Arrangement of data in a particular order.

A = [1 13 9 6 12]

#factors → 1 2 3 4 6

Ascending (default)

Descending

Q → Given an integer array, delete elements from array one by one & the cost of deletion is sum of elements. Find min cost to delete all elements.

A = [~~2~~ 1 4] cost = 2+1+4 = 7

~~1~~ 4 1+4 = 5

4 4 = 4

16

A = [2 1 ~~4~~]

~~2~~ 1 cost = 7+3+1 = 11 ✓

1

A = [4 ~~6~~ 1]

~~4~~ 1 cost = 11+5+1 = 17 ✓

1

A = [~~3~~ ~~5~~ ~~1~~ ~~-3~~] cost = 6+1+(-2)+(-3) = 2 ✓

A = [x y z]

Remove	cost
x	x + y + z
y	y + z
z	z

x + 2y + 3z → total cost

x > y > z ⇒ min cost

Contribution Technique

A = [3 5 1 -3]

	0	1	2	3
N = 4	-3	1	3	5
	4	3	2	1

cost = (-3 * 4) + (1 * 3) + (3 * 2) + (5 * 1) = 2

```
// sort in ascending order → TC = O(N log(N))
                             SC = O(1)
ans = 0
for i → 0 to (N-1) {
    ans += A[i] * (N-i)
}
return ans
```

TC = O(N log(N)) SC = O(1)

Q → Given an integer array with distinct elements, find the count of noble integers i.e A[i] s.t (count of elements < A[i]) = A[i].

	0	1	2	3	4	5
A = [	1	-5	3	5	-10	4]
#elements < A[i] →	2	1	3	5	0	4
	✗	✗	✓	✓	✗	✓

Ans = 3

```
          0   1   2   3
A = [ -3   0   2   5 ]
#el. < A[i] →  0   1   2   3
                       ✓
```

Ans = 1

Bruteforce → ∀i, find the count of elements < A[i] & check if it is noble or not, add the count to get ans.

TC = O(N*N) SC = O(1)

Sol →

```
          0   1   2   3
A = [ -3   0   2   5 ]   // asc. sorted
#el. < A[i] →  0   1   2   3
                       ✓
```

Ans = 1

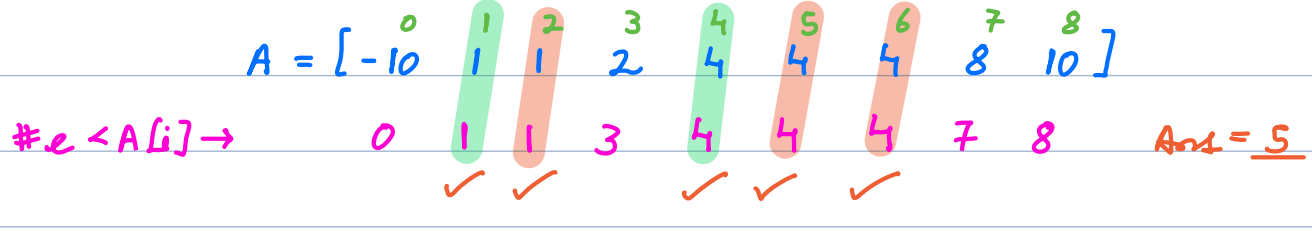
```
// sort in ascending order
cnt = 0
for i → 0 to (N-1) {
    if (A[i] == i)  cnt++
}
return cnt
```

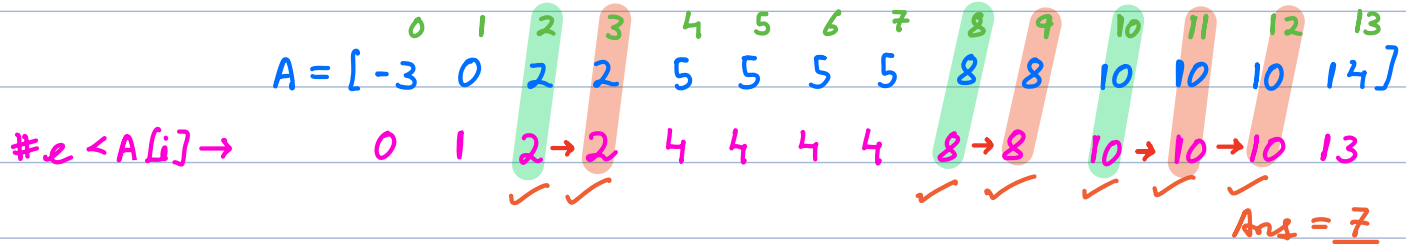
TC = O(N log(N)) SC = O(1)

