

**PROBLEM SET 5**  
**Due: Thursday, April 5**

**Reading:** CLR Section 5.5; Chapter 7; Dynamic Set introduction (pp. 197-199); Sections 11.2 & 11.4; Chapter 13 (skim 13.4)

**Suggested Problems:** 7.1-5; 7.2-1; 7.3-1; 7.4-1 7.4-2; 7.5-3; 7.5-6; 13.1-2, 13.1-2, 13.1-5; 13.2-1; 13-3.5; 13.4-2; 13-2;

**Problem 1 [10]:** Dr. Ima Fleik of the Snake Oil Institute of Learning has just submitted a paper to an algorithms conference in which she claims the following amazing results:

1. An  $O(n)$  worst-case running time comparison-based algorithm for constructing a binary search tree of  $n$  elements.
2. An  $O(n)$  worst-case running time comparison-based algorithm for enumerating the elements of a heap in sorted order.

The program chair of the conference has asked you to read Dr. Fleik's paper and evaluate these two claims. Unfortunately, her algorithms are inscrutable and her analyses are impenetrable. But then it hits you that you can prove both of her claims are false from first principles by applying the decision model for comparison sorts to Tree Sort and to Heap Sort. Write a brief but convincing note to the program chair explaining why Dr. Fleik's claims are false.

**Problem 2 [10]:** Exercise 13.2-2 on p. 250 of CLR. *Note:* one of the sets may be empty.

**Problem 3 [30]:**

- a. Draw as trees the sequence of heaps  $H_0, H_1, \dots, H_{17}$  that results from inserting the following letters into the empty heap  $H_0$ :

T H E Q U I C K B R O W N Y A M S

For example,  $H_1$  should be the result of inserting T into the empty heap  $H_0$ ;  $H_2$  should be the result of inserting H into  $H_1$ ; and so on. Assume that letters appearing earlier in the alphabet have a higher priority than those appearing later. For example, the root node of  $H_{17}$  should be A. You need only show the final tree that results from each insertion; do not show the individual "bubble up" steps that lead to the final tree.

- b. Show how heap  $H_{17}$  would be represented as an array.
- c. Draw as trees the sequence of heaps  $H_{17}, H_{18}, \dots, H_{22}$  that result from extracting and deleting the five highest-priority items of  $H_{17}$  in order of priority (from high to low). You need only show the final tree that results from each deletion; do not show the individual "bubble down" steps that lead to the final tree.

**Problem 4 [20]:**

- a. Draw the binary search tree (BST) that results from inserting the following letters one-by-one (from left to right) into an empty binary search tree:

T H E Q U I C K B R O W N Y A M S

You need only show the final tree that results from inserting all of the letters; do not show the intermediate tree after inserting each letter. To simplify the drawing of trees, do not show any explicit leaves or parent pointers.

- b. Draw the sequence of BSTs  $T_0, T_1, \dots, T_{17}$  where  $T_0$  is the answer to part a and  $T_i$  is obtained from  $T_{i-1}$  by deleting the  $i$ th letter of T H E Q U I C K B R O W N Y A M S. For example, deleting T from  $T_0$  yields  $T_1$ ; deleting H from  $T_1$  yields  $T_2$ ; and so on. When deleting a value  $v$  at a node with two non-empty subtrees, replace  $v$  with its predecessor (not its successor).

**Problem 5 [30]:**

- a. Draw the sequence of height-balanced BSTs  $T_0, T_1, \dots, T_{17}$  that results from inserting the following letters one-by-one (from left to right) into an empty tree:

T H E Q U I C K B R O W N Y A M S

Use the algorithm presented in class for rebalancing height-balanced trees that are unbalanced after insertion. Draw each node as a pair of (1) its value and (2) its height.

- b. Draw the sequence of AVL trees  $T_0, T_1, \dots, T_{17}$  that results from inserting the following letters one-by-one (from left to right) into an empty tree:

T H E Q U I C K B R O W N Y A M S

Use the algorithm presented in class for insertion into AVL trees. Draw each node as a pair of (1) its value and (2) its balance factor.

**Extra Credit [20]:** Exercise 11.4-5 on p. 216 of CLR. Express your algorithm in pseudocode. Your algorithm should print out the keys in an inorder traversal; i.e., the key of every node should be printed out after the key of every node in its left subtree but before every node in its right subtree. Your algorithm should run in  $O(1)$  space (i.e., no recursion or explicit stacks), and you should not modify the tree. However, you may assume that you can compare two pointers for equality. Remember that each node holds a pointer to its parent in addition to its left and right subtrees. *Hint:* if you were exploring a cave system that branched out like a binary tree, what strategy would you use to explore the whole system?

*Problem Set Header Page*  
*Please make this the first page of your hardcopy submission.*

**CS231 Problem Set 5**  
**Due Thursday, April 5, 2001**

Name:

Date & Time Submitted (*only if late*):

Collaborators (*anyone you collaborated with in the process of doing the problem set*):

*In the **Time** column, please estimate the time you spent on the parts of this problem set. Please try to be as accurate as possible; this information will help me to design future problem sets. I will fill out the **Score** column when grading your problem set.*

<b>Part</b>	<b>Time</b>	<b>Score</b>
General Reading		
Problem 1 [10]		
Problem 2 [10]		
Problem 3 [30]		
Problem 4 [20]		
Problem 5 [30]		
Extra Credit [20]		
<b>Total</b>		