CS 332 Exam 1 - Review Problems

Fall, 2011

The exam will be over the following sections of the textbook.

1.1, 1.2
2.1, 2.2, 2.3
3.1
4.1, 4.5,
6.1 - 6.4
7.1 - 7.4
8.1
9.1, 9.2

In addition to the problems given below, you should also look at the "practice problems" from the textbook that were assigned on the course web site.

Problem 1: Let P be a problem. Suppose we know that for any algorithm that solves P, the worst-case time complexity of the algorithm is $O(n^2)$ and also $\Omega(n \lg(n))$. Let A be an algorithm that solves P. Which of the following statements about A is consistent with the information about the complexity of P?

- (a) A has worst-case time complexity of $O(n^2)$.
- (b) A has worst-case time complexity of $O(n^{1.5})$.
- (c) A has worst-case time complexity of $O(n)$.
- (d) A has worst-case time complexity of $\Theta(n^2)$.
- (e) A has worst-case time complexity of $\Theta(n^3)$.

Problem 2: For each of the following questions, briefly explain your answers.

- (a) If I prove that an algorithm takes $O(n^2)$ worst-case time, is it possible that it takes $O(n)$ on some input?
- (b) If I prove that an algorithm takes $O(n^2)$ worst-case time, is it possible that it takes $O(n)$ on all inputs?
- (c) If I prove that an algorithm takes $\Omega(n^2)$ worst-case time, is it possible that it takes $O(n)$ on some input?
- (d) If I prove that an algorithm takes $\Omega(n^2)$ worst-case time, is it possible that it takes $O(n)$ on all inputs?
- (e) If I prove that an algorithm takes $\Theta(n^2)$ worst-case time, is it possible that it takes $\Omega(n^3)$ on some input?

Problem 3: Prove or disprove.

- (a) $2^{(2+n)} \in O(2^n)$
- (b) $2^{(2n)} \in O(2^n)$

Problem 4: Show that $\ln n! \in O(n \ln n)$.

Problem 5: Which is a better worst case time, $O(n)$ or $O((\ln n)^2)$? Give a brief justification.

Problem 6: Below are six C functions that each compute the value of 2^n for a positive integer *n*. Analyze each function for how many addition and multiplication operation are needed as a function of the input *n* (can you find recurrence relations for $M(n)$, the number of multiplications, and $A(n)$, the number of additions?) .Which of the implementations of 2^n is the most efficient?

```
int two1 (int n)
{
   if (n == 0) {return 1;} else return 2*two1(n-1);}
}

int two2 (int n)
{
   if (n == 0) {return 1;} else {return two2(n-1)+two2(n-1);}
}

int two3 (int n)
{
   // Note call to two1.
   if (n == 0) {return 1;} else {return two3(n-1)+two1(n-1);}
}

int two4 (int n)
{
   if (n == 0) {return 1;}
   else if (n % 2 == 0) {int x=two4(n/2); return x*x;}
   else {return 2*two4(n-1);}
}

int two5 (int n)
{
   if (n == 0) {return 1;}
   else if (n%2 == 0) {two5(n/2) * two5(n/2)}
   else {return 2*two5(n-1);}
}

int two6 (int n)
{
   // Note call to two4.
   if (n == 0) {return 1;} else {return two6(n-1)+two4(n-1);}
}
```