- Sorting
  - Why sorting
  - Min cost to remove all Elements
  - Noble Integer → Ver1
    ↘ Ver_2
  - Comparator
  - Count Sort Basics

Sorting : Arranging data in inc / dec order based on parameter

Ex1: 3 8 9 14 19 } Based Value

Ex2: 19 14 9 8 3 } Based Value

Ex3: 1 5 3 9 6 10 12 } Based factors

# factors	1	2	2	3	4	4	6

⇒ Sorting sort() —

* To sort N array Elements pls take NlogN Time ?

→ You can use Inbuilt Sort Function

108) Min Cost to Remove all Elements.

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

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

Find Min cost to remove all Elements?

Ex1: {2, 1, 4}

		Cost
Remove 2	→ {2, 1, 4}	7
Remove 1	→ {1, 4}	5
Remove 4	→ {4}	4
	Cost :	16

		Cost
Remove 4	⇒ {2, 1, 4}	7
Remove 2	→ {1, 2}	3
Remove 1	→ {1}	1
Final ans:	Cost : 11	

Ex2: {4, 6, 1}

		Cost
Remove 6	→ {1, 4, 6}	11
Remove 4	→ {1, 4}	5
Remove 1	→ {1}	1
	Cost : 17	

Ex3: {3, 5, 1, -3}

		Cost
Remove 5	{3, 5, 1, -3}	6
Remove 3	{3, 1, -3}	1
Remove 1	{1, -3}	-2
Remove -3	{-3}	-3
	Cost : 2	

obs: We are deleting elements in desc order to get min cost?

Ex 4: { a, b, c, d }

		Cost
Remove a :	{ a, b, c, d }	a+b+c+d
Remove b :	{ b, c, d }	b+c+d
Remove c :	{ c, d }	c+d
Remove d :	{ d }	d

Total Cost : a + 2b + 3c + 4d

1st Element we delete = 1st Max

2nd Element = 2nd Max

3rd Element = 3rd Max

4th Element to delete = 4th Max

Pseudocode

1) Sort in desc order

TC : (N log N + N)
: O(N log N)

```
sum = 0
for(i = 0; i < N; i++){
    sum = sum + arr[i] * (i+1)
}
```

28) Noble Integer { Data is Distinct }

Given N array elements, Calculate No: of Noble Integers Present

ar[i] is said to be Noble {No: of elements < ar[i] } = ar[i] .

Ex1 :

{ 1, -5, 3, 5, -10, 4 }

#count : 2 1 3 5 0 4

less

Ex: 3 Noble Integers

Ex2:

{ -3, 0, 2, 5 }

{ # Count no < ar[i] } →

Index : 0 1 2 3

0 1 2 3

Ex3:

{ -10, -5, 1, 3, 4, 5, 10 }

{ # Count no of < ar[i] : }

Index : 0 1 2 3 4 5 6

0 1 2 3 4 5 6

// Sol1: For every element get no: of less than arr[i] in complete array.

```
cnt = 0
i = 0; i < N; i++){
    int less=0
    j = 0 ; j < N; j++){
        if( arr[j] < arr[i]){
            less++
        }
    }
    if ( less == arr[i]){
        cnt++
    }
}
```

TC: O(N²) SC: O(1)

Taking time.

For every element to get No: of Elements less than that?

Sol2: Sort & compare arr[i] == i

→ Sort arr[] in inc TC: O(NlogN + N)

→ ans = 0

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

30) Noble Integer Elements can repeat. } Above soln can work.?

→ Special index

Ex1: ⇒ { 0 2 2 4 4 6 }

cut : 0 1 1 3 3 5

index: 0 1 2 3 4 5

Ques1: { -10 1 1 1 4 4 4 7 10 } ans = 7

cut < ar[i] 0 1 1 1 4 4 4 7 8

index: 0 1 2 3 4 5 6 7 8

Ques2: { -10 1 1 2 4 4 4 8 10 } → ans = 5

cut of < ar[i] 0 1 1 3 4 4 4 7 8

index: 0 1 2 3 4 5 6 7 8

Ques3: { -3 0 2 2 5 5 5 5 8 8 10 10 10 14 }

cut of ar[i] 0 1 2 2 4 4 4 4 8 8 10 10 10 13

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

obs1: If element is same as prev
cnt won't change
ar[i] != ar[i-1]

obs2: If element comes first time
no: of elements less than ar[i] = i

1) sort array in increasing order

```
int ans = 0
int less = 0
if (ar[0] == 0) { ans++ }
for (i = 1; i < N; i++) {
    if (ar[i] != ar[i-1]) {
        less = i;
    }
    else { // ar[i] is repeating
        // less won't even change
        // No need to write.
    }
    if (less == ar[i]) {
        ans++;
    }
}
```

→ if i==0 code fails.

