Algorithm Analysis

CSI 5301

# Description: Big Multiplication

# Data Structure: vector<string>

# Algorithm:

1) Store number1 as a vector with single digits as its string elements

2) Store number2 as a vector with single digits as its string elements

3) Apply divide-and-conquer algorithm for integer multiplication from figure 2.1 of textbook. Below is the Algorithm copied from the textbook.

function multiply(x, y)

Input: Positive integers x and y, in binary

Output: Their product

n = max(size of x, size of y)

if n = 1: return xy

xL , xR = leftmost dn/2e, rightmost bn/2c bits of x

yL , yR = leftmost dn/2e, rightmost bn/2c bits of y

P1 = multiply(xL , yL )

P2 = multiply(xR , yR )

P3 = multiply(xL + xR , yL + yR )

return P1 × 2n + (P3 − P1 − P2 ) × 2n/2 + P2

|  |  |
| --- | --- |
| Input N | Integers as string |
| Basic Operation | Basic arithmetic operations  Vector manipulation |
| Recurrance relation | T(n) = 3T(n/2) +O(n) for n>1 |

**Summation or Recurrence Relation:**

1. Say, T(n) is the total time taken by the program,
2. We recursively call the method to calculate P1, P2 and P3 with half the input size each time
3. The other operations are basic loops and custom implementation of vector addition, subtraction and multiplication which we can say are in the order of O(n)

So, from points 1, 2 & 3 above we can write the recurrence relation for this algorithm as:

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

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

If we apply the Master Theorem to this recurrence relation,

T(n) = aT(n/b) + O(nd) ; a = 3, b=2 and d=1

In this case logba = log23so, logba > d and Big-O in this case is O(nloga)

Therefore,

**T(n) = O(nlog3)**