Todays Content:

- → Sum of man of all subsequences ✓
- → Insertion sort ✓
- → Inversion Count ✓
- → Count Sort : Saturday

Q1) Given ar[N] calulate sum of max of every subsequence

Ex: ar[3] = { 3 1 -4 }

Sum of min every Subsequence → TODO

All subsequence →	max
{ }	0
{3}	3
{1}	1
{-4}	-4
{3 1}	3
{3 -4}	3
{1 -4}	1
{3 1 -4}	3
	Sum = 10

$= 3*4 + 1*2 + -4*1 = 10$

Idea1:

Generate all subseq, get max of each sub & add it

TC: $2^N * \{N\} \rightarrow O(2^N * N)$ SC: $O(1)$

Idea2: Contribution:

For every element, calulate no: of Subsequen in which ar[i] max

$$\sum_{i=0}^{n-1} ar[i] * \underline{c_i} \; ?$$

↳ # no: of subseq in which ar[i] max?

Ex: ar[] = { 4 7 2 5 8 10 }

# In how many subseq 7 is max?

elements < 7 : {4 2 5} = 3 ele = 2^3 = subseq

{ 7 }

{ 4 7 }

{ 2 7 }

{ 5 7 }

{ 4 2 7 }

{ 2 5 7 }

{ 4 5 7 }

{ 4 2 5 7 }

obs: We need to cal no: of elements less than that for each element, to do this sorting helps.

ar[] = { 4 7 2 5 8 10 }

$$\begin{array}{lcccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 \\
\text{sortar[] = \{} & 2 & 4 & 5 & 7 & 8 & 10 \text{ \}} \\
\text{\# elem < ar[i] =} & 0 & 1 & 2 & 3 & 4 & 5 \\
\text{\# subseq =} & 2^0 & 2^1 & 2^2 & 2^3 & 2^4 & 2^5
\end{array}$$

# Contribution = 2 + 8 + 20 + 56 + 128 + 320 = ______

```
int summaxSub(int ar[N]) { TC: O(N log N + N) ≈ O(N log N)
    sort(ar) // sort given arr
    sum = 0
    i = 0; i < n; i++ {
        // for ar[i], no: of sub, in which ar[i] is max = 2^i
        sum = sum + ar[i] * (1<<i)
    }
    return sum
}
```

<u>Note</u>: Even, if data repeats above logic works

2Q)

Given ar[N], first n-1 elements are sorted, sort entire arr[]
Expected SC: O(1)

ar[6] = { 2 6 10 14 20 4 }

sort[] = { 2 4 6 10 14 20 }

<u>Ideas</u>: Sort entire arr[] : TC: O(NlogN) SC: O(1)

	0	1	2	3	4	5
ar[6] = {	2	~~6~~	~~10~~	~~14~~	~~20~~	~~4~~ }
		4	~~4~~	~~4~~	~~4~~	20
	2	4	6	10	14	20

	0	1	2	3	4	5 -	6
ar[7] = {	3	~~6~~	~~10~~	~~14~~	~~20~~	~~24~~	~~5~~ }
	3	5	6	10	14	20	24

j	: ar[j] <= ar[j+1] :
5	: no swap
4	: no swap
3	: no swap
2	: no swap
1	: no swap
0	: yes break

<u>Idea</u>: Iterate from back, & compare adj elements, if they are not in correct order we swap: Insertion Step

<u>Insertion Step</u>:

ar[N], sorted from [0, n-2],

```
j = n-2; j >= 0; j--){
    if (ar[j] > ar[j+1]){
        swap ar[j] & ar[j+1]
    }
    else { break}
}
```

<u>TC</u>: O(N) SC: O(1)

3Q) Given ar[ ] sort it unsing Insertion Steps → {Insertion Sort}

	0	1	2	3	4	5
ar[6] : {	10	3	6	8	2	5 }

Step:1 → 3 10 6 8 2 5

Step:2 → 3 6 10 8 2 5

Step3:→ 3 6 8 10 2 5

Step4 → 2 3 6 8 10 5

Step5 → 2 3 5 6 8 10

```
void InsertionSort( int ar[N]){        TC: O(N^2)   SC: O(1)
                                                     ↳ inplace
    i = 1; i< N; i++){
                                                     Stable: Yes
        // insert ar[i] in sorted part [0, i-1]
        j = i-1; j >= 0; j--){
            if (ar[j] > ar[j+1]) ){
                swap (ar[j] & ar[j+1])
            }
            else { break}
        }
    }
}
```

