CSE310 HW04, Thursday, 03/11/2010, Due: Thursday, 03/25/2010

Please note that you have to typeset your assignment using either LaTeX or Microsoft Word. Hand-written assignment will not be graded. You need to submit a hardcopy before the lecture on the due date. You also need to submit an electronic version at the digital drop box. For the electronic version, you should name your file using the format HW04-LastName-FirstName.

1. (10 pt) Give the **exact** number of element-wise comparisons needed to find both the smallest and the largest elements in an array of n un-sorted elements.
   Give the **exact** number of element-wise comparisons needed to find both the smallest and the second smallest elements in an array of n un-sorted elements. Note that you are not supposed to use asymptotic notations to answer these questions.

   Solution: Let $f(n)$ denote the number of comparisons needed to find both the smallest and the largest elements in an array of n elements. First consider the case where n is even. Let $n = 2k$. For $k = 1$, we only need one comparison. Therefore $f(2) = 1$. Suppose we have computed the largest and the smallest elements among the first $2(k-1)$ elements, using $f(2k-2)$ comparisons. For the next two elements, we need 3 comparisons. Therefore $f(2k) = f(2k-2) + 3$. This leads to $\mathbf{f(2k) = 3k - 2}$ for $k = 1, 2, \ldots$.

   Now consider the case where n is odd. Let $n = 2k + 1$. For $k = 0$, no comparison is needed, i.e., $f(1) = 0$. For $k = 1, 2, \ldots$, $f(2k+1) = f(2k) + 2$, because the last element needs to be compared with both the candidate for largest and the candidate for smallest. Therefore $\mathbf{f(2k+1) = 3k}$ for $k = 0, 1, 2, \ldots$. Combining both cases, we have $\mathbf{f(n) = \lceil \frac{3}{2} n \rceil - 2}$.

   Grading:
   4 pts: if the answer is not perfect, but is $\frac{3}{2}n$ plus a constant.
   1 pts: if the answer is $2n - 3$.

   In order to find both the smallest and the second smallest elements in an array, $n + \log n - 2$ comparisons are needed.

   First in $n/2$ comparisons, we have $n/2$ groups of 1^{st} and 2^0 candidate for 2^{nd} smallest in each group. In $n/4$ comparisons, we have $n/4$ groups of 1^{st} and 2^1 candidates for 2^{nd} smallest in each group. $\cdots$ In $n/2^k$ comparisons, we have $n/2^k$ groups of 1^{st} and 2^{k-1} candidates for 2^{nd} smallest in each group. In 1 comparison, we have 1 group of 1^{st} and $2^{\log n}$ candidates for 2^{nd}

smallest in the group. We need $\log n - 1$ comparisons to find the 2^{nd} smallest element in the $2^{\log n}$ candidates. Thus, the total number of comparisons is $n + \log n - 2$.

Grading:

4 pts: if the answer is not perfect, but is $\log n$ plus n plus a constant,

1 pts: if the answer is $2n - 3$.

2. (10 pts) In the linear time select-k algorithm, we used groups of 5. As we discussed in class, we could use groups of r, where $r = 3, 4, 5, 6, 7, \ldots$, although the worst-case running time may not necessarily linear. Let $T_r(n)$ denote the worst-case time complexity of the algorithm with groups of r. Derive the recurrence formula for $T_4(n)$ and $T_{11}(n)$. You have to prove the bounds on the number of elements in the recursive calls to justify your answer.

Solutions: We first concentrate on the derivation of $T_4(n)$. An important part of the derivation is to derive a lower bound of the *upper left portion* of the elements. There are at least $\lceil \frac{\lceil \frac{n}{4} \rceil}{2} \rceil - 2$ columns, each with at least 2 elements that are smaller than the median of medians. Therefore the *upper left portion* contains at least $\frac{n}{4} - 4$ elements that are smaller than the median of medians. This leads to the recurrence

$$T_4(n) \leq T_4(\lceil \frac{n}{4} \rceil) + T_4(\frac{3}{4}n + 4) + \Theta(n).$$

For group size 11, the *upper left portion* contains at least $\lceil \frac{\lceil \frac{n}{11} \rceil}{2} \rceil - 2$ columns, each with at least 6 elements that are smaller than the median of medians. Therefore we have

$$T_{11}(n) \leq T_{11}(\lceil \frac{n}{11} \rceil) + T_{11}(\frac{8}{11}n + 12) + \Theta(n).$$

Grading:

For each of the two cases, $5pts$ for perfect answer, $3pts$ for deriving the correct bound for the *upper left portion*. No penalty for off by an *additive* constant in the derivation.

