Lab07-Trees

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

∗ Name:Jin Zhejian Student ID: 517370910167 Email: jinzhejian@outlook.com

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.

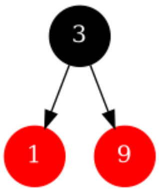
   (c) Suppose that we further insert keys 2, 8 into the red-black tree you get in Problem (1-b). Draw the resulting red-black tree.

   (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.

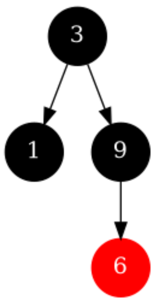
   (e) Suppose that we further insert keys 4, 5 into the red-black tree you get in Problem (1-d). Draw the resulting red-black tree.

   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.

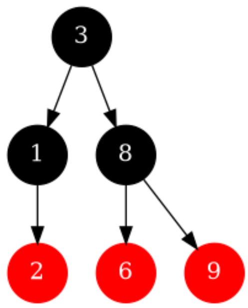
   Solution.



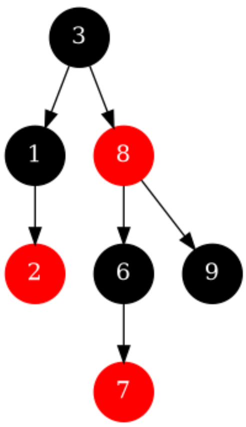
(a)



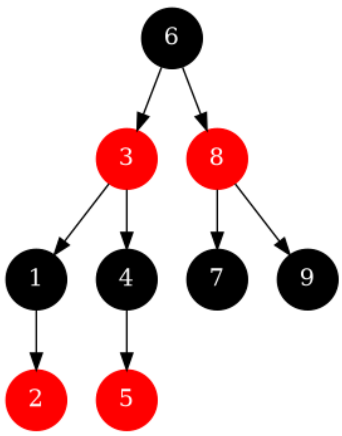
(b)



(c)



(d)