4Q) Stream of numbers, after inserting new number print entire sorted data

Ex: Stream : 6 8 2 4 5 9

output

: 6 8

: 6 8 2

: 2 6 8 4

: 2 4 6 8 5

: 2 4 5 6 8 9

: 2 4 5 6 8 9

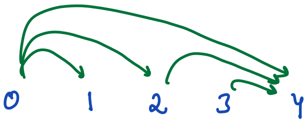
Idea: Perfom insertion step for every new element : TC: $O(N^2)$ SC: $O(1)$

Perfom merge sort after every step:

TC: $N * N\log N \rightarrow O(N^2 \log N)$

Q8) Given ar[N] calculate no: of pairs [i, j] such that $i<j$ & $ar[i] > ar[j]$

inversion count



Ex1:

	0	1	2	3	4
Ex1:	10	3	8	15	6
	3	0	1	1	0

: ans = 5 : inversion count

Ex2:

	0	1	2	3	4	5	6	7	8	9
Ex2:	10	3	8	15	6	12	2	18	7	1
	6	2	4	5	2	3	1	2	1	0

: ans = 26

Idea: Check all pairs

```
c=0
i=0; i<n; i++) {        TC: O(N^2)  SC: O(1)
|   j=i+1; j<n; j++) {
|   |   if( ar[i] > ar[j]) {
|   |   |   c=c+1
|   |   }
|   }
}
return c;
```

Pairs i<j && ar[i] > ar[j]

Idea:

0	1	2	3	4	5	6	7	8	9
10	3	8	15	6	12	2	18	7	1

0	1	2	3	4
10	3	8	15	6

5	6	7	8	9
12	2	18	7	1

Sort:

