**DAA LAB 4**

Code for counting inversions:

import numpy as np

x = np.random.randint(1,6,100)

print(x)

# Counting Inversions (Brute Force Algorithm):

def count\_inversions(x):

    count = 0

    for j in range(100):

        for i in range(j+1,100):

            if(x[j]>x[i]):

                count = count + 1

    return count

a = count\_inversions(x)

print(a)

# Counting Inversions (Divide And Conquer Algorithm):

def merge\_and\_count(arr, new\_arr, left, mid, right):

    i = left

    j = mid + 1

    k = left

    inv\_count = 0

    while i <= mid and j <= right:

        if arr[i] <= arr[j]:

            new\_arr[k] = arr[i]

            i += 1

        else:

            new\_arr[k] = arr[j]

            inv\_count += (mid-i + 1)

            j += 1

        k += 1

    while i <= mid:

        new\_arr[k] = arr[i]

        i += 1

        k += 1

    while j <= right:

        new\_arr[k] = arr[j]

        j += 1

        k += 1

    for i in range(left, right + 1):

        arr[i] = new\_arr[i]

    return inv\_count

def merge\_sort\_and\_count(arr, temp\_arr, left, right):

    inv\_count = 0

    if left < right:

        mid = (left + right) // 2

        inv\_count += merge\_sort\_and\_count(arr, temp\_arr, left, mid)

        inv\_count += merge\_sort\_and\_count(arr, temp\_arr, mid + 1, right)

        inv\_count += merge\_and\_count(arr, temp\_arr, left, mid, right)

    return inv\_count

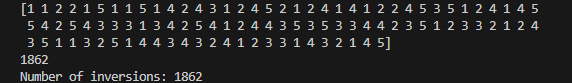
def count\_inversions(arr):

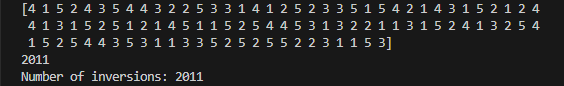
    temp\_arr = [0] \* len(arr)

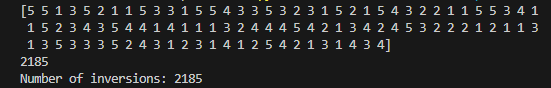
    return merge\_sort\_and\_count(arr, temp\_arr, 0, len(arr) - 1)

print(f"Number of inversions: {count\_inversions(x)}")

Screenshots:







Code for integer multiplication:

//Brute Force Algorithm:

def integer\_multiplication(a,b):

    c = str(a)[::-1]

    d = str(b)[::-1]

    n1 = len(c)

    n2 = len(d)

    if a<0:

        c = str(a)[::-1]

        n1  = n1 - 1

    if b<0:

        d = str(b)[::-1]

        n2 = n2 - 1

    l = []

    for i in range(min(n1,n2)):

        ans = 0

        for j in range(max(n1,n2)):

            ans = ans + int(c[i])\*int(d[j])\*(10\*\*j)\*(10\*\*i)

        l.append(ans)

    result = 0

    for k in range(len(l)):

        result += l[k]

    if (a<0) ^ (b<0):

        return -result

    return result

c = integer\_multiplication(a,b)

print(c)

//Divide and Conquer Algorithm:

def karatsuba\_multiplication(a,b):

    if a < 10 or b < 10:

        return a \* b

    p = max(len(str(a)), len(str(b)))

    q = p // 2

    a1 = a // 10\*\*q

    a0 = a % 10\*\*q

    b1 = b // 10\*\*q

    b0 = b % 10\*\*q

    c2 = karatsuba\_multiplication(a1, b1)

    c0 = karatsuba\_multiplication(a0, b0)

    c1 = karatsuba\_multiplication(a1 + a0, b1 + b0) - c2 - c0

    return (c2 \* 10\*\*(2\*q)) + (c1 \* 10\*\*q) + c0

result = karatsuba\_multiplication(a, b)

print(f"Product of {a} and {b} using Karatsuba multiplication is: {result}")

Screenshot:







