Sorting

TABLE OF CONTENTS

Java

```java
static void change(int a) {
    a = 50;
}

public static void main(String args[]) {
    int a = 10;
    change(a);
    System.out.println(a);
}
```

Python

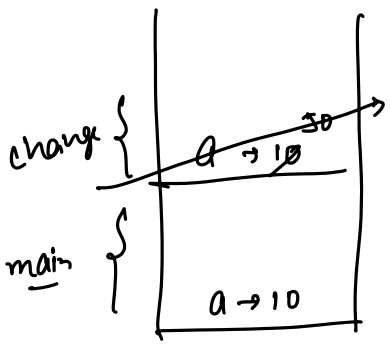
```python
def change(a):
    a = 50

def main():
    a = 10
    change(a)
    print(a)

if __name__ == "__main__":
    main()
```

o/p. → 10

Stack



Java

```java
static int[] fun(int[]a) {
    a = new int[2];
    a[0] = 50; a[1] = 60;
    return a;
}

public static void main(String args[]) {
    int[]a = {10,20,30};
    a = fun(a);
    System.out.println(a[0]);
}
```

Python

```python
def fun(a):
    a = [0, 0]
    a[0] = 50
    a[1] = 60
    return a

def main():
    a = [10, 20, 30]
    a = fun(a)
    print(a[0])

if __name__ == "__main__":
    main()
```

o/p. → 50

Sorting

↳ arrangement of data in any particular order on the basis of some parameter.

arr[] → [2, 3, 9, 12, 17, 19] parameter → value

arr[] → [19, 6, 5, 2, -1, -15] parameter → value

arr[] → [1, 13, 9, 6, 12] parameter → factors count.

factors → 1 2 3 4 6

Why sorting?

→ to make search faster.

→ analyze & represent the data

Java → Arrays.sort(arr);

C++ → sort(arr.begin(), arr.end())

Python → arr.sort()

} check the exact syntax.

T.C → O(N log N)

S.C → depends on inbuilt sorting algo.

Question (Elements Removal)

Given N elements, at every step remove an array element.

Cost to remove an element = Sum of array of elements present in an array

Find minimum cost to remove all elements.

NOTE : First add the cost of removal and then remove it.

arr - [2 1 4]

remove 2 ⇒ 7

remove 1 ⇒ 5

remove 4 ⇒ 4

total cost → 16

[2 1 4]

remove 4 → 7

remove 2 → 3

remove 1 → 1

total cost → 11

arr[] → [4, 6, 1]

remove 6 → 11

remove 4 → 5

remove 1 → 1

total cost → 17

arr[] → [3, 5, 1, -3]

remove 5 → 6

remove 3 → 1

remove 1 → -2

remove -3 → -3

total cost → 2

arr[] → [a b c d]

Remove a → a + b + c + d

Remove b → b + c + d

Remove c → c + d

Remove d → d

total cost → a + 2b + 3c + 4d

a > b > c > d.

to get the minimum cost ⇒ largest element should appear minimum no. of times.

Idea → Sort the arr. in decreasing order. Eg → [6, 4, 1] (0 1 2)

#code →

```
sort( arr, desc.)
 cost = 0;
for( i=0; i<N; i++){
{       cost += (arr[i] x (i+1));
}
return cost;
```

T.C → O(N log N)
S.C → depends on inbuilt sorting algo

Question (Noble Integers) { Distinct data }

Given N array elements, calculate number of noble integers.

An element ele in arr [ ] is said to be noble if { count of smaller elements = ele itself }

arr - [1 , -5 , 3 , 5 , -10 , 4]

count of smaller elements ⇒ 2 1 3 5 0 4

ans = 3

arr - [-3 , 0 , 2 , 5]

count of smaller ele → 0 1 2 3

ans = 1

BF idea. -

for every element, iterate and find the count of elements smaller than current element.

```
ans = 0
for ( i = 0; i < N; i++) {
    count_smaller = 0;
    for ( j = 0; j < N; j++) {
        if ( arr[j] < arr[i] ) { count_smaller ++ }
    }
    if ( count_smaller == arr[i] ) { ans ++ }
}
return ans;
```

T.C → $O(N^2)$
S.C → $O(1)$

optimisation

if array is sorted in ascending order.

