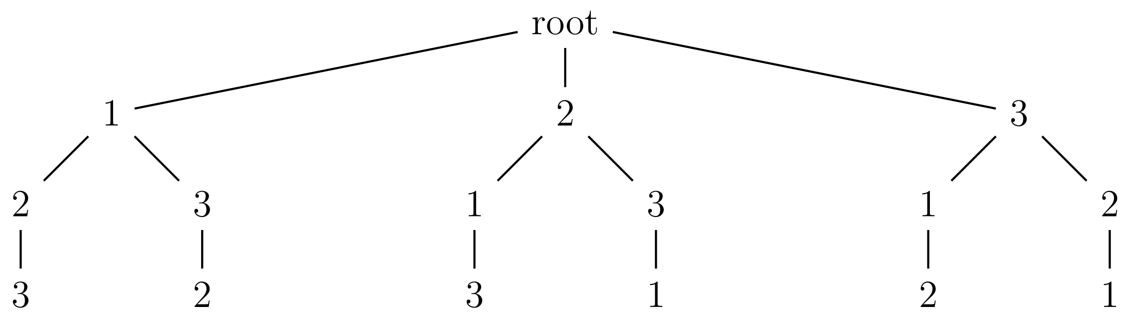
Lab07-Trees

VE281 - Data Structures and Algorithms, Xiaofeng Gao, TA: Qingmin Liu, Autumn 2019

∗ Please upload your assignment to website. Contact webmaster for any questions.

∗ Name:Jintian Ge   Student ID:517021911142   Email: gejintian@sjtu.edu.cn

Hint: You can use the package **tikz** to draw trees.



1. Red-black Tree

   (a) Suppose that we insert a sequence of keys 9, 3, 1 into an initially empty red-black tree. Draw the resulting red-black tree.

   (b) Suppose that we further insert key 6 into the red-black tree you get in Problem (1-a). Draw the resulting red-black tree.

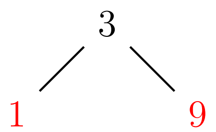
   (c) Suppose that we further insert keys 2, 8 into the red-black tree you get in Problem (1-b). Draw the resulting red-black tree.

   (d) Suppose that we further insert key 7 into the red-black tree you get in Problem (1-c). Draw the resulting red-black tree.

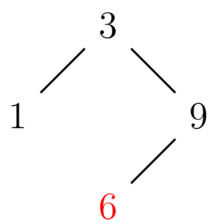
   (e) Suppose that we further insert keys 4, 5 into the red-black tree you get in Problem (1-d). Draw the resulting red-black tree.

   When you draw the red-black tree, please indicate the color of each node in the tree. For example, you can color each node or put a letter **b/r** near each node.

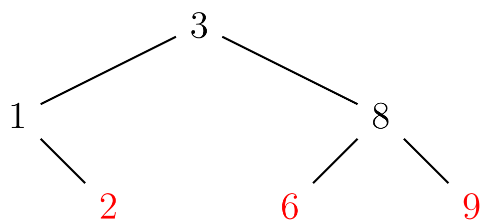
Solution. (a)

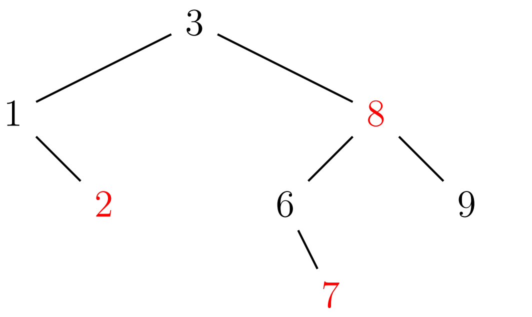


(b)

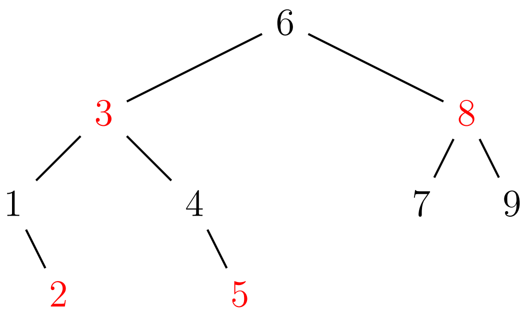


(c)





(d)

