Sorting

Friday - Contest

Bit

Modular

Sorting

25th Apr - 1st May

Hashing 1 & 2

Agenda

- What is sorting ?
- Inbuilt sort method
- Min cost to remove all elements
- Noble Integer
- Comparators

Sorting Algorithms → Will be covered in the Advance Batch

What is sorting ?

Arranging ~~integers~~ data in ~~asc / desc~~ specific order

based on some parameter.

Sort a deck of cards → Value
→ Color
→ Suits

Ex1

3 8 9 14 19 → Ascending By value

Ex2

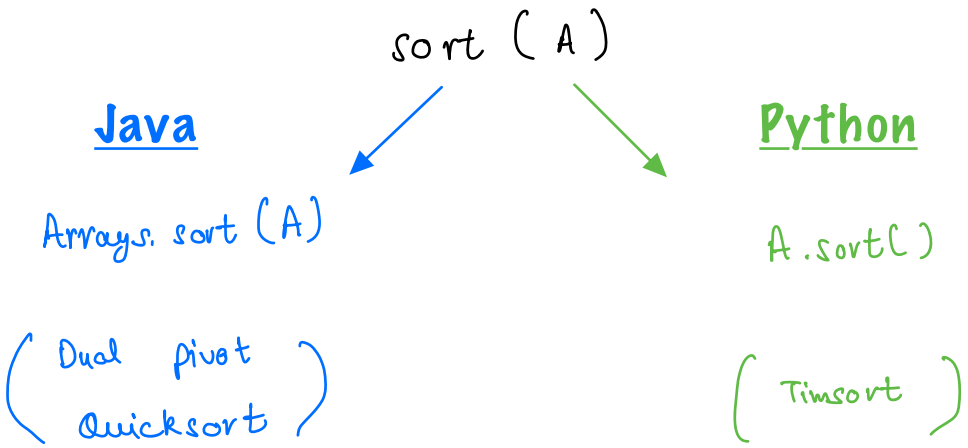
19 14 9 8 3 → Descending By value

Ex3

1	3	5	9	6	10	12
Factors: 1	2	2	3	4	4	6

→ Ascending By no of factors

Inbuilt Sort Methods



To sort an array of N items.

TC: $O(N \log_2 N)$

Min cost to remove all elements

Given N array elements, at every step remove an array element.

Cost to remove element = Sum of array elements present in the array.

Find the min cost to remove all elements.

Example

arr[3] = 2 1 4 (indices 0, 1, 2)

		Cost
Remove 1	[2, 1, 4]	2 + 1 + 4 = 7
Remove 2	[2, 4]	2 + 4 = 6
Remove 4	[4]	4
		17

Remove 4	[2, 1, 4]	2 + 1 + 4 = 7
Remove 2	[2, 1]	2 + 1 = 3
Remove 1	[1]	1
		11

Ans = 11

Example **Quiz 1**

arr[] = 4 6 1

Remove 6	[4, 6, 1]	4 + 6 + 1 = 11
Remove 4	[4, 1]	4 + 1 = 5
Remove 1	[1]	= 1
		17

