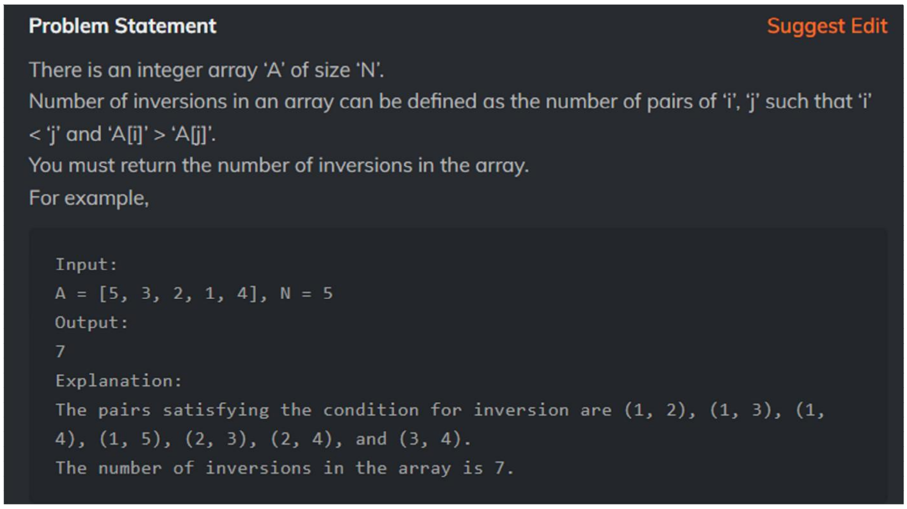
40. Count Inversions in an Array | Brute and Optimal



Problem Statement Suggest Edit

There is an integer array 'A' of size 'N'.
Number of inversions in an array can be defined as the number of pairs of 'i', 'j' such that 'i' < 'j' and 'A[i]' > 'A[j]'.
You must return the number of inversions in the array.
For example,

```
Input:
A = [5, 3, 2, 1, 4], N = 5
Output:
7
Explanation:
The pairs satisfying the condition for inversion are (1, 2), (1, 3), (1,
4), (1, 5), (2, 3), (2, 4), and (3, 4).
The number of inversions in the array is 7.
```

Count Inversion

arr[] = [5 3 2 4 1]

$i < j$ & $(a[i] > a[j])$

no of pairs should be left element > right element

(5, 3) ⇒ $5 > 3$ (4,2) X

(5,2) ⇒ $5 > 2$ because $i < j$ X

(5, 1) ⇒ $5 > 1$ left element > right element

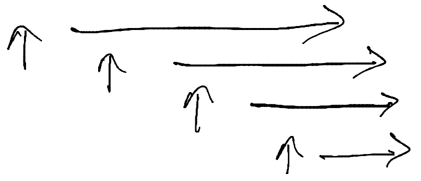
(3,2)   (2,1)

(3,1)   (4,1)

Count = 8 ⇒ return this

1) Brute Force :

arr[] = {5 3 2 4 1}



```
cnt = 0
for( i = 0 ⟶ n-1)
{   for( j = i+1 ⟶ j++)
    {
        if(a[i] > a[j])
            cnt += 1;
    }
}
return cnt;
```

T.C ⇒ ≈ $O(n^2)$ S.C ⇒ $O(1)$

2) Better Approach $n^2 \rightarrow n \log n$ or n

Suppose there are two sorted array:

[2, 3, 5, 6] [2, 2, 4, 4, 8]

Now to find how many no of pair you can form where left element is from left array & right element is from right array and it's greater.

e.g. $5 > 2 \Rightarrow$ pair.

Take two pointers.

[2, 3, 5, 6] [2, 2, 4, 4, 8]

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑

here 1st 2 == 2, so move

left > right

└ we need to find this

$\boxed{3 > 2 \Rightarrow \text{pair}}$

so that means every element right of '3' will make pair with '2' as array is sorted.

