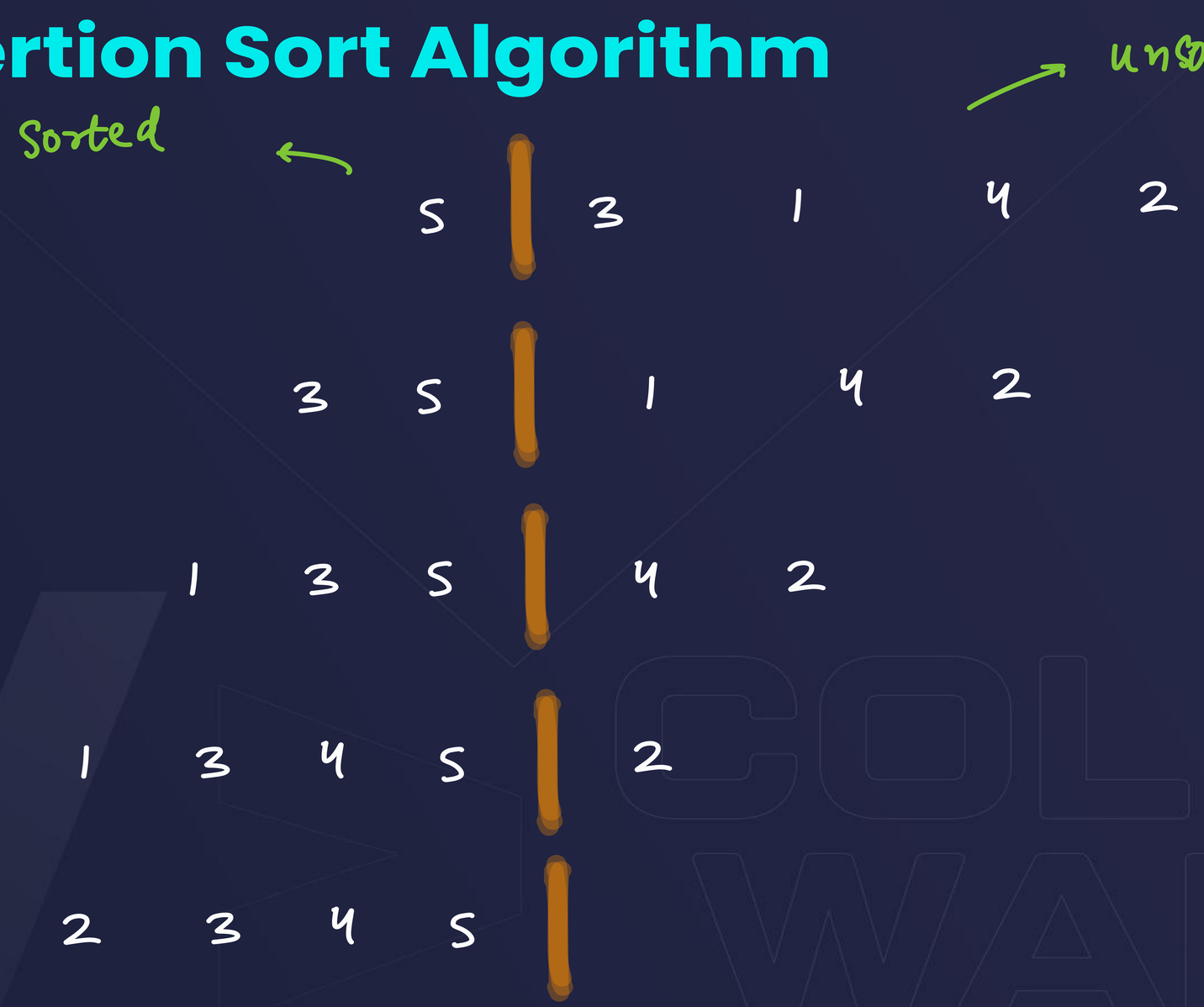
→ unstable

Usecases : Cost of swapping ✓

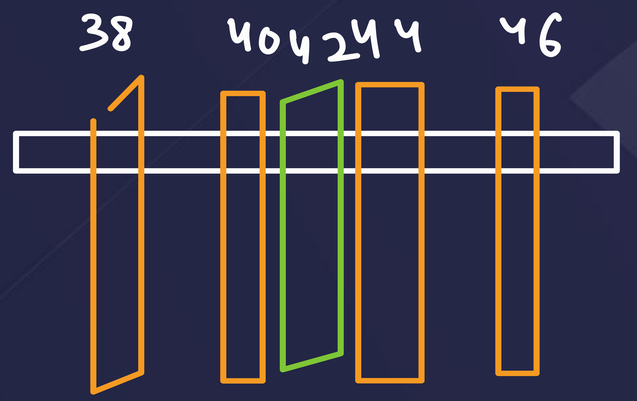
Starting se 'k' min de out of n

If size of array is small

Insertion Sort Algorithm



Insertion Sort Algorithm



5 3 1 4 2

3 5 1 4 2

3 1 5 4 2

1 3 5 4 2

1 3 4 5 2

1 3 4 2 5

1 3 2 4 5

1 2 3 4 5 (Sort)

Insertion Sort Algorithm


```
for(int i=1; i<=n-1 ;i++){
    int j = i ;
    while ( j >= 1 ){
        if(arr[j] >= arr[j-1]) break;
        if(arr[j] < arr[j-1])
            Swap(arr[j], arr[j-1]);
        j--;
    }
}
```

Insertion sort Code and dry run

```
// insertion sort
for(int i=1;i<n;i++){
    int j = i;
    while(j>=1 && arr[j]<arr[j-1]){
        swap(arr[j],arr[j-1]);
        j--;
    }
}
```

→ n-1 times

n = 4

arr =

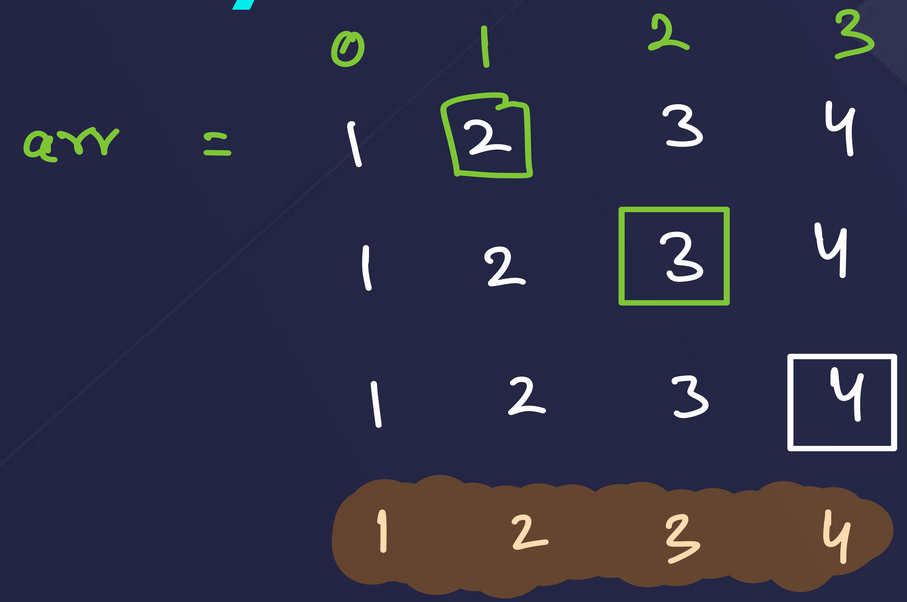
0	1	2	3
4	3	2	1
3	4	2	1
3	2	4	1
2	3	4	1
2	3	1	4
2	1	3	4
1	2	3	4

i = 1 2 3 4

j = 1 0 2 1 0 3 2 1 0

Insertion sort Code and dry run

```
// insertion sort
for(int i=1;i<n;i++){
    int j = i;
    while(j>=1 && arr[j]<arr[j-1]){
        swap(arr[j],arr[j-1]);
        j--;
    }
}
```



i = 1 2 3 4

j = 1 2 3

Time and Space complexity

Worst Case → $O(n^2)$

Avg. Case → $O(n^2)$

Best Case → $O(n)$

Stability of Insertion ~~and Selection~~ Sort

only adjacent swaps just like bubble sort

Stable Sorting Algorithm

4_1 4_2 2 1

4_1 2 4_2 1

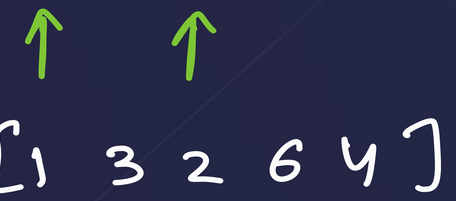
2 4_1 4_2 1

2 4_1 1 4_2

2 1 4_1 4_2

1 2 4_1 4_2 'Stability'

Ques : 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 : Sort a String in decreasing order of values associated after removal of values smaller than X.

Repeat

Classwork

Reverse

THANK YOU