cs201c, Tutorial 2: AVL, 2-3-4, and red-black trees

Instructor: Apurva Mudgal

August 26, 2019

1 Basic properties

1. What are the maximum number of rotations possible in an insert or delete operation on an AVL tree with n nodes. Give worst-case examples.

2. Consider a 2-3-4 tree with n nodes.

   Suppose we do an insert operation on this tree. What is the maximum number of split operations? Give a worst-case example.

   Now suppose we do a delete operation on this tree. What is the maximum number of merge operations? What is the maximum number of rotate operations? Give worst-case examples.

3. Complete the mapping between insert and delete operations on a 2-3-4 tree and insert and delete operations on the corresponding, isometric red-black tree.

   Further, answer the following questions:

   (a) What is the maximum number of rotations when you insert a key into a red-black tree with n nodes?

   (b) What is the maximum number of rotations when you delete a key from a red-black tree with n nodes?

4. Describe in detail procedures for one-pass (top-down) insert and delete operations on a 2-3-4 tree. (By one-pass we mean that you do not trace the root-to-leaf path two times.)

5. Let k be a positive integer greater than or equal to 1. $AVL(k)$ trees are an extension of AVL trees, with the property that the height balance factor at each node now lies in the set $\{-k, -(k-1), \ldots, 0, \ldots, k-1, k\}$. (Note that $AVL(1)$ trees are the same as the AVL trees discussed in class.)

   (a) What is the maximum height of an $AVL(k)$ tree on n nodes?

(b) Design insert and delete operations on an $AVL(k)$ tree and analyze their worst-case performance.

6. (a) Suppose we insert numbers $1, 2, \ldots, n$ in this order starting from an empty AVL tree. Describe the final AVL tree created as a result of this.

   (b) Suppose we insert numbers $1, 2, \ldots, n$ in this order starting from an empty 2-3-4 tree. Describe the final 2-3-4 tree created as a result of this.

2 Augmenting a tree

1. Augment a binary search tree so that given any key k, the sum of all keys in the tree less than or equal to k can be obtained in $O(h)$ time. (Here h is the height of the tree.)

2. Augment AVL trees so that given any integer k ($1 \leq k \leq n$), the k-th smallest key in an AVL tree with n nodes can be found in $O(\log_2(n))$ time.

3. Augment a 2-3-4 tree so that finding k-th smallest key in a 2-3-4 tree with n nodes can be done in $O(\log_2(n))$ time.

3 Merging two trees

Note that the original trees T_1 and T_2 do not exist after the merging operation. Further, you are allowed to augment the trees suitably, if required.

1. Suppose you are given two binary search trees T_1 and T_2, such that every key in tree T_1 is strictly less than every key in tree T_2. Further, assume the heights of trees T_1 and T_2 are h_1 and h_2 respectively.

   Give an $O(h_1 + h_2)$ algorithm to merge T_1 and T_2 into a single binary search tree T.

2. Suppose you are given two AVL trees T_1 and T_2 of heights h_1 and h_2 respectively. Further, assume all keys in tree T_1 are strictly less than all keys in tree T_2.

   Give an $O(h_1 + h_2)$ time algorithm to merge T_1 and T_2 into a single AVL tree T.

3. You are given two 2-3-4 trees T_1 and T_2 of heights h_1 and h_2 respectively. Assume all keys in first tree T_1 are strictly less than all keys in second tree T_2.

   Give an $O(h_1 + h_2)$ time algorithm to merge the two given trees into a single 2-3-4 tree T.