Sorting :- arrangement of data in particular order on the basis of some parameter

1 4 5 6 9 12 — data is sorted in asc of magnitude

9 7 5 4 3 1 — sorted desc

	1	13	9	6	12
count of factors	1	2	3	4	6

data is sorted in asc order no of factors

inbuilt f^n :- sort(_,_) → asc order

T.C : O(n log n)

S.C : O(1) - O(N) depends

why even we need sorting ? → better searching

Given an array of integers. Remove every element from the array.

whenever you remove ≡ there is a cost

↳ sum of all elements present in the array before removal.

Find the min cost to remove all elements.

A: [2 1 4]

[~~2~~ 1 4]	remove 2	2+1+4 = 7
[~~1~~ 4]	remove 1	1+4 = 5
[4]	remove 4	4
		16

[1 2 ~~4~~]	remo 4	7
[1 ~~2~~]	remove 2	3
[~~1~~]	remove 1	1
	total	11

[4 ~~6~~ 1]

[4 ~~6~~ 1]	delete 6	11
[~~4~~ 1]	delete 4	5
[~~1~~]	delete 1	1
		17

$-3 \Rightarrow 6, 9, 8, 3$ → 26 ✗

[3 5 1 -3]

[3 ~~5~~ 1 -3]	delete 5	3+5+1+-3 = 6
[~~3~~ 1 -3]	delete 3	3+1+-3 = 1
[~~3~~, -3]	delete 1	1+(-3) = -2
[-~~3~~]	delete -3	= -3

2

[a b c d]

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

$a + 2b + 3c + 4d$

minimise → $a > b > c > d$

start removing elements in desc order

```
cost = 0;
sort (desc order)
for (i=0; i<n; i++)
{
    cost += arr[i]*(i+1);
}
```

S.C: depends upon sorting algo

desc order

0	1	2	3	4	5	6	7	8	..
9	7	6	5	3	1				
* 1	* 2	* 3	* 4	. . . . .					

arr[i] * (i+1)

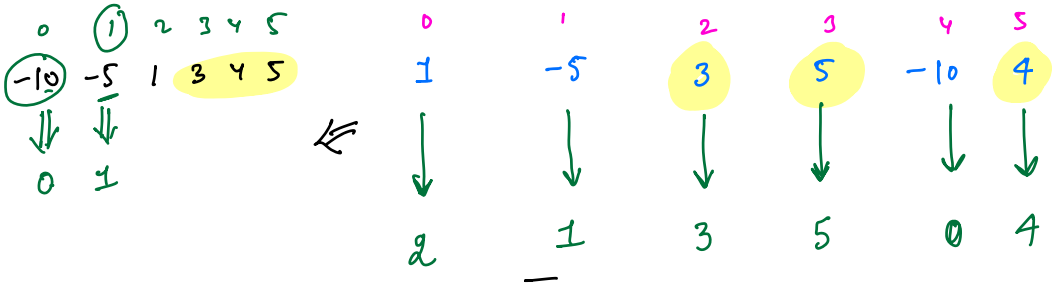
T.C: 1 + nlogn + N

O(n logn)

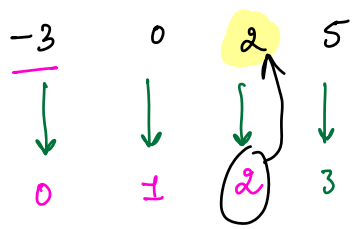
Noble integer

Given N array elements. Find count of noble elements.

A[i] is a noble element if no of elements less than A[i] = A[i] (distinct)



Quiz



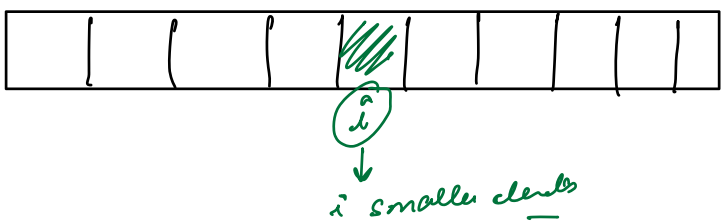
B-F

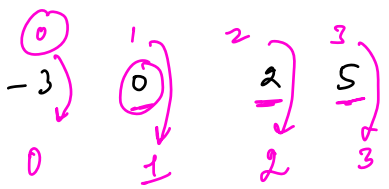
for every element count smaller elements

```
ans = 0
for ( int i = 0; i < n; i++)
{
    int count = 0;
    for ( int j = 0; j < n; j++)
    {
        if ( arr[j] < arr[i])
            count++;
    }
    if (count == arr[i]) ans++;
}
```