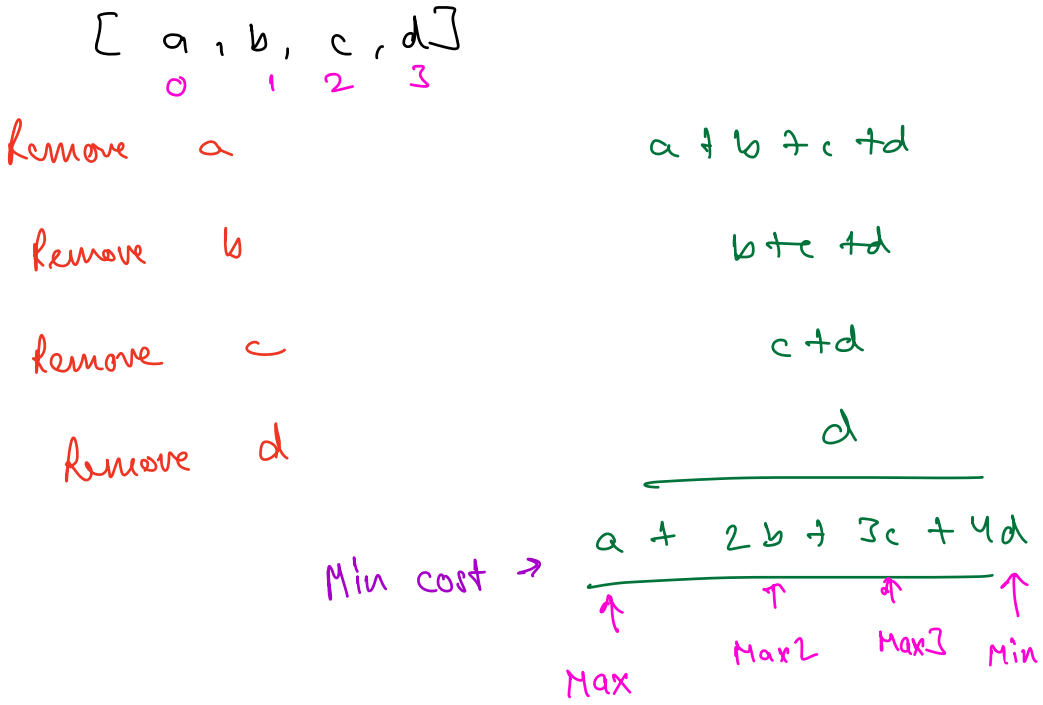
Example **Quiz 2**

arr[] = 3 5 1 -3

Remove 5	[3, 5, 1, -3]	3 + 5 + 1 + -3 = 6
Remove 3	[3, 1, -3]	3 + 1 + -3 = 1
Remove 1	[1, -3]	1 + -3 = -2
Remove -3	[-3]	= -3
		2

Observation

We have to delete the elements in descending order of value to get the min cost.



Pseudocode

Sort in descending order ← TODO
← O(n log n)

```
cost = 0
for ( int i=0; i< N; i++) {
    cost = cost + A[i] * (i+1)
}
return cost
```

TC : O(N log N)
SC : O(1)

Noble Integer

Given N array elements of unique numbers, calculate number of noble integers present in it.

A[i] is said to be Noble if

(No of elements < A[i]) = A[i]

Example

arr[ ] =	1	-5	3	5	-10	4
Count of elements < arr[i]	2	1	3	5	0	4

Ans = 3

Example Quiz 3

Sorted

arr[ ] =	-3	0	2	5
Count of elements < arr[i]	0	1	2	3
Index	0	1	2	3

Ans = 1

Example Quiz 4

Sorted

arr[] =	-10	-5	1	3	4	5	10
Count of elements < arr[i]	0	1	2	3	4	5	6
Index	0	1	2	3	4	5	6

Ans = 3

Brute Force

```
ans = 0
for  i → [0, n-1] : {
     c = 0
     for  j → [0, n-1] : {
          if  arr[j] < arr[i]
               c++
     }
     if (c == arr[i])
          ans++
}
return ans
```

TC : $O(N^2)$

SC : $O(1)$

Optimised Solution

```
ans = 0
sort(A)                          <- O(N log N)
for (i=0; i<N; i++) {
    if (A[i] == i)
        ans++
}
return ans
```

TC : O(N log N)

SC : O(1)

Break till 10:06 PM

Noble Integer 2

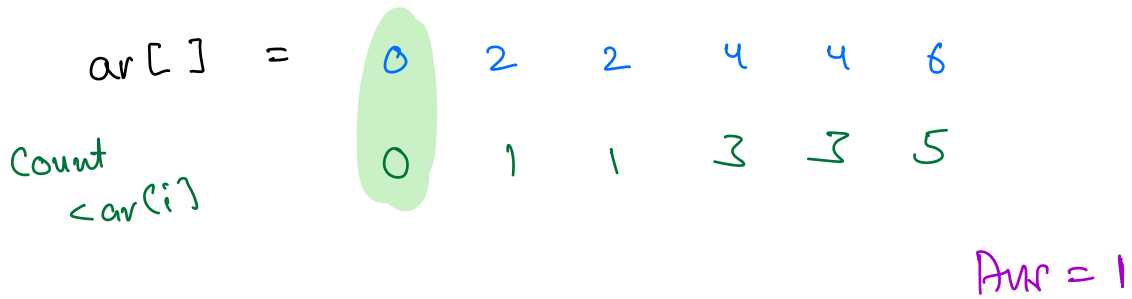
Given N array elements, calculate number of noble integers present in it.

Note: Data can repeat.