Count of smaller elements for arr[i] → i

#code:-

```
sort(arr, asc)
ans = 0;
for(i=0; i<n; i++){
    if(arr[i] == i){ ans++}
}
return ans;
```

T.C → O(NlogN)

S.C → depends on sorting algo

Question (Noble Integers) : { Data can repeat }

arr - [-10 , 1 , 1 , 3 , 100]

0 1 2 3 4

ans = 3

arr - [-10 , 1 , 1 , 2 , 4 , 4 , 4 , 8, 10]

0 1 2 3 4 5 6 7 8

ans = 5

arr - [-3 , 0 , 2 , 2 , 5 , 5 , 5 , 5 , 8 , 8 , 10 , 10 , 10 , 14]

0 1 2 3 4 5 6 7 8 9 10 11 12 13

ans = 7.

count_smaller = 0 1 2 4 8 10 13

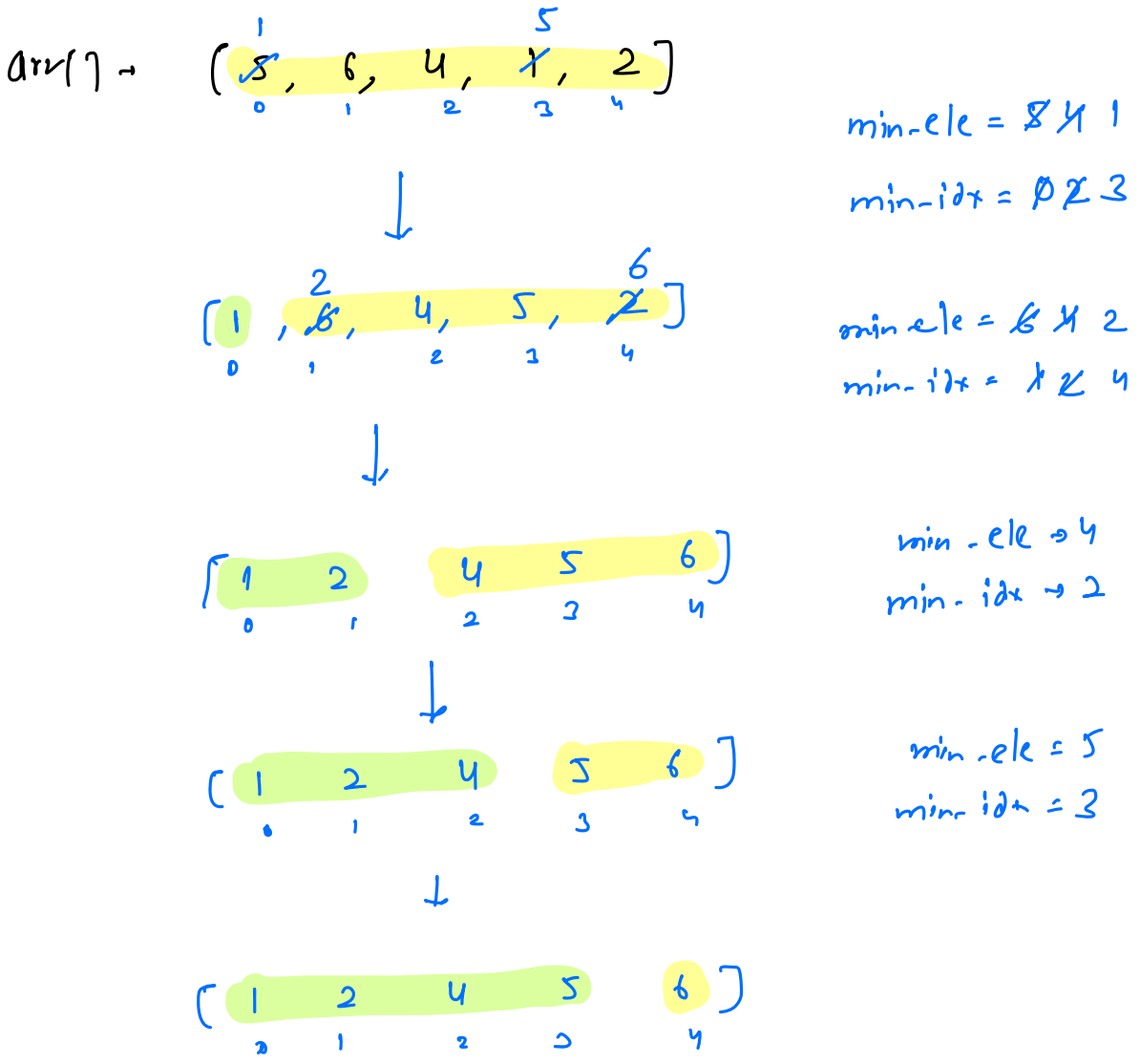
< / > Code

```
Arrays.sort(arr, asc.);
count_smaller = 0, ans = 0;
if( arr[0] == 0) { ans ++ }
for( i = 1; i < n; i++){
        if( arr[i] != arr[i-1]){
            count_smaller = i;
        }
        if( arr[i] == count_smaller) { ans ++ }
}
return ans;
```