0	1	2	3	4
3	6	8	10	15

↑ P1

5	6	7	8	9
1	2	7	12	18

Sort

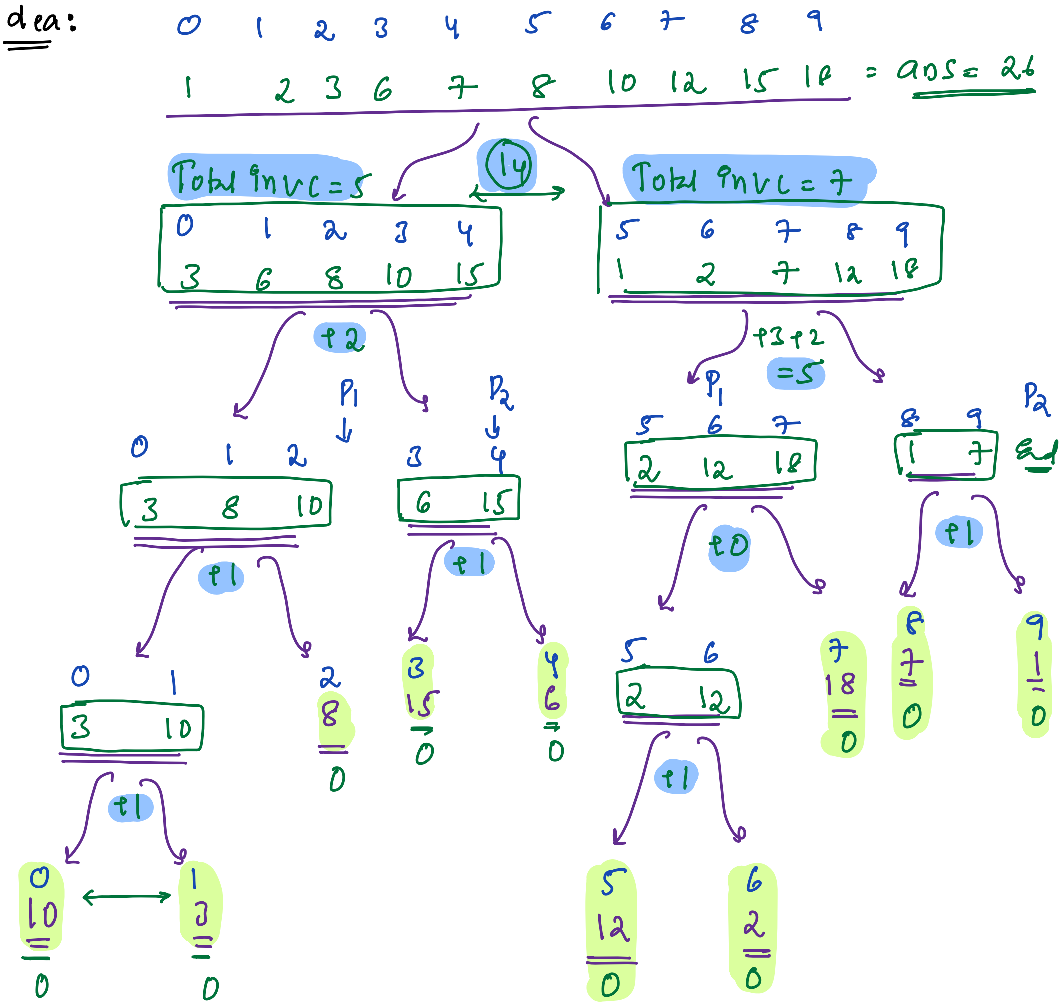
↑ P2

1	2	3	6	7	8	10	12	15	18

#ans = +5 +5 +0 +0 3 0 0 1 0 0 = 14

[Pairs present only in left
&
Pairs present only in right]

Idea:


0 1 2 3 4 5 6 7 8 9
1 2 3 6 7 8 10 12 15 18 = ans = 26
Total invc = 5
14
Total invc = 7
0 1 2 3 4
3 6 8 10 15
5 6 7 8 9
1 2 7 12 18
+2
P1
P2
+3+2 =5
P1
P2
0 1 2
3 8 10
3 4
6 15
5 6 7
2 12 18
8 9
1 7
+1
+1
+0
+1
0 1
3 10
2
8
0
3
15
0
4
6
0
5 6
2 12
7
18
0
8
7
0
9
1
0
+1
+1
0
10
0
1
3
0
5
12
0
6
2
0

```
int ans=0 //global variable, verify in your language of choice
void merge(int A[], int s, int m, int e){
    tmp[e-s+1];
    p1 = s, p2 = m+1, p3 = 0
    while( p1<=m && p2 <= e){
        if( A[p1] <= A[p2] ){
            tmp[p3] = A[p1]; p3++, p1++
        }
        else{
            tmp[p3] = A[p2], p3++, p2++
        }
    }
    while( p1 <= m){ tmp[p3] = A[p1]; p3++, p1++ }
    while( p2 <= e){ tmp[p3] = A[p2], p3++, p2++ }
    //copy tmp[   ] -> ar[s   e]
    i = s, j=0; i <= e; i++, j++ ){
        ar[i] = tmp[j]
    }
}
void mergeSort( int ar[], int s, int e){
    if(s == e){return}
    int m = (s+e)/2
    mergesort(ar, s, m) -> f(N/2)
    mergeSort(ar, m+1, e) -> f(N/2)
    merge(ar, s, m, e) -> N
}
return ans;
```

s [m] m+1 e

↑ p1 ↑ m] : How may elem : m - P1 + 1

no: of remaining elements in left

ans = ans + m - P1 + 1

Dating algo:

	1	2	3	4	5
5 movies:	Harry Potter	Bahubali	Pk	Endgame	3 idiiots

	4	1	5	2	3
Sheetal :	{ Endgame	Harry Potter	3 idiots	Bahubli	Pk }
	1	2	3	4	5

	5	1	4	2	3
invc	{ 3 idiots	HP	EGame	Bahu	Pk }
Girish :	{ 3	2	1	4	5

= ③

	5	4	1	2	3
invc	{ 3 idiots }	{ Egame }	{ HP }	{ BahBa }	Pk
Saiva :	{ 3	1	2	4	5 }

= ②