

## CSE3358 Problem Set 8

03/18/05

Due 03/25/05

### Problem 1: Practice Insertions (10 points)

Show the results of inserting the following keys

F, S, Q, K, C, L, H, T, V, W, R, N, P, A, B, X, Y, D, Z, E

in the order shown into an originally empty

(a) (5 points) Red-Black Tree

(b) (5 points) B-tree with  $t = 2$

### Problem 2: AVL tree (30 points)

In 1962, Adelson-Velskii and Landis invented the AVL tree. An AVL tree is a binary search tree that satisfies the following additional property:

AVL property: For any node  $x$ ,  $|h[\text{left}[x]] - h[\text{right}[x]]| \leq 1$  where  $h[x]$  denotes the height of  $x$ , and we define  $h[NIL] = -1$ .

In other words, the heights of the left and right subtrees (children) of a node differ by at most one.

(a) (5 points) Let  $S(h)$  be the minimum number of nodes in an AVL tree of height  $h$ . Show that  $S(0) = 1$  and  $S(1) = 2$  and that for  $h \geq 2$ ,  $S(h) = S(h - 1) + S(h - 2) + 1$ .

(b) (5 points) Using induction, show that  $S(h) \geq \phi^h$ , where  $\phi = \frac{1+\sqrt{5}}{2}$ .

(c) (5 points) Conclude that the height  $h$  of an AVL tree on  $n$  nodes satisfies  $h = \Theta(\log n)$ .

(d) (15 points) Given that every node  $x$  stores its height  $h[x]$ , an AVL tree can be maintained after an insertion operation. Here's how: After inserting a new node  $x$  as a leaf, we set  $h[x] = 0$  and we go up the tree updating the height of every node on the path from  $x$  to the root by incrementing its height by 1. When we encounter the first node  $y$  for which  $|h[\text{left}[y]] - h[\text{right}[y]]| = 2$  (if any), we fix the AVL property for  $y$  by performing rotations. Depending on the position of  $x$  relative to  $y$ , different rotation(s) must be performed. We have two cases to consider (the other cases are symmetric)

- case 1:  $x$  is in the left subtree of  $\text{left}[y]$
- case 2:  $x$  is in the right subtree of  $\text{left}[y]$

The above two cases suggest that, since node  $y$  violates the AVL property after insertion, the left subtree of  $y$  is higher than the right subtree of  $y$  (the insertion was done in the left subtree of  $y$ ), and the height difference is 2 (more than 1). For each of the two above cases, describe the rotation(s) required to re-establish the AVL property for node  $y$ . Argue that after re-establishing the AVL property for node  $y$ , no more height updates or fixes will be needed, i.e. the resulting tree is AVL.

**Problem 3: The dishonest successor (30 points)**

As you know, the homework policy states that dishonest students are given red tags. Therefore, every student has a grade and a red-tag boolean flag to identifying whether the student is dishonest or not.

We wish to support all dynamic set operations on students such as INSERT, DELETE, SEARCH, MINIMUM, MAXIMUM, SUCCESSOR, PREDECESSOR in addition to the following operation:

Given a student  $x$  with grade  $k$ , find the dishonest student  $y$  (if any) with the lowest grade  $k'$  such that  $k' > k$  in  $O(\log n)$  time, where  $n$  is the number of students. In other words, given a student  $x$ , we wish to find the dishonest successor of  $x$  in  $O(\log n)$  time.

Explain how you can augment a red-black tree to support this operation. In doing so, describe the 4 steps discussed in class:

- (a) (0 points) Choose the underlying data structure (red-black tree in this case) and determine what constitute the keys.
- (b) (10 points) Identify the additional information that will augment the red-black tree
- (c) (10 points) Argue that the additional information can be maintained easily by insertion and deletion and describe how it is maintained by rotations
- (d) (10 points) Provide a pseudocode for the operation DISHONEST-SUCCESSOR( $x$ )

**Problem 4: Some B-tree operations (30 points)**

Given a B-tree  $T$  with minimum degree  $t$ , describe an algorithm that:

- (a) (10 points) Finds  $(x, i)$  in  $O(\log_t n)$  time such that  $key_i[x]$  is the MINIMUM key in  $T$ .
- (b) (10 points) Given  $(x, i)$ , finds  $(y, j)$  such that  $key_j[y]$  is the SUCCESSOR of  $key_i[x]$  in  $T$  in time:
  - $O(\log_t n)$  if  $x$  is a non-leaf
  - $O(\log_2 n)$  if  $x$  is a leaf
- (c) (10 points) Given  $k$ , finds  $(x, i)$  in  $O(\log_2 n)$  time such that  $key_i[x] = k$  in  $T$ .

**Problem 5: Overlapping rectangles (20 points)**

VLSI databases commonly represent an integrated circuit as a list of rectangles. Assume that each rectangle is rectilinearly oriented (sides parallel to the  $x$ - and  $y$ -axis), so that a representation of a rectangle consists of its minimum and maximum  $x$ - and  $y$ -coordinates. Give an  $O(n \log n)$  time algorithm to decide whether or not a set of rectangles so represented contains two rectangles that overlap. Your algorithm need not report all intersecting pairs, but it must report that an overlap exists if one rectangle entirely covers another, even if the boundary lines do not intersect. (*Hint*: Move a “sweep” line across the set of rectangles and maintain their heights in an interval tree).