Complexidade de Algoritmos, 2020/2
Paulo Roberto Albuquerque

1) Resolva a relação de recorrência:

$T(n) = 3T(n/2) + n$
$T(1) = 1 = n - 3^L T(n/2^L)$

Respostas:
$T(n) = 3T(n/2) + n$
$3T(n/2) = 3^2 T(n/2^2) + 3n/2$
$3^2 T(n/2^2) = 3^3 T(n/2^3) + 3^2 n/2^2$
$3^3 T(n/2^3) = 3^4 T(n/2^4) + 3^3 n/2^3$
$\vdots$
$3^{L-1} T(n/2^{L-1}) = 3^L T(n/2^L) + 3^{L-1} n/2^{L-1}$

$n/2^L = 1 \Leftrightarrow n = 2^L \Leftrightarrow L = \log_2 n$

$T(n) = n + 3n/2 + 3^2 n/2^2 + 3^3 n/2^3 + \ldots + 3^L T(1)$

$T(n) = 3^L T(1) + \sum_{i=0}^{l-1} 3^i (n/2^i)$
$T(n) = 3^L T(1) + \sum_{i=0}^{l-1} (3^i n/2^i)$
$T(n) = 3^L T(1) + n \sum_{i=0}^{l-1} (3/2)^i$
$T(n) = 3^L . T(1) + n((1 - (3/2)^l) / (1 - (3/2)))$
$T(n) = 3^L . T(1) + n((1 - (3/2)^l) / (-1/2))$
$T(n) = 3^L . T(1) + n.(-2 + 2.(3/2)^L)$
$T(n) = 3^L - 2n + 2n.(3/2)^L$
$T(n) = 3^{\log 2(n)} - 2n + 2n.(3/2)^{\log 2(n)}$
$T(n) = n^{\log 2(3)} - 2n + 2n.n^{\log 2(3/2)}$
$T(n) = n^{\log 2(3)} - 2n + 2n.(n^{\log 2(3) - 1})$
$T(n) = n^{\log 2(3)} - 2n + 2n.((n^{\log 2(3)})/n)$
$T(n) = n^{\log 2(3)} - 2n + 2n^{\log 2(3)}$

$T(n) = 3n^{\log 2(3)} - 2n \Rightarrow O(n^{\log 2(3)}) = \mathbf{O(n^{1.59})}$

2) Construa a árvore de chamadas para o algoritmo abaixo, destaque os valores de P1, P2, P3 e valor de retorno em cada chamada tendo como entrada os valores de 4 bits 15 e 10 (o retorno pode ter o dobro do número de bits da entrada).

<u>function multiply</u>(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

x_L, $x_R =$ leftmost $\lceil n/2 \rceil$, rightmost $\lfloor n/2 \rfloor$ bits of x
y_L, $y_R =$ leftmost $\lceil n/2 \rceil$, rightmost $\lfloor n/2 \rfloor$ bits of y

$P_1 = \mathtt{multiply}(x_L, y_L)$
$P_2 = \mathtt{multiply}(x_R, y_R)$
$P_3 = \mathtt{multiply}(x_L + x_R, y_L + y_R)$
return $P_1 \times 2^n + (P_3 - P_1 - P_2) \times 2^{n/2} + P_2$

Respostas: O losango simboliza o retorno devido ao caso base, e o retorno abaixo da chamada é o retorno recursivo calculado pela fórmula.