C++
SELECTION & INSERTION SORT

Lecture-22

Raghav Garg

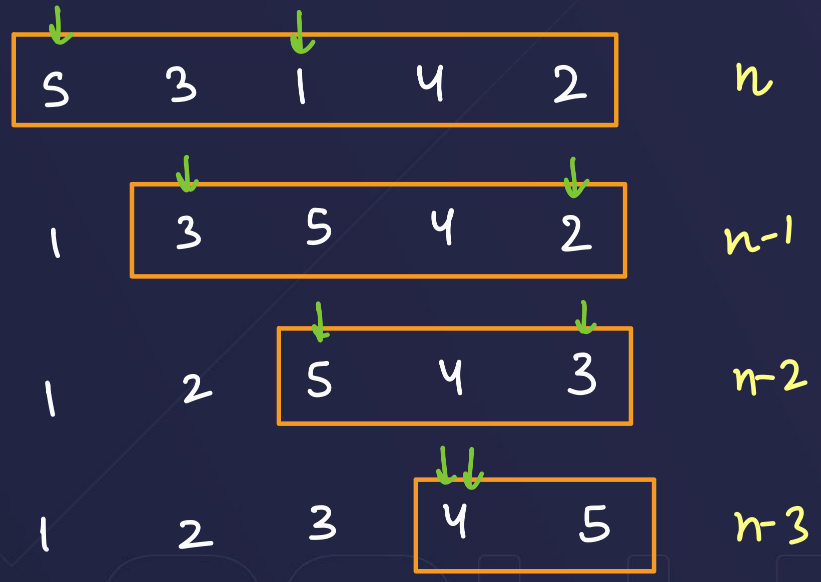
Today’s checklist

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

arr =



5 3 1 4 2 n

1 3 5 4 2 n-1

1 2 5 4 3 n-2

1 2 3 4 5 n-3

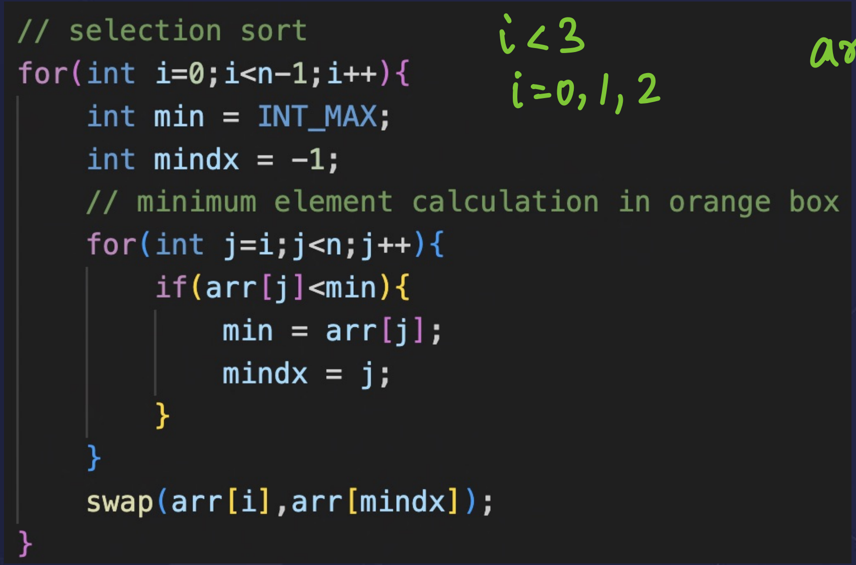
Sorted : 1 2 3 4 5

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

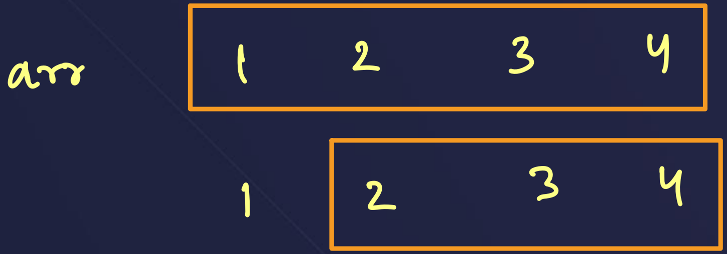
Time Complexity

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

$5_1 \quad 5_2 \quad 1 \quad 3 \quad 2$

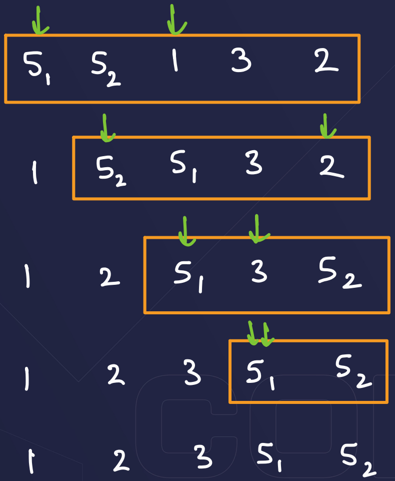
sort

$1 \quad 2 \quad 3 \quad 5_1 \quad 5_2$ stable sort

sort

$1 \quad 2 \quad 3 \quad 5_2 \quad 5_1$ unstable sort

Selection Sort Algorithm



5_1 5_2 1 3 2

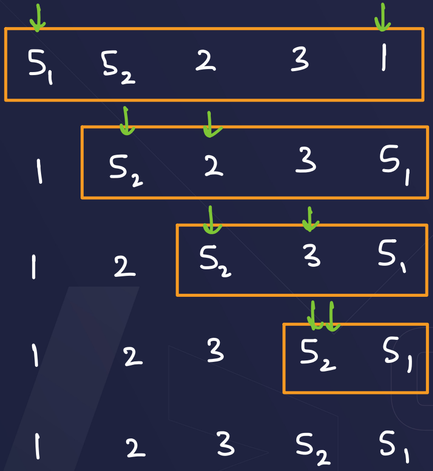
1 5_2 5_1 3 2

1 2 5_1 3 5_2

1 2 3 5_1 5_2

1 2 3 5_1 5_2 → stable

Selection Sort Algorithm



S_1 S_2 2 3 1

1 S_2 2 3 S_1

1 2 S_2 3 S_1

1 2 3 S_2 S_1

1 2 3 S_2 S_1 → unstable

<u>Usecases</u> : Cost of swapping ✓

Starting se 'k' min ele out of n

If size of array is small

Insertion Sort Algorithm

Sorted ←

→ unsorted

5 | 3 1 4 2

3 5 | 1 4 2

1 3 5 | 4 2

1 3 4 5 | 2

