CS161 Handout 09
Summer 2001 July 26, 2001

Problem Set 5: Advanced Data Structures

Due: Thursday, August 2nd at 10:15 a.m.

Problem 1: Joins on Red-Black Trees (15 points, courtesy of CLR, Problem 14-2, p. 278)

The join operation takes two dynamic sets S_1 and S_2 and an element x such that for any $x_1 \in S_1$ and $x_2 \in S_2$, we have $key[x_1] \le key[x] \le key[x_2]$. It returns a set $S = S_1 \cup \{x\} \cup S_2$. Here we'll investigate how to implement this join operation on red-black trees.

- Given a red-black tree T, we store its black-height as the field $bh[T]$. Argue that this field can be maintained by **`RB-Insert`** without requiring any extra storage in the tree and without increasing the asymptotic running times. Show that while descending through T, we can determine the black-height of each node we visit in $O(1)$ time per node visited.

We want to implement the operation **`RB-Join(`**T_1**`,`** x**`,`** T_2**`)`**, which destroys T_1 and T_2 and returns a red-black tree $T = T_1 \cup \{x\} \cup T_2$. Let n be the total number of nodes in T_1 and T_2.

- Assume without loss of generality that $bh[T_1] \ge bh[T_2]$. Describe an $O(\lg n)$-time algorithm that finds a black node y in T_1 with the largest key among all those nodes whose black-height is $bh[T_2]$.
- Let T_y be the sub-tree rooted at y. Describe how T_y can be replaced by $T_y \cup \{x\} \cup T_2$ in $O(1)$ time without destroying the binary-search-tree property.
- What color should we make x so that so that red-black properties 1, 2, and 4 are maintained? Describe how property 3 can be restored in $O(\lg n)$-time.
- Argue that the running time of **`RB-Join`** is $O(\lg n)$.

Problem 2: Listing all Interval Intersections (10 points, courtesy of Exercise 15.3-4)

Given an interval tree T and an interval i, describe how all intervals in T that overlap i can be listed in $O(\min(n, k \lg n))$, where k is the number of intervals in the output list. You may temporarily delete entries from the interval tree if necessary, though a snazzier implementation will neither delete them nor mark previously listed nodes in any particular way.

Problem 3: Minimum Gap between Keys (15 points, courtesy of Exercise 15.3-6)

Show how to maintain a dynamic set Q of numbers that supports the operation **`Min-Gap`**, which gives the magnitude of the difference between the two closest numbers in Q. For example, if $Q = \{1,5,9,15,18,22\}$, then **`Min-Gap(`**Q**`)`** would return 3, because

15 and 18 are nearest neighbors. Make the operations **`Insert`**, **`Delete`**, **`Search`**, and **`Min-Gap`** as efficient as possible, and analyze their running times. (Hint: Jerry's solution stores three extra fields of information per node, and one of them is called **`min-gap[x]`**, which tracks the smallest interval between elements in the sub-tree rooted at **`x`**. To see what other fields are needed, imagine that you need to re-compute the **`min-gap`** field for the root of the tree, and ask yourself: what information needs to be stored in the root's left and right children in order for me to compute **`min-gap[root[T]]`**?)