CS 3343 (Spring 2014) Exam 1
Feb 25, 2014
10:00am - 11:25am (85 minutes)

Name: ____________________ ID: ________________

- Don’t forget to put your name and ID on the cover page
- This exam is closed-book
- 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>Comparing functions</th>
<th></th>
<th>Proof by definition</th>
<th></th>
<th>Analyzing iterative algorithm</th>
<th></th>
<th>Master method</th>
<th></th>
<th>Recursion tree</th>
<th></th>
<th>Substitution method</th>
<th></th>
<th>Analyzing recursive algorithm</th>
<th></th>
<th>(Extra credit) K-way merge sort</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Proof by definition</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Analyzing iterative algorithm</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Master method</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Recursion tree</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Substitution method</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Analyzing recursive algorithm</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>(Extra credit) K-way merge sort</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>120</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1
1. Comparing functions
   (a) (10 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.
   \[ 5n^2 + n \log^2 n, \ n^{2.5}, \ 3n + n \log n, \ 3^n, \ 5n^2 \log n, \ \log_{10}(n!), \ 2^{2n}, \ 2n^2 - 3n + 5. \]

   (b) (10 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>1) ((n^2 + 5n) \log n)</td>
<td></td>
<td></td>
<td>( n^2 + \log n )</td>
<td></td>
</tr>
<tr>
<td>2) (n^2 \log n)</td>
<td></td>
<td></td>
<td>( 5n^3 + n \log n )</td>
<td></td>
</tr>
<tr>
<td>3) (n \log n)</td>
<td></td>
<td></td>
<td>( n + \sqrt{n} )</td>
<td></td>
</tr>
<tr>
<td>4) (n^2 + 10 \log(n!))</td>
<td></td>
<td></td>
<td>( 5n^2 + \sqrt{n} )</td>
<td></td>
</tr>
<tr>
<td>5) (\log_2(n^2))</td>
<td></td>
<td></td>
<td>( n \log_2 n )</td>
<td></td>
</tr>
</tbody>
</table>

2. (10 points) Using the definition of O, prove that \( \sum_{i=1}^{n} i \in O(n^2) \).
3. (10 points) You are given the following iterative algorithm:

```c
Mystery(A[1..n])
//Input: An array A[1..n] of n real numbers
for (i = 1; i <= n-1; i++) {
    for (j = i+1; j <= n; j++) {
        if (A[i] == A[j])
            return false;
    }
}
return true;
```

a. What does this algorithm compute?

b. What is the worst-case time complexity of the algorithm, as a function of \( n \)? Give an input that can lead to the worst-case running time.

c. What is the best-case time complexity of the algorithm, as a function of \( n \)? Give an input that can lead to the best-case running time.
4. (25 points) Assume that $T(1) \in \Theta(1)$. Solve the following recurrence functions using the master method. If the master method cannot be applied, state the reason. Justify your answer.

a. $T(n) = 3T(n/3) + n$;

b. $T(n) = 3T(n/2) + n^2$;
c. $T(n) = 4 T(n/2) + (n + 3 \log n)$;

d. $T(n) = 4 T(n/2) + n^2 \log n$;

e. $T(n) = 2 T(n/2) + \log n$;
5. (10 points) Recursion tree method.

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

$$T(n) = 3T(n - 2) + 1$$
6. (10 points) Substitution method.

Assume that $T(1) \in \Theta(1)$ and $T(n) = 2T(n/2) + 3T(n/3) + n^2$. Prove $T(n) \in O(n^2)$ using the substitution method.
7. (15 points) Analysis of recursive algorithms.

Consider the pseudocode of the following two algorithms. In both algorithms, the input \( A \) is an array of size \( n \), which is then split into 3 subarrays in subsequent recursive calls.

```
AlgX (A[1..n])
    if (n < 3) return sum(A[1..n]);
    p = floor(n/3);
    // r is a random integer between
    // 1 and 3
    r = ceil(3 * rand());
    if (r == 1)
        return AlgX (A[1..p]);
    else if (r == 2)
        return AlgX (A[p+1..2p]);
    else
        return AlgX (A[2p+1..n]);
end

AlgY (A[1..n])
    if (n < 3) return sum(A[1..n]);
    p = floor(n/3);
    r = ceil(3 * rand());
    a = AlgY (A[1..p]);
    b = AlgY (A[p+1..2p]);
    c = AlgY (A[2p+1..n]);
    if (r == 1)
        return a;
    else if (r == 2)
        return b;
    else
        return c;
end
```

a. Write down the recurrence function for the running time of AlgX and solve it. If your analysis relies on any non-trivial assumption, state it.

b. Write down the recurrence function for the running time of AlgY and solve it. If your analysis relies on any non-trivial assumption, state it.

c. Which algorithm is more efficient?
8. (Extra credit - 20 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.
Extra page
Extra page
Extra page