Cover → 1) What is Sorting.
2) Stable sort.
3) Questions
4) Comparators

Not cover → Any sorting Algo.

Sorting → TC = O(N log(N))
SC = O(1) or O(N)
Advance Content

What is sorting ?

Ordered arrangement of objects.

Eg → 1) Books in a library
2) People in order of height.
3) Arranging house items by our Mom.
4) Words in oxford dictionary.

Order

[-5, 10, 2, 3, 0, 1]

Increasing Order → [-5, 0, 1, 2, 3, 10] $A[i] < A[i+1]$ $\forall i$

Decreasing Order → [10, 3, 2, 1, 0, -5] $A[i] > A[i+1]$ $\forall i$

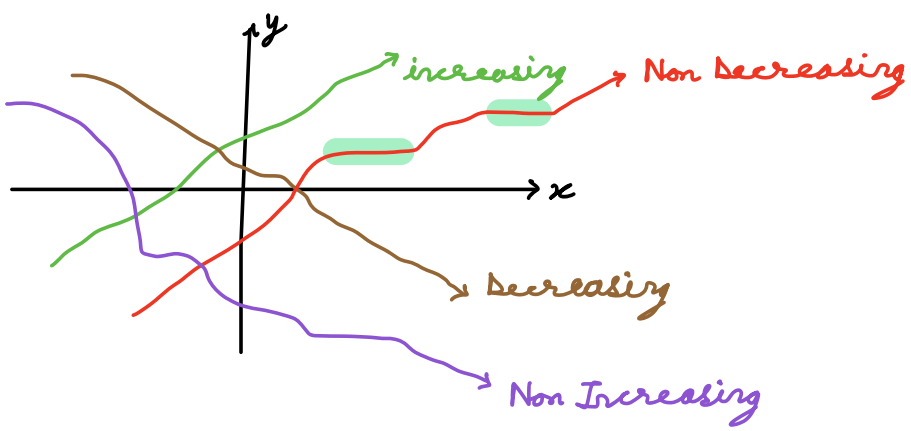
Ascending

[-5, 10, 2, -5, 3, 2, -5, 10]

Descending

Non-Decreasing → [-5, -5, -5, 2, 2, 3, 10, 10] $A[i] <= A[i+1]$ $\forall i$

Non-Increasing → [10, 10, 3, 2, 2, -5, -5, -5] $A[i] >= A[i+1]$ $\forall i$



A → a = 10 b = -5 c = 0

Increasing order → b c a ✓
-5 0 10

Order in Strings

Alphabetic Order / Lexicographical Order {a, b, c ... z}

happy < sad
h < s

success > save
u > a

time > tide
m > d

[dog, bat, apple, rubber] → [apple, bat, dog, rubber]

cool < cooler
null < e

apple > Apple
a > A

ASCII

a → 97	A → 65	'0' → 48
b → 98	B → 66	'1' → 49
⋮	⋮	⋮
z → 122	Z → 90	'9' → 57

Q → Sort the strings based on length. (Ascending order → Default)

0	1	2	3	4	5
[apple,	dog,	bat,	scaler,	kite,	gold]
5	3	3	6	4	4

[dog, bat, kite, gold, apple, scaler]

Desc → [scaler, apple, kite, gold, dog, bat]

Q → Sort the integers based on absolute value.

0	1	2	3	4	5	6
[-10,	6,	2,	-6,	-8,	1,	8]
10	6	2	6	8	1	8
[1,	2,	6,	-6,	-8,	8,	-10]

|-10| = 10
|10| = 10

$|x| \rightarrow x$ if $x >= 0$
$|x| \rightarrow -x$ if $x < 0$

Stability → If x_1 is before x_2 & $x_1 = x_2$
then in sorted order also x_1 should be before x_2.
(compare index for equal elements in array)

Sorting a Collection

1) Java → `ArrayList<Integer> A;`
`Collections.sort(A);` ✓

2) C++ → `vector<int> A;`
`sort(A.begin(), A.end());` ✓

3) Python → `A = [ . .. . ]`
`A.sort()` // A is changed.

`B = sorted(A)` // A is not changed.

K	A	U → preferred member
1	2	3
2	3	1

10:28 PM

Q→ Given an integer array A.
Find min cost of removing all elements from the array.
cost of removing A[i] → sum of all elements present in the array before removing A[i].

