Sorting → Arranging Data based on a certain Rule

1, 3, 10, 15 ✓

10, 5, 2, 1, -10 ✓

1, 13, 9, 6, 12 ✓

factors 1 2 3 4 6

Sorting techniques ✗

Sorting & its use case ✓

TC → O(n log n)

SC ⇒ O(1) or O(n)

- Why is sorting important?

  while solving a ques. you reached

  TC → n^2 ✓ try sorting

  if TC → < nlogn ✗ do not try sorting

  - Python - sorted(Arr)
  - C++ - sort(Arr)
  - Java - Array.sort()

Q. Given an array of N elements, delete all elements using fn. such that i) It removes 1 element at a time
ii) It costs sum of all elements left before deletion
Find the least cost?

eg: 5, 8, 3

delete 5 → 5 + 8 + 3 = 16
delete 8 → 8 + 3 = 11
delete 3 → 3 = 3
30

---

delete 8 → 5 + 8 + 3 = 16
delete 5 → 5 + 3 = 8
delete 3 → 3 = 3
27

Brute force?
- 2 nested for loops { TC (n^2)
  - find greatest number
  - find the cost

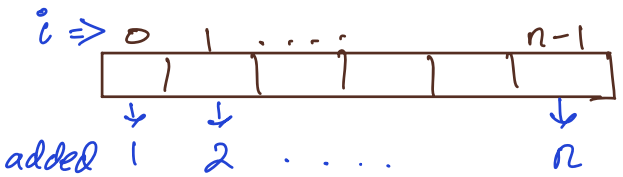
Better solution?

Sorting : 5, 8, 3, 2
sort(desc) → 8, 5, 3, 2

Prefix Sum? TC: (n) + (n log n)
= O(n log n)
SC: O(n)

eg: a, b, c, d { sorted array }

del a a + b + c + d
del b b + c + d
del c c + d
del d d
---
a + 2b + 3c + 4d



for each element
sum = (i+1) (Arr[i])

```
Algo :- . sum = 0
        . Sort (Arr)
        . for (i = 0; i < n ; i++)
              sum += (i+1) (Arr[i])

        return (sum)
```

TC - n(log n) SC = 1 or N

because of sorting

5, -3, 8, 7, 10

(1) Sort 10, 8, 7, 5, -3

1(10) + 2(8) + 3(7) + 4(5) + 5(-3)

= . . . . .

1) 10 + 8 + 7 + 5 + -3
2) 8 + 7 + 5 + -3
3) 7 + 5 + -3
4) 5 + -3
5) -3

Q) Find count of Noble integers in an array of size N
{Distinct} A noble integer is such that

Arr[i] = no. of elements smaller than Arr[i]

eg:- 5, 3, 1, 0, 2, 6

	5	3	1	0	2	6
no. of smaller elements	4	3	1	0	2	5

Brute force :-
- i → 0 - (n-1)
  - j → 0 - (n-1)
    - if Arr[i] > Arr[j]
      - cnt_s ++
  - if cnt_s == Arr[i]
    - nobel_cnt ++

TC = n^2

To optimize we use sorting:

5, 3, 1, 0, 2, 6

sort →	0	1	2	3	5	6
i →	0	1	2	3	4	5

if Arr[i] == i

eg: -3, 5, 3, 7, 25, 22, -2, -10

2 observation, numbers that can never be nobel
i) -ve numbers
ii) numbers >= n

-10	-3	-2	3	5	7	22	25
0	1	2	3	4	5	...	

What if we allow duplicate numbers?

eg:- ① 0, 0, 0, 0, 0

② -5, 1, 3, 3, 5, 7

③ -2 -1 2 2 3 5 5 5

↳ there are 2 numbers smaller than 2?

-2	-1	2	3	5
		2		5
				5

i = 0 1 2 → 4 → 5

flag → count of duplicates

Algo • Sort the array { n log n }

```
cnt = 0
for (i = 1; i < n; i++)
    if arr[i] != arr[i-1]
        cnt = i
    else
        cnt = cnt
```

Take care of i = 0

Arr[0] == 0 ? label ✓ else label X

-3, -2, 2, 2, 3

i = 0, ①, ②, 3, ④

cnt = 0 1, 2 → 2, 4

Q. Sort an array in asc order of their factors, and if the factors are the same sort based on magnitude.

eg: 5, 4, 6, 1, 8

factors 2 3 4 1 4 $\Rightarrow T < \sqrt{N}$

$\Rightarrow$ 1 5 4 6 8

inbuilt sort
- i) Array to sort
- ii) Till where to sort
- iii) Add a comparator

Comparator : Rules to sort

Rules for the question: (x and y)

i) factors(x) < factors(y)
   - x comes first

ii) factors(x) > factors(y)
   - y comes first

iii) factors(x) = factors(y)
   - if (x > y)
     - y
   - else
     - x

Homework: Find how comparator fn. is implemented in your respective languages.

Sort (Array, fn = compare( ))

fn compare (int x , int y) { return type is bool

factor_x = fact(x) } $\sqrt{x}$
factor_y = fact(y) } $\sqrt{y}$

if (factor x < factor y)
return True
else if (factor y < factor x)
return False

{ if True = x
False = y

OR
also returns int

else # fact(x) == fact(y)
if (x > y)
return False
else
return True

Use this compare fn. in the