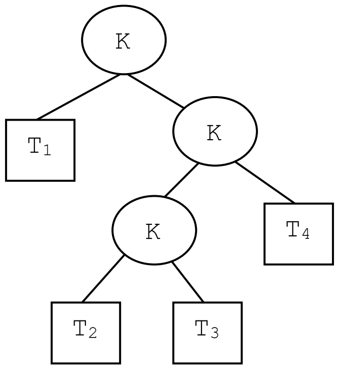
Problem Set 4

Problem 1 Consider a tree storing 100,000 entries. What is the worst-case height of T in the following cases?

a. T is an AVL tree.
b. T is a (2,4) tree.
c. T is a binary search tree
d. T is a splay tree

Problem 2 What does a splay tree look like if its entries are accessed in increasing order of their keys?

Problem 3 Assuming that *T*1, *T*2, *T*3, and *T*4 are AVL trees of height *h*, alter the following binary search tree to yield an AVL tree.



Problem 4 The method findAll(k) in a binary search tree T is to find a set of all the entries in T whose keys are equal to k. Design an algorithm for findAll(k) which runs in time O(h+s), where h is the height of T and s is the size of the set returned.

Problem 5 Show that after the tri-node restructuring operation on the subtree rooted at the node z, the new subtree is an avl tree.

Problem 6 Let T and U be (2,4) trees with n and m entries, respectively, such that all the entries in T have keys less than the keys of all the entries in U. Describe an O(log n + log m)-time algorithm for merging T and U into a single (2,4) tree.

Problem 7 Consider a sequence of keys (5, 16, 22, 45, 2, 10, 18, 30, 50, 12, 1). Draw the final result of inserting the entries with these keys in the given order into

a. An initially empty (2, 4) tree.
b. An initially empty red-black tree.

Problem 8 Consider a binary search tree where all the keys are distinct. Describe a modification to the binary search tree that supports the following two index-based operations in O(h) time, where h is the height of the tree:

- atIndex(i): return the entry in the binary search tree with index i.
- indexOf(e): return the index of entry e.

The index of an entry is the sequence number of the entry in the inorder traversal on the tree. For example, the indices of the first two entries with the smallest key and the second smallest key are 0 and 1, respectively.