LAB-4

Integer multiplication by Divide-and-Conquer

```
Karatsuba (x, y)
// To compute product of 2 nos.
// Input: Two non-negative int x & y
// Output: Product of x & y
if x < 10 or y < 10
    return x * y
else
    n = max length of (x, y)
    a, b = first and second half of x
    c, d = first and second half of y
    compute p = a+b and q = c+d
    recursively compute ac = a.c
    bd = b.d and pq = p.q
    compute ad+bc = pq - ac - bd
    compute 10^n . ac + 10^(n/2) . (ad+bc) + bd
```

TC of Karatsuba Algorithm

This algorithm recursively computes 3 products each of size half of input, adds in $O(n)$ for combining the localised optimal solutions. So recurrence relation will be

$$T(n) = 3T(n/2) + O(n)$$

Applying master theorem,

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

$$3 > b^d \quad 3 > 2$$

$$\therefore T(n) = O(n^{\log_2 3})$$

$$= O(n^{1.585})$$

The Brute-force to calcute the product takes $O(n^2)$ time as each digit is multiplied with all other digits.

∴ Karatsuba multiplication is efficient approach as compared to brute-force technique.

Testcases (Integer multiplication)

Positive

1) $x = 12$ $y = 34$
$xy = 408$

2) $x = 123$ $y = 456$
$xy = 56088$

3) $x = 789$ $y = 123$
$xy = 97047$

4) $x = 12345$ $y = 67890$
$xy = 838102050$

5) $x = 123456789$ $y = 9876543210$
$xy = 1219326311126352690$

Negative

1) $x = -12$ $y = 34$
$xy = -408$

2) $x = 123$ $y = -456$
$xy = -56088$

3) $x = -789$ $y = -123$
$xy = 97047$

4) $x = -12345$ $y = 67890$
$xy = -838102050$

5) $x = -987654321$ $y = -123456789$
$xy = 1219326311126352690$

Counting Inversions Using Divide and Conquer

* This f° is for recursive division step

```
Sort CountInv (A[n])
// Recursively count inversions in left & right subarrays
// Input : array A of n distinct integers.
// Output : Sorted array B and no. of inversions of A.
if n == 0 || n == 1
        return (A, 0).
else
   (C, leftInv) = Sort/CountInv (first half of A)
   (D, rightInv) = Sort/CountInv (second half of A)
   (B, splitInv) = Merge and Sort CountInv (C, D)

   return (B, leftInv + rightInv + splitInv)
```

```
MergeSort CountInv
// To return the sorted array B and counting Inversions
// Input : Sorted array C and D
// Output : Sorted array B and no. of counting Inversions.
// ASSUMPTION : n is even.
i = 1    j = 1 ,  splitInv = 0
for k = 1 to n do
      if C[i] < D[j],
            B[k] = C[i] ; i++
      else
            B[k] = D[j] ; j++
            splitInv += (n/2 - i + 1)
   return (B, splitInv)
```

Time Complexity for Divide and Conquer

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 $C(n) = O(n)$

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

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

Using master theorem

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

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

$$a = b^d$$

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

$$\therefore T(n) = O(n \log n)$$

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 ++;
return numInv;
```

Time Complexity of Brute-Force

The Summation eqn is

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

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

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

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

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

Testcases (counting Inversion)

Positive

1) codes = [84205, 55881, 26415, 66611, 64035]
o/p = Student 1: Brute-force inversions = 6,
Optimised inversions = 6

2) codes = [21648, 86929, 82205, 65035, 80421]
o/p = Student 2: Brute-force inversions = 5,
Optimised inversions = 5

3) codes = [99915, 52246, 49670, 17129, 73238]
o/p: Student 5: Brute-force inversions = 7
optimised inversions = 7

4) codes = [52327, 47445, 12180, 91815, 20328]
o/p: Student 7: Brute force inversions = 6
optimised inversions = 6

5) codes = [75649, 85312, 72434, 11379, 91516]
o/p: student 8: Brute-force inversions = 5
Optimised inversions = 5

Negative

1) codes = [26122, 69915, 24881, 61651, -96661]
o/p: Invalid course code "-96661" found.

2) codes = [F#)3h, 80374, 66321, 27208, 63188]
o/p: Student4: Non-numeric invalid course code "F#)3h" found.

3) codes = [-21203, riq%1, \co-i, 94596, nd]U%J]
o/p: student 6: Non-numeric invalid code "-21205" found.

4) codes = [-57974, 76465, 73778, 95316, C^Wx]
o/p: student 9: Non-numeric invalid code "-57974" found.

5) codes = [97569, 24732, 65650, }([j+, 79904]
o/p: student 10: Non-numeric invalid code "}([j+" found.