Day-17

Insertion Sort

=> Partially sorting the array

Eg:

0	1	2	3	4
5	3	4	1	2

1st pass: 3, 5, 4, 1, 2
i=0

2nd pass: 3, 4, 5, 1, 2
i=1

3rd pass: 1, 3, 4, 5, 2
i=2

4th pass: $\boxed{1, 2, 3, 4, 5}$
i=3

Sorted array

For every index:

Put that index element at the correct index of LHS

Outer loop:

		i	j
Index 1	Pass 1	0	1
Index 2	2	1	
3	3	2	
4	4	3	
5	5	4	

Working:

	i < (n-2)	j > 0
0 1 2 3 4 5, 3, 4, 1, 2 Swap 3<5 3, 5, 4, 1, 2	0	1
3, 5, 4, 1, 2 4<5 Swap 3, 4, 5, 1, 2	1	2
3, 4, 5, 1, 2 Swap 1<5 3, 4, 1, 5, 2 ↓ 1<4 3, 1, 4, 5, 2 ↓ 1<3 1, 3, 4, 5, 2	2	3
1, 3, 4, 5, 2 Swap 2<5 ↓ 1, 3, 4, 2, 5 ↓ 2<4 1, 3, 2, 4, 5 2<3 ↓ Sorted $\boxed{1, 2, 3, 4, 5}$ Sorted Break	3	4

Time complexities:

Best case: Array is already sorted   1 2 3 4 5

(N-1) Linear

$O(N)$

Worst case: Desc sorted   5 4 3 2 1 $\frac{N(N+1)}{2}$ comparisons

$$\frac{(N-1)(N-1+1)}{2} = \frac{N^2 - N}{2} \Rightarrow O(N^2)$$

Why we use?

→ Adaptive: No. of swaps reduced compared to bubble sort

→ Stable:

→ Used for smaller values of N works good when array is partially sorted. In Hybrid Sorting algo

Program:

```
import java.util.Arrays;
public class insertionsort{
  public static void main(String[] args){
      int[] arr={3,1,4,5,2};
      insertion(arr);
      System.out.println(Arrays.toString(arr));
  }
  static void insertion(int[] arr){
      for(int i=0; i<arr.length-1; i++){
          for(int j=i+1; j>0; j--){
              if(arr[j]<arr[j-1]){
                  int temp=arr[j];
                  arr[j]=arr[j-1];
                  arr[j-1]=temp;
              }else{
                  break;
              }
          }
      }
  }
}
```

[1, 2, 3, 4, 5]

if (arr[j] > arr[j-1])

→ [5, 4, 3, 2, 1]

Selection Sort: Select element & put it on its correct index

0 1 2 3 4

(n-1) 4, 5, 1, 2, 3 — Swap

* Check the largest element & put it in correct index place (i.e) index = 4

(n-2) 4, 3, 1, 2, $\boxed{5}$ — Swap

* Check 2nd largest element

(n-3) 2, 3, 1, $\boxed{4, 5}$ — Swap

(n-4) 2, 1, $\boxed{3, 4, 5}$ — Swap

⋮

0 $\boxed{1, 2, 3, 4, 5}$

Sorted array!

Also select minimum element & put it in correct index

0 1 2 3 4

4, 5, 1, 2, 3 — Swap

=> 1, 5, 4, 2, 3

1, 2, 4, 5, 3

1, 2, 3, 5, 4

=> $\boxed{1, 2, 3, 4, 5}$ Sorted

Total comparison: $0 + 1 + 2 \cdots + (n-1)$

$$\frac{(n-1) * (n \cancel{+1} \cancel{-1})}{2}$$

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

$O(n^2)$ Constant removed

Time complexities: Worst case: $O(N^2)$ Stable = No

Best case: $O(N^2)$

Performs well on small list / array.

=> In this sort, largest number move to large index place by swapping method.

=> In best & worst case, both are same $O(N^2)$

Program:

```
import java.util.Arrays;
public class selection{
    public static void main(String[] args){
        int[] arr = {64,25,12,22,11};
        selectionsort(arr);
        System.out.println(Arrays.toString(arr));
    }
    static void selectionsort(int[] arr){
        for(int i = 0; i < arr.length-1; i++){
            int min = i;
            for(int j = i+1; j < arr.length; j++){
                if(arr[j] < arr[min]){
                    min = j;
                }
            }
            int temp = arr[min];
            arr[min] = arr[i];
            arr[i] = temp;
        }
    }
}
```

$i < min$

Asc order

[11, 12, 22, 25, 64]

$j > min$

Desc order

[64, 25, 22, 12, 11]