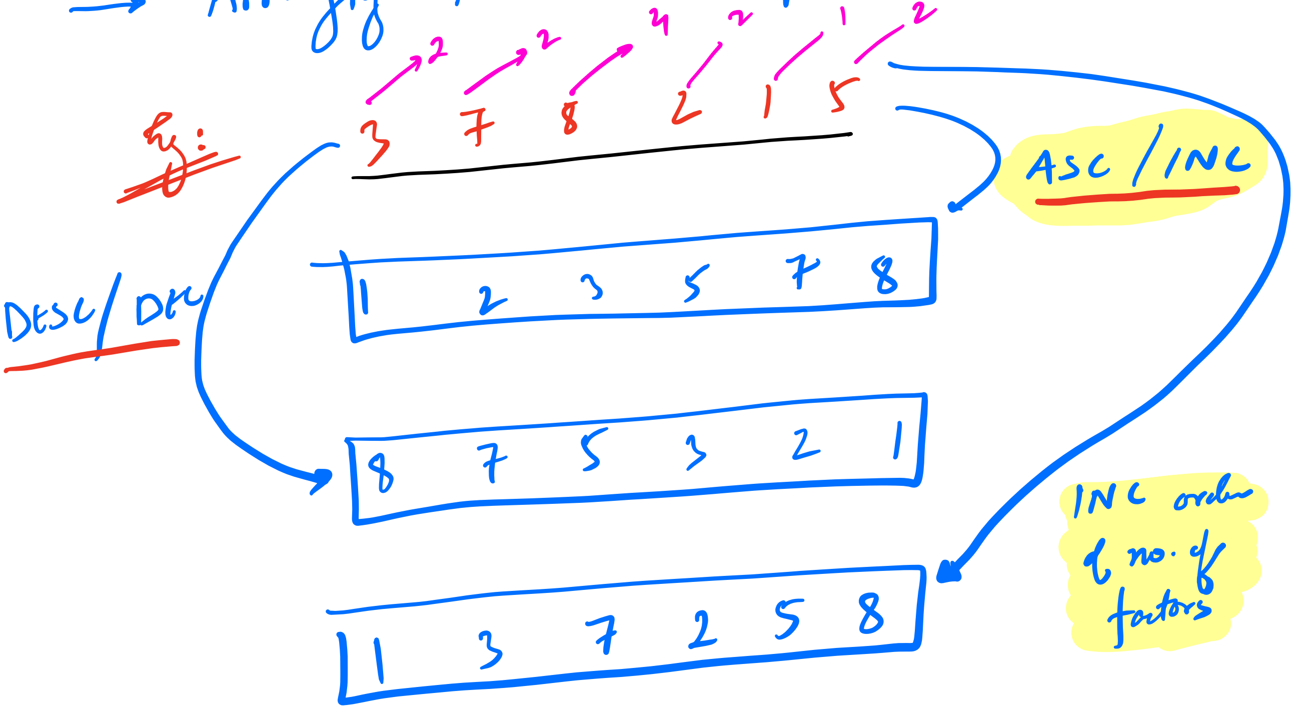
⊙ Sorting ?

→ Arranging the data in particular order!



⊙ Sort() f^n [INBUILT]

$\boxed{TC : O(N \log N)}$

$\boxed{SC : O(1)}$ → heapSort | otherwise $O(N/\log N)$

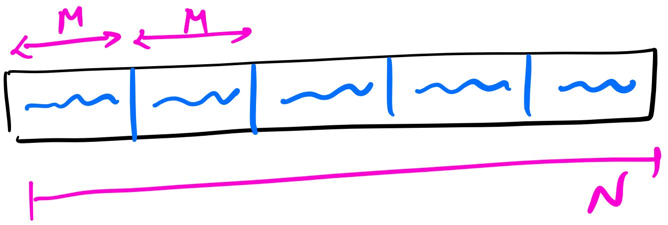
A : [ ]

Sort?

sort(A); → NlgN | O(1)

Array of strings

A :



Sort ? N log N

O(1) ← int vs int

O(M) ← string vs string

s = prakash

t = prakaas

$$\boxed{TC : O(M N \log N)}$$

Q Given N elements, at every step remove an element!
Cost of removing an element : Sum of the elements present in the array beforing removing this element.

