5/3/2024

PAGE No.
DATE / /

Day 8 of DSA

Tasks :-

Checkbox	Sorting :-
[x]	Sorting Algorithms : Insertion Sort, Selection Sort.
[x]	Implement insertion sort, selection sort.
[x]	Searching Techniques : Linear search, binary search
[x]	Practice linear search, basic Binary search

PAGE No.
DATE 1 1

eg. taken from Love Babbar Bhaiya

Insertion Sort :-

eg:

0	1	2	3
4	12	11	20

Inserting Sort 4 12 20
11

mean ;
We have to insert a particular as per its position.

Algorithm :-

(1) Traverse from i=1 to i<n. 1st loop

(2) Take temp & store arr[i];

(3) then Take another loop :
start loop j=i-1 to i>=0

(4) search is arr[j] <> arr[i]
then arr[j+1] = arr[j]

else we break;

(5) After loop we put temp in arr[j+1].

PAGE No.
DATE / /

```
For( int i = 1; i < n; i++)
{
    int temp = arr[i];
    int j = i-1;
    For(j; j >=0; j--)
    {
        IF( arr[j] > arr(i))
        {
            arr[j+1] = arr[j];
        }
        else
        {
            break;
        }
    }
    arr[j+1] = temp;
}
```

Time Complexity : $O(N^2)$
Space Complexity : $O(1)$

PAGE No.
DATE 1 1

* Selection Sort :-

Algorithm :-

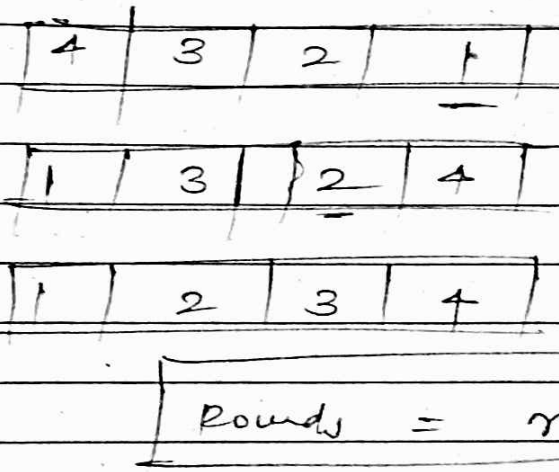
(1) We start traversing Array & checking the min d index from i+1 to n.

(2) & then swap it with current index

(3) That's it.

Basically, selection sort linearly Arrays elements by their size

eg



$$\text{Rounds} = n - 1$$

PAGE No.
DATE 1 1

```
For( int i = 0 ; i < n ; i++)
{
    int minIdx = i;
    For( int j = i+1 ; j < n ; j++)
    {
        if (arr[j] < arr[minIdx])
        {
            minIdx = j;
        }
    }
    Swap( arr[minIdx], arr[i]);
}
```

Time complexity: $O(n^2)$
Space complexity: $O(1)$

PAGE No.
DATE / /

* Linear Search

* we have to traverse whole array & checks every element that is equal or not

$$TC = O(N)$$

* Binary search

* we Find out mid of Array

+ & checks if element is < mid then we search in left

+ otherwise right.

+ & search it using mid - Again & Again

$$TC = O(n \log n)$$

PAGE No.
DATE 1 1

(1) Implement Insertion sort to sort an array of integers in non-decreasing order.

→

```
int i, j, temp;

For( i = 0; i < n; i++)
{
    temp = arr[i];

    for(j = i-1; j >= 0; j--)
    {
        arr[j+1] = arr[j];
    }

    arr[j+1] = temp;
}
```

Time Complexity: O(n2)
Space Complexity: O(1)

PAGE No.
DATE 1 1

(2) Implement Sort of selection for an array of integers in non-decreasing order

→ Approach :-

* Use Basic selection sort.

that is Find min-idx & put it swap with current element.

* ~~Algo~~

```
For( int i = 0; i<n; i++)
{
    int min_idx = i;
    For( int j = i+1; j<n; j++)
    {
        if ( arr[min_idx] > arr[j])
        {
            min_idx = j;
        }
    }
    swap (arr[i], arr[min_idx]);
}
```

Time complexity : $O(n^2)$
Space complexity : $O(1)$

PAGE No.
DATE 11

(3) kth smallest element in an array.
Using Insertion Sort

→

Approach:-

```
For( int i=0; i<n; i++)
{
    int i,j, temp;
    temp = arr[i];
    j = j-1;

    while(j>=0 && arr[j] > temp)
    {
        arr[j+1] = arr[j];
        j--;
    }
    arr[j+1] = temp;
}

return arr[k-1];
```

Time complexity: O(n²)
~~Time~~
Space Complexity: O(1)

PAGE No.
DATE / /

(4) Kth smallest element in an array.
Using Selection sort.

→

Approach

```
For( int i = 0; i < n; i++)
{
    int min_idx = i;
    for(int j = i+1; j < n; j++)
    {
        if (arr[min_idx] >> arr[j])
        {
            min_idx = j;
        }
    }
    swap(arr[min_idx] arr[j]);
}

return arr[K-1];
```

Time complexity : $O(n^2)$
space complexity : $O(1)$