Intro

Arrangement of ~~numbers~~ data is a particular order on the basis of parameters.

2 5 8 10 => Sorted in ASC order on the basis of magnitude

50 15 7 3 => Sorted in DSC order on the basic of magnitude

Quiz: 1 13 9 6 12 , Is this array sorted?

-> Not Sorted based on magnitude.

	1	13	9	6	15	12
	↓	↓	↓	↓	↓	↓
No of Factors	1	2	3	4	4	6

why Sorting?

- → Ease searching
- → Ease Organising (Readability)
- → Ease Analysing (Comparison)
- → Ease Deduplication

Sorting

How to use Sorting to solve problems.

(More Important)

This we will cover here.

Different Sorting Algorithms

↳ 2 Basic Algo

↳ Rest we will cover in ADV module

Q:- Given an array of N integers. Minimize the cost to empty the given array where cost of removing a element is equal to sum of all elements left in the array at the moment.

$$A = \begin{array}{ccc} 0 & 1 & 2 \\ [\,2 & 1 & 4\,] \end{array}$$

Let's remove 4 ⇒ Cost = 2 + 1 + 4 = 7

[2 1]

Let's remove 2 ⇒ Cost = 2 + 1 = 3

[1]

Let's remove 1 ⇒ Cost = 1

[ ]

Total Cost 11 = Ans.

$$A = \begin{array}{ccc} 0 & 1 & 2 \\ [\,2 & 1 & 4\,] \end{array}$$

Let's remove 1 => Cost = 2 + 1 + 4 = 7

[2 4]

Let's remove 2 => Cost = 2 + 4 = 6

[4]

Let's remove 4 => Cost = 4

[ ]

Total Cost = 17

Quiz : minimum cost to remove all elements ?

[4 6 1]

Removing 6 => 4 + 6 + 1 = 11

[4 1]

Removing 4 => 4 + 1 = 5

[1]

Remove 1 => 1

[ ]

17 = Ans.

Quiz: minimum cost to remove all elements?

$[3 \quad 5 \quad 1 \quad -3]$

Remove 5 $\Rightarrow$ Cost $= 3+5+1 \ (-3) = 6$

$[3 \quad 1 \quad -3]$

Remove 3 $\Rightarrow$ Cost $= 3 + 1 + (-3) = 1$

$[1 \quad -3]$

Remove 1 $\Rightarrow$ Cost $= 1 + (-3) = -2$

$[-3]$

Remove -3 $\Rightarrow$ Cost $= -3$

Total Cost $= 6 + 1 + (-2) + (-3)$

$= 2$

Obs: Remove largest element first

$$\underrightarrow{[a \quad b \quad c \quad d]}$$

Remove a $\Rightarrow$ $a + b + c + d$

Remove b $\Rightarrow$ $b + c + d$

Remove c $\Rightarrow$ $c + d$

Remove d $\Rightarrow$ d

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

If I am removing element x at ith (0 based index) position, the cost contributed by x

$$= (i+1) * x$$

Sol: Sort the given array in DSC order.

[3 5 1 -3]

⇒ 5 3 1 -3

↓ ↓ ↓ ↓

$5*1$ $3*2$ $1*3$ $-3*4$

‖ ‖ ‖ ‖

$5 + 6 + 3 + -12 = +2$

```
int calculateCost{ int arr[], int n){
    reverse_sort(arr)
    int ans = 0
    for (int i = 0; i < N; i++) {
        ans += (i+1)*arr[i]
    }
    return ans
}
```

TC: O(N) + Time Complexity of Sorting

= O(NlogN)

For all in-build methods

TC: O(N log N)

SC: Depends on sorting algo

SC: O(1 to N)

Q Given an array of distinct elements. Find the count of nobel integers.

Arr[i] is nobel if count of elements smaller that arr[i] is equal to arr[i] where arr[i] is element and i is index

	[1	-5	3	5	-10	4]
	↓	↓	↓	↓	↓	↓
Smaller than A[i] ⇒	2	1	3	5	0	4
			⇓	⇓		⇓
			nobel	nobel		nobel

ans = 3

Quiz :

	-3	0	2	5
	↓	↓	↓	↓
Elements less than A[i]	0	1	2	3

ans = 1

Brute force : For each element, count the elements less that it and check nobel

TC: $O(N^2)$ SC: $O(1)$

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

How to optimise ?

→ What is extra work you are doing ?

Finding count of smaller elements

→ Can sorting help here ?

YES :)

[1 -5 3 5 -10 4]

!!

	0	1	2	3	4	5
	-10	-5	1	3	4	5
	↓	↓	↓	↓	↓	↓
Numbers smaller than A[i]	0	1	2	3	4	5

After Sorting : Each element have
[0 to i-1] elements
less than A[i]

Count = i

```
int   countNobelIntegers(int arr[], int n){
    sort(arr)
    int ans = 0
    for(int i = 0 ; i < N ; i++){
        if( i == arr[i])
            ans++
    }
    return ans;
}
```

TC: N + Time of Sorting

N + N log N

= O(N logN)

SC: Depends on Sorting Algo used

Q Given an array of integers. Find the count of nobel integers.

