CS 2420-001 Algorithms and Data Structures

Fall Semester, 2018

Assignment 3: Binary Search Trees

Due Date: 1:30 p.m., Monday, Oct. 1, 2018 (at the beginning of CS2420 class)

(**Note:** This assignment has three written questions and a programming question. For the programming part, please submit only your java source file on Canvas. For written questions, you may either submit it on Canvas or write on papers and bring it to classroom before/on the due day.)

1. **(15 points)** Consider the binary tree T shown in Fig. 1.

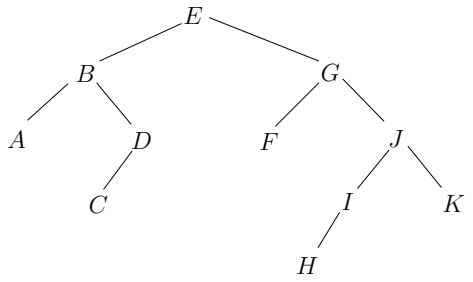

Figure 1: A binary tree T for Question 1

   (a) Recall that the *depth* of a node v is defined to be the number of edges in the path from v to the root of T, and the *height* of a node v is defined to be the number of edges in the path from v to its deepest descendent leaf. The height of the tree is defined to be the height of the root.

   What are the depth and the height of the node G? What is the height of the tree? **(6 points)**

   (b) Give the pre-order, in-order, and post-order traversal lists of T. **(9 points)**

2. **(10 points)** Consider the following binary search tree in Fig. 2. Draw the new tree after remove operation: remove(47). As discussed in class, if v is the node you want to remove and v has two children, then you should use the smallest node in the **right** subtree of v to replace v.

3. **(15 points)** Starting from an empty tree, draw the final binary search tree after the following sequence of operations: insert(20), insert(5), insert(15), insert(30), insert(40), insert(25), insert(10), insert(12), insert(8), insert(28), remove(20), insert(20), remove(5).

   Note: If you want to remove a node v that has two children, then, again, you are required to use the smallest node in the right subtree of v to replace v.

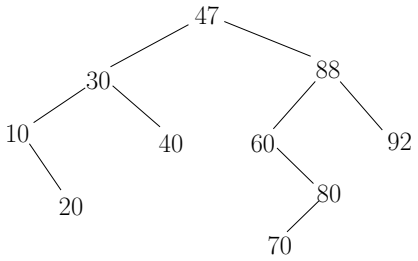

Figure 2: A binary search tree for Question2

4. **(40 points)** In this exercise, we will implement the following operations of binary search trees we discussed in class.

   (a) $insert(x)$: insert a new node with key x to the tree. If there is already a node whose key is x, then we do nothing.

   (b) $remove(x)$: remove the node whose key is x from the tree. If x is not in the tree, then we do nothing.

   Note: As discussed in class, if the node x has two children, then please use the node with the smallest key in the right subtree of x to replace the node x.

   (c) $search(x)$: determine whether the key x is in the tree. If yes, return "true"; otherwise return "false".

   (d) $findMin()$: return the smallest key of the current tree.

   On Canvas, go to the following directory: homework/hw3/question4. There are a starter java file "hw3_Q4.java" and an input file "hw3_Q4_input.txt". Each line of the input file is one of the following types: "insert x", "remove x", "search x", or "findMin". The program reads the input file line by line and perform the operations accordingly. For each search or findMin operation, the result will be output to the console. Finally, the program will print both pre-order and in-order traversal lists of the tree to console (the traversal functions have already been provided).

   You task is to complete the four functions: $insert()$, $remove()$, $search()$, and $findMin()$. For each function, you may write the code using either the iterative or recursive approach we discussed in class. You can also overload these functions, as I did for the traversal functions in the starter java file or as we discussed in class.

   In addition, a file "solution_hw3_Q4_output.txt" with the correct output is given in the same directory.

   Please follow the instructions we used before, e.g., submit your java file to Canvas; do not change the input/output format in your submitted program. If you really do not like my starter code, feel free to change it. You may even start everything from scratch. But please make sure your program accepts my input file and has the similar style of output.

Total Points: 80