Data Structures and Programming

Spring 2014, Midterm Exam.

1. (10 pts) Order the following functions
2.2^n, $log(n^{10})$, 2^{2012}, $25n \times log(n)$, 1.1^n, $2n^{5.5}$, $4 \times log(n)$, 2^{10}, $n^{1.02}$, $5n^5$, $76n$, $8n^5 + 5n^2$
by their asymptotic growth rate, in non-decreasing order. Indicate by circling those functions that are "big-Theta" (i.e., $\Theta(..)$) of each other.
Solution:

$2^{2012} = 2^{10} < log(n^{10}) = 4 \times log(n) < 76n < 25n \times log(n) < n^{1.02} < 5n^5 = 8n^5 + 5n^2 < 2n^{5.5} < 1.1^n < 2.2^n$

2. (8 pts) Give the best asymptotic ("big-Oh", i.e., $O(...)$) characterization of the **best** and **worst** case time complexity of the algorithm $Count(A, B, n)$. Explain in detail how you computed the complexity. Note that $\leftarrow$ denotes the assignment statement.

```
Algorithm $Count(A, B, n)$
Input: Arrays $A$ and $B$ of size $n$.
       A, B store positive integers.
       $i \leftarrow 0$
       $sum \leftarrow 0$
       while $i < n$ do
             if $A[i] < n$ then
               for $j \leftarrow 0$ to $A[i]$ do
                   $sum \leftarrow sum + B[j]$
             $i \leftarrow i + 1$
return $sum$
```

Solution:
Best case: $O(n)$ when "if $A[i] < n$ ..." always false;
Worst case: $O(n^2)$ when when "if $A[i] < n$ ..." always true

3. (10 pts) Consider the use of the exclusive-or encoding method to represent doubly linked lists as discussed in class. Given a node X, let $Link(X)$ be the exclusive-or encoding kept in node X, i.e., $Link(X)$ is the outcome of taking the exclusive-or of the addresses of X's predecessor and successor along the list. Consider a doubly linked list consisting of the following: X_1 X_2 X_3 X_4 X_5 X_6. Suppose pointers P and Q point to nodes X_3 and X_4, respectively. Explain the updates needed to "delete" X_3 and X_4, making the new list X_1 X_2 X_5 X_6. Note that after the deletion, P and Q point to X_2 and X_5, respectively. Use additional temporary pointer(s), if needed. You may use the notation $\begin{pmatrix} x \\ y \\ ... \end{pmatrix} \leftarrow \begin{pmatrix} exp_1 \\ exp_2 \\ ... \end{pmatrix}$ defined in class to describe your algorithm.

Solution:

Before							
Address	Link(Link(P) $\oplus$ Q) $\oplus$ P	Link(P) $\oplus$ Q	P	Q	Link(Q) $\oplus$ P	Link(Link(Q) $\oplus$ P) $\oplus$ Q	
Node	X_1	X_2	X_3	X_4	X_5	X_6	

After					
Address		$P_{new} \leftarrow$ Link(P) $\oplus$ Q	$Q_{new} \leftarrow$ Link(Q) $\oplus$ P		
Node	X_1	X_2	X_5	X_6	
Link		(Link(Link(P) $\oplus$ Q) $\oplus$ P) $\oplus$ (Link(Q) $\oplus$ P)	(Link(Link(Q) $\oplus$ P)) $\oplus$ Q) $\oplus$ (Link(P) $\oplus$ Q)		

Hence, the operations are

$$\begin{pmatrix} \text{P} \\ \text{Q} \\ Link(Link(\text{P}) \oplus Q) \\ Link(Link(\text{Q}) \oplus P) \end{pmatrix} \leftarrow \begin{pmatrix} Link(\text{P}) \oplus Q \\ Link(\text{Q}) \oplus P \\ (Link(Link(\text{P}) \oplus Q) \oplus P) \oplus (\text{ Link}(Q) \oplus P) \\ (Link(Link(\text{Q}) \oplus P)) \oplus Q) \oplus (\text{ Link}(P) \oplus Q) \end{pmatrix}$$

4. (10 pts) Suppose that we have numbers between 1 and 1000 in a binary search tree and want to search for the number 363. For each of the following sequences, can it be the sequence of nodes examined? **Yes or No?** No explanations are needed.