A = [2 4 1]
0 1 2

Removed Element

2 → 2+4+1 = 7
4 → 4+1 = 5
1 → 1 = 1
13

Removed Element

4 → 2+4+1 = 7
2 → 2+1 = 3
1 → 1 = 1
11 ✓

Removed Element

1 → 2+4+1 = 7
4 → 2+4 = 6
2 → 2 = 2
→ 15

A = [3 5 2]
(indices: 0 1 2)

3 5 2 → (3+5+2) + (5+2) + 2 = 19
3 2 5 → (3+2+5) + (5+2) + 5 = 22
5 3 2 → (3+2+5) + (3+2) + 2 = 17 (Ans)
5 2 3 → (3+2+5) + (3+2) + 3 = 18
2 3 5 → (3+5+2) + (3+5) + 5 = 23
2 5 3 → (3+5+2) + (3+5) + 3 = 21

Observation → [a b c d] (indices: 0 1 2 3)

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

1.a + 2.b + 3.c + 4.d

min cost → largest . . . . smallest

Solution Steps →
1) Sort in Descending order
2) ans = 0
   for i → 0 to (N-1)
       ans += A[i] * (i+1)
   return ans

A = [3 5 2] ⟶ [5 3 2] (indices: 0 1 2)

ans = 5*1 + 3*2 + 2*3 = 5+6+6 = 17

5 → (5 + 3 + 2)
3 → (3 + 2)
2 → (2)

Q → Given an array of size N,
count the number of noble integers present.

Noble Integer → Any element A[i] is noble iff number of elements less than A[i] is equal to A[i].

```
          0   1   2   3   4   5   6
A = [ 1  -5   3   5   5  -10   4 ]
# <A[i] → 2   1   3   5   5   0   4        Ans = 4
```

```
          0   1   2   3   4
A = [-10,  1,  3,  1,  100]
#<A[i] →  0   1   3   1   4        Ans = 3
```

Observations → 1) Can -ve number be noble integer? → No

A[i] → count of integers >= 0

2) Distinct Elements →

```
A = [ 10   1   3   -5   4   -10   5 ]
```

Sort in Ascending order →

```
[-10  -5   1   3   4   5   10 ]
  0   1   2   3   4   5   6
```

```
#elements < A[i] → 0   1   2   3   4   5   6
```

Solution Steps → 1) Sort in ascending order.

2) ans = 0

```
for i → 0 to (N-1)
    if (A[i] == i)
        ans += 1
return ans
```

3) Non-Distinct Elements →

Sorted → [-10 1 1 2 4 4 4 8] Ans = 5

```
          0   1   2   3   4   5   6   7
# elements < A[i] → 0   1   1   3   4   4   4   7
```

1) Sort
2) ans = 0
   cnt = 0 // #elements < A[i]
   for i → 0 to (N-1)
      if (i > 0 && A[i] != A[i-1])
         cnt = i
      if (A[i] == cnt)
         ans += 1
   return ans

A = [0 0 2 2 4 6]
idx: 0 1 2 3 4 5
cnt = 0 0 0 2 2 4 5
ans = 0 + 1 + 2 + 3 + 4 + 5 ✓

A[i-1] < A[i]
0 ____ (i-1) < A[i]

[count of elements < A[i]] <= (N-1)

10 A[i] → 10 elemese < A[i]

Comparators →

```
int compare(Integer a, Integer b){

    // If we want a on left of b in o/p → return -ve number.
    // If we want a on right of b in o/p → return +ve number.
    // If we don't want to change order → return 0.
}
```

```
ArrayList<Integer> A;
Collections.sort(A, mysort);
```

```
array A[] = _ _ _
Arrays.sort(A, comp_);
```

```
Comparator<Integer> mySort = new Comparator<Integer>(){

    @Override
    public int compare(Integer a, Integer b){
        return a - b;
    }
```

a = 10 b = 5 a - b → +ve 5 ____ 10

a = 2 b = 4 a - b → -ve 2 ____ 4

ascending order

Descending order → return b-a;

Stability ✓

a = 6 b = 6 a - b = 0

abs(x) = |x|

Ascending order based on absolute value →

return abs(a) - abs(b);

---

a = -10 b = 5

5 ____ -10

abs(a - b) = abs(-15) = 15 ✓

abs(a) - abs(b) = 10 - 5 = 5 ✓

a = -5 b = 10

-5 ____ 10 ←

abs(a - b) = abs(-5 - 10) = abs(15) = 15 ✓

abs(a) - abs(b) = 5 - 10 = -5 ✓

10 ____ -5 ✗

|5| < |10|

-5 is on left of 10