Sorting

Agenda

- Why sorting ?
- Problems

Problem Solving
Session 2
on 13th (Saturday)
Optional

What is sorting ?

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

based on some parameter

Sort a deck of cards → Suits
→ Color
→ value

Ex1

3 8 9 12 16 21

Ascending By value

Ex2

48 32 31 23 19 10

Descending By value

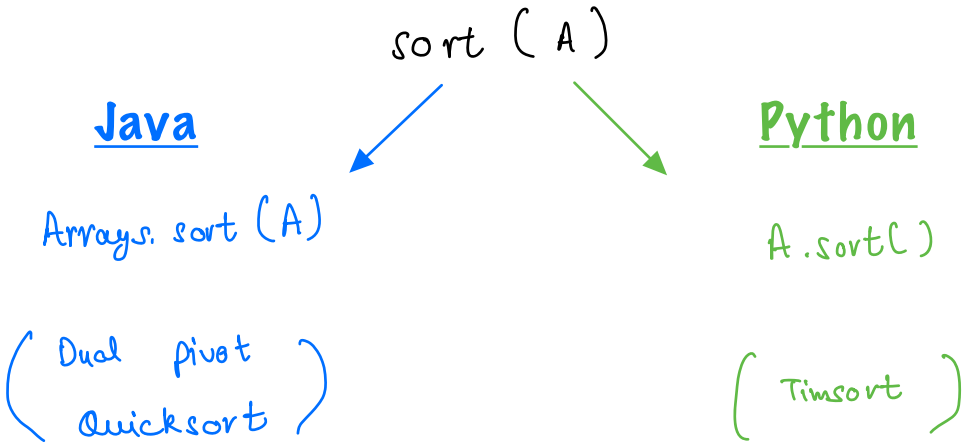
Ex 3

1 5 3 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: $O(N \log_2 N)$

A = [4, 2, 3]

Python specific

B = sorted(A)

A remains unchanged

B will get a sorted copy of A

A .sort( )

A get sorted

Q1. 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

	0	1	2
arr[3] =	2	1	4

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

		Cost
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] 6

Remove 3 [3, 1, -3] 1

Remove 1 [1, -3] -2

Remove -3 [-3] -3

2

Observation

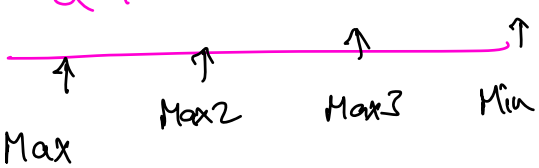
We have to delete the elements in descending order of value to get the min cost.

[a , b , c , d]

Remove a	[a, b, c, d]	a + b + c + d
Remove b	[b, c, d]	b + c + d
Remove c	[c, d]	c + d
Remove d	[d]	d

Min cost

a + 2b + 3c + 4d



(index + 1) * arr[i]

Pseudocode

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

```
cost = 0

for( i=0; i < N; i++ ) {          ← O(n)

    cost = cost + (i+1) * arr[i]

}

return cost
```

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

Q2. 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

Unsorted

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

Ans = 3

Example Quiz 3

Sorted

ar[ ] =	-3	0	2	5
Count of elements < ar[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

For every element, check if it is noble.

```
ans=0
for (i=0; i<N; i++) {
        c=0
        for (j=0; j<N; j++) {
                if ( ar[j] < ar[i])
                        c++
        }
        if ( ar[i] == c)
                ans++
}
return ans
```

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

Optimised Solution

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

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

Break till 10:10 PM

Q3. Noble Integer 2

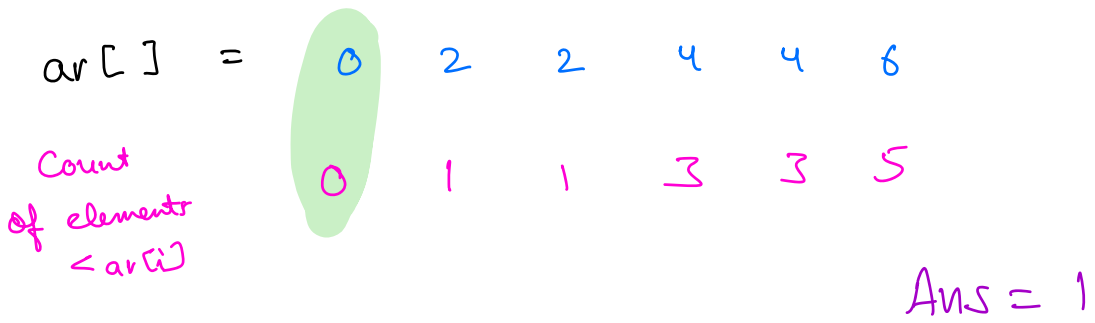
Given N array elements, calculate number of noble integers present in it.