   (a) 2, 252, 401, 398, 330, 344, 397, 363

   (b) 924, 220, 911, 244, 898, 258, 362, 363

   (c) 925, 202, 911, 240, 912, 245, 363

   (d) 2, 399, 387, 219, 266, 382, 381, 278, 363

   (e) 935, 278, 347, 621, 299, 392, 358, 363

   Solutions: (c) and (e): NO.

5. (12 pts) Consider the following AVL tree. Perform operation delete(1) and rebalance the tree if necessary. You have to use exactly the same algorithm as in class. Show all the intermediate trees. **Solution**

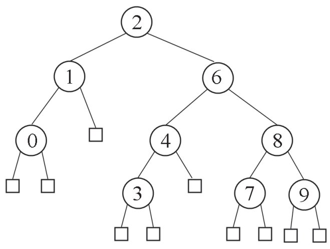
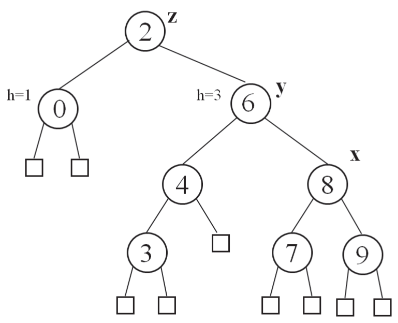
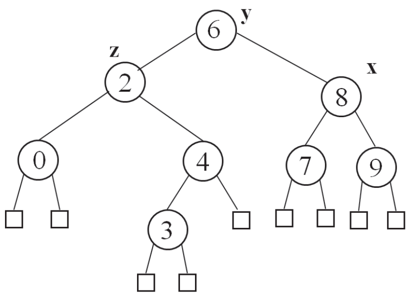


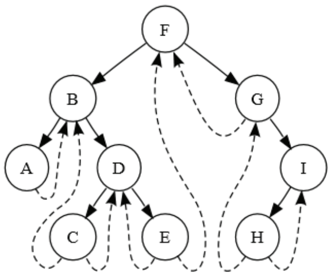
Figure 1: An AVL tree



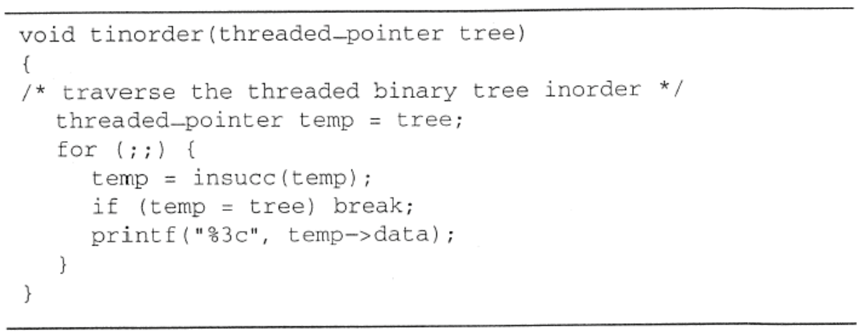


6. (10 pts)

(a) (4 pts) Define threaded binary trees.
Solution: A binary tree is threaded by making all right child pointers that would normally be null point to the inorder successor of the node, and all left child pointers that would normally be null point to the inorder predecessor of the node.