3. (10 pts) Let T be the binary search tree illustrated in the rightmost tree in Figure 12.4(b) in the textbook. In the following questions, each question refers to this tree T, not the resulting tree after some operations.

(2 pts) Print out the results of pre-order traversal.
Solution: 15, 5, 3, 12, 10, 6, 7, 13, 20, 18, 23.

(2 pts) Which node is the successor of the node with value 7?
Solution: The node with value 10.

(2 pts) Show the tree after inserting 14 into tree T.
Solution: $15, 5, 3, 12, 10, 6, 7, 13, 14, 20, 18, 23$.

(2 pts) Show the tree after deleting 12 from tree T.
Solution: $15, 5, 3, 13, 10, 6, 7, 20, 18, 23$.

(2 pts) List the element-wise comparisons (in the correct order) for searching 9 in T.
Solution: 9 is compared with the following elements: $15, 5, 12, 10$

Grading: 0/2 for each of the five subquestions.

4. (20 pts) Fig. 1 illustrates a red-black tree, where a thick circle represents a black data-bearing node, a thin circle represents a red data-bearing node, and a thick rectangle represents a black nil node. In your work, you have to write **black** near a black node and write **red** near a red node to indicate the colors.

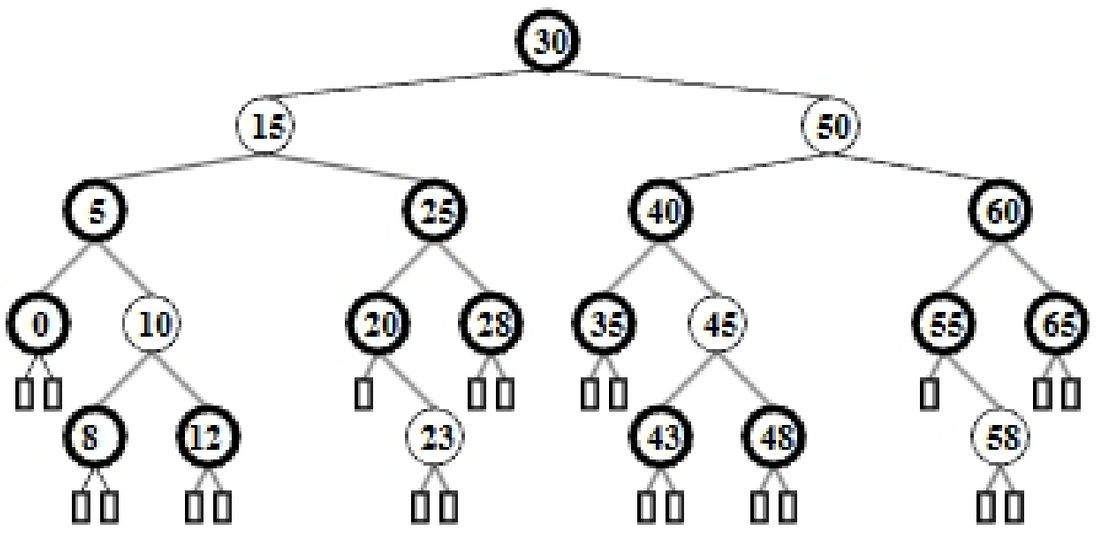

Figure 1: A red-black tree.

(4 pts) Draw the **resulting red-black tree after inserting 56** into the red-black tree in Fig. 1. You only have to draw the portion of the tree that is affected.
Solution: First, 56 (red) is a left child of 58, after the binary search tree insertion. This is case-2. We perform a rotation (rotating away) to get to case-3. Now 56 (red) is the right child of 55, and 58 (red) is the right child of 56. We perform a rotation again. Finally 56 (black) is the left child of 60, 55 (red) is the left child of 56, and 58 (red) is the right child of 56.

(4 pts) Draw the **resulting red-black tree after deleting 40** from the red-black tree in Fig. 1. You only have to draw the portion of the tree that is affected.
Solutions: Since node with value 40 has two children, we slice out its successor, the node with value 43 and copy the content of the sliced out node to the node originally with value 40. This is case-2, because 48 is black and has two black (nil) children. We