7200. Analysis of Algorithms

Midterm Exam

October 14, 2016

• Answer all three questions.
• You have exactly two hours.
• Answer a question only within the given space by using a readable normal size font text.
• For Problems 2 and 3: you get 20% of the credit if you leave the answer blank; you get no credit for a wrong answer.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Maximum Points</th>
<th>Your Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Good Luck!
1. Let $n = 2^k$ be a power of 2 for $k \geq 0$ and let $T(n)$ be a recursive function defined as follows

$$ T(1) = 1; \quad T(n) = aT(n/2) + f(n) $$

for a constant $a$ that can be either 1, or 2, or 4 and a function $f(n)$ that can be either 0, or 1, or $n$. Match the following nine possible values for $T(n)$:

(a) $T(n) = 1$
(b) $T(n) = \log_2(n) + 1$
(c) $T(n) = n$
(d) $T(n) = 2n - 1$
(e) $T(n) = 2n - 1$
(f) $T(n) = n(\log_2(n) + 1)$
(g) $T(n) = n^2$
(h) $T(n) = \frac{4n^2 - 1}{3}$
(i) $T(n) = 2n^2 - n$

with each of the following combinations of $a$ and $f(n)$:

• $f(n) = 0$ and $a = 1$: $T(n) =$
• $f(n) = 0$ and $a = 2$: $T(n) =$
• $f(n) = 0$ and $a = 4$: $T(n) =$
• $f(n) = 1$ and $a = 1$: $T(n) =$
• $f(n) = 1$ and $a = 2$: $T(n) =$
• $f(n) = 1$ and $a = 4$: $T(n) =$
• $f(n) = n$ and $a = 1$: $T(n) =$
• $f(n) = n$ and $a = 2$: $T(n) =$
• $f(n) = n$ and $a = 4$: $T(n) =$

Note that you need to assign $2n - 1$ to two different recursive functions.

You earn two points for each correct match. You earn two extra points for having at least five correct answers. You lose one point for each wrong match.
2. **Definition:** An array $A$ of size $n$ is called a $k$-sorted array:

   if for any index $i$, if $j$ is the location of $A[i]$ when the array is sorted then $|i - j| \leq k$.

In other words,

$A[i]$ is at most $k$ locations away from its position in the sorted version of array $A$.

**Example:** Consider the array

$$A = [5, 3, 8, 1, 21, 2, 13, 34] \; .$$

The sorted version of $A$ is

$$A' = [1, 2, 3, 5, 8, 13, 21, 34] \; .$$

It follows that

- $A[1] = 5$ is 3 locations away from its position at $A'[4]$,
- $A[2] = 3$ is 1 location away from its position at $A'[3]$,
- $A[3] = 8$ is 2 locations away from its position at $A'[5]$,
- $A[4] = 1$ is 3 locations away from its position at $A'[1]$,
- $A[5] = 21$ is 2 locations away from its position at $A'[7]$,
- $A[6] = 2$ is 4 locations away from its position at $A'[2]$,
- $A[7] = 13$ is 1 location away from its position at $A'[6]$,
- $A[8] = 34$ is 0 locations away from its position at $A'[8]$.

As a result, $A$ is a 4-sorted but not a 3-sorted.

Note that a sorted array is a 0-sorted array and that any array is an $(n - 1)$-sorted array but maybe a $k$-sorted array for $k < n - 1$. 
Let $A$ be an arbitrary array.

Using comparisons, describe an efficient algorithm to make it an $n/2$-sorted array.

What is the worst-case number of comparisons made by your algorithm? Use the $O$-notation.

Justify the correctness and complexity of your algorithm.
If you are not sure about your previous algorithm, describe any (possibly not efficient) correct algorithm that solves the problem.
What is the worst-case number of comparisons made by your algorithm? Use the $O$-notation.
Justify the correctness and complexity of your algorithm.
3. **Definition:** An array $A$ with $n$ distinct integers is called a hill array if there exists an index $m$ such that the array is in an increasing order from index 1 to index $m$ and in a decreasing order from index $m$ to index $n$:


**Example:**

$$A = [2, 4, 6, 8, 7, 5, 3, 1]$$

is a hill array for which $m = 4$.

**Example:**

$$A = [13, 34, 21, 8, 5, 3, 2, 1]$$

is a hill array for which $m = 2$.

Note that a sorted array is a hill array for which $m = n$ and that a reversed sorted array is a hill array for which $m = 1$. 
Let $A$ be a hill array. Describe an efficient algorithm to find the index $m$.
What is the worst-case number of comparisons made by your algorithm? Use the $O$-notation. Justify the correctness and complexity of your algorithm.
If you are not sure about your previous algorithm, describe any (possibly not efficient) correct algorithm that solves the problem.

What is the worst-case number of comparisons made by your algorithm? Use the $O$-notation.

Justify the correctness and complexity of your algorithm.