Agenda

1) Inversion count **

2) custom comparison
   └ sort on the basis of no. of factors etc.

3) Largest number

Q.1 Given an A[], count total no. of inversions. (i, j) is an inversion if i < j but A[i] > A[j].

A[] = [2 3 0 1]
      0 1 2 3

value based =

(2,0) (2,1)
(3,0) (3,1)

A[] ⇒ [8 5 3 4 1 6 2]
      0 1 2 3 4 5 6

cnt = 15

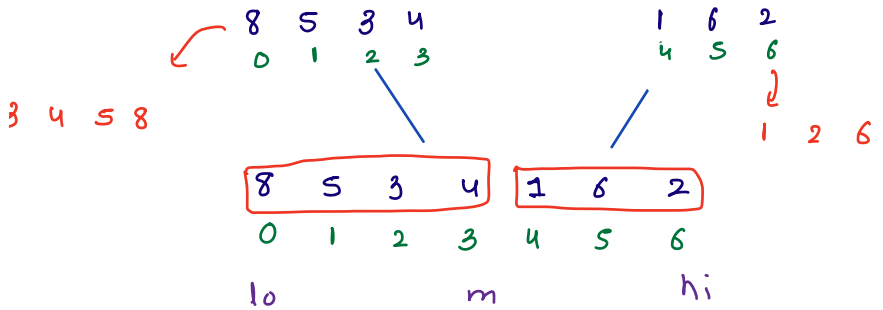
(8,5) (5,3) (4,1)
(8,3) (5,4) (4,2)
(8,4) (5,1) (6,2)
(8,1) (5,2)
(8,6) (3,1)
(8,2) (3,2)

i) Brute idea : $O(n^2)$

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

TC: $O(n^2)$

Expected TC: $O(n \log n)$



A = 3 4 5 8

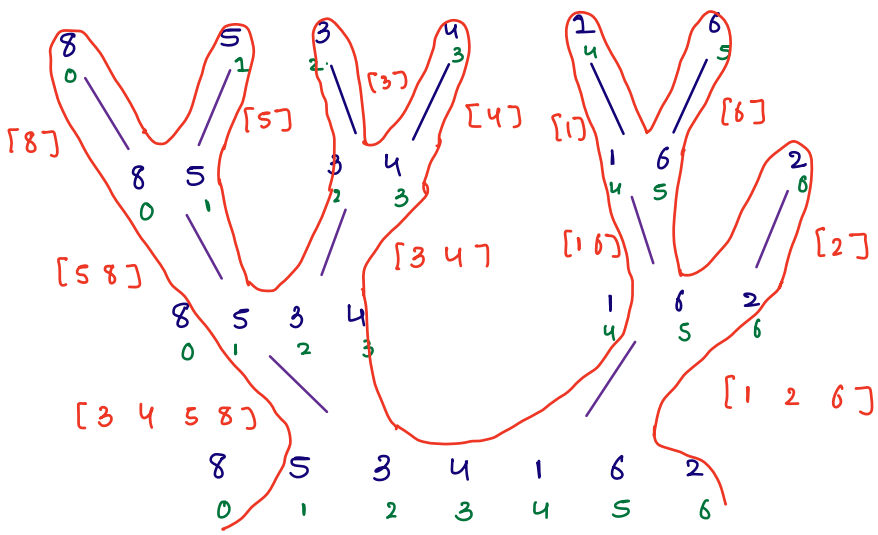
B = 1 2 6

ans = 1 2 3 4 5 6 8

A[i] > B[j]

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

inversions: (8,5) (5,3) (8,3) (5,4) (8,4) cnt = 1+2+2+1
(6,2) (3,1) (4,1) (5,1) (8,1) +4+4+1
(3,2) (4,2) (5,2) (8,2) (8,6)



A = [3 4 5 8]

B = [1 2 6]

ans = 1 2 3 4 5 6 8

A[i] > B[j]
(inversions)

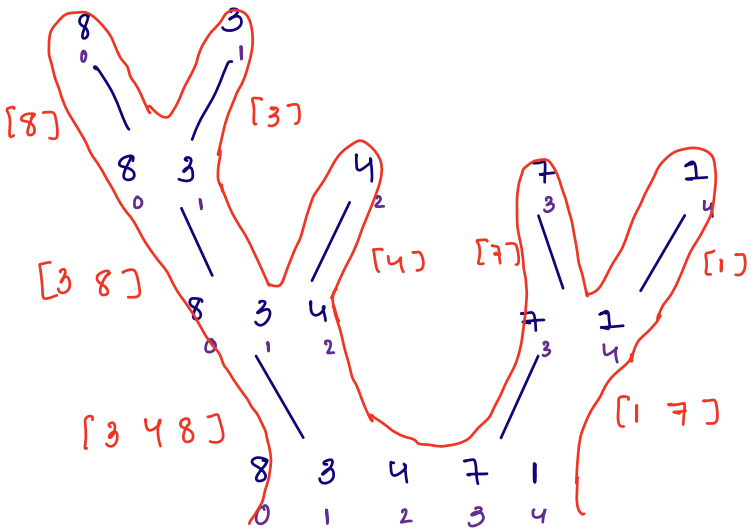
1 2 3 4 5 6 8

(8,3) (8,4) (7,1)
(3,1) (4,1) (8,1) (8,7)

cnt += 1+1+1+3+1

A : [3 4 8]

B : [1 7]



merge { A[i] > B[j] ; cnt += (n-i) }

(n = A.length)

↳ 1 3 4 7 8

TC : O(nlogn)

Custom comparison

```
int [] A = { 2, 3, 1, 13, 10 }

Arrays.sort(A); // Ascending order on the basis of value of element
```

```
public int compare(a,b)
```

- if you want to keep a before b return -ve
- if you want to keep a after b return +ve
- otherwise return 0.

<, >

Doubts

Insertion sort using recursion

```
void solve(int[] A) {
    helper(A, A.length-1);
}

void helper(int[] A, int idx) {
    if(idx==0) {
        return;
    }
    helper(A, idx-1);
    // do your task, to insert A[idx] in sorted region
    for(int j=idx-1; j>=0; j--) {
        if(A[j] > A[j+1]) {
            // swap A[j], A[j+1];
        }
        else {
            return;
        }
    }
}
```

A	1	2	3	4
41c	0	1	2	3

```
static void fun(int x){
    if(x>0){
        fun(--x);
        sop(x);
        fun(--x);
    }
}
psv main(){
    int a=4;
    fun(a);
}
```

o/p 0 1 2 0 3 0 1