Universidade Federal de Ouro Preto
Departamento de Computação – DECOM
BCC241 – Projeto e Análise de Algoritmo
Prof. Anderson Almeida Ferreira
Exercícios – Divisão e Conquista

Turma: 11
Matrícula: 20.1.4003

1. multiply(1001, 0110)
   n = max(size of 1001, 0110) = 4
   $n \neq 1$
   $X_L = 10$, $X_R = 01$, $Y_L = 01$, $Y_R = 10$
   P1 = multiply(10, 01)
    n = max(size of 10, 01) = 2
    $n \neq 1$
    $X_L = 1$, $X_R = 0$, $Y_L = 0$, $Y_R = 1$
    P1 = multiply(1, 0)
     n = max(size of 1, 0) = 1
     n = 1
     return 1 * 0 = 0
    P2 = multiply(0, 1)
     n = max(size of 0, 1) = 1
     n = 1
     return 0 * 1 = 0
    P3 = multiply(1, 1)
     n = max(size of 1, 1) = 1
     n = 1
     return 1 * 1 = 1
    return $2^n * 0 + 2^{n/2} + (1 - 0 - 0) + 0 = 000 + 10 + 0 =$ **10**
   P2 = multiply(10, 01)
    n = max(size of 01, 10) = 2
    $n \neq 1$
    $X_L = 0$, $X_R = 1$, $Y_L = 1$, $Y_R = 0$
    P1 = multiply(0, 1)
     n = max(size of 0, 1) = 1
     n = 1
     return 0 * 1 = 0
    P2 = multiply(1, 0)
     n = max(size of 1, 0) = 1
     n = 1
     return 1 * 0 = 0
    P3 = multiply(1, 1)
     n = max(size of 1, 1) = 1
     n = 1
     return 1 * 1 = 1
    return $2^n * 0 + 2^{n/2} + (1 - 0 - 0) + 0 = 000 + 10 + 0 =$ **10**
   P3 = multiply(11, 11)
    n = max(size of 11, 11) = 2
    $n \neq 1$
    $X_L = 1$, $X_R = 1$, $Y_L = 1$, $Y_R = 1$
    P1 = multiply(1, 1)
     n = max(size of 1, 1) = 1
     n = 1
     return 1 * 1 = 1
    P2 = multiply(1, 1)
     n = max(size of 1, 1) = 1

n = 1
return 1 * 1 = 1
P3 = multiply(10, 10)
n = max (size of 10, 10) = 2
$n \neq 1$
$X_L = 1$, $X_R = 0$, $Y_L = 1$, $Y_R = 0$
P1 = multiply(1, 1)
n = max(size of 1, 1) = 1
n = 1
return 1 * 1 = 1
P2 = multiply(0, 0)
n = max(size of 0, 0) = 1
n = 1
return 0 * 0 = 0
P3 = multiply(1, 1)
n = max(size of 1, 1) = 1
n = 1
return 1 * 1 = 1
return $2^n * 1 + 2^{n/2} + (1 – 1) + 0$ = 100 + 00 + 0 = **100**
return $2^n * 1 + 2^{n/2} + (100 – 1) + 0$ = 100 + 100 + 0 = **1001**
return $2^n * 1 + 2^{n/2} + (1001 – 10 - 10) + 10$ = 100000 + 10100 + 10 = **110110**

2. a. T(n) = 5T(n/2) + O(n)
a = 5, b = 2, d = 1
$\log_2 5 > 1$ -> T(n) = **O(n^log₂5)**

b. T(n) = 2T(n-1) + O(1)
T(n) = 2(2T(n-2) + O(1)) + O(1) = 4T(n-2) + 3O(1)
T(n) = 2(T(n-3) + 3O(1)) + O(1) = 8T(n-3) + 7O(1)
.
.
.
$T(n) = 2^k T(n-k) + \sum_{i=0}^{k-1} 2^i O(1)$, n = k
$T(n) = O(2^n) + O(n) = \mathbf{O(2^n)}$

c. $T(n) = 9T(n/3) + O(n^2)$
a = 9, b = 3, d = 2
$\log_3 9 = 2$ -> $T(n) = \mathbf{O(n^2 \log n)}$

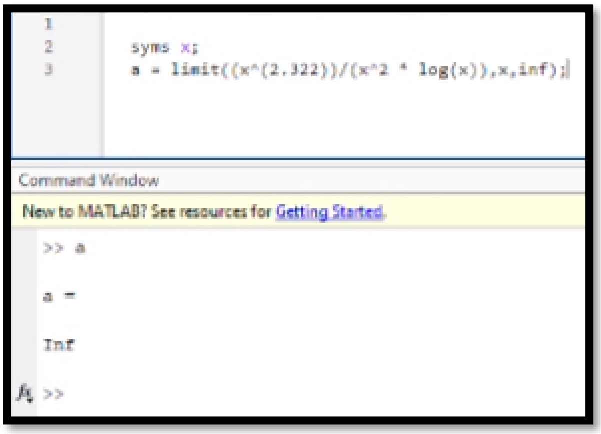

Figura 1 - Limite de A/C

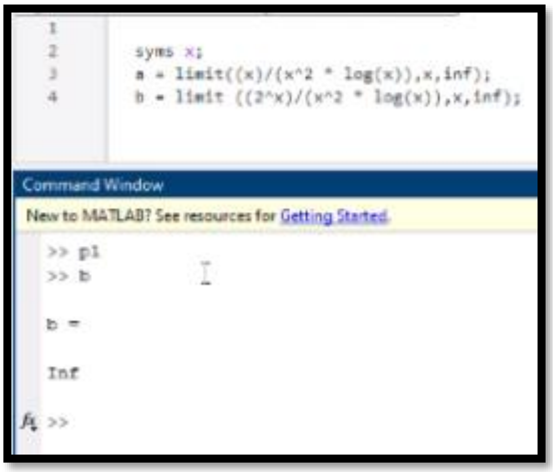

Figura 2 - Limite de B/C

Como $n^2\log n$ é dominado assintoticamente por 2^n e $n^{\log_2 5}$, eu escolheria o algoritmo C.