- Selection
- Bubble
- Merge

Sorting —

Arrangement of Data in increasing / decreasing order. on basis of a criteria

Ex 1: 1 5 9 13 21 ⟵ > Criteria : Value

Ex 2: 21 16 5 2 −1 ⟵

Ex 3: 1 2 11 9 6 10 ⟵ ⇒ Criteria : no of factors

Factors 1 2 2 3 4 4

#

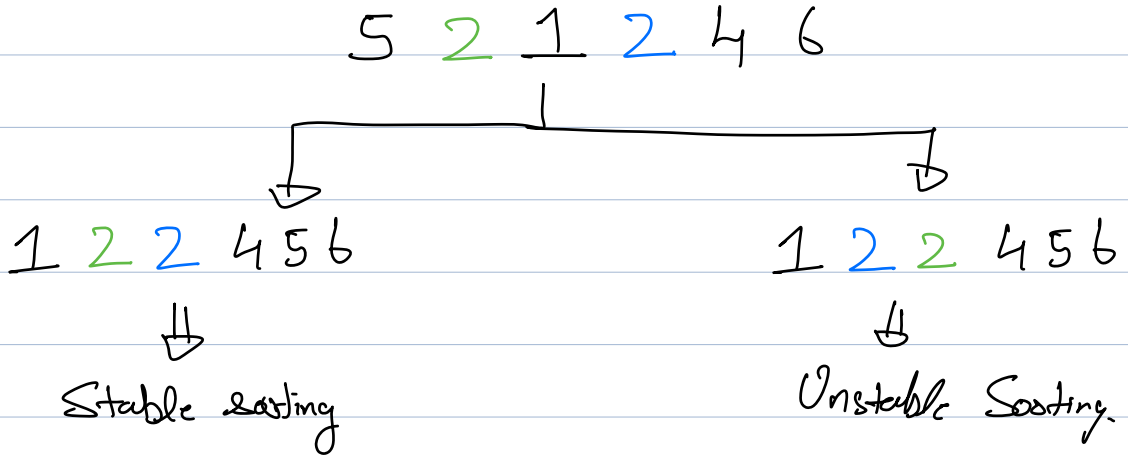
Virat Kohli ⇒ 250
Jos Buttler ⇒ 400
Hardik Pandya ⇒ 400
Rohit Sharma ⇒ 600
Shreyas Iyer ⇒ 250
Rishabh Pant ⇒ 300

Stable Sorting →

Virat Kohli ⇒ 250
Shreyas Iyer ⇒ 250
Rishabh Pant ⇒ 300
Jos Buttler ⇒ 400
Hardik Pandya ⇒ 400
Rohit Sharma ⇒ 600

Stable Sorting : Maintaining the relative order of elements having similar value.

Stable Sorting



In-place

We call an algorithm inplace if SC: O(1).

Q1) Given N array elements. Find the k^{th} smallest element.

$k < \log n$

Ex1 arr[5] : 3 8 2 4 -1

k = 2 : ans : 2

Approach 1 : 1) Sort the arr. in ascending. | TC: O(nlogn)

2) return arr[k-1];

Approach 2 : Find minimum element one by one.

arr[5] : 3 8 2 4 -1

1) Find 1st min : -1 [0, n-1] TC: O(nk)

-1 8 2 4 3 < nlogn

2) Find 2nd min : 2 [1, n-1]

-1 2 8 4 3

3) Find 3rd min ⇒ 3 [2, n-1]

-1 2 3 4 8

4) Find 4^{th} min $\Rightarrow$ 4 $[3, n-1]$

-1 2 3 4 8

Inplace ↪

Stable X

Ex 1 : 1 3 2 2 [choosing last occurrence minimum]

↓

1 3 2 2 X

↓

1 2 2 3

Ex 2 : 1 2 2 1 [choosing first + occurrence minimum]

↓

1 2 2 1

↓

1 1 2 2

↓

1 1 2 2

Pseudo Code

```
for (int i=0 ; i<(n-1) ; i++) {

    int min => arr[i]
    int ind => i;

    for (int j=i+1 ; j<n ; j++) {

        if ( arr[j] < min ) {
            min = arr[j]
            ind = j
        }
    }

    swap (arr[i], arr[ind]);

}
```

$$TC : (n)+(n-1)+(n-2)+ \ldots \quad 1$$
$$\simeq n^2$$

$$TC : O(n^2) \qquad SC : O(1)$$

Q2 Given an array of n elements. Sort it. You can only swap adjacent element.

5-0-1

$< n-i-1$

$n-i$

Ex1:

0	1	2	3	4
3	-1	6	2	4

Iteration 1 : -1 3 2 4 6

Iteration 2 : -1 2 3 4 6

$(n-1)$

Array is sorted.

Inplace : ⌙

Stable : ⌙

Ex1

2 1 2 4 5

1 2 2 4 5

$(n-i-1)$

Pseudo Code

```
for (int i=0 ; i<(n-1) ; i++ ) {

    bool swap = false

    for (int j=0 ; j<(n-i-1) ; j++) {

        if (arr[j+1] < arr[j]) {
            swap (arr[j+1], arr[j]); swap=true
        }
    }
    if (!swap)
        break;
}
```

Tc : $O(n^2)$

SC : $O(1)$

Q3 Given A[n] and B[m]. Both are sorted. Make an C[n+m] which is sorted and contains all element from A & B.