TC: O(NlogN + N)
= O(NlogN)

10:37 mins
break.

→ Comperator } → Please check syntax for this in your language of choice

Q) Given N array elements, sort them in increasing order of their { No: of factors }
If 2 elements have same no: of factors, element with less value should come first.

→ Note: You cannot use any extra space

Ex1: { 9, 3, 4, 8, 16, 37, 6, 13, 15 }

	9	3	4	8	16	37	6	13	15
# factors	3	2	3	4	5	2	4	2	4

output: 3 13 37 4 9 6 8 15 16

Ex2: { 1, 21, 6, 23, 10, 14, 25 }

	1	21	6	23	10	14	25
# factrs:	1	4	4	2	4	4	3

output 1 23 25 6 10 14 21

→ No extra space ✗

→ Sorting ✗ → modify Inbuilt sort using comperator

→ sort( ) }

Ex1:

25 → 3

16 → 5

In final order 25 should come befn 16.

Ex2:

10 → 4

9 → 3

In final order 9 should come before 10

Ex3:

49 → 3

25 → 3

Since facts are same, 25 should come befn 49.

remember Today →

We always 2 parameters to comparator func, it will always return T|F

```
bool comp (int a, int b){
    // Note: If we want a to come befa b in final order, in that case function should return True.
    int f1 = factors(a)
    int f2 = factors(b)
    if( f1 < f2 ) { return True, a should come befa b}
    else if( f1 == f2 && a < b) return True
    else return false, b should come a
}
```

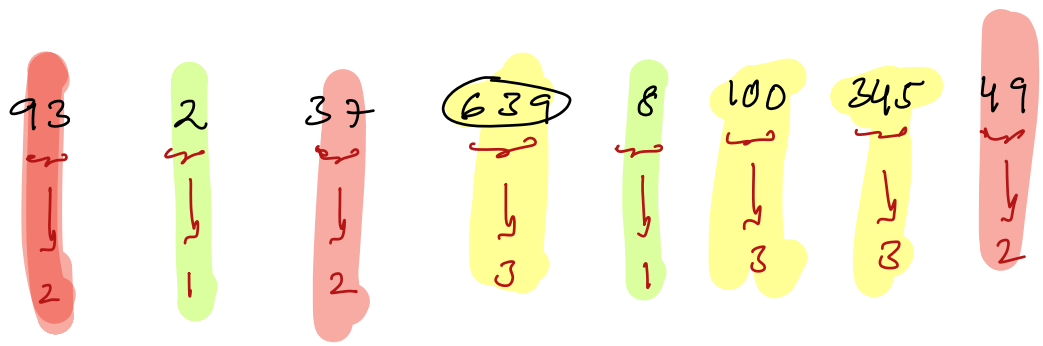
```
sort( arr[], comp)
```

→ Advanced Batch /MergeSort/ Why this above works/

We pass all elements,

48) Given N array elements, sort them in increasing order of No: of digits, if 2 elements have same no: of digits, Element with more value should come first.

Eg1:



output : 8 2 93 49 37 639 345 100

```
bool comp (int a, int b){
    // if we want a to come before b in final order
       return true, else return false
    int d1 = countdigits(a)
    int d2 = count digits(b)
    if ( d1 < d2) { return True }
    else if ( d1 == d2 && a > b) return True }
    else return False
}
```

sort (arr[] comp)

// Inbuilt sort function will pass parameters to the compare function.

{Assignments: Will add by 12PM}

Doubts

$> N/2$ → { If we delete 2 distinct elements}

at max (1) major

$> N/3$ → { If we delete 3 distinct elements}

at max (2)

↳ We can have 2 majority Elements

- { ele1 freq1 ele2 freq2 }
- check if ele1 is ans
- check if ele2 is ans