$\boxed{5 \geq 2}$, $\boxed{6, 2}$ Cnt += 3

So, now move the right array to 2

again all of the three element will make

pair so again cnt += 3 ⇒ 3 + 3 = 6

So, now move the right array to 4

(3 < 4) X so move left array

(5 > 4), so all the element from '5' will

make pair with the '4'

(5, 4)   (6, 4)   Cnt = 6 + 2 = 8

Now move the right array again it's '4'.

(5 > 4) so cnt += 2 ⇒ 8 + 2 = 10

move the left array,

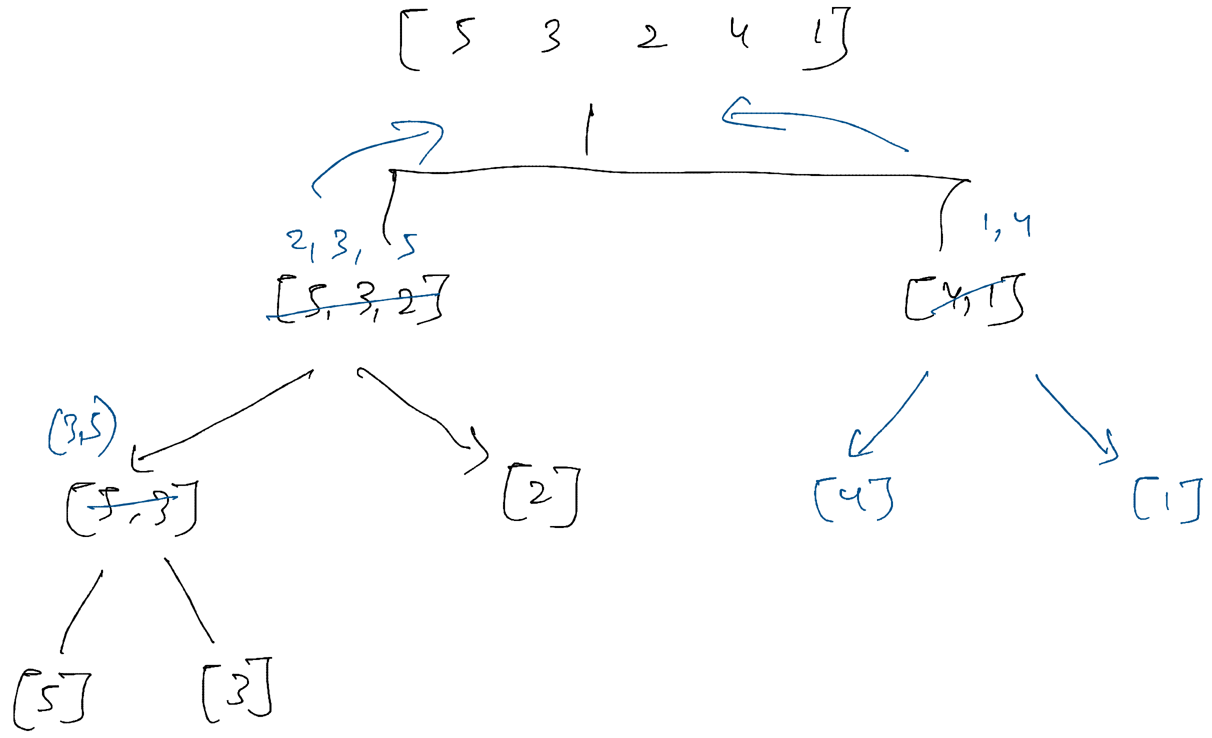
(6 > 8) ✗ so array got exhaused.

2 2 2 3 4 4 5 6 8

So now if we map the array into some sorted

So now if we map the array into some sorted form and get the left section & right section sorted, so here we think of merge sort algorithm

Approach + Dry Run



This is array in itself

[5]   [3]

everyone → right of 5 can form a pair.
here only 3 is there, so cnt +1.

[3, 5]   [2]
^

[3, 5]   [2]
↑   ~~↑~~ ↑

3 can pair with 2.

so that mean all the element will form pair now +2,

+1

[2, 3, 5]   [1, 4]

T.C ⇒ $O(n \log n)$

S.C ⇒ $O(N)$

Watch this video again