Ex1 : A[] : { -1, 2, 6 }
B[] : { 3, 4, 5 }

C[6] : { -1, 2, 3, 4, 5, 6 }

Approach 1 : 1) Put all elements in C
2) Sort C

Tc : ((n+m) log (n+m))

Approach 2

A[] : { -1, 2, 6, 7, 8 } ← P_1
B[] : { 3, 4, 5 } ← P_2

C[6] ⇒

0	1	2	3	4	5	6	7
-1	2	3	4	5	6	7	8

↑ P_3

Tc : O(n+m)

```
merge ( A[], n, B[], m) {

    int c[n+m];
    int p1 => 0;   int p2 => 0 , int p3 =>;

    while ( P1 < n && P2 < m ) {

        if ( A [P1] ≤ B [P2] ) {
            C [ P3 ] = A [P1];
            P1++ ; P3++
        } else {
            C [P3] = B [P2]
            P2 ++ , P3++ :
        }
    }

    while (P1 < n) {
        C[P3] = A[P1]
        P3++; , P1 ++:
    }

    while (P2 < m) {
        C[P3] - B[P2]
    } P3++; P2++:
}
```

Tc : O(n+m)

10:35pm

Q Given N array elements & 3 indices s, m, e.

→ $s \leq m$

Note: Subarray $[s, m]$ is sorted
Subarray $[m+1, e]$ is sorted.

You need to sort subarray $[s, e]$

Ex1:

0	1	2	3	4	5	6	7	8	9
4	8	10	11	12	5	6	7	21	

$s \Rightarrow 2$, $m \Rightarrow 4$, $e \Rightarrow 7$

Ans

0	1	2	3	4	5	6	7	8	9
4	8	5	6	7	10	11	12	21	

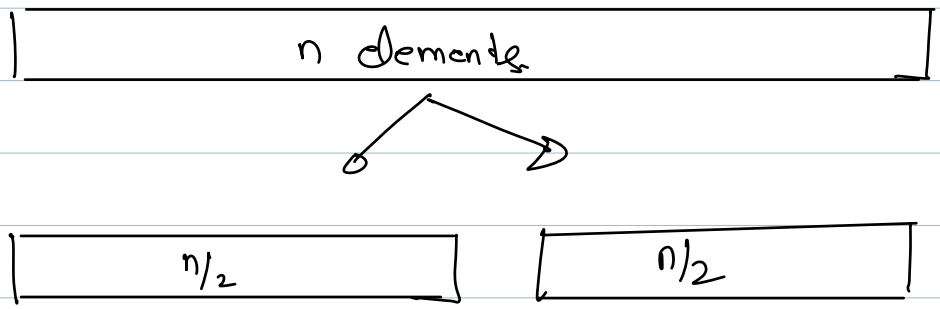
Pseudo Code :

```
merge ( A, s, m, e) {

    int temp [ e - s + 1 ]
    int p1 => s
    int p2 => m+1 , int p3 => 0.

    while ( P1 ≤ m && p2 ≤ e) {

        if (arr [P1] < arr [P2] ) {
            temp [P3] = arr [P1];
            P3++, P1++;
        } else {
            temp [P3] = arr [P2];
            P3++, P2++;
        }
    }

    while ( P1 ≤ m)
        temp[P3] = arr [P1]; P3++, P1++;
    while (P2 ≤ e)
        tem [P3] = arr [P2] ; P3++ P2++

    // Copy from temp
    to array.
}
```

TC : O(n)
SC : O(n)

$n = 100$

Selection $\Rightarrow$ 10000

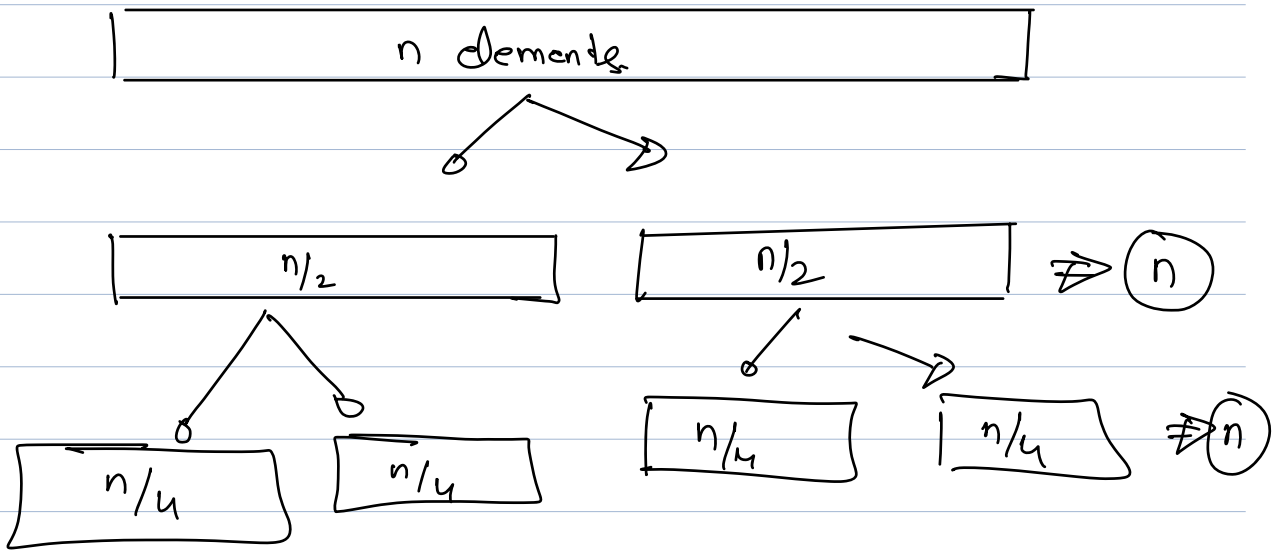


Step $\Rightarrow \left(\frac{n^2}{4}\right) + \left(\frac{n^2}{4}\right) + (n)$

$\left(\frac{n^2}{4}\right) + \left(\frac{n^2}{4}\right)$: 2 time selection

(n) : merge

$\Rightarrow \frac{n^2}{2} + n \Rightarrow 5100$

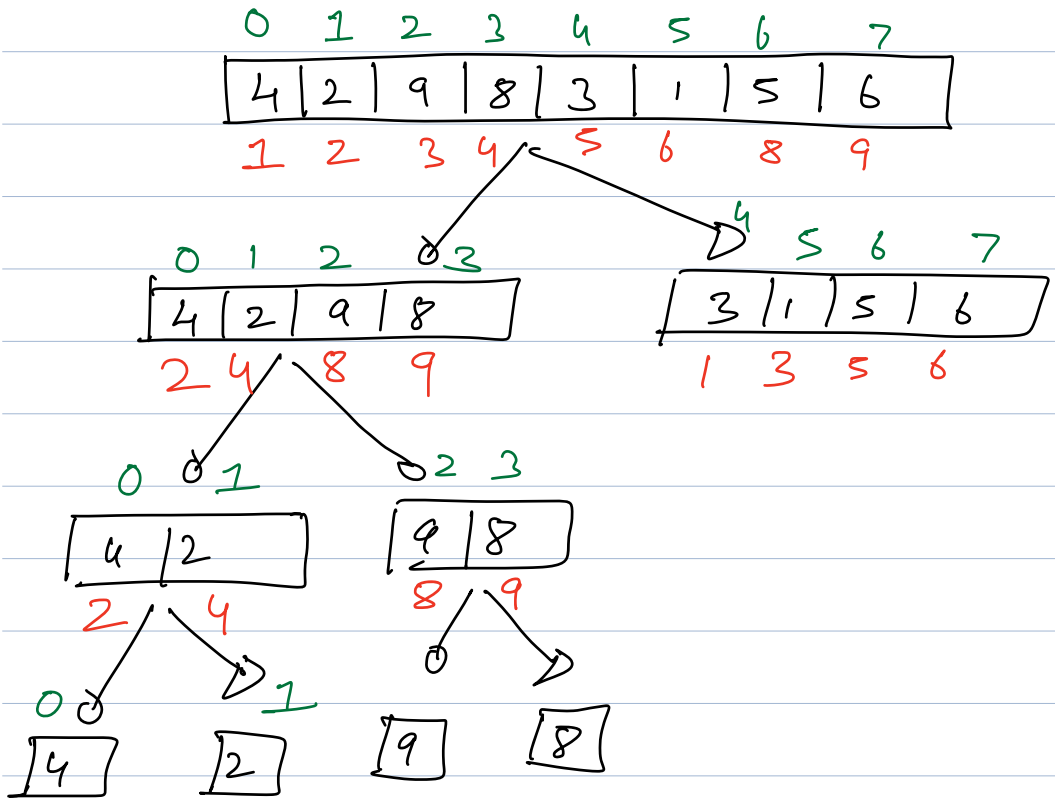


Steps $\Rightarrow$ $\left(\frac{n}{4}\right)^2 \times 4 + n + n$

$\Rightarrow$ $\frac{n^2}{16} \times 4 + 2n$

$\frac{n^2}{4} + 2n \Rightarrow$ 2700

$s \neq 0$, $c = 7$ $\Rightarrow$ $\frac{(s+c)}{2}$



Call stack $\Rightarrow$ $O(\log n)$

Temporary space $\Rightarrow$ $O(n)$

Pseudo Code

```
void mergeSort ( int arr[], int s, int e)

    if (s==e)
        return ;

    int mid => (s+e)/2

    mergeSort (arr, s, mid);
    mergeSort (arr, mid+1, e);

    merge (arr, s, mid, e);
```

3

$T(n) \Rightarrow 1 + 2T(n/2) + n$

Tc : $O(n \log n)$

Sc : $O(n)$

Stable

Inplace : X