0.1 a. f = Θ(g)

d. f = Θ(g)

e. f = Θ(g)

f. f = Ω(g)

m. f = Ω(g)

n. f = O(g)

o. f = Ω(g)

q. f = Ω(g)

0.3 a. F0 = 0, F1 = 1, Fn = Fn-1 + Fn-2

Prove Fn ≥ 2.5n when n ≥ 6

F0 = 0

Fk ≥ 2.5k => Fk-1 + Fk-2 ≥ 2.5k

F(k+1) ≥ 2.5(k+1)

F(k+1)-1 + F(k+1)-2 ≥ 2.5k+.5

Fk + Fk-1 ≥ 2.5k+.5

* 1. n = dh – 1

(

1.7 Since an n-bit number takes O(n2) time just as an m-bit number does, and since the initial formula takes 2 n-bit numbers , which factors in as 2n (which naturally falls out), this case takes (n+m) resulting in O((n+m)2) time.

Anders Maraviglia

2.1 x = 10011011, y = 10111010, n = 8

XL = 1001, xr = 1011, yL = 1011, yr = 1010

xy = 2nxLyL + 2n/2(xLyr + xryL) + xryr

xy = 28(1001\*1011) + 24(1001\*1010 + 1011\*1011) + 1011\*1010

xy = 28(9\*11) + 24(9\*10+11\*11) + 11\*10

xy = 28830

xy = 0111000010011110

2.4 Algorithm A

T(n) = 5 \* T( n/2 )

a = 5, b = 2, d = 1

d < logb a => 1 < log2 5

O( nlog(5)/log(2)) = O( n2.332)

Algorithm B

T(n) = 2 \* T( n - 1 )

T(1) = 0

T(2) = 2\*T(1)

T(3) = 2\*T(2) = 2\*2\*T(1)

T(4) = 2\*T(3) = 2\*2\*2\*T(1)

O(n2)

Algorithm C, and I would choose this one

T(n) = 9 \* T( n/3 )

a = 9, b = 3, d = 2

d = logba => 2 = log3 9

O(n log n)

2.5 a. Θ(n(log3)/(log2))

b. Θ(n)

d. Θ(n(log3)/(log9))

g. Θ(n)

j. Θ(2n)

2.12 The function prints Θ(n) lines.

2.13 a. B3 = 1, B5 = 2, B7 = 5. Even n values like 4 are left out because that would not make a full binary tree.

b. The recurrence relation for Bn is the summation of all possible values of the tree.

2.15 One must loop three times; once to swap all values less than v while keeping track of the index value, once for all values equal to v and swapping, and once for all values greater than v and swapping. If still confused, refer to lab two question three.   
2.23 a. Loop over the array and at each position run a binary search to find all elements that are equal, and if that count is greater than the current maximum found number of elements, replace it with the new found value.

b. My apologies, but I cannot give a linear time algorithm at this time.