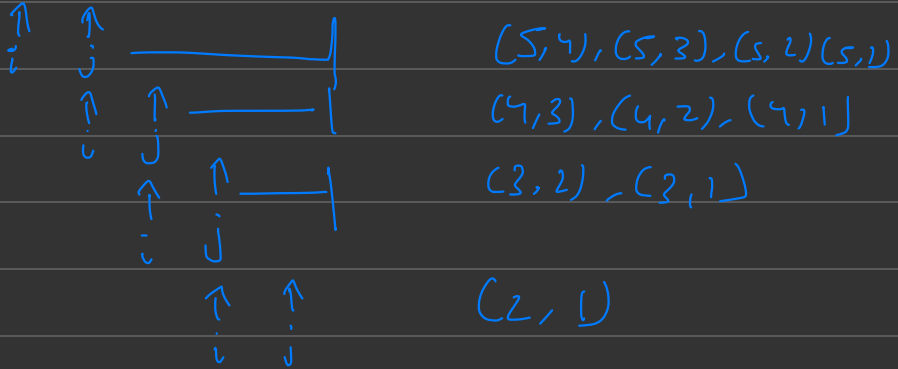


Brute force

An element $a[i]$ can form pair with further elements if $a[i] > a[j]$

[5, 4, 3, 2, 1]



inversion_ct = 10 //

T.C $\rightarrow O(n^2)$, SC $\rightarrow O(1)$

Optimised approach (using Merge sort)

- In merge sort, while merging two halves, one thing is sure that both the halves are sorted

eg: $[2, 3, 5] + [1, 4]$

The diagram illustrates the merging process. The first array is $[2, 3, 5]$ and the second is $[1, 4]$. The first array is divided into two halves: $[2, 3]$ and $[5]$. The second array is $[1, 4]$. The indices i and j are marked at the start of each array. A bracket labeled $(mid-i)$ indicates the range of elements in the first array that are greater than the current element in the second array.

(if $a[i] > a[j]$, $a[j]$ forms inversion pairs with all the elements on the right of i to end of first arr i.e. mid .)

$$\therefore \text{comb_inv} += (mid - i)$$

In the above example, when being copied to temp array inv pairs $\rightarrow (1, 2), (1, 3), (1, 5), (4, 5)$

$$T.C \rightarrow O(n \log n) \quad S.C \rightarrow O(n)$$

[5, 3, 2, 1, 4]

[5, 3, 2]

[1, 4]

[5, 3]

[2]

[1]

[4]

[5]

[3]

$a[i] \ a[j]$
 $5 > 3$

[3, 5]

count = (1-0)
= 1

[2, 3, 5]

$a[i] \ a[j]$
 $2 > 3$

count += (2-0)

count = 3

[1, 2, 3, 4, 5]

[1, 4]

$a[i] \ a[j]$
 $1 < 4$

no change in count

$a[i] \ a[j]$

$1 > 2$

count += (3-0)

count = 6

$4 > 5$

count += (3-2)

count = 7

```
int mergeSort (int arr[], int temp[], int l, int r)
    int mid, inv_count = 0;
```

```
while (l < r) {
    mid = (l+r)/2;
```

```
    inv_count += mergeSort(arr, temp, l, mid);
```

```
    inv_count += mergeSort(arr, temp, mid+1, r);
```

```
    inv_count += merge(arr, temp, l, mid+1, r);
```

```
}
```

```
return inv_count;
```

```
}
```

```
int merge (int arr[], int temp[],  
           int left, int mid, int right)
```

```
{ int i, j, k;
```

```
  int inv_count = 0;
```

```
  i = left ; j = mid ; k = left;
```

```
  // arr 1 → l to mid-1    // arr 2 → mid to r
```

```
  while ((i <= mid-1) && (j <= right)) {
```

```
    if (arr[i] <= arr[j]) { // No swapping
```

```
      temp[k++] = arr[i++]; // copy to temp & i++
```

```
    } else { // swapping & inv_count count
```

```
      temp[k++] = arr[j++]; // copy to temp & j++
```

```
      inv_count += (mid - i); // inv_count count
```

```
    }
```

```
  }
```

```
  // If elems in arr[l] to arr[mid-1] are left, copy them
```

```
  while (i <= mid-1) temp[k++] = arr[i++]
```

```
  // If elems in arr[mid] to arr[r] are left, copy them
```

```
  while (j <= right) temp[k++] = arr[j++]
```

```
  for (i = left, i <= right; i++) arr[i] = temp[i]
```

```
  return inv_count;
```

```
}
```

↑
copy temp arr elems
to arr.