CS 3343.002 (Fall 2016) Exam 1
Oct 4, 2016
2:30pm - 3:50pm (80 minutes)

Name: ______________________ ID: ____________________

• Don’t forget to put your name and ID on the cover page
• This exam is closed-book, but you can have a one-page cheat sheet.
• If you have a question, stay seated and raise your hand.
• Please try to write legibly – if I cannot read it, you may not get credit.
• Do not waste time – if you cannot solve a question immediately, skip it and return to it later.
• Try your best to answer each question. Partial credits will be given if you show that you have some ideas – but not according to the length of your answer.

<table>
<thead>
<tr>
<th></th>
<th>1) Comparing functions</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>2) Proof of asymptotic notations</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>3) Analyzing an iterative algorithm</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>4) Master theorem</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>5) Analyzing recursive algorithms</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>6) Recursion tree method</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>7) Substitution method</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8) K-way merge sort (Extra credit)</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>120</td>
<td></td>
</tr>
</tbody>
</table>
1. Comparing functions (13 points)
   (a) (8 points) Order the following functions according to their order of growth from the lowest to the highest. If you think that two functions are of the same order (i.e. \( f(n) \in \Theta(g(n)) \)), put them in the same group.
   
   \( 5n + \log n, n^{2.5}, n + n^2 \log n, 2^n, n^2 \log n, (n + 1)!, 2^{2n}, n^3 + 2n + 5 \).

   (b) (5 points) For each pair of functions in the table below, determine whether \( f(n) \in O(g(n)) \), \( f(n) \in \Omega(g(n)) \), \( f(n) \in \Theta(g(n)) \), or all of them. It is NOT necessary to justify your answer.

<table>
<thead>
<tr>
<th>( f(n) )</th>
<th>( O )</th>
<th>( \Omega )</th>
<th>( \Theta )</th>
<th>( g(n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n + 5) \log n</td>
<td></td>
<td></td>
<td></td>
<td>( n^2 + \log n )</td>
</tr>
<tr>
<td>( n^2 \log n )</td>
<td></td>
<td></td>
<td></td>
<td>( 5n^2 + n \log n )</td>
</tr>
<tr>
<td>( n \log n )</td>
<td></td>
<td></td>
<td></td>
<td>( n\sqrt{n} )</td>
</tr>
<tr>
<td>( n^2 + 10 \log n )</td>
<td></td>
<td></td>
<td></td>
<td>( n^2 - \sqrt{n} )</td>
</tr>
<tr>
<td>( \log_3(n^3) )</td>
<td></td>
<td></td>
<td></td>
<td>( 2 \log_2 n )</td>
</tr>
</tbody>
</table>

2. (12 points) Proof of asymptotic notations
   (a) (7 points) Use the basic definition of \( \Omega \), prove that \( 2n^2 - n + 1 \in \Omega(n^2) \)

   (b) (5 points) Given two asymptotically positive functions \( f(n) \) and \( g(n) \), prove that the following statement is correct: \( f(n) \in O(g(n)) \) implies that \( f(n) + g(n) \in \Theta(g(n)) \).
3. (12 points) Analysis of iterative algorithm. You are given the following sorting algorithm:

```c
//Input: An array A of n real numbers
1 ExchangeSort(A[1..n]) {
2     for (i = 1; i <= n - 1; i++) {
3         for (j = i+1; j <= n; j++) {
4             if (A[i] > A[j])
5                 exchange(A[i], A[j]);
6         }
7     }
8 }
```

a. In order to argue for the correctness of the algorithm, what can be claimed at the end of $i$-th iteration of the outer for loop? (Hint: what properties are satisfied by the first $i$ numbers in the array?)

b. How many times will line 4 be executed, as a function of $n$, in the best case and worst case, respectively?

c. How many times will line 5 be executed, as a function of $n$, in the best case and worst case, respectively?

d. What is the asymptotic running time of this algorithm?
4. (35 points) Assume that \( T(1) \in \Theta(1) \). Solve the following recurrence functions using the **master method**. Mark the correct answer. You do NOT need to show all details.

a. \( T(n) = 2T(n/2) + n^2; \)
   - (a) \( \Theta(n) \)
   - (b) \( \Theta(n \log n) \)
   - (c) \( \Theta(n^2) \)
   - (d) Master Method cannot be applied.
   - (e) None of the above.

