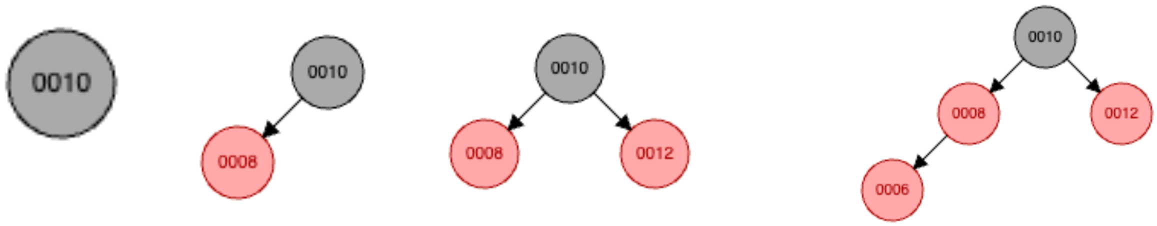
Lab 6

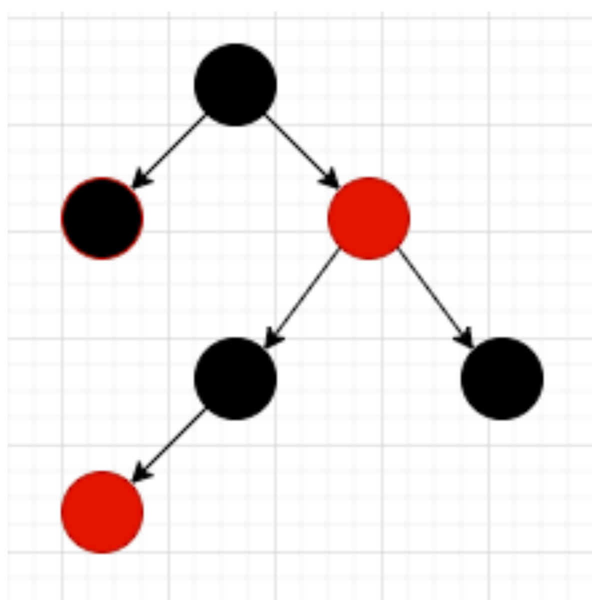
1. A red-black tree is said to be derivable if it is obtained from an insertion sequence of nodes, using the rules for insertions starting from an empty tree. Give an example to show that not every red-black tree is derivable. (In other words, you can build a BST that satisfies the four conditions for a red-black tree, and yet there is no way to obtain this tree by successively inserting nodes using the insertion algorithm rules.)

- Example: 10, 8, 12, 6, 7



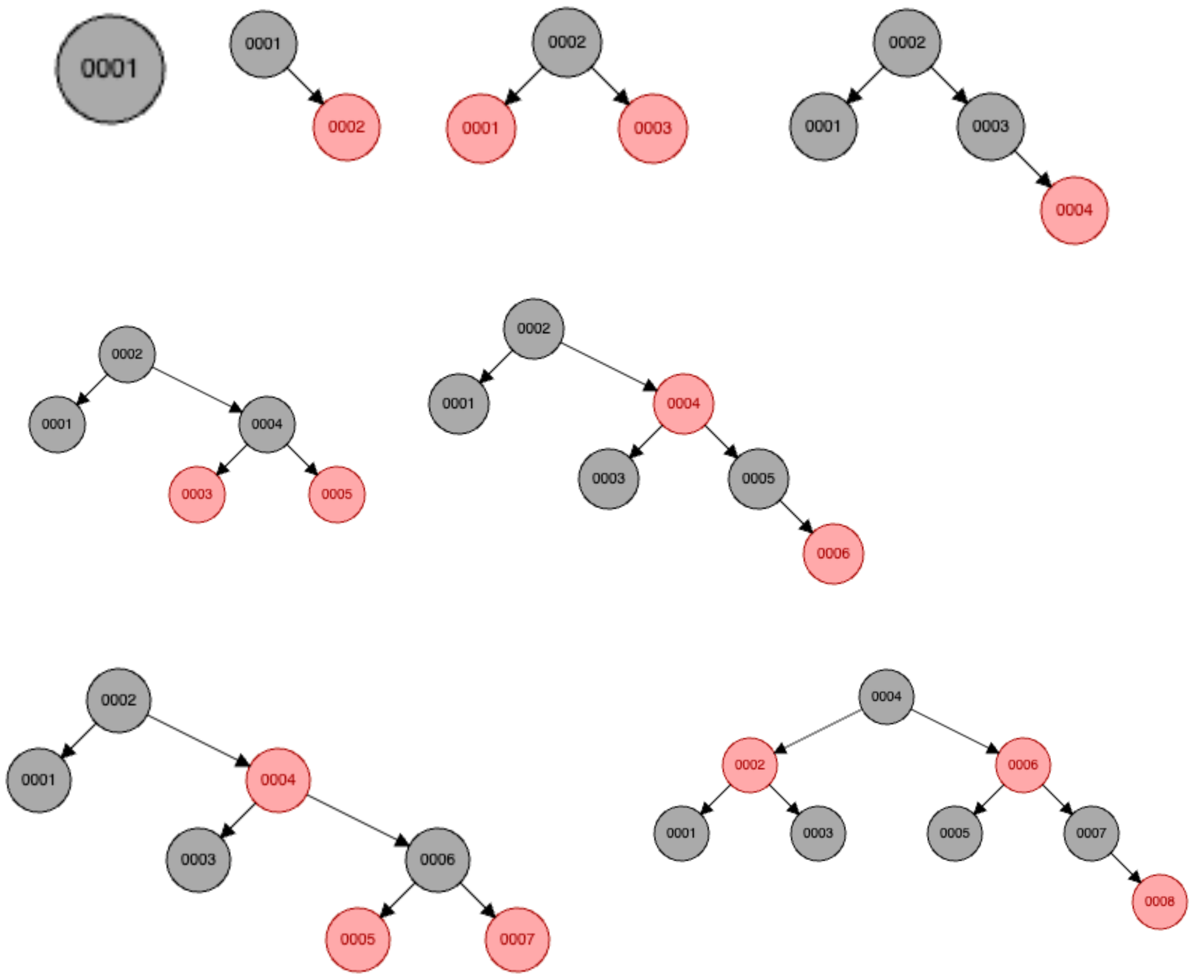
Insertion result in outer grandchild red-red violation
-> change color of P & G, but G need to be Black, cannot
change P to Red

