1. **Abstract**

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

1. **Problem description**,

The file “IntegerArray.txt” included in this project folder contains all the

100,000 integers between 1 and 100,000 (inclusive) in some order, with no

integer repeated. The task is to compute the number of inversions in the file

given, where the i-th row of the file indicates the i-th entry of an array. Because

of the large size of this array, we should implement a divide-and-conquer

algorithm.

1. **Algorithm (pseudocode)**,

Function conquer(array, A, B)

i = 0

j = 0

k = 0

counter = 0

While i < length of A and j < length of B

If A[i] <= B[j]

array[k] = A[i]

i++

k++

End if

Else

array[k] = B[j]

j++

k++

counter += (length of A) - i

End else

End while

Return counter

End function

Function divide(array)

Inversion = 0

If length of array > 0

mid = (length of array ) / 2

A = sub array of array from left to mid

B = sub array of array from mid + 1 to right

Inversion = Inversion + divide(A

Inversion = Inversion + divide(B)

Inversion = Inversion + conquer(array, A, B)

End if

Return Inversion

End function

Function main()

Read array of integer from file

Call divide function

Print number of inversion

Write array of integer to file

End function

1. **Implementation details**,

**import** sys  
**import** time  
  
*# increase the maximum recursion depth*sys.setrecursionlimit(10000)  
  
  
**def** conquer(array: [], tmp: [], left: int, mid: int, right: int) -> int:  
 *"""  
 conquer step, collect all result* **:param** *array: array of integer* **:param** *tmp: temporary array* **:param** *left: left index* **:param** *mid: mid index* **:param** *right: right index* **:return***: number of insertion  
 """* count = 0  
 ll = left  
 mm = mid  
 tt = left  
  
 *# until reaching one of 2 ending* **while** ll < mid **and** mm <= right:  
 *# a[i < a[j], no rul is broken* **if** array[ll] <= array[mm]:  
 tmp[tt] = array[ll]  
 tt += 1  
 ll += 1  
 **pass  
 else**:  
 *# a[i] > a[j] , j > i* tmp[tt] = array[mm]  
 tt += 1  
 mm += 1  
 *# sum up the* count += mid - ll  
 **pass  
 pass** *# copy the rest to tmp array* **while** ll < mid:  
 tmp[tt] = arr[ll]  
 tt += 1  
 ll += 1  
 **pass** *# copy the rest to tmp array* **while** mm <= right:  
 tmp[tt] = arr[mm]  
 tt += 1  
 mm += 1  
 **pass** *# copy tmp arry to the array* ll = left  
 **while** ll <= right:  
 array[ll] = tmp[ll]  
 ll += 1  
 **pass  
  
 return** count  
 **pass  
  
  
def** divide(array: [], tmp: [], left: int, right: int) -> int:  
 *"""  
 divide array and collect insertion number* **:param** *array: the array* **:param** *tmp: the temporary array* **:param** *left: left index* **:param** *right: right index* **:return***:  
 """* count = 0  
 **if** left < right:  
 *# get mid index* mid = int((left + right) / 2)  
  
 *# work on left sub array* count += divide(array, tmp, left, mid)  
 *# work on right sub array* count += divide(array, tmp, mid + 1, right)  
  
 *# collect the rest of insertion while merging 2 sub arrays* count += conquer(array, tmp, left, mid + 1, right)  
 **pass  
 return** count  
 **pass  
  
  
def** inversion\_count(array: []) -> int:  
 *"""  
 count number of inversion* **:param** *array: the array* **:return***: number of inversion  
 """  
 # temporary array* tmp = [**None**] \* len(array)  
  
 *# call the function* **return** divide(array, tmp, 0, len(array) - 1)  
 **pass  
  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 *# read file line by line  
 # then convert each line to an integer* arr = [int(line.rstrip(**'\n'**)) **for** line **in** open(**'IntegerArray.txt'**)]  
 *#arr = [2, 4, 1, 3, 5]  
 # print result* start\_time = time.time()  
 print(**'inversion count:'**, inversion\_count(arr))  
 print(**"Running time: %s seconds"** % (time.time() - start\_time))  
  
 *# sorted array is saved in output.txt* **with** open(**'output.txt'**, **'w'**) **as** output:  
 **for** item **in** arr:  
 output.write(**"%d\n"** % item)  
 **pass  
  
 pass**

1. **Running results and analysis**,
   1. Result

Running result

* 1. Analysis
     1. The running time is O(n logn)
     2. The naive algorithm uses O(n2), divide and conqueror algorithm O(n logn) with the help of merge sort algorithm
     3. The worst case scenario takes 25000000000 inversion, n(n - 1) / 2

1. **Conclusions on what you have learned in the project**,
   1. I’ve learned a lot of thing in the project,
   2. I know how to work on recursive call better
   3. I know how to thing like the computer does
2. **References**
   1. https://medium.com/the-andela-way/count-inversions-5fe3288f11fb