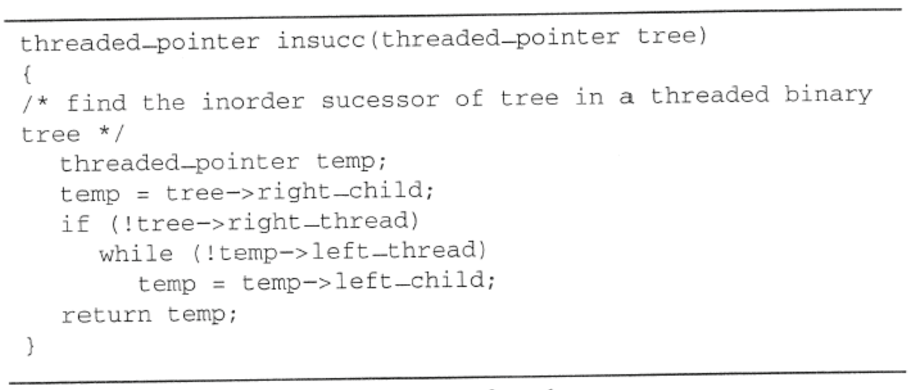


(b) (6 pts) Design a non-recursive algorithm to do inorder traversal of a threaded binary tree. Explain your algorithm in Chinese or English. Also use an example to show how your algorithm works.
Solution:



```
void tinorder(threaded_pointer tree)
{
/* traverse the threaded binary tree inorder */
   threaded_pointer temp = tree;
   for (;;) {
      temp = insucc(temp);
      if (temp = tree) break;
      printf("%3c", temp->data);
   }
}
```

Program 5.11: Inorder traversal of a threaded binary tree



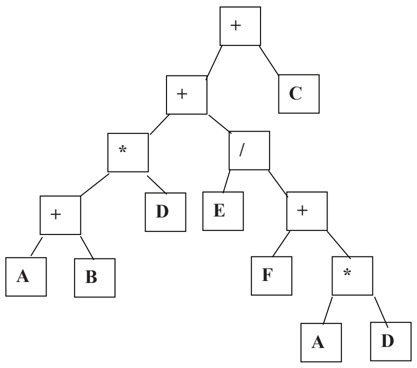
```
threaded_pointer insucc(threaded_pointer tree)
{
/* find the inorder sucessor of tree in a threaded binary
tree */
   threaded_pointer temp;
   temp = tree->right_child;
   if (!tree->right_thread)
      while (!temp->left_thread)
         temp = temp->left_child;
   return temp;
}
```

Program 5.10: Finding the inorder successor of a node

7. (10 pts) Draw the expression tree and write the prefix and postfix forms for the following expression.

$$(A + B) * D + (E/(F + A * D)) + C.$$

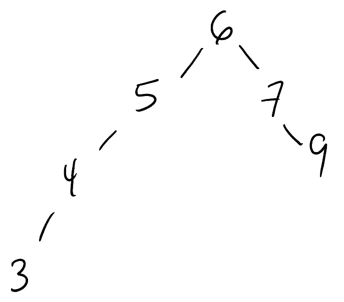
Solution



Prefix: + + * + A B D / E + F * A D C
Postfix: A B + D * E F A D * + / + C +

8. (15 pts)

(a) (10 pts) Draw the Splay tree that results from inserting the keys 4, 9, 3, 7, 5, 6 in that order into an initially empty Splay tree. Show all the intermediate trees (i.e., draw the tree after each insertion is carried out). Also find the potential for each intermediate tree using the potential function discussed in class (i.e., potential of a tree T is $\sum_{node\ x\ in\ T} \lfloor \log(size(x)) \rfloor$, where $size(x)$ is the number of nodes in the subtree rooted at x.
Solution



Potential of the final tree is
$\lfloor \log(size(x_3)) \rfloor + \lfloor \log(size(x_4)) \rfloor + \lfloor \log(size(x_5)) \rfloor + \lfloor \log(size(x_6)) \rfloor + \lfloor \log(size(x_7)) \rfloor + \lfloor \log(size(x_9)) \rfloor =$
$0 + 1 + 1 + 2 + 1 + 0 = 5$