FIND the MIN cost to remove all elements

		Cost
A :	[2, 1, 4]	7
−2	[1, 4]	+ 5
−1	[4]	+ 4
−4	X	$\boxed{16}$

A : [2, 1, 4]

-4 [2, 1]

-2 [1]

-1 x

Cost

7

+3

+1

$\boxed{11}$

vs

Observation

(a, b, c d) → a + b + c + d

-a Largest (b, c, d) → b + c + d

-b (c, d) → c + d

-c (d) → d

-d smallest x

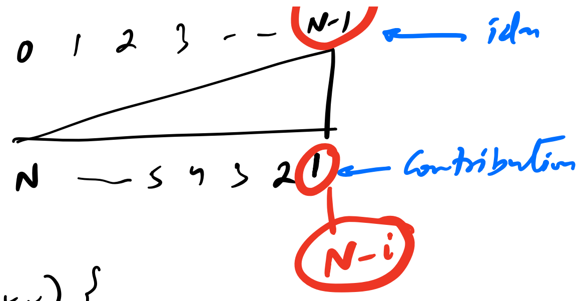
Total Cost → $\boxed{a + 2b + 3c + 4d}$

1 2 10 100

a : Largest

d : Smallest

```
//A
A.sort();
 sum = 0;
f(i=0; i<N; i++){
    sum += (N-i) x A[i];
}
ret sum;
```



$$\boxed{TC : O(N \log N)}$$

$$\boxed{SC : O(1)}$$

Q. Noble Integer **[DISTICT]**

Given N array elements. Calculate the no. of Noble elements.

Noble element : $\boxed{A[i] \left\{ \begin{matrix} \text{No. of elements} \\ < A[i] \end{matrix} \right\} = A[i]}$

A :	[1,	-5,	3,	5,	-10,	4]	→ 3
#cnt lesser	2	1	3	5	0	4	

I BF

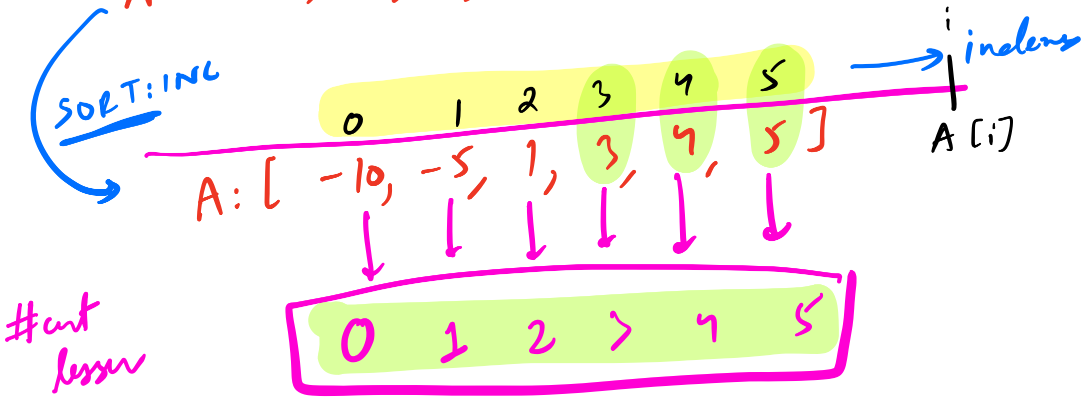
```
ANS=0
f(i=0; i<N; i++){
    cnt=0;
    f(j=0; j<N; j++){
        if( A[j] < A[i]){
            cnt++;
        }
    }
    if(cnt == A[i]){
        ANS++ ;
    }
}
ret ANS;
```

$\boxed{TC=O(N^2)}$

$\boxed{SC=O(1)}$

II

A: [1, -5, 3, 5, -10, 4]



