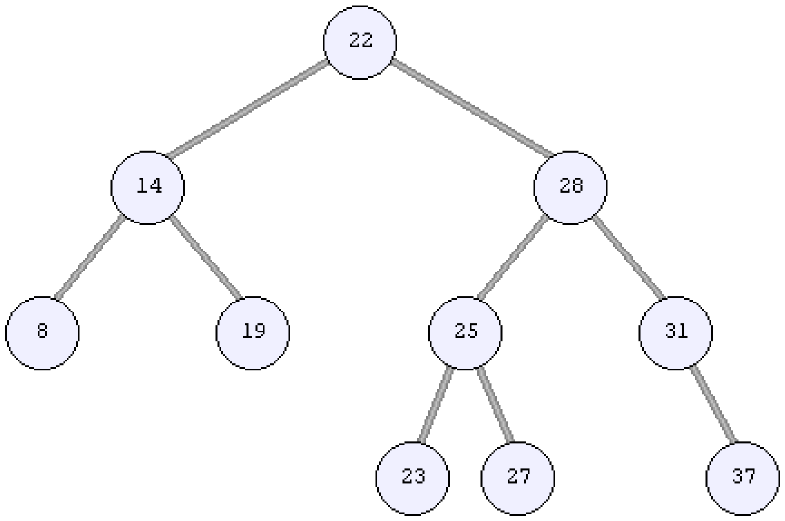
CSE332 Data Abstractions, Summer 2010

Homework 3

Due Friday, July 16, 2010 at the beginning of class.

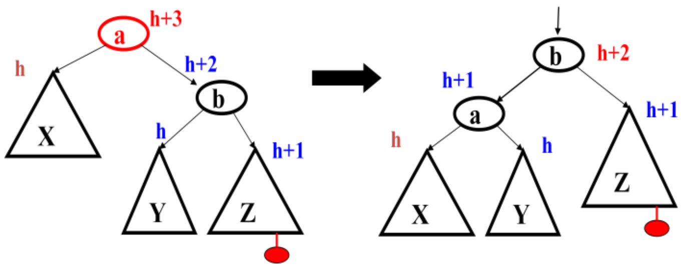
Problem 1: More AVL Insertion

Starting with the AVL tree below, insert the following values: 26, 34, 40, 29, 33, 32. Show the resulting tree after each insertion.

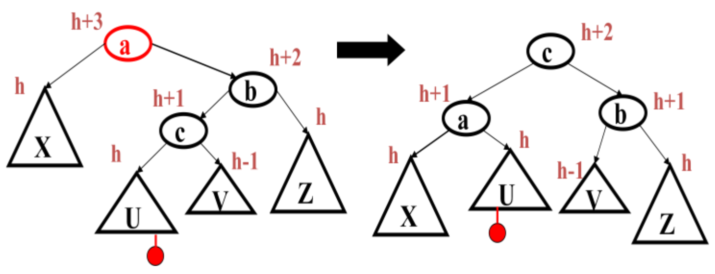


Problem 2: AVL Rotation Scenarios

a. The diagram below shows the general case for performing a case 4 (right-right) rotation; an insertion has taken place in subtree Z, and an imbalance has been detected at node 'a'. Assume subtree Z has height h before the insertion, and height h+1 after the insertion, as shown below. Argue that subtrees X & Y must also have height h prior to the insertion; that is, show that it is not possible for either to have height h-1 or h+1.



b. The diagram below shows the general case for the case 3 (right-left) rotation; an insertion has taken place in subtree U, and an imbalance has been detected at node 'a'. Assume subtree U has height h-1 before the insertion, and h afterwards, as shown. Argue that subtree X must have height h prior to the insertion, V must have height h-1 and Z must have height h.



Problem 3: B-Tree Insertion

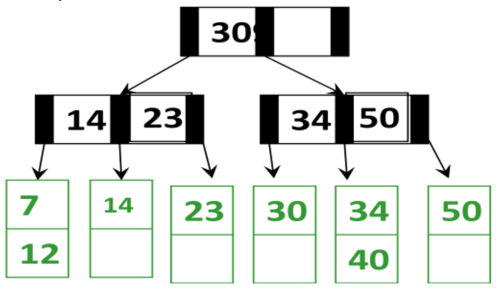
Show the result of inserting 28, 12 17, 4, 31, 34, 8, 14 & 16 in that order into an initially empty B tree with M = 3 and L = 2. (Recall the text, lecture, and this problem call a B tree what many call a B+ tree.) Show the tree after each insertion, clearly labeling which tree is which. In an actual implementation, there is flexibility in how insertion overflow is handled. However, in this problem, follow these guidelines:

- • Always use splitting and not adoption.
- • Split leaf nodes by keeping the smallest 2 elements in the original node and putting the 1 largest element in the new node.
- • Split internal nodes by keeping the 2 children with the smaller values attached to the original node and attach the 2 children with the larger values to the new node.

Problem 4: B-Tree Deletion

Starting with the B-Tree below, with M=3 & L=2, delete the following values one at a time, in order: 14, 23, 7, 50 & 30. Show the B-Tree after each deletion. When performing deletions, use the procedure described in lecture:

- • If a node underflows, try to use adoption before merging nodes.
- • Try to adopt from the left neighbor with the same parent, then, if that fails, the right neighbor with the same parent.



Problem 5: B-Tree Predecessor

In this problem, assume every B tree node has a reference to its parent (except for the root node, as it has no parent) as well as its children (except for leaves), so that it is easy to "move up or down" the tree. Suppose you have a direct reference to the leaf holding a data item with a key k.

a. Describe in English an algorithm for finding the predecessor of k in the B tree (or determining that k is the minimum element and therefore has no predecessor).

b. Give the worst-case number of total nodes accessed by your algorithm in terms of the tree height h. Describe briefly a worst-case situation.

c. Give the best-case number of total nodes accessed by your algorithm. Explain briefly why most keys would exhibit the best-case behavior.