T.C → O(N log N)
S.C → depends on sorting algo

Selection Sort

idea : Select the minimum element and send that elements to correct position by swapping.



</> Code

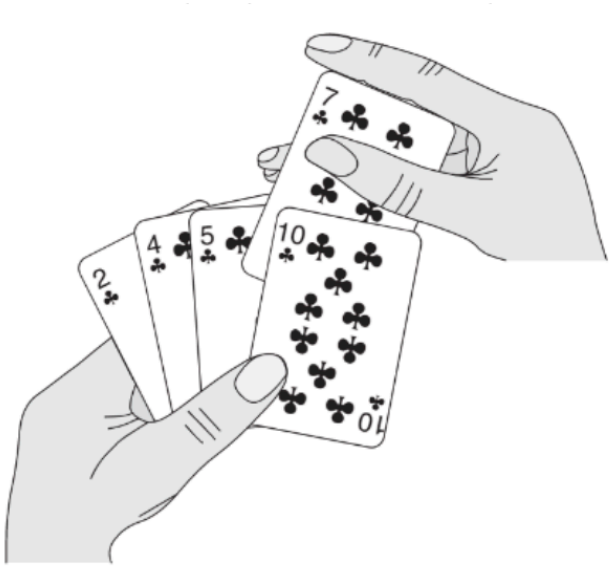
```
for( i = 0; i < N-1 ; i++) {
    min_element = arr[i];
    min_idx = i;
    for( j = i+1; j < N; j++){
        if( arr[j] < min_element) {
            min_element = arr[j];
            min_idx = j;
        }
    }
    swap( arr[i] with arr[min_idx]);
}
```

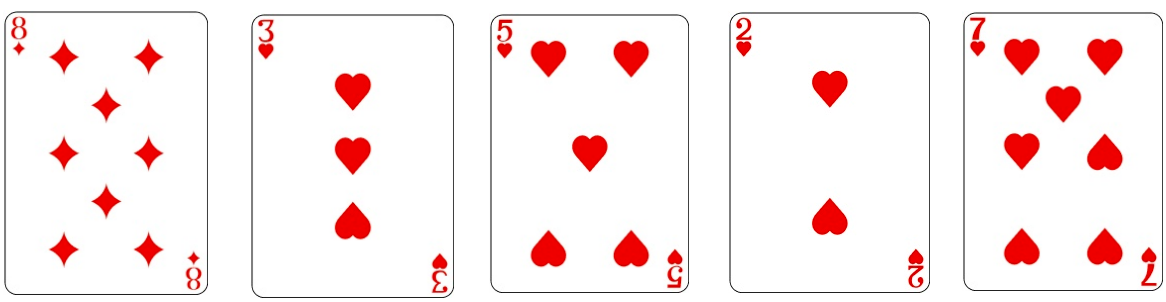
selecting the min element from idx → i to N-1

