
1: Splay Trees

1. True or False:

- (a) Splay trees cannot commonly have duplicate key values.
- (b) Each of the *Find*, *Insert*, and *Delete* operations takes $O(\log n)$ time in a splay tree.
- (c) After searching for a value in a splay tree, searching for it again will restore the original tree shape.
- (d) After every splay operation, the splay tree will be balanced (left and right subtrees of the root differ by at most 1 height).

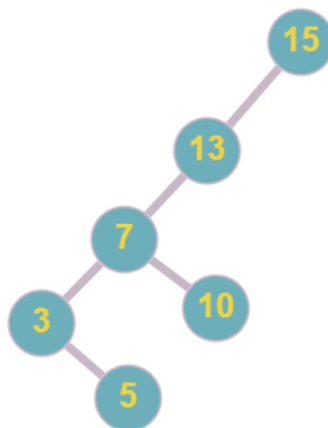
2. Draw the splay tree that would result from inserting the following key values into an empty splay tree in the following order: 7, 8, 2, 4, 3.

Give the pre-order traversal and post-order traversal of nodes in your splay tree.

3. Draw the two splay trees that would result from:

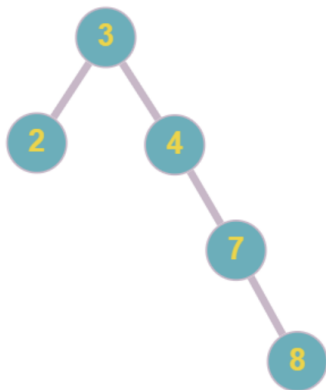
- (a) deleting key 3
- (b) inserting a new key 8 (not deleting key 3)

from the following tree:



Splay Trees

1.
 - (a) True; depending on the implementation splay trees may be altered to handle duplicate keys, but for traditional splay trees attempting to insert a key that is already in the tree will just result in that key being made the root.
 - (b) False; $O(\log n)$ is the amortized time complexity, not the individual operation cost.
 - (c) False; a splay tree's dynamic structure does not guarantee two consecutive search operations will have the same shape.
 - (d) False; unlike other balancing trees (AVL trees), the splay tree is not always balanced across intermediate operations.
2. The splay tree should look similar to the following:



Pre-Order Traversal: 3, 2, 4, 7, 8

Post-Order Traversal: 2, 8, 7, 4, 3

3. The splay trees should look similar to the following:

