Sorting → is arrangement of data in a particular order on the [basis of some parameter.]

A[] = [2, 3, 9, 12, 17, 19]

on what basis

comparing the magnitude

→ ascending order.

A[] = [19, 6, 5, 2, -1, -19]

wrt magnitude. descending order.

Quiz 1

Is the array sorted {1, 15, 9, 6, 12} sorted?

yes

count of factors → 1 2 3 4 5

→ Why is sorting required?—

- organizing ✓
- analyzing ✓
- searching. ✓
- presenting data efficiently. etc ✓

Qus1 Given an array of n integers, minimize the cost to empty given array where cost of removing an element is equal to sum of all elements in an array.

A[ ] = { 2, 1, 4 }

Ans = 11

(1) minimize the cost

2, 1, 4

remove 4 → 2 + 1 + 4 = 7

remove 2 → 2, 1 = 3

remove 1 = 1 = 1

11

=> ~~2~~, ~~1~~, 1

3 → 3 + 5 + 1 = 9

5 → 5 + 1 = 6

1 → 1 = 1

Quiz2 Minimum cost to remove all elements from array {4, 6, 1} ?

4, 6, 1

4 → 4 + 6 + 1 = 11

6 → 6 + 1 = 7

1 → 1 = 1

19

~~4~~ ~~6~~. 1.

6 → 4 + 6 + 1 = 11

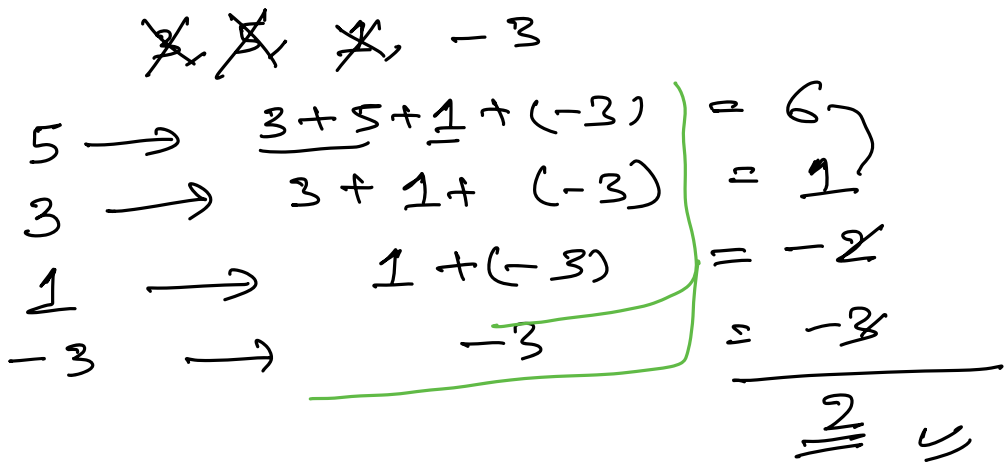
4 → 4 + 1 = 5

1 → 1 = 1

17

Quiz 3

Minimum cost to remove all elements from array[] = {3, 5, 1, -3}



Obs^{v} [, , , d]

Remove a →
Remove b →
Remove c →
Remove d →

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

Sum =

⇒ 1a + 2b + 3c + 4d →

Minimize sum.

Minimize = = a > b > c > d.

Obs^{v} We will have to remove the largest element first.

n log n ← Sort → O(n log n)

arr = [] = [1, 5, 3]

[5, 3, 1]

T.C: = n(log n) [reverseSort(arr);]

S.C: O(1)

```
sum = 0;
for (int i = 0; i < n; i++)
    sum += arr[i] * (i+1);
return sum;
```

i = 0
a[i] = 5 × (0+1) = 5

i = 1
a[i] = 3 × (1+1) = 6

i = 2

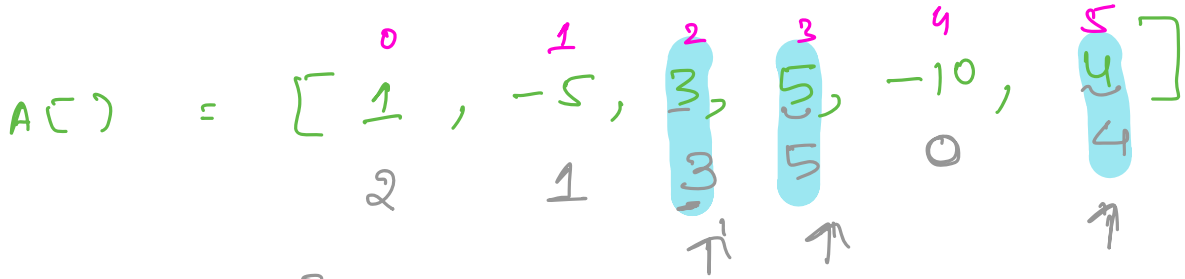
a[i] = 1 1 × 0+2
= 3

Q3 Given an array of distinct elements of size n, find the count of noble integers.