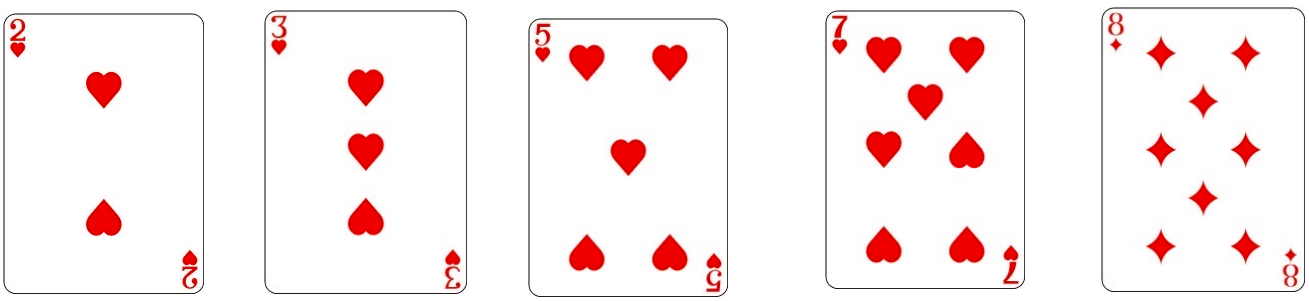
T.C → $O(N^2)$

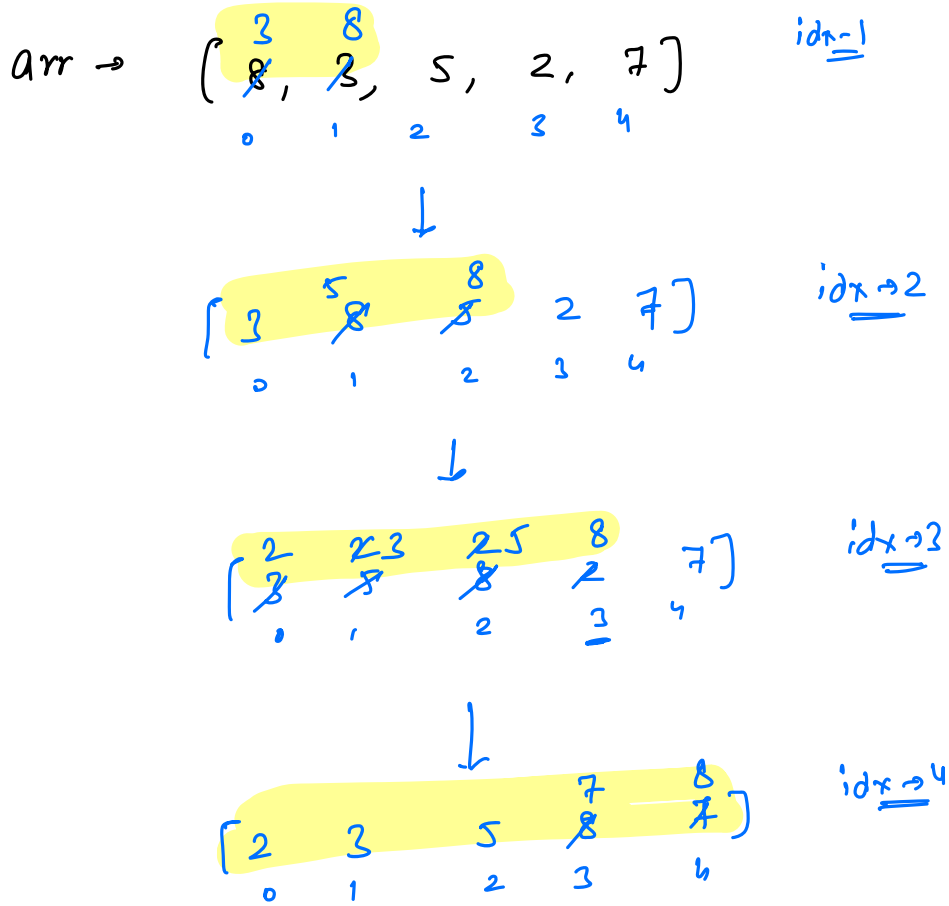
S.C → $O(1)$

Insertion Sort (Arrangement of playing cards)





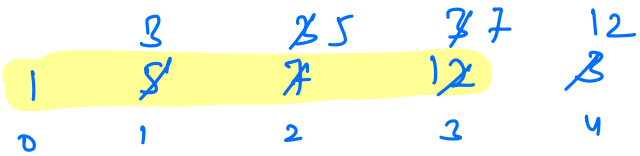




#code→

```
for( i=1 ; i<N; i++){
    for( j=i-1 ; j≥0 ; j--){
        if( arr[j] > arr[j+1] ){
            swap( arr[j] with arr[j+1] );
        }
        else{
            break;
        }
    }
}
```

[T.C → O(N²)
S.C → O(1)]



arr[] = [1 5 12 15 20]
0 1 2 3 4

But if array is already sorted, insertion sort will take N iterations only.

otherwise in worst case T.C of insertion sort will be N^2.