Note: Data can repeat.

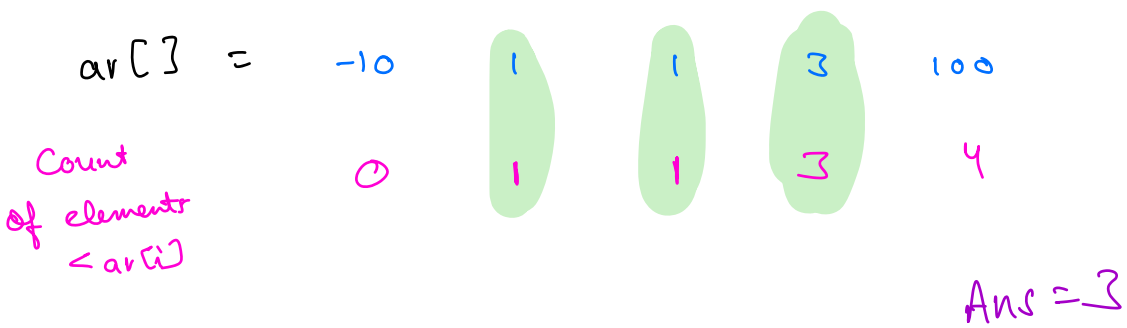
A[i] is said to be Noble if

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

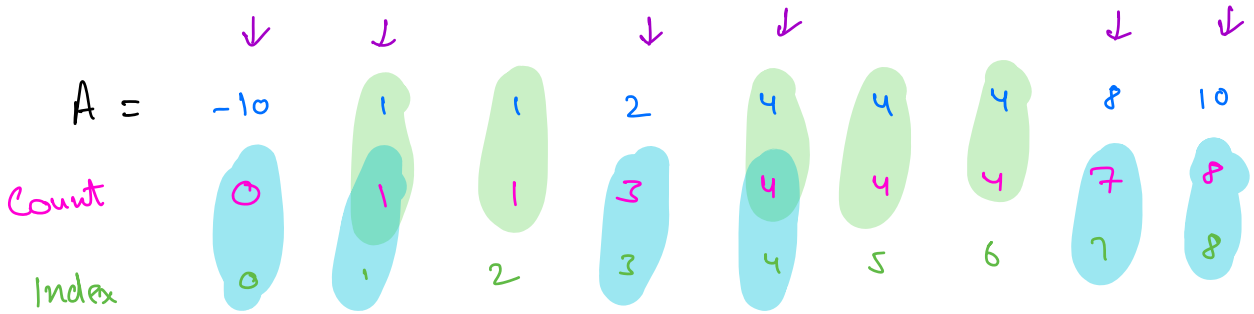
Example



Example Quiz 5

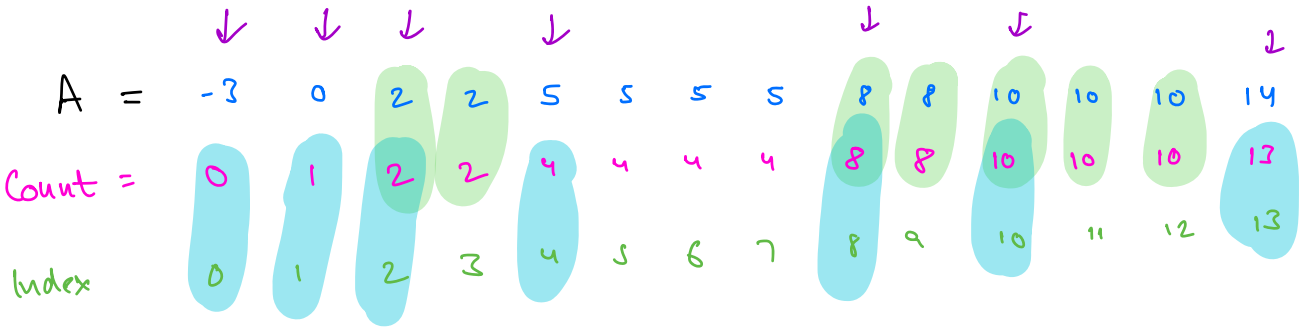


Example Quiz 6



Ans = 5

Example Quiz 7



Ans = 7

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

Optimised Sol - will not work

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 will
    not change
```

New element positions.

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[]) {
        int n = A.length
        int ans = 0
        sort(A)
        if ( A[0] == 0 )
                ans ++
        for(i=1; i<N; i++) {
                if(A[i] == A[i-1])
                        Count will not change
                if ( A[i] != A[i-1] )
                        count = i
                if ( A[i] == count)
                        ans ++
        }

        return ans
}
```

