Review Sheet 7b

CS 70: Data Structures and Program Development

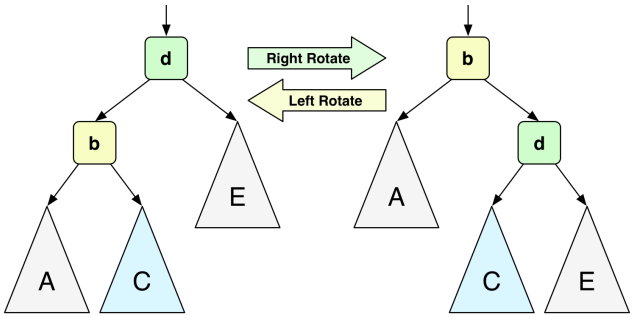
Thursday, March 5, 2020

Learning Targets

1. I can simulate left and right rotations (on paper)
2. I can simulate `insertAtRoot` (on paper)
3. I can simulate Randomized Binary Tree insertion.
4. I can represent a tree using C++ classes and structs.
5. I can implement `insert`, `lookup`, `rotate` in C++.

Review

1. Recall: Tree Rotations



2. Exercises #1 and #2 on exercise sheet

3. BST `insertAtRoot` pseudocode

```
insertAtRoot(tree, x):
   if tree is empty:
      make x its new root.

   else if x < tree's root:
      insertAtRoot(left subtree, x)
      do right rotation at tree's root.

   else if tree's root < x:
      insertAtRoot(right subtree, x)
      do left rotation at tree's root.
```

1. Exercise #3 on sheet

2. BST randomized insert pseudocode

```
randomizedInsert(tree, x):
   if tree currently has n elements,
   with probability 1/(n+1):
      insertAtRoot(tree, x)
   else if x < tree's root:
      randomizedInsert(left subtree, x)
   else if tree's root < x:
      randomizedInsert(right subtree, x)
```

3. Exercise #4 and #5 on sheet

   Do #4 and #5 on exercise sheet.

   Probabilities for insertAtRoot

   - 1/6: if rolled die = 1
   - 1/4: if rolling die twice gives two even numbers
   - 1/2: if rolling die is even
   - 1/1: no need to roll

Representing Trees

1. "A binary tree is empty, or has a root and two (possibly empty) subtrees"

```
class IntTree {
 public: ...
 private:
    // "struct" == ("class" + public by default)
    struct Node {
       int value_;
       Node* left_;
       Node* right_;
    };
    Node* root_;
};
```

2. Insertion Code

1. How would we add "exists"?

2. Rotation Code: