DAA LAB 4

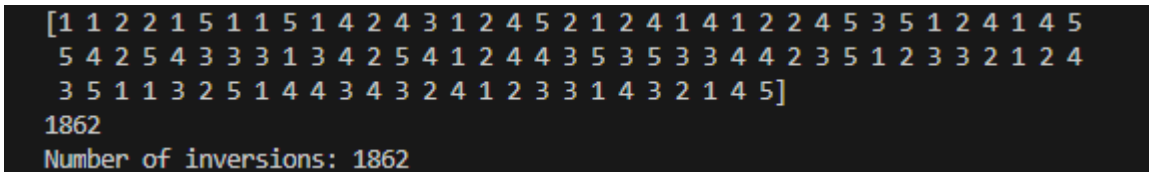
Code for counting inversions:

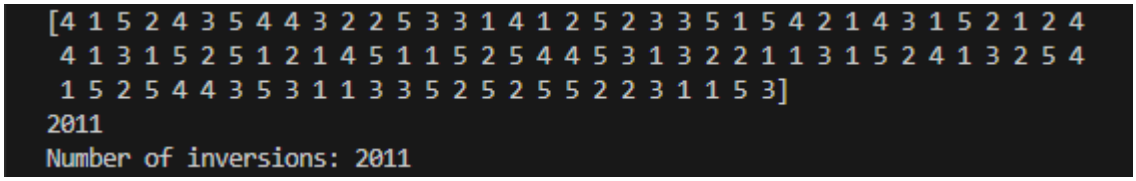
```
import numpy as np

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:
```

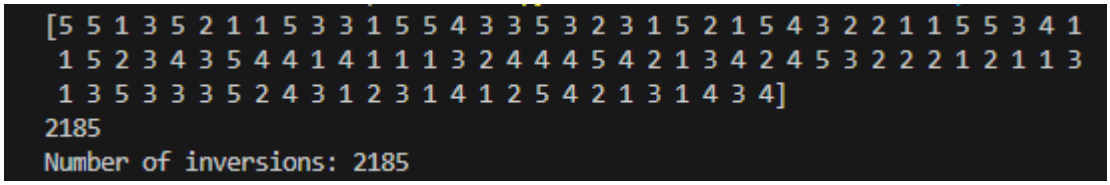
```
            new_arr[k] = arr[j]
            inv_count += (mid-i + 1)
            j += 1
        k += 1

    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: