Name: Ayush Bothra
Roll No: 231070011
Batch: A (1-20)

DAA LAB-04

DAA Lab - 04

Task - 1 :-

Algorithm :

```
count_inversions ( array) :
// A method that requires :
   input : n-sized array.
   and
   output : the number of inversion counts
            and the sorted array.

   if array.length <= 1 do :
       return 0, array
   middle ← Larray.length / 2 ⌋
   left_inv , left := count_inversions (array [ : mid ])
   right_inv, right := count_inversions (array [mid+1 : n])
   split_inv , merged := find_split_inversions ( left, right)
   total_inversions := lft_inv + split_inv + right_inv
   return total_inversions , array


find_split_inversions ( left, right)
 // A method that computes :
    input : left-half and right-half of n-sized
            array
    output : split inversions and sorted array.
```

P.T.O →

```
i, j := 0, 0
while i ≤ left.length AND j ≤ right.length do:
    if left[i] <= right[j] do:
        append left[i] to merged
        i := i+1
    else do:
        append right[j] to merged
        count := count + (left.length - i + 1)
        j := j + 1

while i ≤ left.length do:
    append left[i] to merged
    i := i + 1
while j ≤ right.length do:
    append right[j] to merged
    j := j + 1

return count, merged.
```

Time complexity of brute force :-

There are two loops running for n time, $T(n) = \sum_{i=0}^{n-1} \sum_{0}^{n-1} c$

$$= \sum_{0}^{n-1} c \cdot n$$

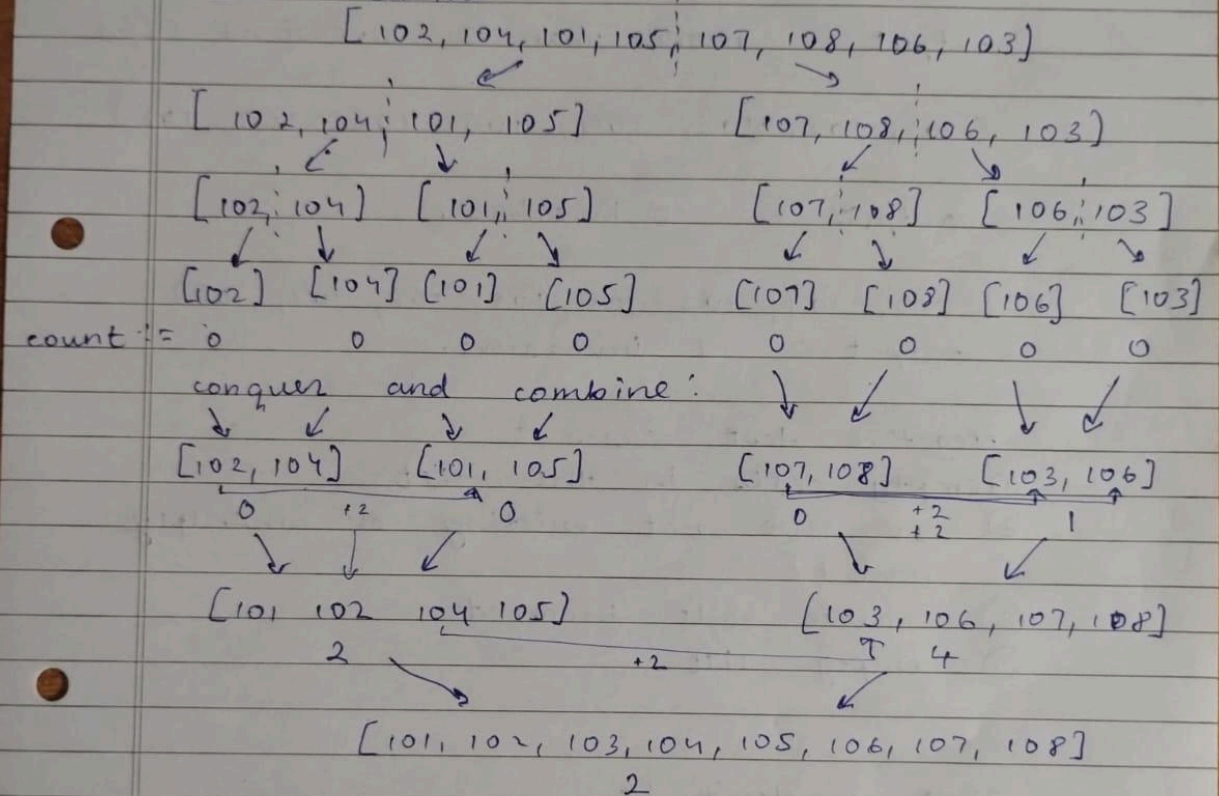
$$= c \cdot n^2$$

$\therefore$ $O(n^2)$ is the time complexity.

Test case :-

let array = [102, 104, 101, 105, 107, 108, 106, 103]

then :- divide :



When returned, all counts will be added.

∴ total - inversions = 2 + 2 + 4 + 1

= 9

further test cases :-

1. [101, 103, 106, 104, 105, 102, 107, 108]

   output : more than 3 inversions [6]

2. [101, 102, 103, 104, 105, 106, 107, 108]

   output : 0 inversions.

3. [102, 101, 103, 104, 105, 106, 107, 108]
   output : 1 inversion.

Negative test cases :-
① empty cells in the list
② csv file not existing or wrongly named
③ course code incorrect
④ empty list

Time complexity :-
The recurrence is $T(n) = 2\,T(\frac{n}{2}) + O(n)$

↑ dividing ↑ combining

conquering takes constant time.
∴ master method :- $T(n) = a\,T(\frac{n}{b}) + O(n^d)$
∴ $a = 2$, $b = 2$, $d = 1$
∵ $a = b^d$ $O(n^d \log n) = O(n \log n)$ is the time complexity.

Task - 2 :-

Algorithm 1 : Brute force :-

```
brute_force (num1, num2)
// takes two int numbers and finds
   their product using grade-school
    multiplication.
   str_num1  =  str(num1)
   str_num2  =  str(num2)
   result := [0] * (length of str_num1 + str_num2)
   for i in 0 to str_num1.length:
      for j = 0 to str_num2.length:
         product := int(str_num1[i]) *
                    int(str_num2[j])
         result[i+j] := product
         if result[i+j] >= 10:
            result[i+j+1] := [result[i+j] / 10]
            result[i+j] := result[i+j] % 10
   result := int(''.join(result))
   return result
```

Time complexity :

The inner loop runs for $\sum_{j=0}^{n-1} O(1)$ times.

and the outer loop runs for $\sum_{i=0}^{n-1} \sum_{j=0}^{n-1} \Theta(1)$

$\therefore$ on solving : $\sum_{i=0}^{n-1} n \Rightarrow cn^2 = T(n)$

Therefore the time complexity is $O(n^2)$

here, c is constant operation cost.

Test case :-

112 * 112 :-

Then, str1 = 211 ; str2 = 211
reversed here for easier understanding.
now according to the loops :-

result : [(1*1)] ~~(1*1), (1*2)~~
∴ [(1*1), (1*1), (1*2)]
result : [(1), (1+1), (2+1), (2*1)]
result : [(1), (2), (3+2), (2+2), (2*2)]
result : [1, 2, 5, 4, 4]

then final result => 12544

negative test case would be floating point numbers.
This algorithm cannot deal with those.

algorithm 2 : Karatsuba :-

```
div_and_conq ( num1, num2)
// input : two integers
// output : their product

  if num1 < 10 or num2 < 10 :
      return num1 * num2

  m := max ( L log10(num1)+1 J, L log10(num2)+1 J) // 2
  high1, low1 = divmod ( num1, 10**m)
  high2, low2 = divmod ( num2, 10**m)

  z0 = div_and_conq ( low1, low2)
  z1 = div_and_conq ( low1 + high1, low2 + high2)
  z2 = div_and_conq ( high1, high2)

  result = ( z2 * (10**(2*m)) +
           ((z1 - z2 - z0) * (10**m)) +
            z0

  return result
```

Time complexity :-

here, we are calling the function 3 times after dividing each number into 2.

$\therefore \quad T(n) = 3\,T\left(\frac{n}{2}\right) + O(n)$

↑ combining time.

$\therefore$ by master theorem,

$T(n) = a\,T\left(\frac{n}{b}\right) + O(n^d)$

$\therefore \quad a = 3, \quad b = 2, \quad d = 1$

$\therefore \quad a > b^d \rightarrow T.C \Rightarrow (n^{\log_b a})$

$\therefore \quad T.C \Rightarrow O(n^{\log_2 3})$

$= O(n^{1.585})$

Test case :-

15 84

↓

1 5 8 4

n1 l1 n2 l2

then $z_0 = 5 \times 4 = 20$

$z_1 = 6 \times 12 = 72$

$z_2 = 1 \times 8 = 8$

$\therefore$ ~~result~~ $\Rightarrow$ ~~$8 \times 10^2 + (81 - 20 - 8) \times 10 + 20$~~

$= 400 +$

$8 \times 100 + (72 - 20 - 8) \times 10 + 20$

$= 1260$

This continues further for larger nos.