Arr[i] is nobel if count of elements smaller that arr[i] is equal to arr[i] where arr[i] is element and i is index

Quiz:

	0	1	2	3	4
	-10	1	1	3	100
Count of Elements less than A[i]	0	1	1	3	4

ans = 3

Quiz:

	0	1	2	3	4	5	6	7	8
	-10	1	1	2	4	4	4	8	10
Count of Elements less than A[i]	0	1 →	1	3	4 →	4 →	4	7	8

obs: The previous logic still works for first occurance of repeated number

Quiz:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
	-3	0	2	2	5	5	5	5	8	8	10	10	10	4
Count of Elements less than A[i]	0	1	2 →	2	4 →	4 →	4 →	4	8 →	8	10 →	10 →	10	13

ans = 7

```
int   CountNobelIntegers(int arr[], int n){
    sort(arr)
    int ans = 0
    int count = 0
    if(arr[0] == count)
            ans++

    for(int i = 1; i<N; i++){
            if(arr[i] != arr(i-1]){
                    count = i

            if(count== arr[i])
                    ans++
    }
    return ans;
}
```

0 1 2 3
-3 0 2 2

ans = ~~0~~ ~~1~~ 2

cont = ~~0~~ ~~1~~ 2

TC: N + Time of Sorting

N + N log N

= O(N LogN)

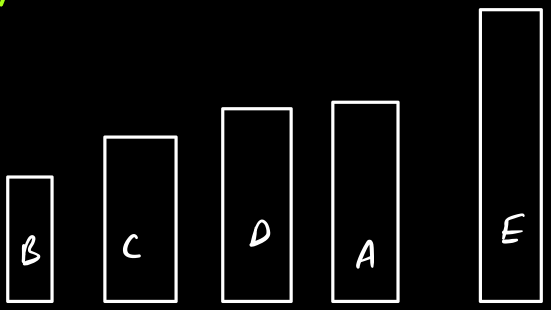
SC: Depends on Sorting Algo used

Selection Sort

Maddala's is very obidient kid.
He became monitor on the class.

Class teacher asked Maddala to arrange all the students in ASC a of thir height.

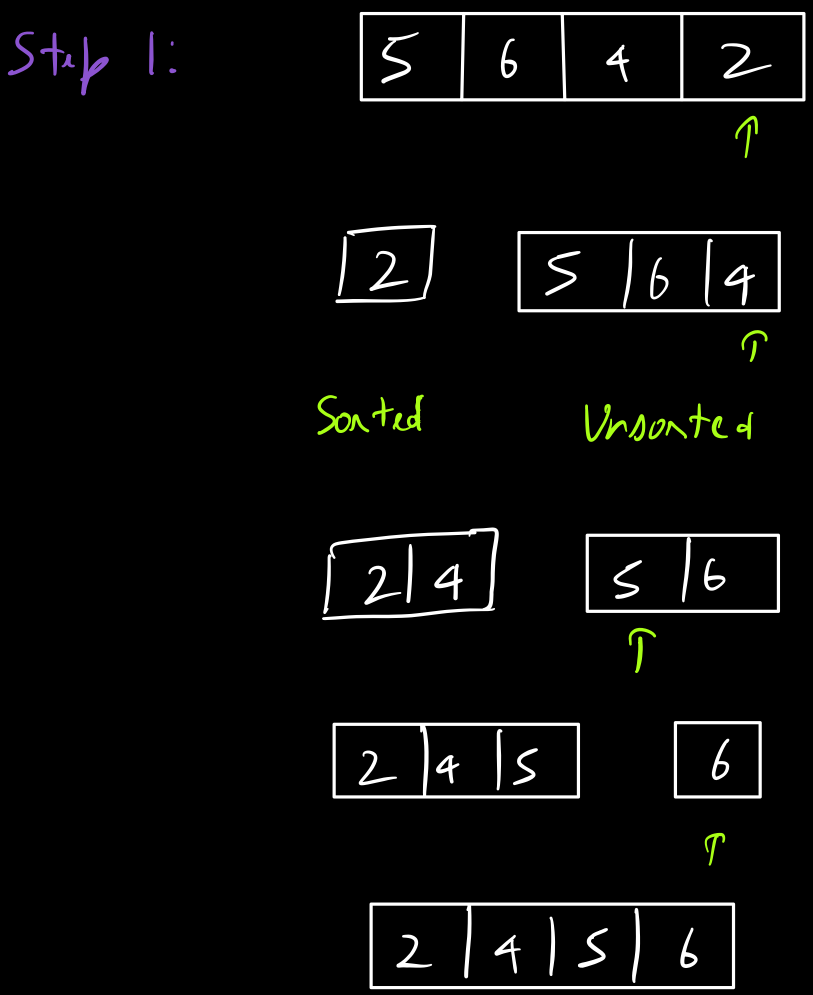
Find smallest student and put in back of line.