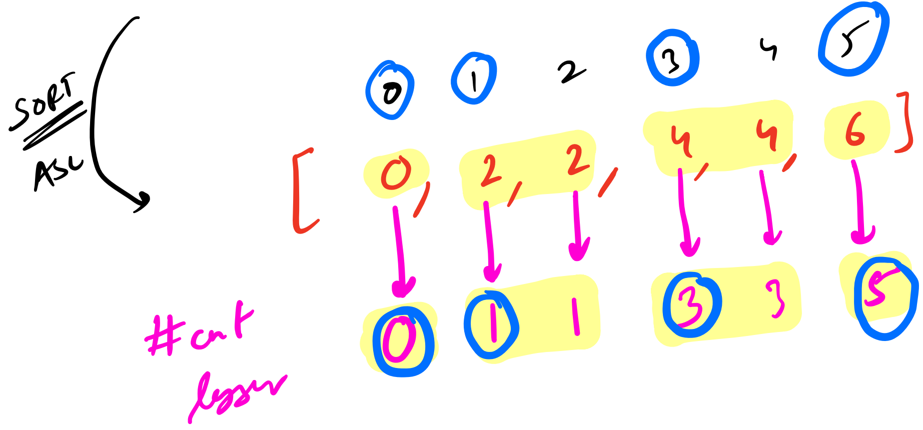
```
// A[]
    A.sort()  →  ASC

    ANS = 0;
    f(i=0; i<N; i++){
        if(A[i] == i){
                ANS++;
        }
    }
    ret ANS;
```

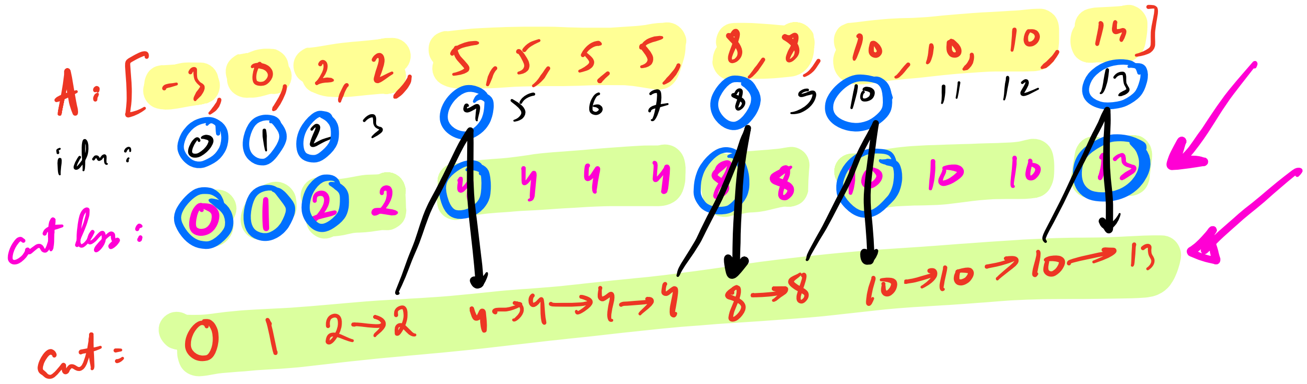
TC = $O(N \lg N)$

SC = $O(1)$

Q SAME AS PREV [elements could repeat]

A : [4, 6, 4, 2, 2, 0]




A : [-3, 0, 2, 2, 5, 5, 5, 5, 8, 8, 10, 10, 10, 14]
idx : 0 1 2 3 4 5 6 7 8 9 10 11 12 13
cnt less : 0 1 2 2 4 4 4 4 8 8 10 10 10 13
cnt : 0 1 2→2 4→4→4→4 8→8 10→10→10→13

```
// A[]
    A.sort();  ⟶  Nlg N
    cnt = 0;
    ANS = 0;
    f(i=0; i< N; i++) {  ⟶  N
        if(i==0 || A[i] != A[i-1]) {
            cnt = i;
        }
        if (A[i] == cnt){
            ANS++;
        }
    }
    ret ANS;
```

TC = O(Nlg N)

