Today's agenda

- understanding Sorting
- Problems on Sorting
- Sorting techniques

Sorting: Arranging data in increasing / decreasing.

↳ on the basis of which Parameter

Ex1: 2 4 10 15 27 → true

Ex2: 20 7 7 -5 -8 → true

Ex3: 1 2 3 7 4 9 6 → not sorted on the basis val.

#factors: 1 2 2 2 3 3 4 → sorted on the basis of factor count.

→ How to sort an array

↳ bubble sort

↳ selection sort

↳ insertion sort

↳ Quick sort

↳ merge sort

↳ :

!

arr[7]: 1 2 3 7 4 9 6

↓

Arrays.sort(arr); → inc. order

↳ T.C: O(N log N)

↓

1 2 2 4 6 7 9

dec. order

Arrays.sort(arr, reverseOrder());

Q) Order of Removal

↳ Given N elements at every step remove an array element. Cost to remove element = Sum of array elements present. Find min cost to remove all elements.

Note: Add cost first and then remove.

Ex1: arr[3] = {3 2 5}

Remove 2: 10

Remove 5: 8

Remove 3: 3

21

arr[3] = {3 2 5}

Remove 2: 10

Remove 3: 8

Remove 5: 5

23

Ex2: arr[4] = {7 6 2 4}

Remove 7: 4+6+2+7

Remove 2: 4+6+2

Remove 6: 4+6

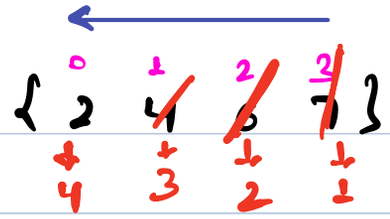
Remove 4: 4

max contribⁿ in total cost = 0th index = min ele

2nd max contribⁿ in total cost = 1st index = 2nd min ele

2nd index = 3rd min ele.

-> Array should be in inc. order.

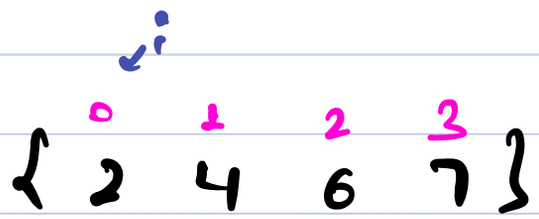


ans += (removal freq) * (N-i)

remove 7 : 2 + 4 + 6 + 7

remove 6 : 2 + 4 + 6

remove 4 : 2 + 4



ans = 0 + (7*1) + (6*2) + (4*3) + (2*4)

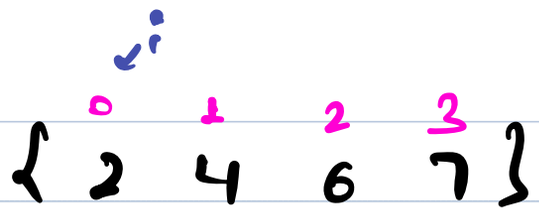
//Psuedo code

```
int removalOrder (int arr[N]) {

    Arrays.sort (arr);

    int ans = 0;

    for (int i = N-1; i >= 0; i--) {

        ans = ans + (arr[i] * (N-i));

    }

    return ans;
}
```

T.C: O(NlogN) + O(N) ≈ O(NlogN)

S.C: O(1)

```
int removalOrder (int arr[N]) {
    Arrays.sort (arr);
    int ans = 0;
    for (int i = N-1; i >= 0; i--) {
        ans = ans + (arr[i] * (N-i));
    }
    return ans;
}
```



ans = 0 + 7 + 12 + 12 + 8

Remove 7 : 2 + 4 + 6 + 7

Remove 6 : 2 + 4 + 6

Remove 4 : 2 + 4

Remove 2 : 2

Q) Good Integer {only distinct}

↳ Given arr[N], Calculate no. of good integers.

