DATA ANALYTICS

Mohamed AbdelHalem_1180182

Assignment_3

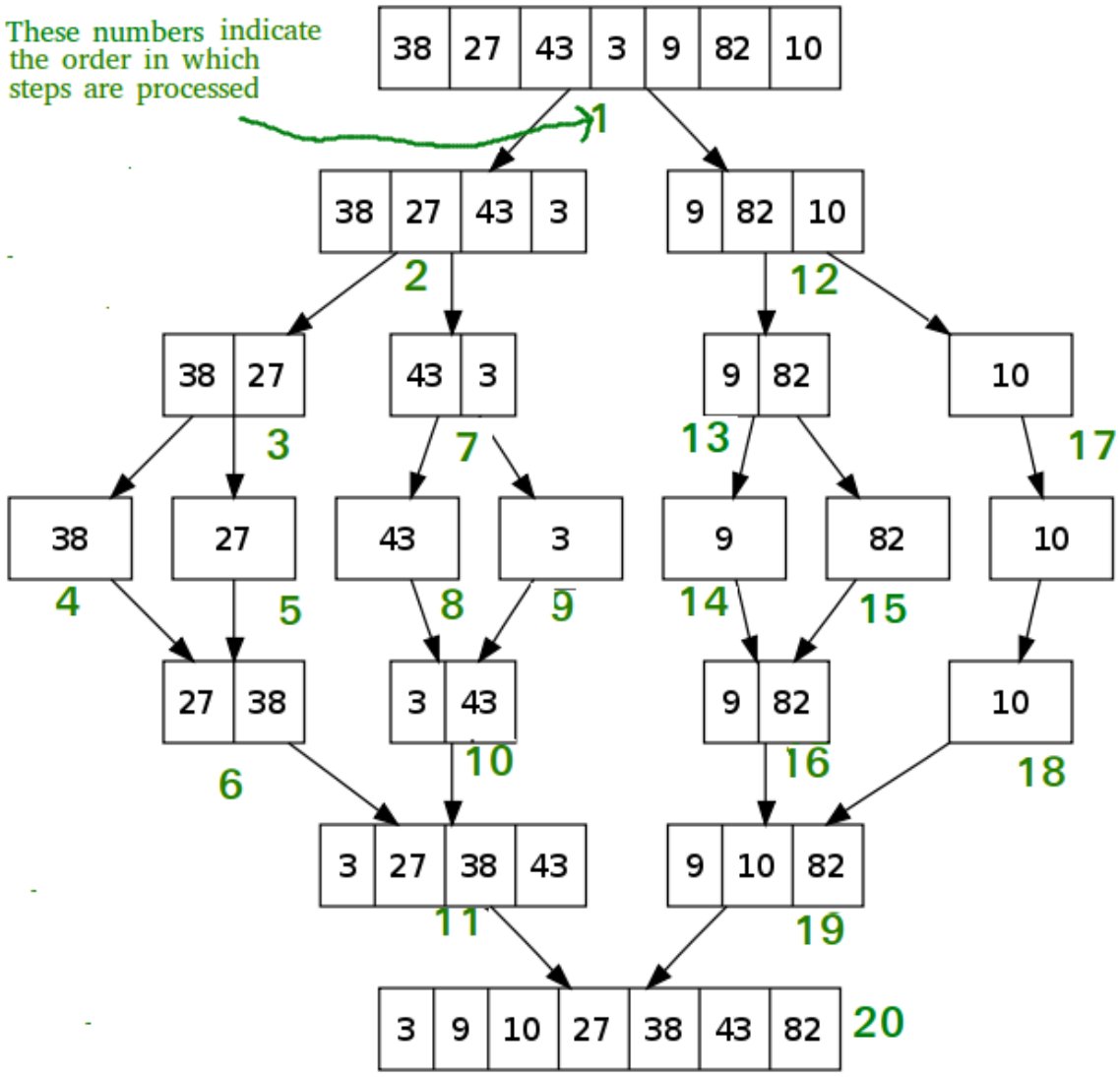
Divide and Conquer for Number of Inversions:

The way I solved this problem:

On understanding Divide and conquer I divided the array into sub arrays left and right and calculated number of inversions for each subarray and I used merge sort to merge each two sub arrays into a sorted subarray and in the merging sort function we first put the conditions in order not to exceed our sub arrays limit and compare to check if we have inversion or not and if we do we do add one to mid-I which is used because left and right sub arrays are sorted so we need from mid - I and after merging the two sub arrays we return the number of inversions we got through the process and recursion repeats this process until we have all number of inversions.

So in summary it's about using merging sort and while using it if some element to be exchanged with another we count number of inversions through mid-I as left and right are sorted and we keep the sorted array to continue the process in recursion and count the other inversions and add them up together and then return them.

