Inversion Count

Tuesday, September 12, 2023 9:21 AM

Algorithm of Inversion Count

We can count inversions in an array by using the merge sort algorithm with minor changes highlighted in green. We r keep the count of total number of inversions. Idea is to divide the array into two equal halves and count the inversion right side of the mid and then call a function that will compute the overall inversions in the array.

```
Merge_Sort(Arr[], left, right )
{
    Inv_Count = 0
    If(left < right)                          ==> $O(1)$
    {
        M = (left + right)/2                  ==> $O(1)$
        Inv_Count += Merge_Sort(Arr, left, M)       ==> $T(N/2)$     recursive call with input size N/2
        Inv_Count += Merge_Sort(Arr, M+1, right)    ==> $T(N/2)$
        Inv_Count += Merge (Arr, left, M, right)    ==> $O(N)$
    }
    Return Inv_Count
}
```

Main Steps of Merge 1: calculate Size, **2:** Create temp Arrays, **3:** Copy data in temp Arrays, **4:** comparison of temp Ar copying pending elements 6: delete temp arrays

Merge (Arr[], left, M, right)

```
{
    Inv_Count = 0
    n1 = M - left +1     //size of sub array1
    n2 = right - M     //size of sub array2
    A[n1]                // temp Arrays definitely on heap
    B[n2]
      copy (Arr, A, left, M)           // By coping the data in temporary arrays, the original input space will get free and
update the data
                                         inside original array
      copy (Arr, B, M+1 right)
    i = j = 1              // iterators for temp Arrays
    k = left             //  iterator for original array
    while (i < n1 AND j < n2)
    {
        If(A[i] <= B[j])
            Arr[k++] = A[i++]
        Else
            Arr[k++] = B[j++]
            Inv_Count += (n1 - i + 1)
    }
    //Either array A or B will get exhausted so we need to copy the pending elements in sorted order in the original a
    while (i < n1)
        Arr[k++] = A[i++]
    while (j < n2)
        Arr[k++] = B[j++]

    Delete A and B
    Return Inv_Count
    Since n1+ n2 = N so the overall complexity of this function will be O(N)
}
```

Open in OneNote App
OneNote works better if you switch to the app.
Try it now

Inversion count is a measure of how far an array is from being sorted12. It is the number of pairs of elements in the array that are in the wrong order2. If the array is already sorted, the inversion count is 0. If the array is sorted in reverse order, the inversion count is the maximum12. Formally, an inversion occurs when a [i] > a [j] and i < j2.

Open in OneNote App
OneNote works better if you switch to the app.
Try it now