An element is said to be good if

{No. of element < ele :: ele itself}

	0	1	2	3	4	5
Ex1:	-1	-4	3	5	-15	4
#less:	2	1	3	5	0	4

→ ans = 3

Ideal

↳ for every number, count smaller no. and check if count == no.

T.C: $O(N^2)$

Idea 2

	0	1	2	3	4	5
	-1	-4	3	5	-15	4

	0	1	2	3	4	5
	-15	-4	-1	3	4	5
		1	2			5

After sorting,

{No. of element < ele :: ele itself}

i → No. of element

arr[i] → ele itself

//Pseudo code

```
int goodInteger(int arr[N]){
    Arrays.sort(arr);

    int ans = 0;

    for(int i=0; i<N; i++){
        if(arr[i] == i){
            ans++;
        }
    }

    return ans;
}
```

T.C: O(NlogN)

S.C: O(1)

Break till 10:29 pm

{No. of element < ele :: ele itself}

Good Integers : { Data can repeat }

	0	1	2	3	4	5	
Ex1:	0	2	2	3	3	8	→ ans = 3
#less:	0	1	1	3	3	5	

After sorting

Ex2:

	0	1	2	3	4	5	6	7	8	9	10	11	12
	-4	-4	2	2	5	5	5	5	8	8	8	10	17
	0	0	2	2	4	4	4	4	8	8	8	11	12

Obs1: All the same elements will be either good or bad (count of no. < ele will remain same)

Obs2:

if ele is 1st occ. → arr[i] != arr[i-1]

No. of ele < ele : : ?

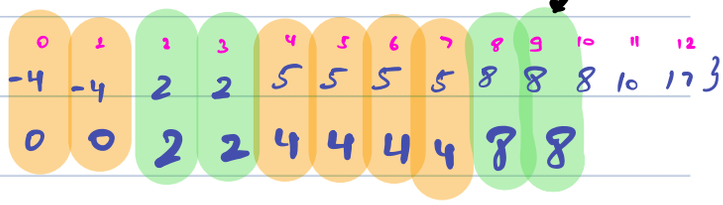
//Pseudo Code

```
int goodintegersDeplicate (int arr[N]){

    Arrays.sort(arr);
    int ans=0;
    →//Handle for 0th index →H.W
    int lessCount=0;
    for(int i=1; i<N; i++){
        if(arr[i]!=arr[i-1]){ //first occ.
            lessCount=i;
        }
        else{
        }
        if(arr[i]==lessCount){ans++;}
    }
}
```

T.C: O(NlogN)

S.C: O(1)

ans = 6 2 2 3 4



lessCount = 6 2 4 8

```
int goodintegerDeplicate (int arr[n]){

    Arrays.sort(arr);
    int ans=0;

        int lessCount=0;
    for(int i=1; i<n; i++){
        if(arr[i]!=arr[i-1]){ //first occ.
            lessCount=i;
        }
        else{ //next occ.

        }
        if(arr[i]==lessCount){ans++;}
    }
}
```

// Sorting techniques

① Bubble Sort

↳ Sort the array in asc. order but we can swap adjacent elements only.

arr[8] = { 5 7 5 4 10 -2 6 3 }
(0 1 2 3 4 5 6 7)

(j, j+1)

iter 1: { 5 7 5 4 10 -2 6 3 } → {0, N-2}
(i, j)
5 5 4 7 -2 6 3 10

iter 2: { 5 5 4 4 -2 6 3 10 } → {0, N-3}
(i, j)
4 5 -2 6 3 7

iter 3: { 4 5 -2 5 6 3 7 10 } → {0, N-4}
(i, j)
4 5 -2 5

Total no. of iter = 7 = N-1

{0, N-2} / {1, N-1}

iter

//Pseudo code

```
void bubbleSort (int arr[N]){

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

        for(int j=0; j<=N-2-i; j++){
            if(arr[j] > arr[j+1]){
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }

    }

}
```

T.C: $O(N^2)$

S.C: $O(1)$

int → -2^{31} $2^{31}-1$

$-(-2^{31}) \rightarrow 2^{31} = -2^{31}$