- Sorting
- Q. Min cost to remove all elements
- Q. Noble integer
  - Distinct
  - Duplicates.
- → Comparator function

Sorting : Arranging data in inc/des order. based on priority

Ex1 : 3 8 9 12 100 568

Ex2 : 19 14 9 8 6 3

Ex3 : 1 5 9 6 10 12 [inc order based on no of factors]

factors : 1 2 3 4 4 6

Sort(arr, N) TC : O(N log N)

Q. Min cost to remove all element.

Given an array N, at every step you can remove 1 element.

Cost to remove = sum of all elements present in arr.

Find min cost to remove all elements.

Ex {2, 4, 1}

	Cost	Resultant
Remove 2 :	{2+4+1} = 7	{4, 1}
Remove 1 :	{1+4} = 5	{4}
Remove 4 :	{4} = 4	{}

7+5+4 = 16

Remove 4	{1+2+4} = 7	{1, 2}
Remove 2	{1+2} = 3	{1}
Remove 1	{1} = 1	{}

7+3+1 = 11

{a, b, c, d}

Remove a	a+b+c+d
Remove b	b+c+d
Remove c	c+d
Remove d	d

a + 2b + 3c + 4d

a → max, 2b → 2nd max, 3c → 2nd max, 4d → min

Check syntax in your language

```
int findMinCost(int arr[], int N)
```

Arrays.sort(arr) [NlogN time]

reverse(arr)

Assume you have sorted in array in ting

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

calc. ↓ Sorting ↓

TC: O(N + NlogN)

TC.: O(NlogN)

SC: O(1)

Q. Noble Integer [Distinct elements]

Given an array N. find no. of noble integers in the array.

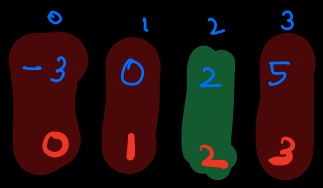
a[i] is noble if (no of elements < arr[i] = arr[i])

Ex

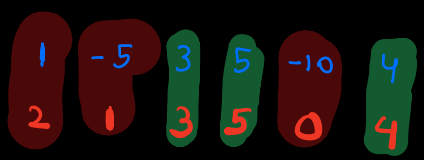
1	-5	3	5	-10	4
2	1	3	5	0	4

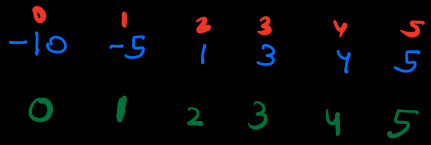
ans = 3

Ex



ans = 1






```
int noblend (arr[], N)
{
    // sort the array in ↑ing order
    cnt = 0
    for(i=0; i<N; i++)
    {
        if( arr[i] == i)  cnt++
    }
    return cnt
}
```

TC: O(N + NlogN)
TC: O(NlogN)
SC: O(1)

Q. Noble Integer (Duplicate can be present).

	0	1	2	3	4	5
	0	2	2	4	4	6
	0	1	1	3	3	5

ans = 1

No of elements < arr[i]

C

arr[i] == arr[i-1]

Count will not change

arr[i] != arr[i-1]

C = index

	0	1	2	3	4	5	6	7	8
	-10	1	1	2	4	4	4	8	10
	0	1	1	3	4	4	4	7	8

```
int nobleno (arr[], N)

// sort the array in ting order
ans=0, c=0
for(i=0; i<N; i++)
{
    if(i!=0 && A[i]==A[i-1])
    { Do nothing ✓ Continue ✗
    else
        c=i
    if(arr[i]==c)
        ans++
}
```

→ c

→ if noble

return ans

TC : O(NlogN)
SC : O(1)

Comparator function.

Sorting will be based on this function.

Sort (arr, comp)

if (arr[i] > arr[j])

{ Then arr[i] should come before arr[j]

Sort (arr, comp)

bool comp (int a, int b)

{ Return True
a should come before b

Return False
b should come before a.

Return True
both are equal

Sort in ascending.

```
Sort( arr, comp)
bool comp ( int a, int b)
{
    if( a ≤ b) return True
    else return False
}
```

Sort in descending

```
Sort( arr, comp)
bool comp ( int a, int b)
{
    if ( a ≥ b) return True
    else return False
}
```

Based on no of factors.

```
Sort ( arr, comp )
bool comp( int a, int b )
{
```

f_a = factors of a

f_b = factors of b

if ($f_a < f_b$) return True

else if ($f_a > f_b$) return False

else

{

if ($a \leq b$) return True

else return False