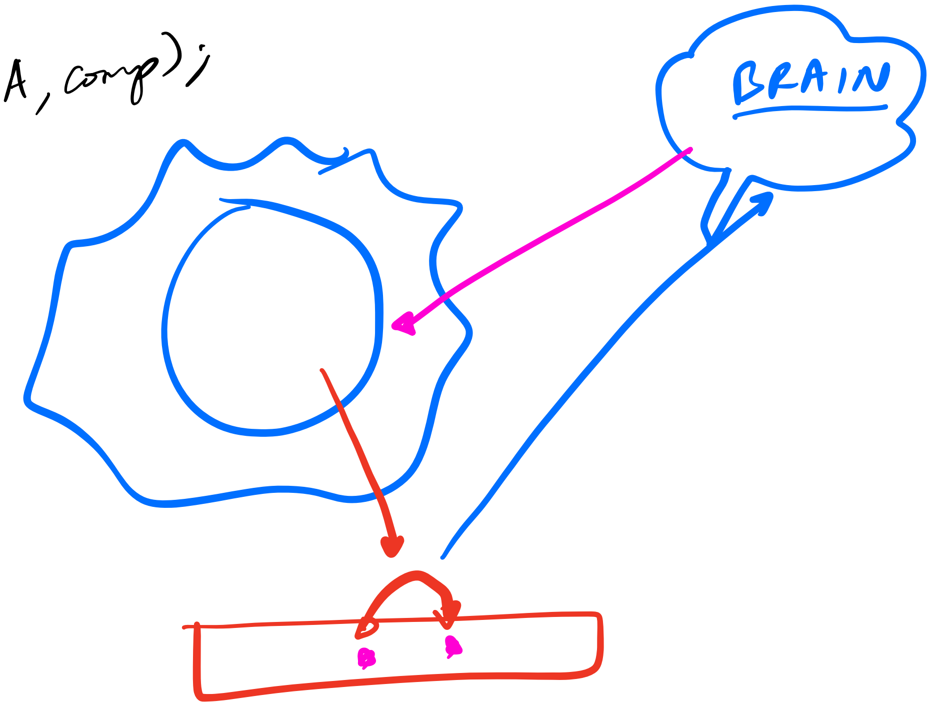
SC = O(1)

⊙ Custom Comparators

→ It will help your sorting Algo to decide the order!

→ f^ns that you write!

Sort (A, comp);



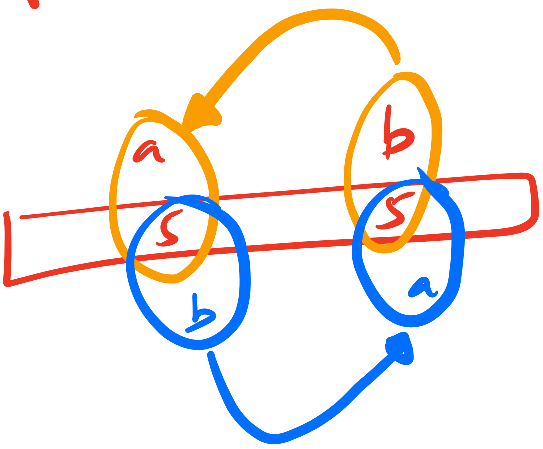
```
bool comp(int a, int b) {  → SORT ASC
    // Return true if a before b
    // Return false otherwise!                SORT DES
    if (a < b) ret true;  →  if(b<a) . —
                              — —
    else ret false;

}
```

Q Sort the Array in DEC order of No of factors.
if there is a tie, the smaller should come first!

```
bool comp( int a, int b){
    int fa = cntfactors(a);
    int fb = cntfactors(b);
    if ((fa > fb)||(fa == fb && a < b)){
            ret true;
    }
    ret false;
}

Sort(A, comp);
```

SORT ASC

```
bool comp( a, b) {      ✗
    if (a <~~=~~ b) ret true;
            ✓
    ret false;
}
```



NOTE :

- ⊙ Return true when you strongly want to place a before b !

- ⊙ For equal priority elements ALWAYS return false !