b. \( T(n) = 2T(n/3) + n^{3/2}; \)
   - (a) \( \Theta(n^{\log_3 2}) \)
   - (b) \( \Theta(n^{3/2}) \)
   - (c) \( \Theta(n^{\log_2 3}) \)
   - (d) Master Method cannot be applied.
   - (e) None of the above.

c. \( T(n) = 8T(n/2) + n^3; \)
   - (a) \( \Theta(n^3 \log n) \)
   - (b) \( \Theta(n^4) \)
   - (c) \( \Theta(n^3) \)
   - (d) Master Method cannot be applied.
   - (e) None of the above.

d. \( T(n) = T(n/3) + \frac{n}{\log n}; \)
   - (a) \( \Theta(n/\log n) \)
   - (b) \( \Theta(n) \)
   - (c) \( \Theta(n \log n) \)
(d) Master Method cannot be applied.
(e) None of the above.

\[ T(n) = 2T(n/4) + \sqrt{n} \log n; \]

(a) \( \Theta(\sqrt{n}) \)
(b) \( \Theta(\sqrt{n} \log n) \)
(c) \( \Theta(\sqrt{n} \log^2 n) \)
(d) Master Method cannot be applied.
(e) None of the above.

\[ T(n) = \frac{n}{2} T(n/2) + n \log n; \]

(a) \( \Theta(n) \)
(b) \( \Theta(n \log n) \)
(c) \( \Theta(n \log^2 n) \)
(d) Master Method cannot be applied.
(e) None of the above.

\[ T(n) = 4T(n - 2) + n^2; \] Show details if necessary.

(a) \( \Theta(n^2 4^n) \)
(b) \( \Theta(n^4 2^n) \)
(c) \( \Theta(2^n) \)
(d) \( \Theta(4^n) \)
(e) None of the above.
5. (13 points) Analysis of recursive algorithm. Consider the pseudocode of the following two algorithms for computing the $n$-th power of three, where $n$ is a non-negative integer.

```
Alg1 (n)
    if (n == 0) return 1;
    return 3 * Alg1(n - 1);
end

Alg2 (n)
    if (n == 0) return 1;
    m = floor (n / 2);
    p = Alg2(m);
    p = p * p;
    if (n % 2 == 1) // n is an odd number
        return 3 * p;
    else // n is an even number;
        return p;
    end
end
```

a. (3 points) Using induction, prove that alg1 will correctly compute $3^n$.

b. (4 points) Let $A(n)$ and $B(n)$ be the running time of Alg1 and Alg2, respectively, as a function of $n$. Write down the recurrence for $A(n)$ and $B(n)$. 
c. (6 points) Solve $A(n)$ and $B(n)$ to obtain the asymptotic running time of Alg1 and Alg2, using any method you learned in class, and determine which algorithm is more efficient.

6. (10 points) Assume that $T(1) \in \Theta(1)$. Solve the following recurrence function using the **recursion tree method** to get an asymptotically tight bound.

$$T(n) = 3T(n/2) + n$$
7. (10 points) Assume that $T(1) \in \Theta(1)$ and $T(n) = 2T(n/3) + T(n/4) + n$. Prove that $T(n) \in O(n)$ using the substitution method.

8. (Extra credit - 15 points) K-way merge sort

In the classic merge sort algorithm, an input array of size $n$ is split into two subarrays, which are recursively sorted and then combined with a linear time merge function. Now consider a generalized idea that splits the input array into $k$ equal-sized subarrays, where $k \geq 2$. As in the classic merge sort, each subarray is sorted recursively, and the sorted subarrays are merged to produce the final sorted array. See pseudocode below.

```
KwayMergeSort(A[1..n]);
    if (n <= 1) return A;
    Split A into k approximately equal-sized subarrays A1, A2, ... Ak.
    B = KwayMergeSort(A1);
    for (i = 2; i <= k; i++)
        B = merge(B, KwayMergeSort(Ai));
    end
    return B;
end
```

a. Define the running time of the algorithm as a function of both $n$ and $k$. (Hint: pay closer attention to the total time needed to merge the sorted subarrays.)
b. Solve the recurrence using the **recursion tree** method. For the tree height, keep the base of the logarithm function explicit.

c. How does the running time change as \( k \) increases? In particular, in the most extreme case, when \( k = n \), what is the running time of the algorithm? Does it make intuitive sense? Explain.
Scratch paper
Scratch paper
Scratch paper