Note: arr[i] is noble if count of elements smaller than arr[i] is equal to arr[i] where arr[i] is element at index i.

A[ ] = { 1, -5, 3, 5, -10, 4}

Ans = 3



ans = 3

Quiz 4

Count the number of noble integers in the array. A = { -3, 0, 2 , 5 }

-3	0	2	5
0	1	2	3

ans = 1

Brute Force

T.C $O(n^2)$

S.C : $O(1)$

pseudo

1. Iterate over the element and select 1 by 1.
2. For that element iterate over array to find the count of smaller
3. If count is equal to that element array increment the ans count.

Optimize

(1) [5, 4, -1, 0]

[-1, 0, 4, 5] ✓

0 1 2 3

⇒ index

[-1, 0, 2, 5]

```
ans = 0
arr = [ ]
sort (arr);
for ( i = 0 ; i < n ; i++ )
{
    if (A[i] == i)
        ans++;
}
print (ans);
```

T.C: n log(n) + n ← nlogn

S.C: O(1)

i = 0 1 ans = 0 1

A[0] = -1 == 0

A[1] = 0 == 1

A[2] = 2 == 2

A[3] = 5 == 3

Q3

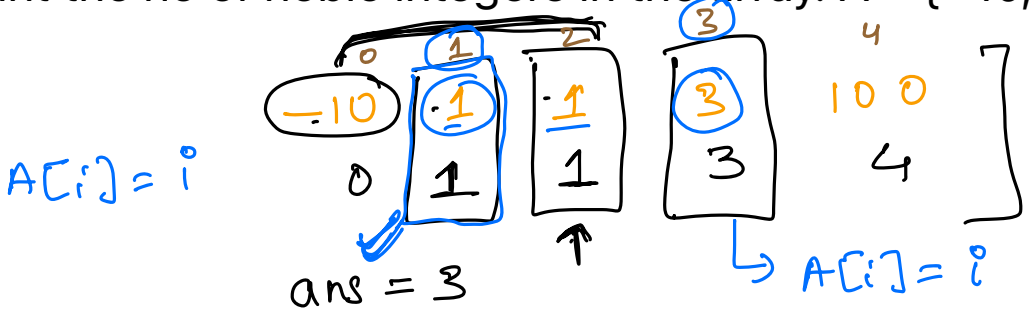
Given an array of size n, find the count of noble integers.

Note: Same as previous question, but all elements need not to be distinct

[5, 4, -1, 2, 2]

Quiz5

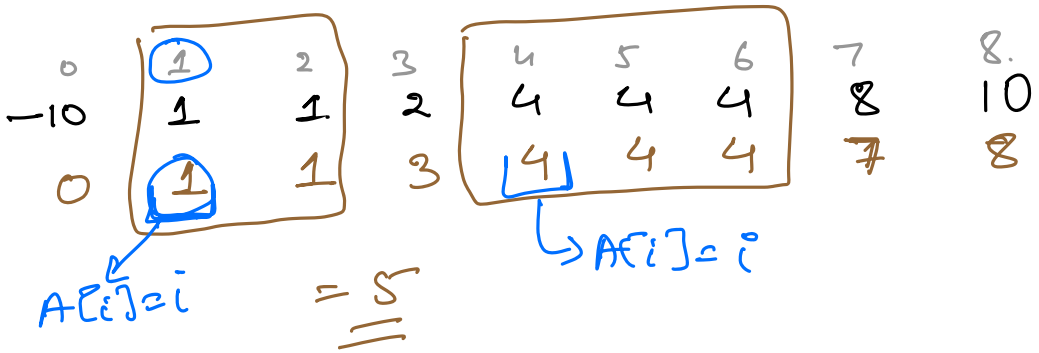
Count the no of noble integers in the array. A = { -10, 1, 1, 3, 100 }