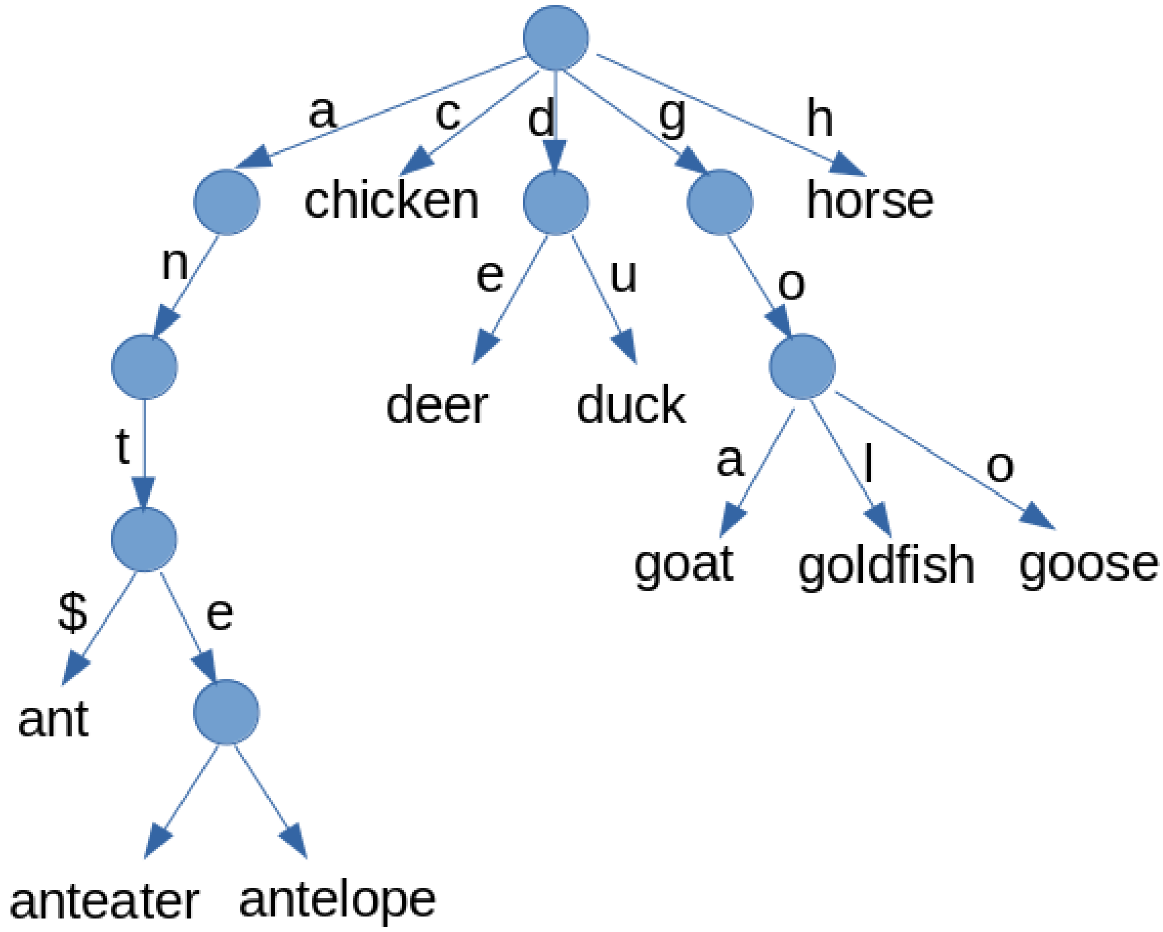


(e)

□

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

Solution. □



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.

a). Each right rotation will increase the length of the rightmost chain by at least 1. Therefore, for an arbitrary n-node BST, in the worst case (the right most chain has only 1 node), it needs $n-1$ right rotations to transform the BST into a right-skewed BST.

b). Suppose we start from T_1, and transform T_1 into T_2. From T_1 to right-skewed BST, we need m right rotations, and From T_2 to right-skewed BST, we need k right rotations. Since rotation is reversible, we can do k steps of reverse right rotations to transform right-skewed BST to T_2. Therefore, in total, we need $m + k < (n-1) + (n-1) = 2n - 2 = O(n)$ rotations to transform T_1 into T_2.

□

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 fix 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 fix the violation, the balance factor of node A cannot be 0.

Solution.

(a) Suppose before insertion, $B_P = h_{P,l} - h_{P,r}$, $B_A = h_{A,l} - h_{A,r}$,.
After insertion, $B'_P = h'_{P,l} - h'_{P,r}$, $B'_A = h'_{A,l} - h'_{A,r}$.
Since the key is inserted into the left subtree of P, $h'_{P,r} = h_{P,r}$, and $0 \leq h'_{P,l} - h_{P,l} \leq 1$.
Therefore, we have $0 \leq B'_P - B_P \leq 1$,
Since P is balanced before insertion, $-1 \leq B_P \leq B'_P \leq B_P + 1 \leq 2$.
But since P is not balanced after insertion, $|B'_P| \geq 2 \Rightarrow B'_P = 2$.
Then $2 \leq B_P + 1 \leq 2 \Rightarrow B_P = 1$.

(b) We have $B_P = h_{P,l} - h_{P,r} = \max\{h_{A,l}, h_{A,r}\} - h_{P,r} = 1$.
$B'_P = \max\{h'_{A,l}, h'_{A,r}\} - h'_{P,r} = \max\{h'_{A,l}, h'_{A,r}\} - h_{P,r} = 2$.
Therefore, $\max\{h'_{A,l}, h'_{A,r}\} = \max\{h_{A,l}, h_{A,r}\} + 1$.
Suppose the key is inserted into the left subtree of node A:
    Then, $h'_{A,r} = h_{A,r} \Rightarrow \max\{h'_{A,l}, h_{A,r}\} = \max\{h_{A,l}, h_{A,r}\} + 1$
    If $h'_{A,l} < h_{A,r}$
        then $h_{A,r} = \max\{h_{A,l}, h_{A,r}\} + 1$
        $\Rightarrow h'_{A,l} < h_{A,r} = h_{A,l} + 1$
        which is impossible because the key is inserted into the left subtree of A.
    So, $h'_{A,l} = \max\{h_{A,l}, h_{A,r}\} + 1$
    So, $h'_{A,l} - h_{A,l} = \max\{h_{A,l}, h_{A,r}\} - h_{A,l} + 1 \geq 1 \Rightarrow h'_{A,l} - h_{A,l} = 1$
    Therefore, $B'_A = B_A + 1 \leq 1$.
    If $B_A = -1$, then the key must be inserted into the left of node A, or it will violate the balance condition.
    Then, $h'_{P,l} = h_{P,l}$, $B'_P = B_P = 1$, which contradicts to what we proved in (a).
    Therefore, $B_A = 0, B'_A = 1$.
Similarly, if the key is inserted into the right of node A, we can get $B_A = 0, B'_A = -1$.

□