Q → Given an integer array ~~with distinct elements~~, find the count of noble integers i.e A[i] s.t (count of elements < A[i]) = A[i].

```
          0   1   2   3   4
A = [ -10   1   1   3   100 ]
#e < A[i] →  0   1   1   3   4
             x   ✓   ✓   ✓   x
```

Ans = 3





Bruteforce → ∀i, find the count of elements < A[i] & check if it is noble or not, add the count to get ans.

TC = O(N x N) SC = O(1)

Sol → if (A[i] > A[i-1]) ⇒ (#elements < A[i]) = i

// sorted

A[i] > all elements from 0 to (i-1)

```
// sort in ascending order
ans = 0
if (A[0] == 0)  ans++
cnt_e = 0   // # elements < A[i]
for i → 1 to (N-1) {
    if (A[i] > A[i-1])  cnt_e = i
    if (A[i] == cnt_e)  ans++
}
return ans
```

TC = O(N log(N)) SC = O(1)

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

ans = 0 1 2 3 4 5 ✓

cnt_e = 0 1 3 4 7 8

10:40 PM

Selection Sort

A = [7 8 1 0 5 4 2]
0 1 2 3 4 5 6

1) Find largest element → TC = O(N) SC = O(1)

2) Find second largest element → TC = O(2N) = O(N)
SC = O(2) = O(1)

3) Find third largest element → TC = O(3N) = O(N)
SC = O(3) = O(1)

⋮

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

SC = O(K) → use same array

SC = O(1)

A = [~~7~~ ~~8~~ 1 0 (5) ~~4~~ ~~2~~]
0 1 2 3 4 5 6
4 2 7 8

m1 = 8
m2 = 7
m3 = 5

K = 3 ✓

K = 5 A = [~~7~~ ~~8~~ ~~1~~ ~~0~~ 5 ~~4~~ ~~2~~]
4 x 2 4 7 8
0 1

largest K elements are present in sorted order ✓

K = N-1 = 6 A = [~~7~~ ~~8~~ ~~1~~ ~~0~~ 5 ~~4~~ ~~2~~]
4 x 2 4 7 8
0 1

sorted array

SC = O(1) TC = O(K * N) → O(N²)

```
for i → (N-1) to 1 {   // swap with index i
    mi = 0
    for j → 1 to i {   // 0 — i max element
        if (A[j] > A[mi])  mi = j
    }
    swap(A, i, mi)
}
```

Selection Sort

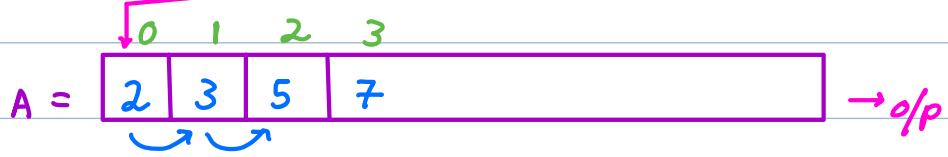
Bubble Sort → swap adjacent elements.

TC = O(N²) SC = O(1)

H.W

Insertion Sort → useful in running data i.e. even if full data is not present at the start.

3 ✓ 5 ✓ 2 ✓ 7 ✓ . . .



min # shift → 0 (insert largest element)

max # shift → N-1 (insert smallest element)

TC for 1 insertion → O(1) to O(N)

10, 8, 7, 3, 1 . . .

⇒ Insert N elements → TC = O(N²)

SC = O(1)

```
n = 0
for ∀input x {
    for i → (n-1) to 0 {        // for(i=n-1; i>=0; i--)
        if (A[i] > x)   A[i+1] = A[i]        (i+1)
        else   break                          // 0 — i <= x
    }
    A[i+1] = x                  // n=0, i = n-1 = -1
    n++                         // i+1 = 0
}
```

3 ✓ 2 ✓ 4 ✓ 8 ✓ 0 ✓

A = [0 2 3 4 8]
    0 1 2 3 4

n = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

i = n-1 = ~~-1~~ ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~2~~ ~~1~~ ~~0~~ -1