7200. Analysis of Algorithms

Midterm Exam

October 23, 2019

Id: ............................................................................................................................

- You have 2 hours to answer all 4 questions.
- Answer a question only within the given space by using a readable normal size font text.
- You get 20% of the credit if you leave the answer blank. You get no credit for a wrong answer.
- Problems 3 and 4:
  - You will get partial credit for less efficient but correct solutions.
  - Words are preferable to codes!
  - Don't provide details for known algorithms.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Part</th>
<th>Maximum Points</th>
<th>Your Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>d</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>d</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

Total 100

Final Grade

Good Luck!
1. **TRUE (T) or FALSE (F)?** Justify your answers with 2-3 sentences. If the answer is FALSE then a counter-example is a perfect justification.

(a) $f = \Omega(g)$ implies $f = O(g)$.

(b) $f = O(g)$ implies $f = \Theta(g)$.

(c) $f = \Theta(g)$ implies $f = \Omega(g)$. 
(d) $f = O(g)$ implies $g = \Omega(f)$.

(e) $f = O(g)$ implies $g = O(f)$.

(f) $f = \Theta(g)$ implies $g = O(f)$. 
2. Define recurrence formulas for the following six exact solutions. Each formula should specify the initial value and the recurrence equation. Justify your answer with 2-3 sentences.

(a) \( T(n) = \log_2(n) \) for all non-negative powers of 2 (1, 2, 4, 8, ...).

(b) \( T(n) = n \) for all positive integers (1, 2, 3, ...).

(c) \( T(n) = n \log_2(n) \) for all non-negative powers of 2 (1, 2, 4, 8, ...).
(d) \( T(n) = n^2 \) for all positive integers \( 1, 2, 3, \ldots \).

(e) \( T(n) = 2^n \) for all positive integers \( 1, 2, 3, \ldots \).

(f) \( T(n) = n! \) for all positive integers \( 1, 2, 3, \ldots \).

(a) Assume that $A$ contains only 2 distinct integers.
   Design an algorithm that sorts the array with $O(n)$ comparisons between two array’s integers.
   You may not use hashing or radix-sort like sorting algorithms.
   Justify the correctness of the algorithm and explain why the complexity is $O(n)$. 

If you are not sure about your previous algorithm, describe a less efficient but correct algorithm that solves the problem.

What is the worst-case number of comparisons made by your algorithm (you may use the $O$-notation)? Justify the correctness of the algorithm and your complexity claim.
(b) Assume that \( n = 2^k - 1 \) for some positive integer \( k \) and that the array contains exactly \( k \) different integers. Furthermore, the largest number appears \( (n+1)/2 \) times, the second largest number appears \( (n+1)/4 \) times and so on until the smallest number which appears exactly once.

**Example:** Let \( n = 2^4 - 1 = 15 \). Then \( A \) contains exactly \( k = 4 \) distinct integers. The largest appears 8 times, the second largest appears 4 times, the third largest appears 2 times, and the smallest number appears once. A could be the following array \([34, 34, 21, 8, 13, 34, 34, 21, 21, 34, 34, 34, 34, 13, 34]\) in which 34 appears 8 times, 21 appears 4 times, 13 appears twice, and 8 appears once.

Design an algorithm that sorts the array with \( O(n) \) comparisons between two array’s integers. You may not use hashing or radix-sort like sorting algorithms.

Justify the correctness of the algorithm and explain why the complexity is \( O(n) \).
If you are not sure about your previous algorithm, describe a less efficient but correct algorithm that solves the problem. What is the worst-case number of comparisons made by your algorithm (you may use the $O$-notation)? Justify the correctness of the algorithm and your complexity claim.

**Remark:** For the following two tasks, the running time should take into account both comparisons between array integers and operations between integers.

(a) Design an efficient algorithm that determines if the sum of all the integers is negative or positive? What is the running time of your algorithm? Justify the correctness of the algorithm and your complexity claim.
If you are not sure about your previous algorithm, describe a less efficient but correct algorithm that solves the problem.
What is the running time of your algorithm?
Justify the correctness of the algorithm and your complexity claim.
(b) Design an efficient algorithm that determines if the product of the integers is negative or positive? What is the running time of your algorithm? Justify the correctness of the algorithm and your complexity claim.
If you are not sure about your previous algorithm, describe a less efficient but correct algorithm that solves the problem.
What is the running time of your algorithm?
Justify the correctness of the algorithm and your complexity claim.