Quiz6

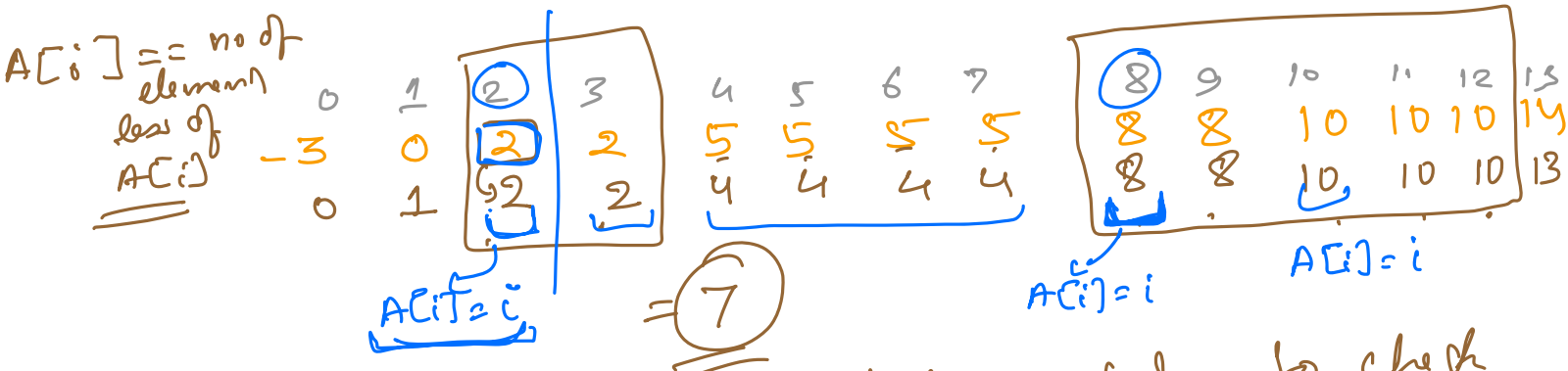
Count the no of noble integers in the array

A = { -10, 1, 1, 2, 4, 4, 4, 8, 10 }



Count the no of noble integers in the array

A = { -3, 0, 2, 2, 5, 5, 5, 5, 8, 8, 10, 10, 10, 14 }



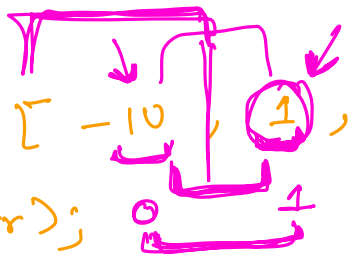
Obs^{v}

1. element needs to be sorted to check how many element are smaller.
2. The first matched element have A[i] = i
3. no. of element smaller than a number is constant irrespective of its duplicates

Soln

(1) I can find the first element using distinct element approach.

(2) check how many number is repeating and add to that count.




```
sort(arr);
count = 0. = 1
ans = 0 = 2
if(arr[0] == 0) ans++;
for(i = 1; i < n; i++)   i = 2
```

T.C. O(nlogn)

S.C O(1)

```
if (arr[i] != arr[i-1])
{
    count = i;
}
if (count == arr[i])
{
    ans++;
}
return ans;
```

A[i] = 0

Selection Sort

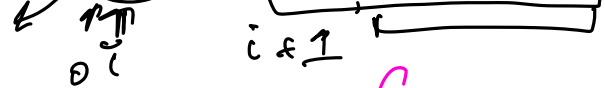
$\Rightarrow$ [5, 4, 1, 10, 3]

1 = [1, 4, 5, 10, 3]

2 $\Rightarrow$ [1, 3, 5, 10, 4]

3 $\Rightarrow$ [1, 3, 4, 10, 5]

4 $\Rightarrow$ [1, 3, 4, 5, 10]




```
selection sort ( arr , n )
{
    int minIndex;
    for ( int i = 0; i < n-1; i++ )
    {
        minIndex = i;
        for ( j = i+1 ; j < n ; j++ )
        {
            if ( arr[j] < arr [minIndex] )
            {
                minIndex = j ;
            }
        }
        swap ( arr [minIndex], arr[i] )
    }
```

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

S.C : $O(1)$

Algo

```
for ( i = 1 ;  i<n;  i++)
```