Merge Sort Picture Explanation



So Merge sort is about dividing array into left and right one and dividing until we compare 2 elements only with each other and sort them in a sub array and keep doing that then use the sorted sub arrays to get to the two halves you made first and sort them together to get the final array sorted

Pseudo Code:

A is the input array and b is the temp array and left is first index and right is last one and ave is mid point

GetNinversions(a,b,left,right)

Number of inversions=0

If right – left<=1:

Return number of inversions

Ave=left+right//2

N_Inversions+=getNinversions(a,b,left,ave)

N_inversions+=getNinversions(a,b,ave,right)

N_inversions+=Merge(a,b,left,ave,right)

Merge(a,b,left,ave,right):

Merge(a,b,left,mid,right)

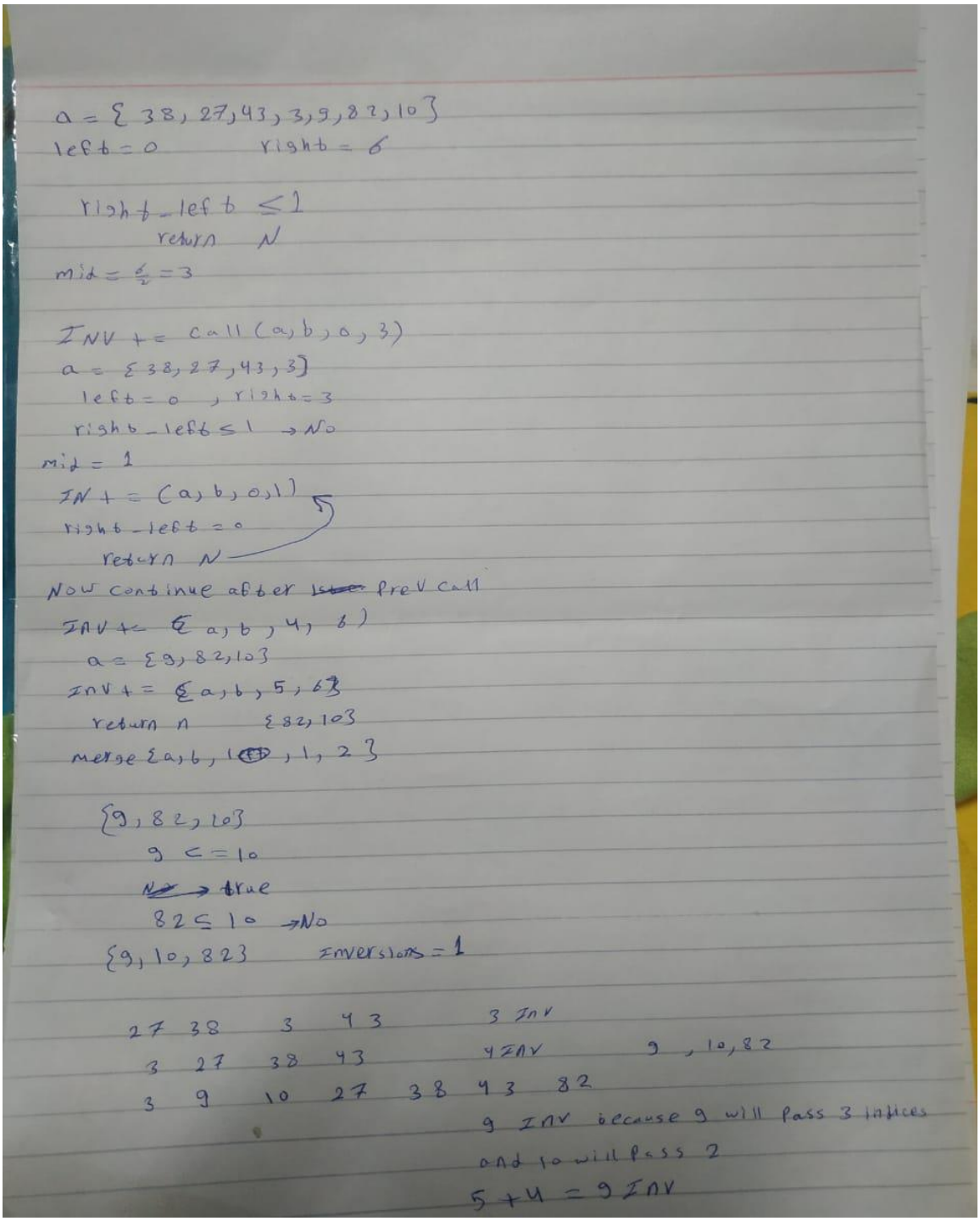
```
i = left
    j = mid + 1
    k = left
  inv_count = 0


    while left <= mid and j <= right:
if a[i] <= a[j]:
    b[k] = a[i]
    k += 1
    i += 1
 else:
  temp_arr[k] = arr[j]
  inv_count += (mid-i + 1)
  k += 1
  j += 1
```

Put any remaining elements in the left in the temp left subarray and for the right in the temp right sub array

And in the end put the sorted on in the original subarray

And return INV_Count

Algorithm Tracing (simple tracing of recursion):



`Complexity: Nlog(N)`

Sources used in this problem: https://www.geeksforgeeks.org/counting-inversions/

Image source: https://www.geeksforgeeks.org/merge-sort/