← Do nothing

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
```

Time - O( )

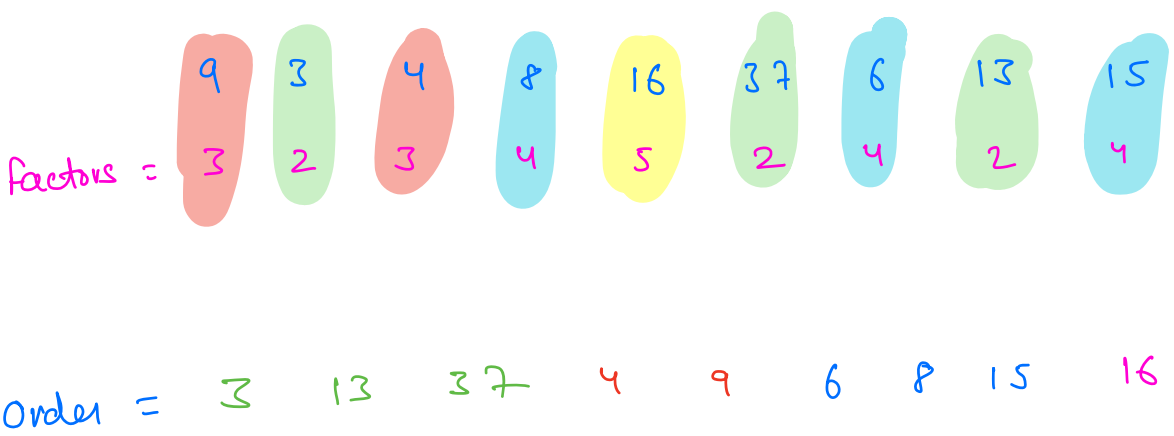
Space - O( )

Comparators

Q4. 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.

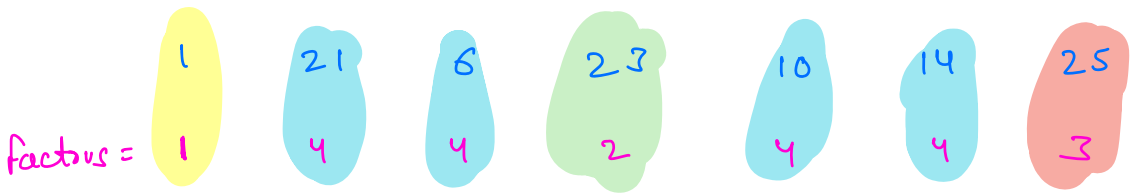
Note: No extra space allowed.

Example



Order = 3 13 37 4 9 6 8 15 16

Example



Order = 1 23 25 6 10 14 21

---

sort (A , ———)

↑ Array

↖ Comparator

Allows us to sort based on some parameter

Concept of Comparator

Sort on no of factors

Ex 1

a = 25 → 3

b = 16 → 5

25 comes first

Return -ve

Ex 2

a = 10 → 4

b = 9 → 3

9 comes first

Return true

Ex 3

a = 49 → 3

b = 25 → 3

25 comes first

Return true

Ex 4

a = 10

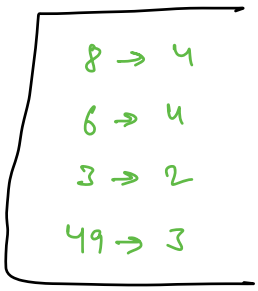
b = 10

Any order

Return 0

Example

Input : 8 6 3 49



Sorted Array : 3 49 6 8

At every step, sorting algorithm is going to take 2 elements at a time and it compares them. it will rearrange them based on the comparison result.

above process is done until the entire array is sorted.

Python, Java

If you want a to come first → Return -ve

If you want b to come first → Return +ve

Pseudocode

Sorting based on no of factors

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

return fa - fb

return a - b

Value Comparison

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()
```

Sorting Comparisons — $n \log n$

In each comparison, you are going to call countFactors — $O(N)$

TC : $O(n^2 \log n)$

SC : $O(1)$

Doubts

Good Night

Thank You

Friday