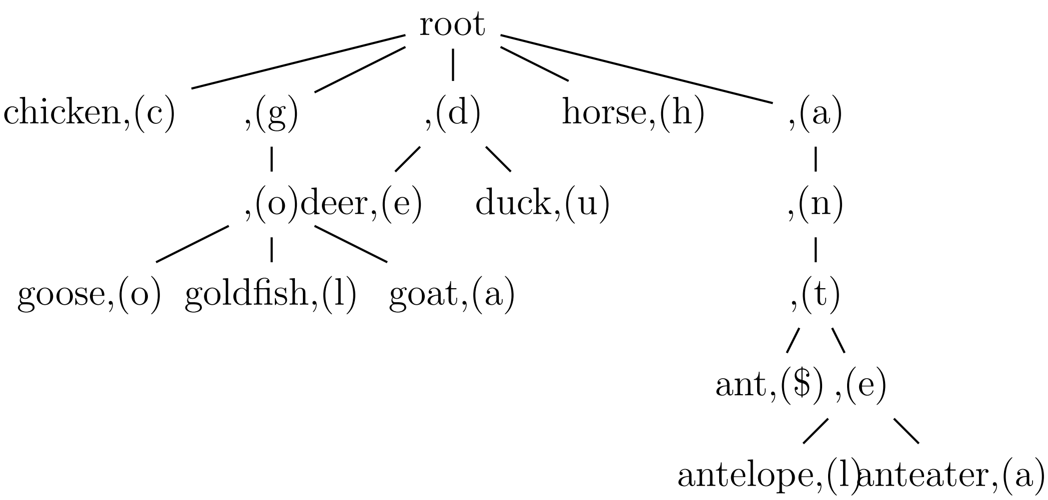


(e)

□

2. Show the alphabet trie for the following collection of words: {chicken, goose, deer, horse, antelope, anteater, goldfish, ant, goat, duck}.

Solution. In this tree, I use $(*)$ to indicate the alphabet in the path.



□

3. Show that any arbitrary n-node binary search tree can be transformed into any other arbitrary n-node binary search tree using $O(n)$ rotations.

Hint: First show that at most $n-1$ right rotations suffice to transform the tree into a right-skewed binary search tree.

Solution. First we show that at most $n-1$ right rotations suffice to transform the tree into a right-skewed binary search tree:

From the most right node, we begin to search upwards. For any nodes with a left subtree, we apply right rotation. In a right rotation, it will rotate one node into the right most path. So, at most $n-1$ rotations can ensure that the BST becomes a right-skewed BST.

Now, for any trees with the same nodes, they will have the same right-skewed BST convertion. Suppose we have two trees A and B. We convert both A and B into a right-skewed tree, and remember the process of B, and finally convert the right-skewed tree which is converted by A into B. □

4. Suppose that an AVL tree insertion breaks the AVL balance condition. Suppose node P is the first node that has a balance condition violation in the insertion access path from the leaf. Assume the key is inserted into the left subtree of P and the left child of P is node A. Prove the following claims:

   (a) Before insertion, the balance factor of node P is 1. After insertion and before applying rotation to x the violation, the balance factor of node P is 2.

   (b) Before insertion, the balance factor of node A is 0. After insertion and before applying rotation to x the violation, the balance factor of node A cannot be 0.

Solution. Notation:

$'$ denotes the tree after insertion. For example, A' denotes tree A after insertion.

$H(*)$ denotes height of the tree *.

(a) We use B to denote the right child of P. B could be null. Then, before insertion, P is balanced. So we have

$$|H(A) - H(B)| \leq 1 \Rightarrow |B(P)| \leq 1$$

Since each insertion will increase the height of a tree by at most 1, we have

$$H(A') \leq H(A) + 1,$$

$$B(P') \leq B(P) + 1$$

So, $-1 \leq B(P') \leq 2$. After insertion, P is unbalanced, so we have:

$$B(P') = 2$$

$$B(P) = 1$$

(b) P' is unbalanced means $H(A)$ increases after insertion. This indicates that the subtree of A with greater height has increased.

We assume that $H(A.left) \geq H(A.right)$. Then, H(A.left) will be increased by one, such that:

$$B(A') = H(A.left') - H(A.right) = H(A.left) + 1 - H(A.right) \neq 0$$

Since after insertion, A' is still balanced, $|B(A')| \leq 1$. This can be converted to:

$$H(A.left) + 1 - H(A.right) \leq 1$$

In the beginning we assume that $H(A.left) \geq H(A.right)$. So finally we have:

$$H(A.left) - H(A.right) = 0 \Rightarrow B(A) = 0$$

□