CS 457: Data Structures and Algorithms 1
Fall 2003-2004 Homework

5. (25 Pts.) Join operation on redblack trees

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 $\text{key}[x_1] \leq \text{key}[x] \leq \text{key}[x_2]$. It returns a set $S = S_1 \cup \{x\} \cup S_2$. In this problem, we investigate how to implement the join operation on redblack trees.

(a) Given a redblack tree $T$, we store its blackheight as the field $\text{bh}[T]$. Argue that this field can be maintained by $\text{RBINSERT}$ and $\text{RBDELETE}$ (as given in the textbook) without requiring extra storage in the nodes of the tree and without increasing the asymptotic running times. Show that while descending through $T$, we can determine the blackheight of each node we visit in $O(1)$ time per node visited.

We wish to implement the operation “$\text{RBJOIN}(T_1, x, T_2)$”, which may destroy $T_1$ and $T_2$ and returns a redblack tree $T = T_1 \cup \{x\} \cup T_2$. Let $n$ be the total number of nodes in $T_1$ and $T_2$.

(b) Assume that $\text{bh}[T_1] \geq \text{bh}[T_2]$. Describe an $O(\log n)$-time algorithm that finds a black node $y$ in $T_1$ with the largest key from among those nodes whose blackheight is $\text{bh}[T_2]$.

(c) Let $T_y$ be the subtree rooted at $y$. Describe how $T_y \cup \{x\} \cup T_2$ can replace $T_y$ in $O(1)$ time without destroying the binarysearchtree property.

Consider the following redblack properties:

- every node is either red or black
- every leaf is black
- for each node, all paths from the node to descendant leaves contain the same number of black nodes

(d) What color should we make $x$ so that the above redblack properties are maintained?

Consider the following redblack properties:

- the root is black
- if a node is red, then both its children are black

(e) Describe how the above two properties can be enforced in $O(\log n)$ time.

(f) Argue that the running time of “$\text{RBJOIN}$” is $O(\log n)$.