15-122 : Principles of Imperative Computation

Summer 1 2012

Assignment 6

(Theory Part)

Due: Monday, June 18, 2012 in class

Name: ______________________________

Andrew ID: ______________________________

Recitation: ______________________________

The written portion of this weeks homework will give you some practice working with binary search and AVL trees. You can either type up your solutions or write them *neatly* by hand, and you should submit your work in class on the due date just before lecture begins. Please remember to *staple* your written homework before submission.

Question	Points	Score
1	6	
2	4	
3	10	
Total:	20	

1. **Treaps** Famous Fred Hacker is a huge fan of both binary search trees and heaps. Given an ordered sequence of $a_1 < a_2 < ... < a_n$, there are many different binary search trees that can be produced out of them. Now, suppose we introduce an additional requirement on the BST: it has to satisfy the min-heap order property, producing a treap. More precisely, we assign a priority p_i to each node a_i and insist that the tree:

   - is a BST with respect to the key values a_i
   - has the min-heap order property with respect to the priorities p_i

(3) (a) Draw a treap over elements $a < b < c < d$ if we assign priorities 4,1,2,3 to a, b, c, d respectively?

Solution:

(3) (b) What treap do we get if we assign priorities 3,4,1,2 to a, b, c, d respectively?

Solution:

2. **Binary Search Trees**

(4) (a) Famous Fred Hacker enjoys climbing trees, but he is afraid to do so if the tree has height greater than 0. Recall that the height of a binary tree is the maximum number of nodes (inclusive) in the path from the root to a leaf. For example, a binary tree of just one node has height 1. Write a recursive function

```
int bst_height(bst B)
```

that returns the height of a binary search tree. This function itself should not be recursive, but you will need to write a helper function that *is* recursive. See the BST code developed in lecture for examples of wrapper functions and recursive helper functions.

Solution:

3. **AVL Trees**

(4) (a) Famous Fred Hacker also likes mudkips. Draw the AVL tree that results after successively inserting the following keys in the order shown

`7 12 10 16 4 2`

into an initially empty tree, maintaining and restoring the invariants of a BST and the additional balance invariant required for an AVL tree after every insert. Your answer should show the tree after each key is successfully inserted.

Solution:

(b) Famous Fred Hacker has a propensity for coming up with ingenious algorithms, but he has a poor sense of balance. One day, he was walking across the street and tripped. We want to show that the height of an AVL tree storing n keys is $O(\log n)$. For this purpose we count the minimum number of nodes n in an AVL tree of height h.

(3) i. Fill in the table below:

h	n
0	1
1	2
2	
3	
4	

(3) ii. Recall that the Fibonacci numbers F are defined by

$$F(n) = F(n-1) + F(n-2)$$

$$F(0) = 0, F(1) = 1$$

How is the minimum number of nodes $n = T(h)$ in an AVL tree of height h related to the Fibonacci numbers?

Solution:

$T(h) =$