CS200 Fall 2015 written homework 2

name: Tanner Pflager
id: 830312414

1. Using the Master Theorem:

Let f be an increasing function that satisfies
$f(n) = a \cdot f(n/b) + c \cdot n^d$
whenever $n = b^k$, where k is a positive integer, $a \geq 1$, b is an integer > 1, and c and d are real numbers with c positive and d nonnegative. Then

$$f(n) = \begin{cases} O(n^d) & \text{if } a < b^d \\ O(n^d \log n) & \text{if } a = b^d \\ O(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

What are the big-O bounds recurrence relations? (Simplify logs and exponents.)

a) f(n) = 4 f(n/2) + n $f(n) = O(n^{\log_2 4}) = O(n^2)$

b) f(n) = 2 f(n/4) + n f(n) = O(n)

c) f(n) = 4 f(n/4) + n f(n) = O(n log n)

d) f(n) = 2 f(n/2) + n f(n) = O(n log n)

e) f(n) = 2 f(n/2) + 1 f(n) = O(n)

f) f(n) = f(n/2) + 1 f(n) = O(log n)

2. Which of the above describes the complexity of

a) Binary Search f) f(n) = O(log n), describes the complexity of Binary Search
the function does not represent Binary Search, f(n/2)+2 does.

---

b) Merge Sort d) f(n) = O(n log n), describes the complexity of Merge Sort
c) also describes the complexity of Merge Sort, however the function itself does not.

3. Given the following method:

```
public int recMax (int[] A){
     return recMax(A,0,A.length-1);
}
private int recMax(int[]A, int lo, int hi){
     if(lo==hi) return A[lo];
     else{
          int mid = (lo+hi)/2;
          int m1 = recMax(A,lo,mid);
          int m2 = recMax(A,mid+1,hi);
          return Math.max(m1, m2);
     }
}
```

a) Derive a recurrence rM(n) relation for recMax(A, lo, hi), where n = hi-lo+1.

rM(n) = 1 for n = 1

rM(n) = 2rM(n/2) + n for n > 1

b) Use the Master Theorem to solve the recurrence and obtain the big O complexity of recMax.

rM(n) = O(n logn)

4. Find a solution to the following recurrence relation, using repeated substitution:

```
f(1) = 2000
f(n) = 1.1 f(n-1)   for n>1
```

f(1) = 2000
f(2) = 2200.0
f(3) = 2420.0
f(4) = 2662.0

f(4) = 1.1 * f(3)
= 1.1 * (1.1 * f(2))
= 1.1 * (1.1 * (1.1 * (f(1))))
= 1.1 * 1.1 * 1.1 * 2000
= 2662

guess: $f(n) = 1.1^{n-1} * 2000$

Informal checks:
f(n) = 1.1 * f(n-1)
$= [1.1^{((n-1)-1)} * 2000] * 1.1$
$= 1.1^{n-1} * 2000$

$f(1) = 1.1^{1-1} * 2000$
= 1 * 2000
= 2000

$\boxed{f(n) = 1.1^{n-1} * 2000}$