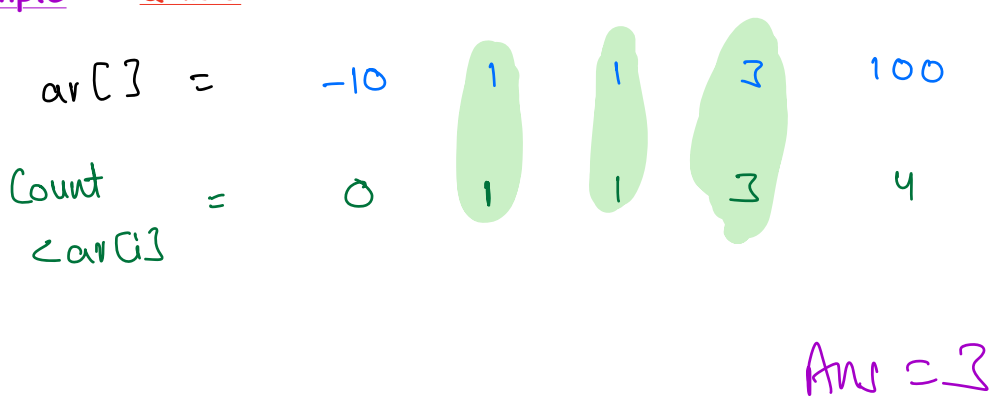
A[i] is said to be Noble if

(No of elements < A[i]) = A[i]

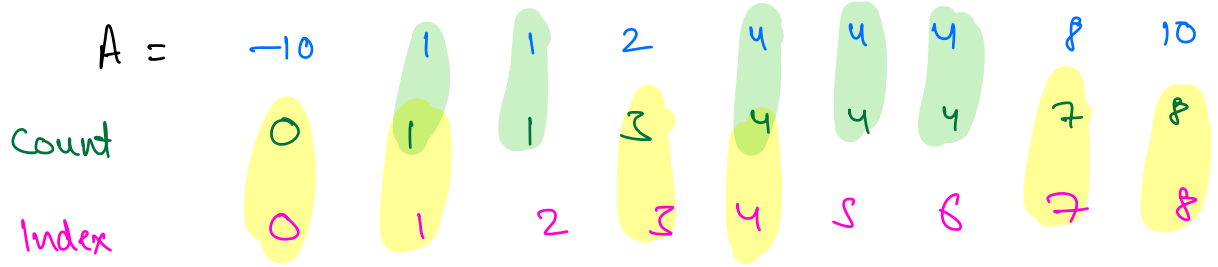
Example



Example Quiz 5

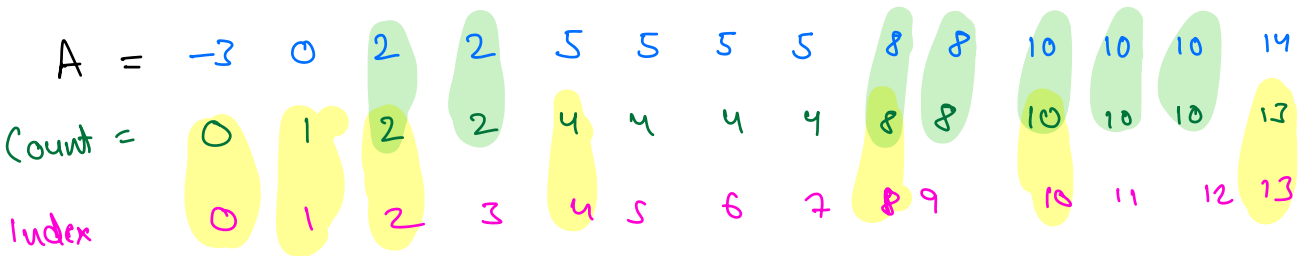


Example Quiz 6



Ans = 5

Example Quiz 7



Ans = 7

Brute force - will work - $O(N^2)$

Optimised sol - will not work
(Previous question)

Observations

If element is same as previous, Quiz 8

A. Count will increment by 1
B. Count will not change
C. Count will be same as index
D. Count will be same as element

if A[i] == A[i-1]

Count won't change

When an element comes for the first time,

No. of elements less than A[i] = index

count

if (A[i] != A[i-1])

count = i

Pseudocode

```
int nobleIntegers(int A[]) {
        n= A.len
        ans = 0
        Sort (A)    =  Ascending order
        if (A[0] == 0)
            ans = 1
        for (i=1; i< N; i++) {
            if A[i] == A[i-1]
                Do nothing
            if (A[i] != A[i-1])
                count = i
            if (A[i] == count)
                ans ++
        }
        return ans
}
```

TC : O(N log N)

SC : O(1)

Java

```java
int nobleInteger2(int[] A) {
    int n = A.length;
    Arrays.sort(A);
    int c = 0, ans = 0;
    if (A[0] == 0)
        ans = 1;
    for (int i = 1; i < n; i++) {
        if (A[i] != A[i - 1])
            c = i;

        if (A[i] == c)
            ans++;
    }

    return ans;
}
```

