Today's Content:

1. Inversion Count

2. Custom Comparison

Question:

Given an array A of integers, count the total no. of inversions.
(i,j) is an inversion if $i<j$ and $A[i]>A[j]$.

$N \leq 10^5$

A: 2 3 0 1 (indices 0 1 2 3)

(i,j) = (2,3)	(i,j) = (3,2)	(i,j) = (0,2)	(i,j) = (1,2)
✓ (2,0)	✓ (3,0)	✗ (0,3)	✗ (1,3)
✓ (2,1)	✓ (3,1)	✗ (0,1)	✗ (1,0)
✗ (2,2)	✗ (3,3)	✗ (0,0)	✗ (1,1)

Ans: 4

$i<j$ and $A[i]>A[j]$.

A: 8 5 3 4 1 6 2 (indices 0 1 2 3 4 5 6)

8 6 5 4 3 2 1

6

Ans: 15

(8,5) (5,3) (3,1) (4,1) (6,2)
(8,3) (5,4) (3,2) (4,2)
(8,4) (5,1)
(8,1) (5,2)
(8,6)
(8,2)

Brute Force:

```
int inversionCount(int[] A)
{ int n = A.length;
  int count=0;
  for(i=0; i<n ; i++)
  {
     for(j=i+1; j<n ; j++)
     {
        if(A[i] > A[j])
        { count++; }
     }
  }
  return count;
}
```

TC: $O(N^2)$

SC: $O(1)$

TLE

	0	1	2	3	4	5	6
A:	8	5	3	4	1	6	2

Can we sort? NO

$i < j \quad A[i] > A[j]$

	0	1	2	3	4	5	6
A:	8	5	3	4	1	6	2

count = ~~0 4 7 8 11 12 13 14~~ 15

Left Half

	0	1	2	3
	8	5	3	4

L: 8 5 R: 3 4

: 5 8 : 3 4

i j

	0	1
	8	5

L: 8 R: 5

i j

	0	1
	3	4

L: 3 R: 4

i j

Solid

	0	1	2	3
	3	4	5	8

i

Right Half

	0	1	2
	1	6	2

L: 1 R: 6 2

: 1 : 2 6

i j

	0	1
	6	2

L: 6 R: 2

i j

Solid

	0	1	2
	1	2	6

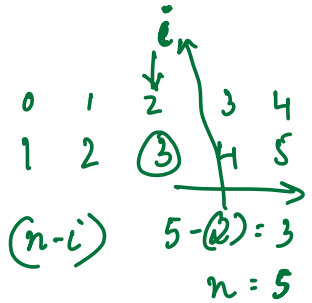
j

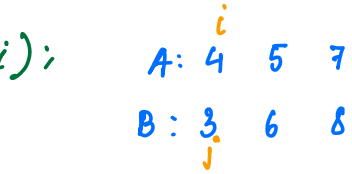
Steps during sorting using merge sort.

1. Sort left half
2. Sort right half.
3. Merge the 2 halves.

Code:

```
int count = 0;
int[] merge(int[] A, int[] B)
{
    int m = A.length, n = B.length;
    int[] ans = new int[m+n];
    int i=0, j=0, k=0;
    while (i<m && j<n)
    {
        if(A[i] <= B[j])
        {
            ans[k] = A[i];
            i++; k++;
        }
        else if (A[i] > B[j])
        {
            count = count + (m-i);
            ans[k] = B[j];
            j++; k++;
        }
    }
    while (j < n)
    {
```





```
            ans[k] = B[j];
            j++; k++;
        }
        while (i < m)
        {
            ans[k] = A[i]
            i++; k++;
        }
        return ans;
    }
```

```
int [] mergeSort(int[]A, int low, int high)
{
    if (low == high)
    { int[] ans = new int[1];
      ans[0] = A[low];
      return ans;
    }
    int mid = (low + high) / 2;
    int [] left = mergeSort(A, low, mid);
    int [] right = mergeSort(A, mid+1, high)
    int [] ans = merge(left, right);
    return ans;
}
```

Break till 10:40.

Is this array sorted (ordered based on some criteria) ?

A:	13	7	6	8	12	36
	2	2	4	4	6	>6

Arrays.sort()

Custom Comparison:

1. Sort by value

2. Sorting using a comparator

a) Sort an array of integers based on the no. of factors.

b) Sort a 2D array of points on a plane
   (i) Point with lesser value of x co-ordinate is lesser.
   (ii) If x co-ordinates of the points are equal then compare value of y co-ordinates.

Doubts :

```java
public static int[] merge(int[] A, int[]B) {
    int m = A.length;
    int n = B.length;

    int[] ans = new int[m+n];

    int i=0, j=0, k=0;

    while(i<m && j<n) {
        if(A[i] <= B[j]) {
            ans[k] = A[i];
            k++; i++;
        }
        else if(A[i] > B[j]) {
            count = count+(m-i);
            ans[k] = B[j];
            k++; j++;
        }
    }
    while(i<m) {
        ans[k] = A[i];
        k++; i++;
    }
    while(j < n) {
        ans[k] = B[j];
        k++; j++;
    }
    return ans;
}
```

```java
public static int[] mergeSort(int[] A, int low, int high) {
    if(low == high) {
        int[] ans = new int[1];
        ans[0] = A[low];
        return ans;
    }
    int mid = (low+high)/2;
    int[] left = mergeSort(A, low, mid);
    int[] right = mergeSort(A, low: mid+1, high);
    return merge(left, right);
}
```

```java
public static void main(String[] args) {
    int[] input = {45, 10, 15, 25, 50};

    int[] ans = mergeSort(input, low: 0, high: input.length-1);

    for(int i=0; i<ans.length; i++) {
        System.out.println(ans[i]);
    }

    System.out.println("The total number of inversions = "+count);
}
```

```java
public static void main(String[] args) {
    Integer[] A = {12, 13, 6, 8, 36, 7};
    Arrays.sort(A, new Comparator<Integer>() {
        @Override
        public int compare(Integer o1, Integer o2) {
            //return a positive value if o1 should appear after o2 in the sorted array
            //return a negative value if o2 should appear after o1 in the sorted array
            //return 0 if the order os o1 and o2 doesn't matter
            int c1 = countFactors(o1);
            int c2 = countFactors(o2);
            if(c1 < c2) {
                return -1;
            }
            else if(c1 > c2) {
                return 1;
            }
            return 0;
        }
    });

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

```java
//Sort these points sucht that all the points with lesser x co-ordinate appear first
//If x coordinate is equal - compare y co-ordinates.
Integer[][] A = {
        {1, 2}, {3, 4}, {-1, -1}, {4, 5}, {4, 7}
};
Arrays.sort(A, new Comparator<Integer[]>() {
    @Override
    public int compare(Integer[] o1, Integer[] o2) {
        //return a positive value if o1 should appear after o2 in the sorted array
        //return a negative value if o2 should appear after o1 in the sorted array
        //return 0 if the order os o1 and o2 doesn't matter
        int x1 = o1[0];
        int y1 = o1[1];

        int x2 = o2[0];
        int y2 = o2[1];

        if(x1 > x2) {
            return 1;
        }
        else if(x1 < x2) {
            return -1;
        }
        else {
            return y1-y2;
        }
    }
});
```