(b) (5 pts) Suppose we want to insert 10 numbers ($\{1, 2..., 10\}$) into an initially empty Splay tree. What will be the order of insertions what will result in a splay tree of height 10 (i.e., becoming a list). Why?
Solution: The keys of value 10, 9 ,.., 4, 3, 2, 1 are inserted in this order in a splay tree. The resulting tree is:

```
1
  2
    3
      ...
        10
```

9. (10 pts) A red-black binary search tree (BST) is called a left-leaning red-black BST if red links lean left (i.e., each red node is the left child of its parent). See Figure 2 for an example. Figure 3 summarizes the rebalancing rules for left-leaning red-black trees w.r.t. insertion. Suppose we insert key 99 into the tree in Figure 2. Show how to rebalance the tree. Show your derivation in sufficient detail.

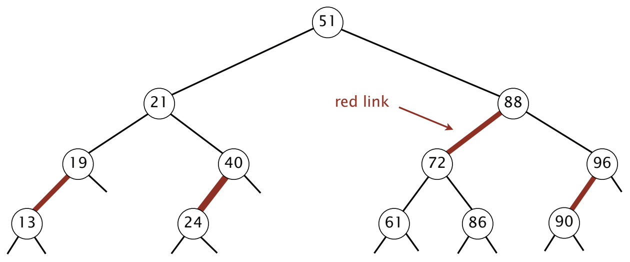


Figure 2: A left-leaning red-black BST

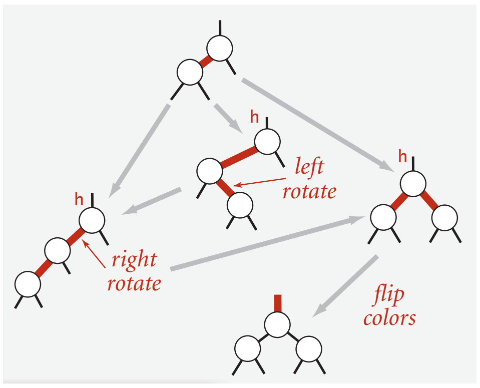
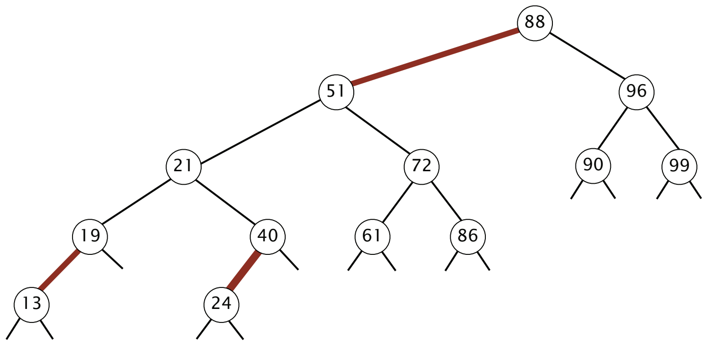


Figure 3: Rebalancing rules

Solution:

To insert 99, we do two color flips, followed by one left rotation. Here is a drawing of the resulting red-black BST.



10. (5 pts) Let A be an array of N integers. Give an algorithm to output all the distinct numbers in A in ascending order. Your algorithm must finish in $O(N \log T)$ time, where T is the number of distinct integers in A. For example, given $A = \{35, 10, 59, 17, 61, 17, 52, 10\}$, your algorithm should output 10, 17, 35, 52, 59, 61. Describe your algorithm in English or Chinese, and explain why your algorithm takes $O(NlogT)$ time.
Solution: Scan the array and maintain all the distinct numbers in a binary tree. After reading the next element, perform dictionary search to check if it already exists in the tree, and output it if not (in which case we insert the element in the tree). As the tree indexes at most T elements at any moment, the total cost is $O(NlogT)$.