CSCI 2270 - Data structures and algorithms
Instructor: Hoenigman
Assignment 8
Due Wednesday, March 16 by 3pm

Trees, trees, trees

In this assignment, you will be completing three Moodle programming questions on binary search trees and red-black trees as well as a small writing exercise on trees in general. In the writing exercise, you will need to draw on your knowledge of every data structure we have covered this semester to discuss different approaches to implementing a data structure that we haven't covered. You don't need to implement the new data structure, just discuss what would need to be considered in an implementation.

Even though there is no COG for this assignment, the questions will still require time and thought. It is recommended that you start early. There is no option for interview grading if your Moodle programming questions don't run.

Moodle programming questions

There are three questions on Moodle under the heading Assignment 8 Quiz. Those three questions are as follows:

1. Write a function that builds a linked list from all nodes in a binary search tree within a specified value range. The function takes the root of the tree and the lower and upper values of the range as arguments and returns the head of the list. The nodes in the list should be in alphabetical order, A-Z.

2. Write a function to determine if the right and left sub-trees for a particular node are height balanced, which means that the heights differ by no more than 1. The function takes the root of the tree as an argument and returns true if the sub-trees are height balanced and false if they are not.

3. Write a function to apply left or right rotations to a binary search tree based on the height of the left and right sub-trees of the root. The function should first determine if a binary search tree is height balanced, and if not, rotate the tree until it is. Your algorithm may need to apply a left or right rotation multiple times. You will not need to apply a left and right rotation to any tree. The function should return the root of the tree.

Building a tree with an arbitrary number of children

Our discussion of trees has focused almost exclusively on binary trees and binary search trees, where any node in the tree can have at most two children. However, there are other tree structures where the number of children for any node is not restricted. Nodes can have up to *n* children, where *n* is not pre-defined and is

determined by the data being stored. These trees are called *n-ary* trees. A binary tree is an n-ary tree where n = 2.

Consider the following scenario:
A local company has asked you to keep track of the hierarchical structure of the staff at its organization. The structure looks like this:

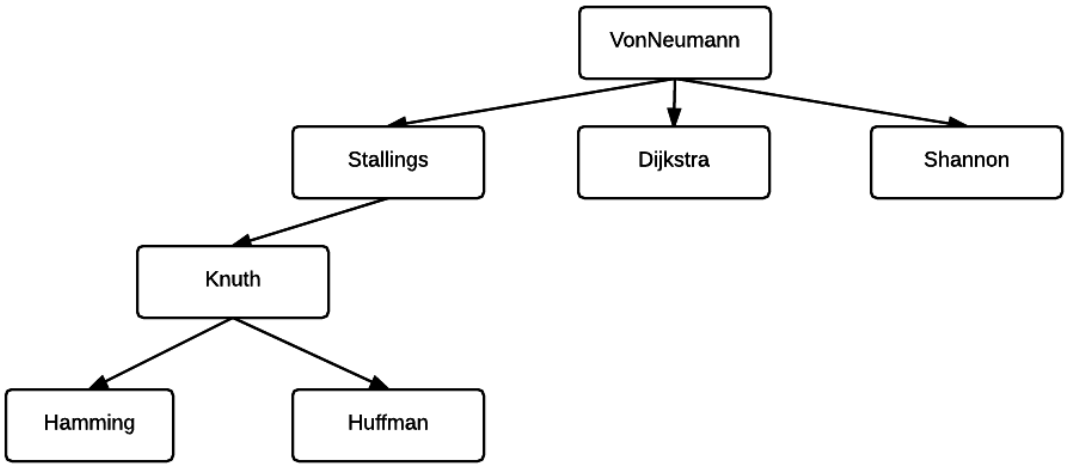

Figure 1. Image adapted from Y. Wu and J. Wang, Data Structure and Practice for Collegiate Programming Contests and Education. Taylor and Francis Group, Boca Raton, FL. 2016.

The company CEO is VonNeumann, shown as the root of the tree. The CEO has three employees who report directly to him - Stallings, Dijkstra, and Shannon. These employees are his children in the tree. Stallings has one direct report - Knuth, and Knuth has two direct reports - Hamming and Huffman. In the organization, each person can have an arbitrary number of employees that report directly to them, and those direct reports appear as children in the tree.

The structure of the organization is not static. New employees are added to the company through hiring and therefore need to be added to the tree. The new employees need to have a manager. Employees are also removed from the company if they resign or are fired, and they need to be removed from the tree. Each of these employees may have direct reports that will need to be reassigned to other managers, or become the manager themselves. The direct reports for any employee are listed by seniority from left to right. For example, Stallings has more seniority than Dijkstra who has more seniority than Shannon. However, none of these employees has more seniority than VonNeumann.

This tree is not a binary tree
The structure shown in Figure 1 is a tree in that there is a hierarchy of parent and child nodes. However, it differs from the binary search trees we have studied because there is no predefined maximum number of children and there is no left and right ordering in relation to the parent.

The operations we have studied on binary search trees - insert, search, delete - use the left and right child properties of the nodes in the tree to perform the operation. For example, to insert a node, we can traverse left or right from a node and compare key values to find the appropriate location for a new node in the tree. We can print all values in the tree by traversing left and then right until all nodes have been visited because we know that left and right are the only options from any node in the tree.

Explain how to build an n-ary tree

For this portion of the assignment, you need to explain how you could build an n-ary tree using the concepts we have learned in the class so far, including arrays, dynamic memory, classes, structs, linked lists, and trees. You do not need to actually implement the tree, and in fact, if you submit an implementation without an explanation, you will receive a zero. It's easy to copy an implementation from the Internet, but that's not the purpose of the assignment and it's cheating.

Your explanation needs to include the following considerations:

1. How is the data stored in this tree as compared to data stored in a binary search tree? Explain why you chose your approach over a different approach by evaluating the properties of the data structures you selected and why those properties are needed for this problem. Also explain why you didn't select other data structures by showing how those features don't fit the requirements of the problem.
2. How do the insert, search, delete operations change for this tree as compared to those operations for a binary search tree? e.g. How would we find a particular employee in this tree and list the employee's direct reports? How do we traverse the tree to show all employees in the company?
3. Any other information you think is relevant to explain your tree.

What to submit

Your answer must be typed, single-spaced, using a 12-point font, and have a minimum length of one page and a maximum length of two pages. Submit a pdf to the Assignment 8 Submit link on Moodle.