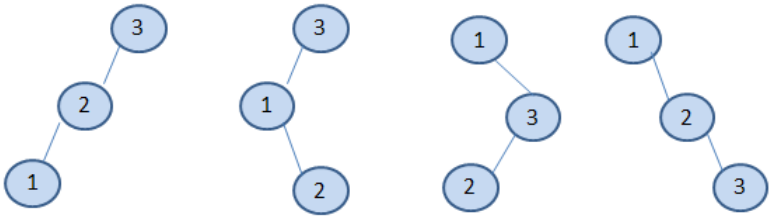
COS 226 – Data Structures and Algorithms
Fall 2014 – Flipped Lecture Section
Group Worksheet week 6 – 10.16.14
30 minutes

Problem #1: AVL and Red-black Trees

An AVL tree is a balanced tree such the height invariant is satisfied between left and right sub trees of any node. That is, |height (T_L) – height (T_R)| <=1 for all nodes T, where T_L and T_R denotes left and right sub trees. There are 4 different scenarios that occur in AVL trees when inserting a new entry as shown below.



(a) Two of them require a single rotation (a left or right rotation) and other two requires double rotation (left followed by right or vice versa). Identify which ones are which and show the balanced tree after single or double rotations.

(b) Insert the keys 7, 6, 5, 4, 3, 2, 1 in that order into an red-black tree while performing balance operations. How many rotations, and color flips are required to create a balanced tree of the 7 nodes.

(c) Now create a balance tree using AVL operations in Part (a). Note that when a new element is inserted the balance of the tree can be broken in one of the 4 ways listed. Perform the rotations as necessary to maintain the AVL height invariant. How many rotations were performed in the case of AVL trees? Is it more or less than red-black tree operations?

(d) **(Challenge)** Prove that the height of an AVL tree is of order $\log_2 n$ for any tree with n nodes. Hint: Let N_h denote the number of nodes in an AVL tree of depth h and convince yourself that following equation holds based on AVL tree invariant.

$$N_h \geq N_{h-1} + N_{h-2} + 1$$

Problem #2: **Hashing with Quadratic Probing**

Hashing with quadratic probing

Given an array based hash table, quadratic probing allows one to find an available spot using the formula $h(x_i) = x_0 + i^2$, where x_0 is the original spot in the table and x_i is the i-th probe to find a spot for the entry.

(a) Show the table after inserting the keys with hash codes 45, 56, 78, 34 23, 88, 65, 75, 44 into a table of size 10 using Quadratic probing.

(b) The quadratic probing can be expensive if one needs to compute squares to do probes to find a place to insert. Suggest ways to avoid computing squares in each probe.

(c) Construct a set of hash codes (use table size = 10), where it is not possible to find a spot for every key using quadratic probing. What would you do in a case like this?

Bonus Problems

Problem #3

Find a sequence of keys to be inserted into a BST and into a red-black BST such that the height of the BST is less than the height of the red-black BST, or prove that no such sequence is possible.

Problem 4: (Cuckoo hashing)
Develop a ST implementation that maintains two hash tables and two hash functions. Any given key is in one of the hash tables, but not both. When inserting a new key, hash to one of the tables, if the position is occupied, replace the key with the new key and hash the old key into the other table (again kicking out a key that may be residing there). If the process cycles, restart. Keep the tables less than half full. Show that this method uses amortized constant time for insert.