2. An AVL Tree is a BST that satisfies a different balance condition, namely: The AVL Balance Condition For each internal node x, the height of the left child of x differs from the height of the right child of x by at most 1. (Equivalently, the heights of the left and right subtrees of x differ by at most 1.) Create a red-black tree that does not satisfy the AVL Balance Condition.

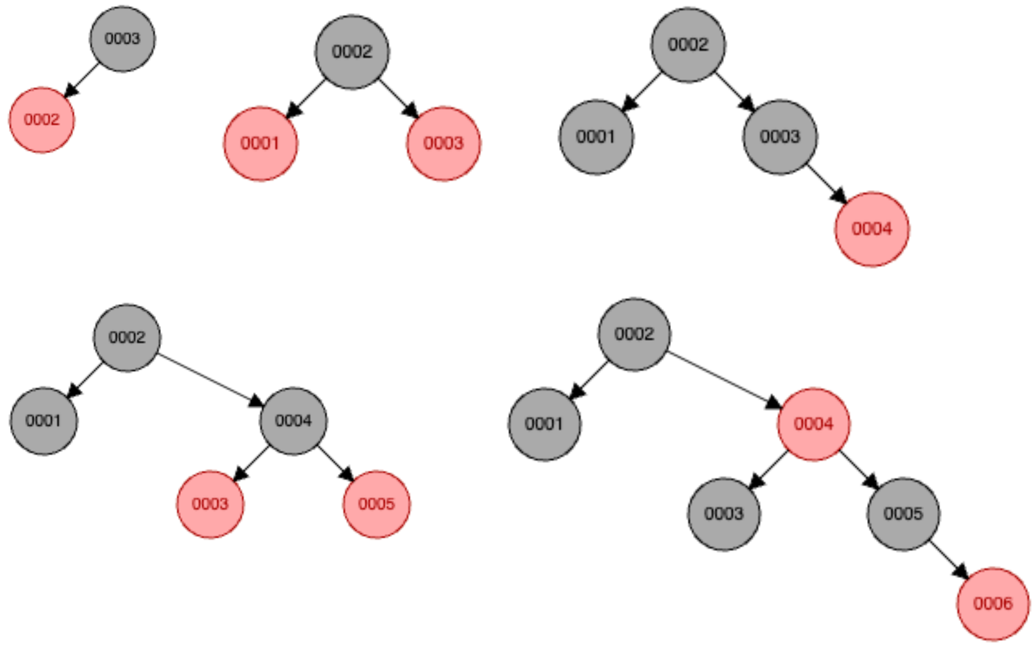


3. Use the insertion algorithm for red-black trees to successively insert the following nodes, starting with an empty tree. a. 1, 2, 3, 4, 5, 6, 7, 8 b. 3, 2, 1, 4, 5, 6 Note on Part (a): Recall that an already sorted insertion sequence is a worst case for an ordinary BST. Notice how the red-black balancing operations handle this to remain balanced.

A.


0001
0001
0002
0002
0001
0003
0002
0001
0003
0004
0002
0001
0004
0003
0005
0002
0001
0004
0003
0005
0006
0002
0001
0004
0003
0006
0005
0007
0004
0002
0006
0001
0003
0005
0007
0008

B.


0003
0002
0002
0001
0003
0002
0001
0003
0004
0002
0001
0004
0003
0005
0002
0001
0004
0003
0005
0006

4. Interview Question. Give a o(n) (“little-oh”) algorithm for determining whether a sorted array A containing n distinct integers contains an element m for which A[m] = m. You must also provide a proof that your algorithm runs in o(n) time

- finding A[m] takes O(logn) time.

$$\lim_{n \to \infty} \frac{logn}{n} = \lim_{n \to \infty} \frac{1}{n^2} loge = 0$$

-> logn is o(n)