CSE 3101Z Design and Analysis of Algorithms
Professor: James Elder
Winter 2009
Assignment 2
Due 11:59pm Monday April 13

First Person:  
Family Name: 
Given Name: 
Student #: 
Email:  

Second Person:  
Family Name: 
Given Name: 
Student #: 
Email: 

Guidelines:

• You may complete this assignment alone or in groups of two. Do not get solutions from other pairs. Though you are to teach & learn from your partner, you are responsible to do and learn the work yourself. Write it up together.

• Please make your answers clear and succinct.

• Relevant Readings:
  – CLRS Ch. 4, 6, 7, 31.1, 21.2
  – Edmonds Ch. 8-10

• This page should be the cover of your assignment.

<table>
<thead>
<tr>
<th>1 (18 marks)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (20 marks)</td>
<td></td>
</tr>
<tr>
<td>3 (20 marks)</td>
<td></td>
</tr>
<tr>
<td>4 (42 marks)</td>
<td></td>
</tr>
<tr>
<td>Total (100 marks)</td>
<td></td>
</tr>
</tbody>
</table>
1. **Recurrences** (18 marks) Provide tight bounds on $T(n)$ for each of the following recurrences. Assume that $T(n)$ is constant for sufficiently small $n$. You may use the Master Theorem, if you can show that it applies. Otherwise, use the alternative techniques we discussed in class to prove the bounds.

   (a) (4 marks) $T(n) = 7T(n/3) + n^2$
   (b) (4 marks) $T(n) = 4T(n/2) + n^2 \sqrt{n}$
   (c) (4 marks) $T(n) = T(n-2) + 2 \log n$
   (d) (6 marks) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

2. **Building a heap using insertion**, adapted from CLRS 6-1 (20 marks)

   The algorithm BUILD-MAX-HEAP described in CLRS builds a heap with the aid of the routine MAX-HEAPIFY. An alternate design (BUILD-MAX-HEAP'), uses MAX-HEAP-INSERT (see below).

   (a) (10 marks) Do the procedures BUILD-MAX-HEAP and BUILD-MAX-HEAP' always create the same heap when run on the same input array? Prove that they do, or provide a counterexample.

   (b) (5 marks) What is the best-case running time for the two methods? Justify your answer.

   (c) (5 marks) What is the worst-case running time for the two methods? Justify your answer.

   **BUILD-MAX-HEAP(A)**
   
   heap-size[A] = length[A]
   
   for $i = \lfloor \text{length}[A]/2 \rfloor$ downto 1
   
   MAX-HEAPIFY(A, i)

   **BUILD-MAX-HEAP'(A)**
   
   heap-size[A] = 1
   
   for $i = 2$ to length[A]
   
   MAX-HEAP-INSERT(A, A[i])

   **MAX-HEAPIFY(A, i)**
   
   $l$=LEFT($i$)
   
   $r$=RIGHT($i$)
   
   if $l \leq \text{heap-size}[A]$ and $A[l] > A[i]$
   
   largest = $l$
   
   else
   
   largest = $i$
   
   if $r \leq \text{heap-size}[A]$ and $A[r] > A[\text{largest}]$
   
   largest = $r$
   
   if largest $\neq i$
   
   exchange $A[i] \leftrightarrow A[\text{largest}]$
   
   MAX-HEAPIFY(A, largest)
MAX-HEAP-INSERT($A$, $key$)

$heap-size[A] = heap-size[A] + 1$

$A[heap-size[A]] = -\infty$

HEAP-INCREASE-KEY($A$, $heap-size[A]$, $key$)

HEAP-INCREASE-KEY($A$, $i$, $key$)

if $key < A[i]$

error “new key is smaller than current key”

$A[i] = key$

while $i > 1$ and $A[PARENT(i)] < a[I]$

exchange $A[i] \leftrightarrow A[Parent(i)]$

$i = PARENT(i)$

3. Binary search trees (15 marks)

You are to design a recursive algorithm that will return values in a binary search tree that bracket a specified key. Specifically, given a binary search tree $A$ containing $n$ elements, a key $k$ and a constant integer $b$, your algorithm will return two sets $S_-$ and $S_+$. $S_-$ is to contain the largest $b$ elements of $A$ that are smaller than or equal to $k$. If there are fewer than $b$ elements smaller than or equal to $k$, $S_-$ should contain all of them. $S_+$ is defined similarly, but will contain the smallest $b$ elements that are strictly larger than $k$.

(a) (10 marks) Provide pseudocode for your algorithm. Clearly specify the precondition and post-condition in your pseudocode. Clearly indicate the base case and how it is handled. How many ‘friends’ are you using? Are they being given a smaller input?

(b) (5 marks) Specify and solve the recurrence relation for the running time $T(n)$ of your algorithm.

4. Signal Detection (42 marks)

A communications channel carries a 1 GHz digital (binary) time signal $A(t)$, i.e., $A(t) \in \{0,1\}$, $t \in [1,2,...,n]$ nanoseconds. An edge is defined to be a transition of the signal from one state to another. The edge may be of positive polarity, i.e., a transition from 0 to 1, or of negative polarity, i.e., a transition from 1 to 0. The delay of the edge is defined as the midpoint of the transition. For example, if $A[7] = 0$ and $A[8] = 1$ then the delay of the edge is $t = 7.5$ nsec.

Given a time signal $A[1...n]$, you are asked to write an efficient algorithm for time-stamping an edge in $A$, i.e., reporting its delay. If there is more than one edge in $A$, the delay of any one may be reported. If no edges are present in $A$, this fact should be reported.

(a) (5 marks) Show that the problem is $\Omega(n)$, i.e., that all correct algorithms are $\Omega(n)$. Hint: Imagine this as a game in which some adversary gives you a sub-linear algorithm and you must find an input array $A$ for which this algorithm yields the wrong answer.
(b) (5 marks) Now suppose that front-end processing guarantees that the first and last value of the signal are different, i.e., \( A[1] \neq A[n] \). Design a recursive sublinear time algorithm \( \text{findedge} \) which reports the delay of an edge. Clearly state the pre- and post-conditions of your algorithm and verify that they are met on each recursive call. Also verify that the size of the input declines monotonically as a function of the depth of recursion. Use pseudocode to describe your algorithm.

c) (2 marks) Write a recurrence equation for your algorithm.

d) Determine tight asymptotic bounds for your algorithm using
   i. (5 marks) The method of substitution
   ii. (5 marks) The recursion tree method
   iii. (5 marks) The master method

e) (2 marks) If \( A[1] = 0 \) and \( A[n] = 1 \), there is at least one positive edge. Does your algorithm ever report the delay of a negative edge for this input?

f) (5 marks) Suppose that front-end processing guarantees that \( A[1] = 0 \) and \( A[n] = 1 \) and you must report the delay of a negative polarity edge, if it exists. What is the time complexity of this problem?

g) (5 marks) You are now asked to write an efficient algorithm for estimating the duration of any pulse in the binary time signal. For example, the signal \( \ldots 0000111000 \ldots \) contains a positive pulse of duration 3 nsec. The signal may contain zero or more pulses. Show that, in general, no sublinear algorithm exists for this problem. Does the problem become easier if the boundary conditions \( A[1] \) and \( A[n] \) are fixed?

h) (5 marks) Suppose that front-end processing guarantees that \( A[1] = A[n] \), that the signal contains at most one pulse, and that this pulse, if it exists, is at least \( n/a \) nsec in duration, where \( a > 1 \) is a constant. Design an algorithm that uses your \( \text{findedge} \) routine to solve the problem in sublinear time. What is the asymptotic running time of your algorithm?