Counting Inversions using divide & conquer

Course choice arrays of students
Output: Inversion count

```
Count(L, R):
    i, j = 0, 0
    n1, n2 = len(L), len(R)
    count = 0
    ans = []

    while i != n1 or j != n2 do:
        if L[i] > R[j] then
            add R[j] to ans
            j++
            count = n1 - i + count
        else:
            add L[i] to ans
            i++

    if i == n1 then
        while j != n2 do
            add R[j] to ans
            j++
    if j == n2 then
        while i != n1 do
            add L[i] to ans
            i++

    return count
```

```
merge and count (arr):
    if len(arr) == 1:
        return 0

    mid = len(arr) // 2

    left = arr[:mid]
    right = arr[mid:]

    inversions = merge and count (left)
    inversions += merge and count (right)
    inversions += count (left, right)

    return inversions
```

Time complexity for D & C

Divide: It computes the middle of the subarray which takes constant time $O(1)$

Conquer: We recursively solve 2 subproblems, each of size $n/2$, which contributes to $2T(n/2)$

Combine: Here, the entire merge procedure takes $O(n)$ such that $C(n) = O(n)$

$$\therefore \quad T(n) = \begin{cases} \Theta(1), & \text{if } n=1 \\ 2T(n/2) + \Theta(n), & n>1 \end{cases}$$

$$T(n) = \begin{cases} c, & \text{if } n=1 \\ 2T(n/2) + cn, & \text{if } n>1 \end{cases}$$

Using master theorem

$T(n) = aT(n/b) + O(n^d)$

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

$T(n) = O(n^d \log n)$

$\therefore \boxed{T(n) = O(n \log n)}$

Counting Inversion using Brute-Force technique

```
CountInv (A[0,1,....,n])
// count the total no. of inversion in given array
// Input : Array A of n distinct integers
// Output : The number of inversions of A
numInv : 0

for i=1 to n-1 do
    for j=i+1 to n do
        if (A[j] > A[i])
            numInv ++j

return numInv;
```

Time Complexity of Brute-Force

The Summation eq^n is

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=i+1} 1$$

$$= \sum_{i=1}^{n-1} (n - i + 1 - 1)$$

$$= \sum_{i=1}^{n-1} (n-i)$$

$$n \sum_{i=1}^{n-1} 1 - \frac{n(n-1)}{2}$$

$$= \frac{3n^2}{2} - \frac{3}{2}n$$

∴ Time Complexity is $O(n^2)$

Testcases:

Positive

1) codes = [85, 44, 20, 65, 63]
o/p = 6

2) codes = [12, 65, 34, 423, 78]
o/p = 2

3) codes = [9, 8, 7, 6, 5, 4]
o/p = 10

4) codes = [123, 643, 237, 43, 23]
o/p = 8

5) codes = [34, 75, 34, 87, 12]
o/p = 5

Negative:

1) codes = [av, 3, 4, 5, 6]
o/p = error

2) codes = [-1, 3, 53, 6, 4]
o/p = 3

3) codes = [34, a, b, 5, 3]
o/p = error

4) codes = [-1, -2, -3, 4-, 5]
o/p = error

5) codes = [1, 2, 3, 4, 5, 6]
o/p = error

Multiplication of 2 integers (Brute Forced)

Input: 2 integers
Output: Multiplicative product

```
Mul(a,b):
    x ← min(a,b)
    y ← max(a,b)
    ans = 0
    i = 0
    for each bit of y do
        partial ← bit * x
        partial ← partial × 10^i
        i = i+1
        ans = ans + partial
    return ans
```

Time Complexity:

Each integer is 'n' bit long

∴ the loop runs n times and in the loop the line where the partial products are added also cost O(n) time as there are n additions to be performed

∴ Time complexity is $O(n^2)$

Integer Multiplication (divide and conquer)
Input: 2 integers
Output: Multiplication ans of 2 integers

Mul(a,b):

if a<10 or b<10:
return a×b

n_1 = no. of digit in a
n_2 = no of digit in b

$n = \max(n_1, n_2)//2$
$x_1 = a // 10^n$
$x_0 = a \% 10^n$
$y_1 = b // 10^n$
$y_0 = b \% 10^n$

$p = mul(x_1, y_1)$
$q = mul(x_0, y_0)$
$r = mul(x_1+x_0, y_1+y_0) - p - q$

return $p \times 10^{2n} + r \times 10^n + q$

Time Complexity:
Karatsuba Algorithm

$T(n) = 3T(n/2) + n$ → addition of p, q, r

↓

3 subproblems of length n/2

$T(n) = aT(n/b) + f(n)$
$a=3 \quad b=2 \quad f(n) = n^1 = n^d \quad \therefore d=1$

By Master theorem.

Time Complexity = $\boxed{O(n^{\log_2 3})}$

Testcases

A) Positive:

1) a = 12
b = 32
o/p: 384

2) a = 123455
b = 234567
o/p = 2147483647

3) a = 634532
b = 53423
o/p = 2147483647

4) a = 0
b = 12356457389
o/p = 0

5) a = 235321235
b = 5646546434
o/p = 2147483647

6) a = 3453453
b = 345345
o/p = 2147483647

Negative:

1) a = 1.1
error invalid data type

2) a = -1
b = 23
o/p : -23

3) a = 0.9
error invalid data type

4) a = 0
b = ad
error

5) a = -1
b = -36543634
o/p: 36543634

18)