1 2 3 4 5 |

Insertion Sort Algorithm

5 3 1 4 2

3 5 1 4 2

3 1 5 4 2

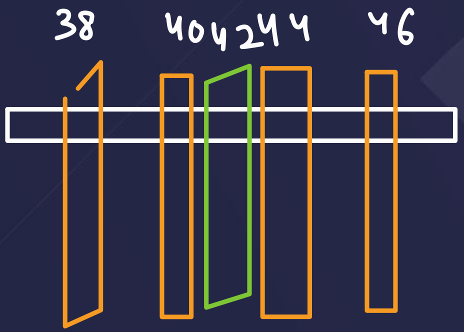
1 3 5 4 2

1 3 4 5 2

1 3 4 2 5

1 3 2 4 5

1 2 3 4 5 (Sort)



Insertion Sort Algorithm

5 $\boxed{3}$ 1 4 2

3 5 $\boxed{1}$ 4 2

3 1 5 4 2

1 3 5 $\boxed{4}$ 2

1 3 4 5 $\boxed{2}$

1 3 4 2 5

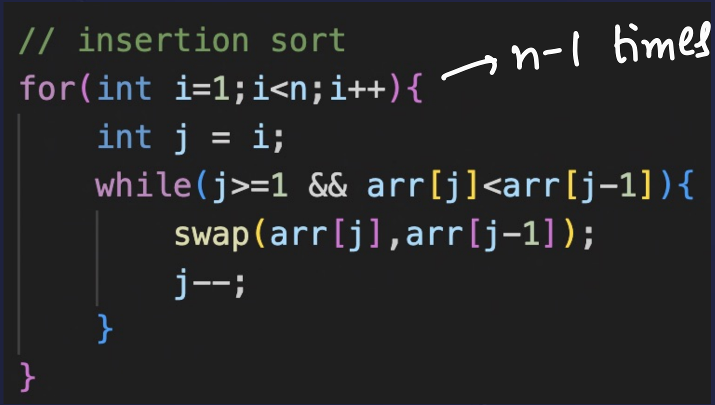
1 3 2 4 5

1 2 3 4 5

```
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--;
    }
}
```

n=4

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--;
    }
}
```

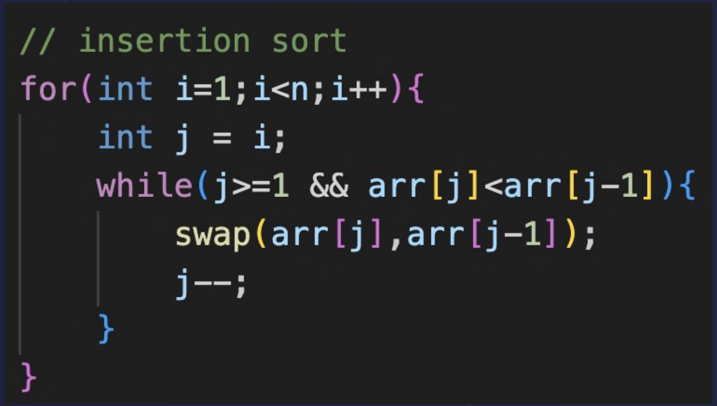
→ n-1 times

	0	1	2	3
arr =	4	$\boxed{3}$	2	1
	3	4	$\boxed{2}$	1
	3	2	4	1
	2	3	4	$\boxed{1}$
	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--;
    }
}
```

$$\begin{array}{ccccc} & 0 & 1 & 2 & 3 \\ arr = & 1 & \boxed{2} & 3 & 4 \\ & 1 & 2 & \boxed{3} & 4 \\ & 1 & 2 & 3 & \boxed{4} \\ & 1 & 2 & 3 & 4 \end{array}$$

$i = \cancel{1}\ \cancel{2}\ \cancel{3}\ 4$

$j = \cancel{1}\ \cancel{2}\ 3$

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]

↑ ↑

[1 3 2 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.

↓

Classwork

↓

Reverse

Repeat

THANK YOU