-) Place everything unarranged queue.

-) Search the shortest student and bring them front

-) Repeat the same thing untill whole class is done



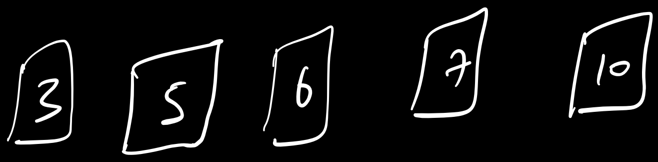
```
void selectionSort(int arr[], int size){
    int i, j, minIndex;
    for (i=0; i<size-1, i++){
        minIndex = i
        for(j=i+1; j<size; j++){
            if(arr[j] < arr[minIndex])
                minIndex = j
        }
        swap(arr[minIndex], arr[i])
    }
}
```

TC: $O(N^2)$

SC: $O(1)$

Insertion Sort

⇒ Very similar to sorting cards.



Inserting a new element in a sorted Array.

Sorted	Unsorted
—	3 5 2
3	5 2
3 5	2
2 3 5	

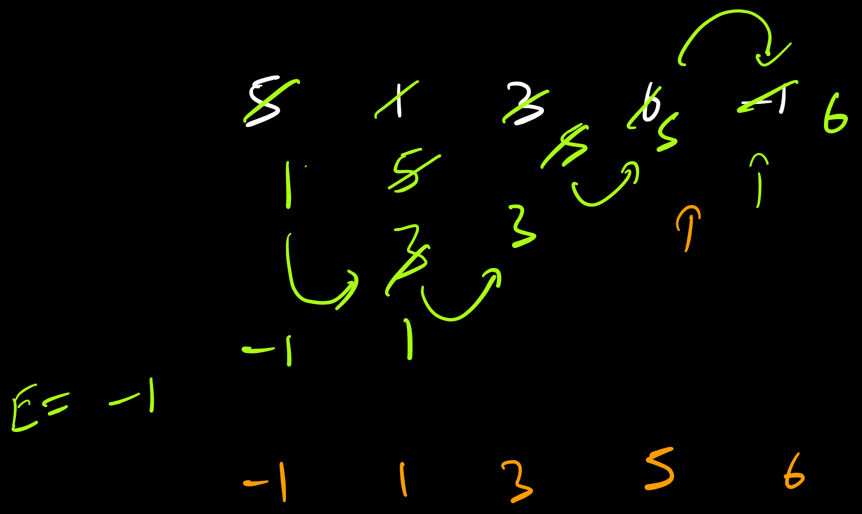
→ We don't process the first, there is nothing left to it

→ Loop from i till the end to process each element

- → Extract the position i element.
  Let call arr[i] as E
- → Compare E with left elements
  i-1 to 0
- → If E is lesser, then move arr[j] to

right by 1

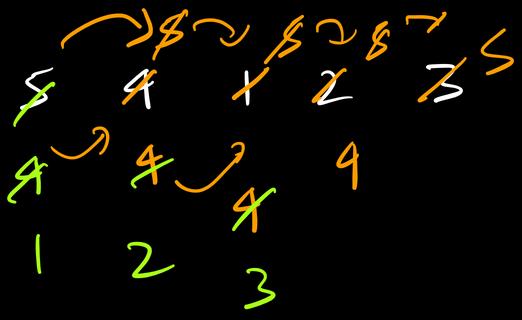
-> Once we found the position for E. Place it there



```
voin   insertionSont( int arr[] , int n){

    for(i=1; i<N; i++){
        int curEle = arr[i]
        # Find the right place for curEle and move elements by 1
        j = i-1
        while(j>=0 && arr[j] > curEle){
            arr[j+1] = arr[j];
            j = j-1;
        }
        arr[j+1] = curEle;
    }
}
```

TC: $O(N^2)$ SC: $O(1)$



cur Ele = 3

=) 1 2 3 4 5

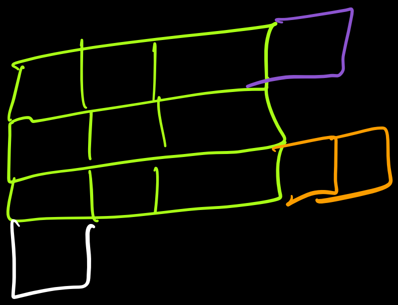
Next Class: Strings

- -) Text Handling
- -) Data Represention
- -) Input and Output
- -) Text Processing
- -) File Handling
- -) Users Interfaces
- -) Error Messages

Doubts.

→ 2D array will have fix N * M size

→ Array list extend size anytime



```
mat[0].push_back(5);

mat[2].push_b.  (x);
mat[2].push   .(x);

mat.pushback(ArrayList<>{1,5})
```