Python

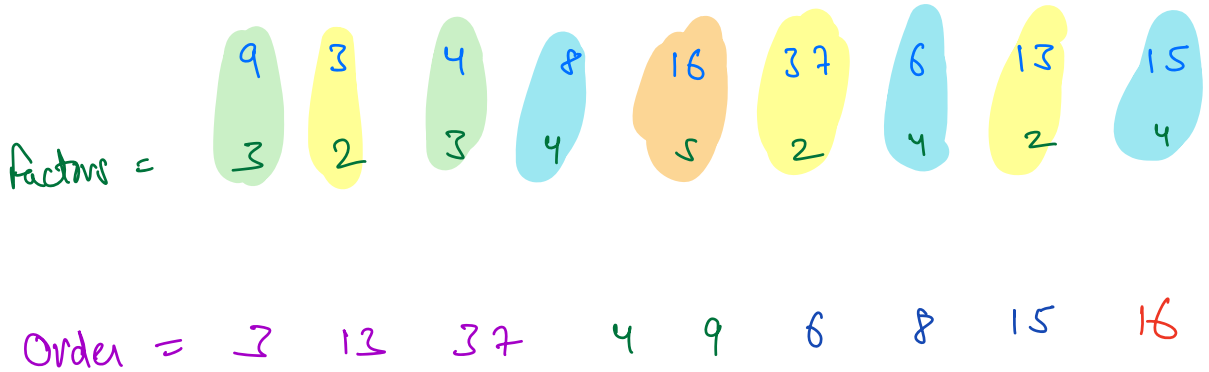
```python
def nobleInteger2(A):
    n = len(A)
    A.sort()
    ans = 0
    c = 0
    if A[0] == 0:
        ans = 1
    for i in range(1, n):
        if A[i] != A[i - 1]:
            c = i

        if A[i] == c:
            ans += 1

    return ans
```

Comparators

Given N array elements, sort them in increasing order of their No of factors.
If 2 elements have same no. of factors, element with less value should come first.

Note: No extra space allowed.

Example



Example

Order = 1 23 25 6 10 14 21

Sort (A , ———)

A → Array

——— → Comparator

Allows us to sort based on some custom parameter

Concept of Comparator

Sort based on no of factors

Ex 1

a: 25 → 3

b: 16 → 5

25 will come first

Return -ve

Ex 2

a: 10 → 4

b: 9 → 3

9 will come first

Return true

Ex 3

a: 49 → 3

b: 25 → 3

25 will come first

Return true

Ex 4

a: 10

b: 10

Return 0

Example

Input : 8 6 3 49

8 → 4
6 → 4
3 → 2
49 → 3

Sorted Array : 3 49 6 8

At every step, sorting algo will pick 2 elements & it will compare them. Based on the comparison, it will decide whether to rearrange them or not.

Above process will repeat till the data is sorted.

Java, Python

If you want a to come first — Return -ve

If you want b to come first — Return +ve

Pseudocode

```
comparator ( int a , int b ) {
    fa = factors(a)
    fb = factors(b)
    if ( fa < fb)
        return -ve
    else if ( fa > fb)
        return +ve
    else {                          ← Compare a & b
        if(a < b)
            return -ve
        else if ( a > b)
            return +ve
        else
            return 0
}
```

return fa - fb

return a - b

Java

```java
import java.util.Arrays;
import java.util.Comparator;

class MyFactorComparator implements Comparator<Integer> {
    int countFactors(int n) {
        int c = 0;
        for (int i = 1; i <= n; i++) {
            if (n % i == 0)
                c++;
        }
        return c;
    }

    public int compare(Integer a, Integer b) {
        int factorsOfA = countFactors(a);
        int factorsOfB = countFactors(b);

        if (factorsOfA == factorsOfB)
            return a - b;
        else
            return factorsOfA - factorsOfB;
    }
}

class Main {
    public static void main(String[] args) {
        Integer[] A = { 9, 3, 4, 8, 16, 37, 6, 13, 15 };
        MyFactorComparator c = new MyFactorComparator();
        Arrays.sort(A, c);

        for (int i = 0; i < A.length; i++) {
            System.out.print(A[i] + " ");
        }
    }
}
```

Python

```python
from functools import cmp_to_key


def countFactors(n):
    c = 0
    for i in range(1, n + 1):
        if n % i == 0:
            c += 1
    return c


def myFactorComparator(a, b):
    factorsOfA = countFactors(a)
    factorsOfB = countFactors(b)

    if factorsOfA == factorsOfB:
        return a - b
    else:
        return factorsOfA - factorsOfB


def main():
    A = [9, 3, 4, 8, 16, 37, 6, 13, 15]
    A.sort(key=cmp_to_key(myFactorComparator))

    print(A)


main()
```

Doubts

Thank you

Best sorting algo - O(N log N)

Good Night

Thank you

Friday

Contest on friday

Announcements on slack channels