CS2302 - Data Structures

Spring 2020

Lab 4

Due Monday, March 30, 2020

A binary search tree can be implemented as a list of lists T as follows:

- If the tree is empty, T = *None*
- Otherwise T = *[data, left, right]*, where *data* is the item stored in the root node and *left* and *right* are the (possibly empty) left and right subtrees of T, represented in the same way as T.

Thus if T represents a non-empty binary search tree, $T[0]$ is the item stored in the root node of the tree, $T[1]$ is the left subtree, and $T[2]$ is the right subtree. Figures 1 to 4 show example binary search trees and their corresponding list representations. Notice that if T is not empty, *len(T)=3*.

For this lab, you will implement some of the functionality of binary search trees using a list implementation. Your task consists of implementing the following functions to be added to the provided *bst_list* program:

1. *size(T)* - Returns the number of data items in the tree.
2. *minimum(T)* - Returns the smallest item stored in the tree.
3. *maximum(T)* - Returns the largest item stored in the tree.
4. *height(T)* - Returns the height of the tree. Recall that the height is depth of the deepest leaf in the tree. For example, the trees in figures 1 to 4 have heights of 1, 2, 3, and 3.
5. *inTree(T,i)* - Boolean function, returns *True* if item i is in the tree and *False* otherwise.
6. *printByLevel(T)* - Prints the data items in the tree ordered by depth.
7. *tree2List(T)* - Returns a sorted list containing all the items in the tree (this must run in $O(n)$ time).
8. *leaves(T)* - Returns a list of the items in the tree that are stored in leaf nodes.
9. *itemsAtDepthD(T,d)* - Returns a list of the items that are stored at depth d in the tree.
10. *depthOfK(T,k)* - Returns the depth of the node that contains k, or -1 if k is not in the tree.
11. *draw(T)* - Draws the tree.

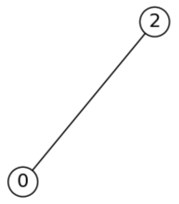

Figure 1: *T=[2, [0, None, None], None]*

As usual, write a report describing your work. Show results of every function with various types of inputs, including an empty tree, a tree with a single node, and balanced and unbalanced tress of different sizes. For every function, write its running time as a comment in the code.

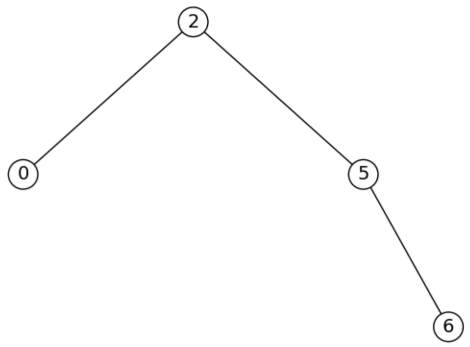

Figure 2: *T=[2, [0, None, None], [5, None, [6, None, None]]]*

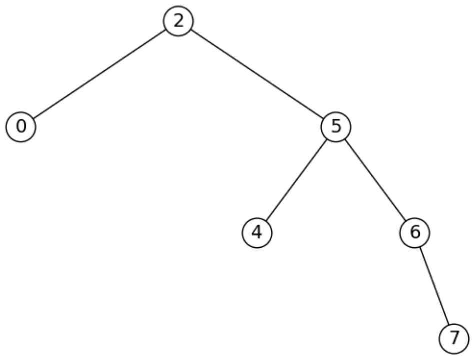

Figure 3: *T=[2, [0, None, None], [5, [4, None, None], [6, None, [7, None, None]]]]*

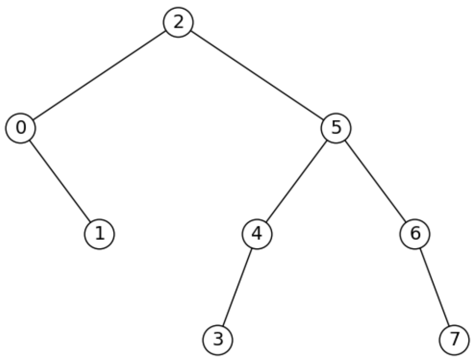

Figure 4: *T=[2, [0, None, [1, None, None]], [5, [4, [3, None, None], None], [6, None, [7, None, None]]]]*