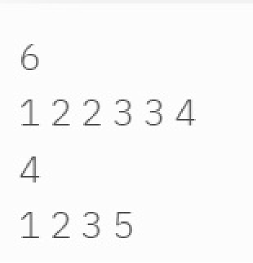
Union Of Two Sorted Arrays

T : O(n+m)

S : O(n+m)



```
6
1 2 2 3 3 4
4
1 2 3 5
```

A : 1 2 2 3 3 4 → n

B : 1 2 3 5 → m

merged : 1 1 2 2 2 3 3 3 4 5

1 2 3 4 5

A : 1 2 2 3 3 4

B : 1 2 3 5 5 6 6 6

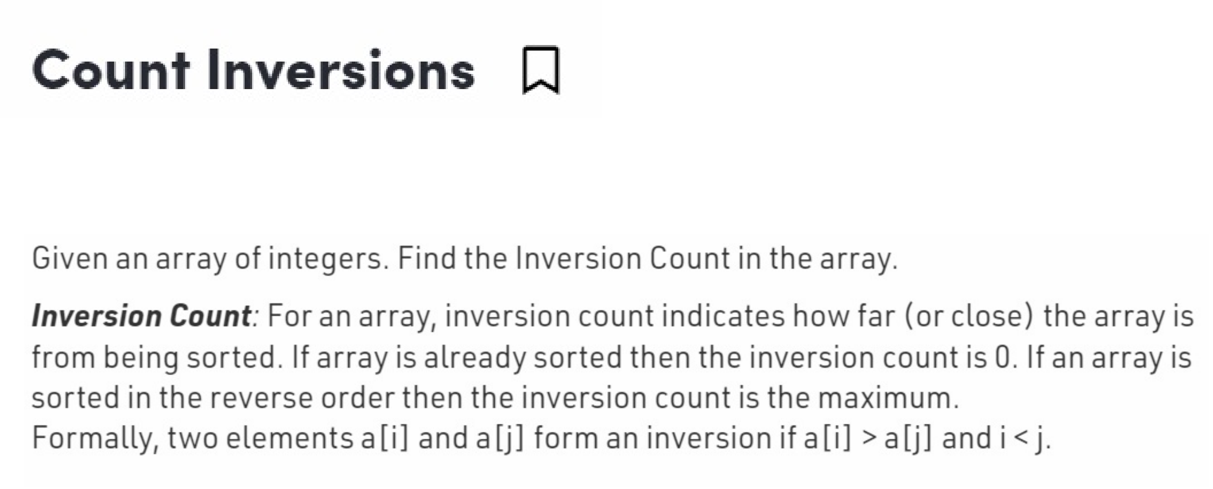
union: 1 2 3 4 5 6

(Ad)

T: O(n+m)

S: O(1)

```
if( A[i] ≤ B[j] ){
        use A[i];
        i++;
}
else {
        use B[j];
        j++;
}
```



Count Inversions

Given an array of integers. Find the Inversion Count in the array.

Inversion Count: For an array, inversion count indicates how far (or close) the array is from being sorted. If array is already sorted then the inversion count is 0. If an array is sorted in the reverse order then the inversion count is the maximum.
Formally, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j.

merge sort

arr : 2 4 1 3 5

order of merge sort

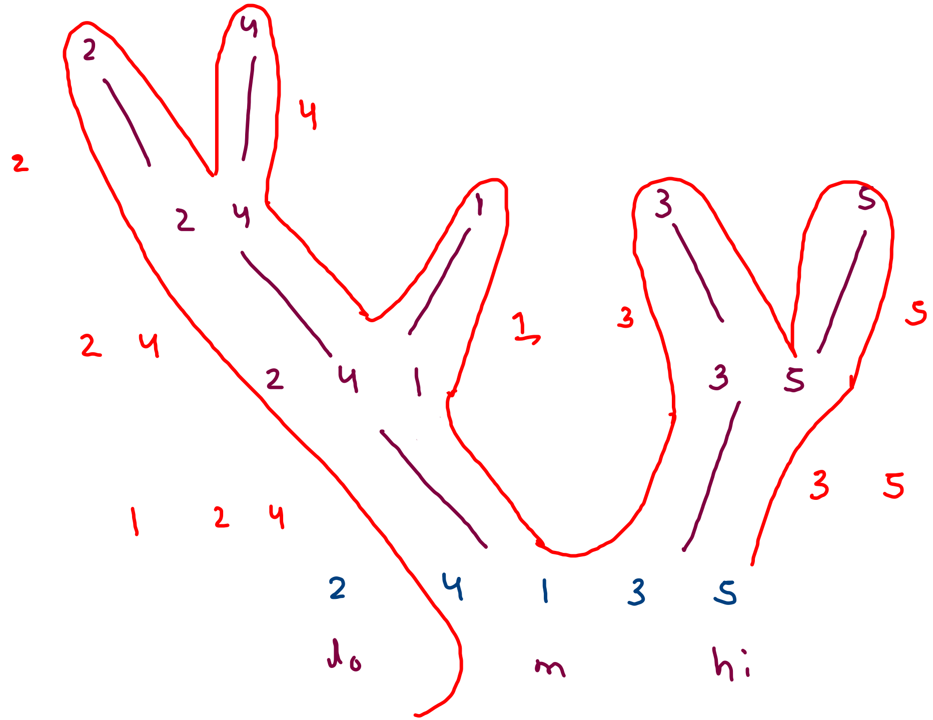
arr : 2 4 1 3 5

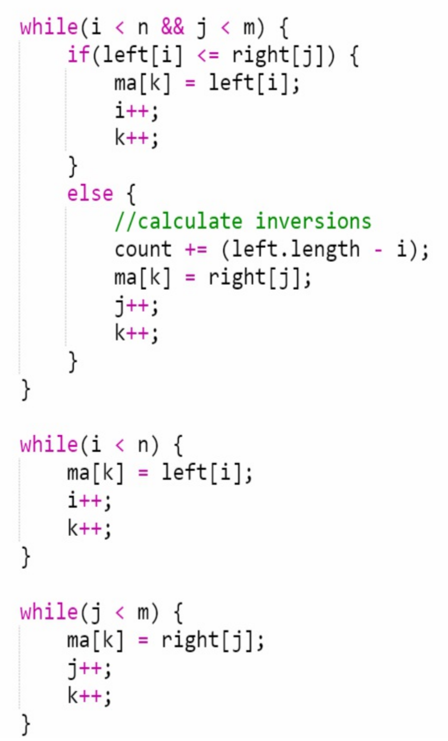
3 inversions

2, 1

4, 1

4, 3



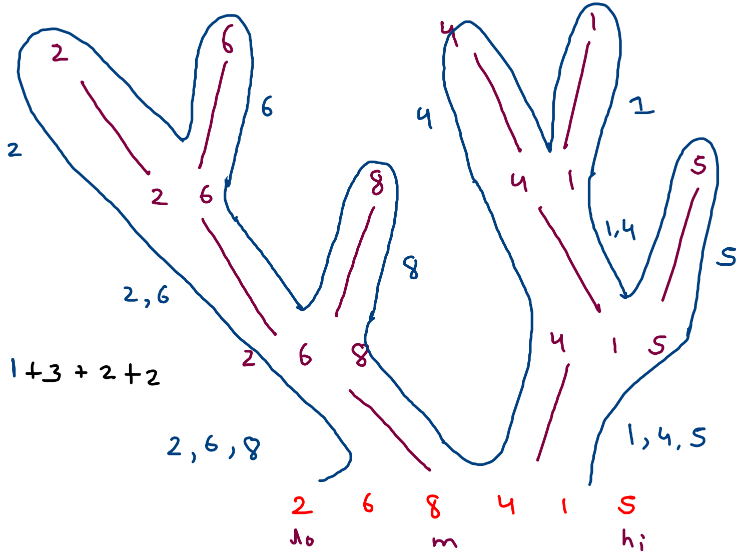


```
while(i < n && j < m) {
    if(left[i] <= right[j]) {
        ma[k] = left[i];
        i++;
        k++;
    }
    else {
        //calculate inversions
        count += (left.length - i);
        ma[k] = right[j];
        j++;
        k++;
    }
}

while(i < n) {
    ma[k] = left[i];
    i++;
    k++;
}

while(j < m) {
    ma[k] = right[j];
    j++;
    k++;
}
```

arr : 2 6 8 4 1 5



count = 1+3+2+2

ans : 8

2	6	8

i

1	4	5

j

1	2	4	5	6	8