Sorting → Arrangement of data in particular order.

A = [3 5 9 12]

(Ascending Order)

A = [12 9 7 1]

(Descending Order)

A = [1 13 9 6 12]

#factors → 1 2 3 4 6 sorted wrt #factors.

Q → Given an integer array, minimize the cost to empty the given array where cost of removal of an element is equal to sum of all elements left in the array before removal.

$A = [\overset{0}{\cancel{2}}\ \ \overset{1}{1}\ \ \overset{2}{\cancel{4}}]$

Remove	Cost
4	2+1+4 = 7
2	2+1 = 3
1	1
	11 ✓ (Ans)

Remove	Cost
4	2+1+4 = 7
1	2+1 = 3
2	2
	12

$A = [\overset{0}{\cancel{4}}\ \overset{1}{\cancel{6}}\ \overset{2}{\cancel{1}}]$

Remove	cost
6	$4+6+1 = 11$
4	$4+1 = 5$
1	1
	17 (Ans)

$A = [\overset{0}{\cancel{3}}\ \overset{1}{\cancel{5}}\ \overset{2}{\cancel{1}}\ \overset{3}{\cancel{-3}}]$

Remove	cost
5	$3+5+1-3 = 6$
3	$3+1-3 = 1$
1	$1-3 = -2$
-3	-3
	2 (Ans)

$A = [a\ \ b\ \ c\ \ d]$

Remove	cost
a	$a+b+c+d$
b	$b+c+d$
c	$c+d$
d	d
	$1.a + 2.b + 3.c + 4.d$

$\underset{\text{largest}}{1.a} + 2.b + 3.c + \underset{\text{smallest}}{4.d}$

SC = O(1)

TC = O(N log(N))

```
// sort (A) ⟶ sort in descending order
cost = 0
for i ⟶ 0 to (N-1) {
    cost += A[i] * (i+1)
}
return cost
```

TC = O(N log (N) + N)

= O(N log (N))

SC = O(1)

Q → Given an integer array of distinct values. Find the count of Nobel integers.

Nobel integer → A[i] is nobel if #elements < A[i] is equal to A[i].

$$\begin{array}{lccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A = [& 1 & -5 & 3 & 5 & -10 & 4 & 12] \\ \boxed{\#\text{elements} < A[i]} \rightarrow & 2 & 1 & 3 & 5 & 0 & 4 & 6 \end{array}$$

$\underline{>= 0}$

Ans = $\underline{3}$

$$\begin{array}{lcccc} & 0 & 1 & 2 & 3 \\ A = [& -3 & 0 & 2 & 5] \\ \#\text{elements} < A[i] \rightarrow & 0 & 1 & 2 & 3 \end{array}$$

Ans = $\underline{1}$

$$\begin{array}{lccccc} A = [& -3 & 1 & 0 & 2 & 5] \\ \rightarrow & 0 & 2 & 1 & 3 & 4 \end{array}$$

Ans = $\underline{0}$

Bruteforce → TC = $\underline{O(N^2)}$

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

Sorted data

$$\begin{array}{lccccc} & 0 & 1 & 2 & 3 & 4 \\ A = [& -3 & 0 & 1 & 2 & 5] \\ \# < A[i] & 0 & 1 & 2 & 3 & 4 \end{array}$$

```
cnt = 0
for i → 0 to (N-1) {
    if ( A[i] == i ) {
        cnt ++
    }
} return cnt
```

TC = $O(N \log(N) + N) = \underline{O(N \log(N))}$

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

$$\begin{array}{lcccc} & 0 & 1 & 2 & 3 \\ A = [& -3 & 0 & 4 & 5] \\ \# < A[i] \rightarrow & 0 & 1 & 2 & 3 \end{array}$$

(Ans=0)

Q → Given an integer array.

Find the count of Nobel integers.

<u>Nobel integer</u> → A[i] is nobel if #elements < A[i] is equal to A[i].

	0	1	2	3	4
A =	[-10	1	1	3	100]
#elements < A[i] →	0	1	1	3	4

Ans = 3

	0	1	2	3	4	5	6	7	8
A =	[-10	1	1	2	4	4	4	8	10]
#elements < A[i] →	0	1	1	3	4	4	4	7	8

Ans = 5

A =	[-3	0	2	2	5	5	5	5	8	8	10	10	10	14]
#elements < A[i] →	0	1	2	2	4	4	4	4	8	8	10	10	10	13

Ans = 7

<u>Sorted data</u>

```
ce = 0
ans = 0
if (A[0] == 0)
    ans = 1
for i → 1 to (N-1) {
    if ( A[i] != A[i-1])
        ce = i
    if (ce == A[i])
        ans++
}
return ans
```

$TC = O(N\log(N) + N) = O(N\log(N))$

$SC = O(1)$

	0	1	2	3	4
A =	[-10	1	1	3	100]
#elements < A[i] →	0	1	1	3	4

Ans = 3

Q → Find largest element in array → $TC = O(N)$

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

Find second largest element → $TC = O(2N) = \underline{O(N)}$

$SC = \underline{O(2)} = \underline{O(1)}$

⋮

Find K^{th} largest element → $TC = O(N * K)$

$SC = \boxed{\underline{O(K)}}$

$$A = [\ \overset{0}{\underset{1}{\cancel{2}}}\ \Big|\ \overset{1}{\underset{\underset{2}{\cancel{1}}}{\cancel{10}}}\quad \overset{2}{5}\quad \overset{3}{7}\quad \overset{4}{\underset{10}{\cancel{1}}}\ ]$$

$K = \underline{3}$ Ans $= \underline{5}$

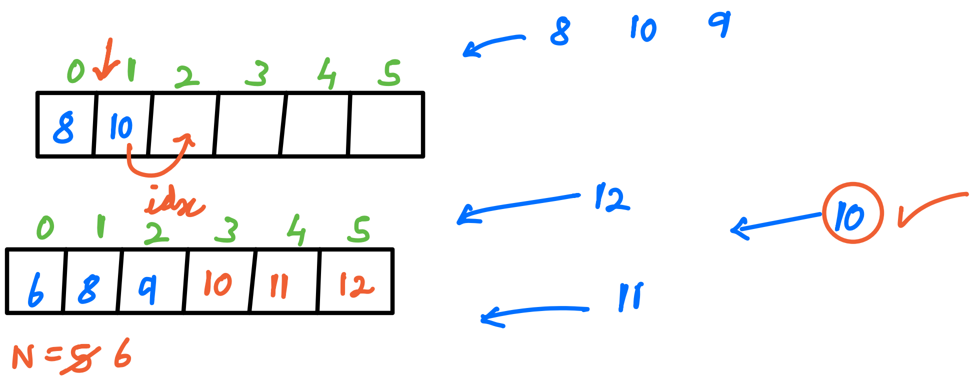
Selection sort

```
$\boxed{last = N-1}$
for i → 1 to (N-1) {   //N-1 times
      idx = 0
      for j → 1 to last {
            if (A[j] > A[idx])   idx = j
      }
      t = A[idx]
      A[idx] = A[last]
      A[last] = t
      last --
}
```

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

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

Insertion Sort (sort running stream of integers)



```
N = 0 // #elements in array
for ( ∀x : input) {  ← can be from input array
    idx = N-1
    while (idx >= 0 && A[idx] > x) {
            A[idx+1] = A[idx]
            idx--
    }
    A[idx+1] = x        N++
} return A
```

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

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