T.C: $O(n^2)$

sort your data in asc order





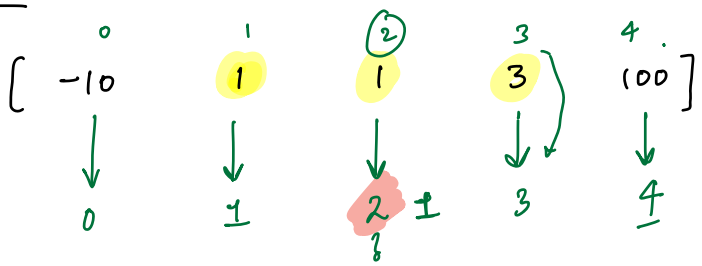
arr[i]
i

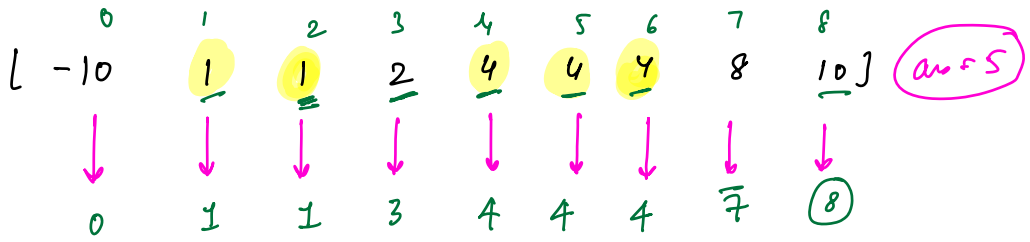
```
// sort the data in asc order        } NlogN
int ans=0;
for(int i=0; i<n; i++)
{
    // i' smaller elements
    if(i == arr[i])
        ans++;
}
```

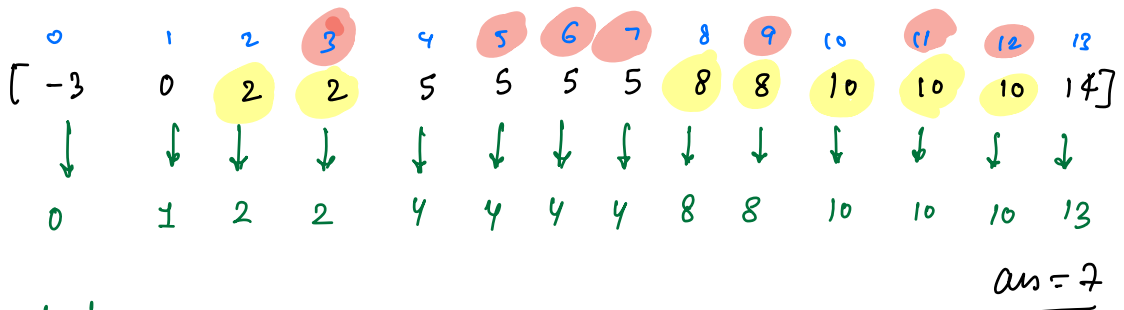
} N

T.C: N + NlogN
≈ O(N logN)

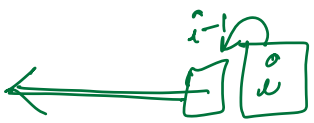
duplicates!







sorted



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

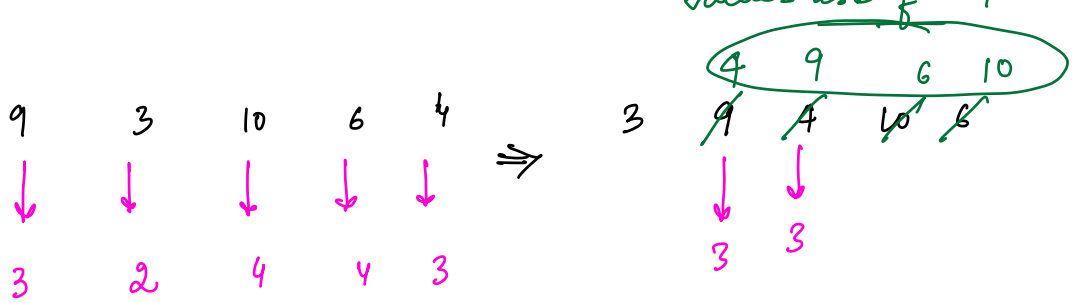
no of smaller elements

```
int less = 0;
if( arr[0] == less) ans++;
for( int i = 1; i < n; i++)
{
    if( arr[i] != arr[i-1])
    {
        less = i;
    }
    if( arr[i] == less)
        ans++;
}
```

asc of magnitude

sort the data in asc order of no of factors

if 2 value have same factors — sort those values asc of magnitude



comparator

Comparison

arr[i] < arr[i+1]

before

sort (___ , ___ , comp) ;

rules

from to

bool comp (<datatype> int a , <datatype> int b)

{

int cnta = count(a);

int cntb = count(b);

if (cnta < cntb)

return true;

else if (cnta == cntb)

{ if (a < b)

return true;

else

return false;

}

sort in asc order of factors

first argument

return true

second argument

return false;

```
        else {
            return false;
        }
    }

    if( count of div of a  < count of div of b)
        // "a" should come first
        return true;
```