Homework #3 –Chapter 15 & 16 Trees

Due Date: Sunday, October 31, 2021, 11:30 PM (Late submissions are not allowed.)

Grades depend on neatness and clarity. Write your answers with enough detail about your approach and concepts used, so that the grader will be able to understand it easily.

1. (100 points) Textbook p 470 – 472, Exercises 1 – 8 & 10 – 11
2. (50 points) Textbook p 510, Exercises 1 – 5

CPSC 250

For example, the person preparing the previous sequence of names to add to the tree might well decide to "help you out" by arranging the names into sorted order. This arrangement, as has been mentioned, would lead to a tree of maximum height. Thus, while in many applications you can expect the behavior of a binary search tree to be excellent, you should be aware of the possibility of poor performance due to some characteristic of a given application.

Is there anything you can do if you suspect that the operations might not occur in a random order? Similarly, is there anything you can do if you have an enormous number of items and need to ensure that the height of the tree is close to $\log_2 n$? Chapter 19 presents variations of the basic binary search tree that are guaranteed always to remain balanced and therefore be of minimum height.

Figure 15-17 summarizes the order of the retrieval, addition, removal, and traversal operations for the ADT binary search tree.

FIGURE 15-17 The Big O for the retrieval, addition, removal, and traversal operations of the ADT binary search tree

Operation	Average case	Worst case
Retrieval	O(log *n*)	O(*n*)
Addition	O(log *n*)	O(*n*)
Removal	O(log *n*)	O(*n*)
Traversal	O(*n*)	O(*n*)

SUMMARY

1. Binary trees provide a hierarchical organization of data, which is important in many application
2. Traversing a tree is a useful operation. Intuitively, traversing a tree means to visit every node in Because the meaning of "visit" is application dependent, you can pass a client-defined `visit` fu the traversal operation.
3. The binary search tree allows you to use a binary search-like algorithm to search for an item with value.
4. Binary search trees come in many shapes. The height of a binary search tree with n nodes can a minimum of $\lceil \log_2(n+1) \rceil$ to a maximum of n. The shape of a binary search tree determi ciency of its operations. The closer a binary search tree is to a balanced tree (and the farther linear structure), the closer the behavior of the `search` algorithm will be to a binary search (ther it will be from the behavior of a linear search).
5. An inorder traversal of a binary search tree visits the tree's nodes in sorted search-key order.

EXERCISES

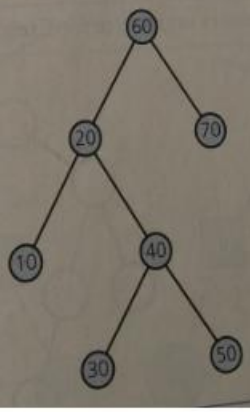
1. Consider the tree in Figure 15-18. What node or nodes are
   a. The tree's root?
   b. Parents?
   c. Children of the parents in part *b*?

d. Siblings?
e. Ancestors of 50?
f. Descendants of 20?
g. Leaves?

2. What is the height of the tree in Figure 15-18?
3. Starting with an empty binary search tree, in what order should you add items to get the binary search tree in Figure 15-18?
4. Using the binary search tree in Figure 15-18, trace the search algorithm when it searches for
   a. 30
   b. 15

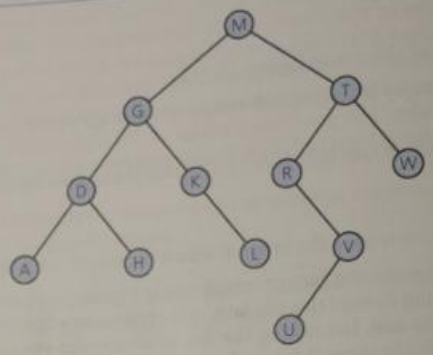
   In each case, list the entries in the order in which the search visits them.
5. Suppose that you traverse the binary search tree in Figure 15-18 and write the data item in each node visited to a file. You plan to read this file later and create a new binary search tree by using the ADT binary search tree operation add. In creating the file, in what order should you traverse the tree so that the new tree will have exactly the same shape and nodes as the original tree?
6. Consider the binary search tree in Figure 15-18. What tree results after you add the entries 80, 65, 75, 45, 35, and 25, in that order?
7. What are the preorder, inorder, and postorder traversals of the binary tree in Figure 15-19?
8. Is the tree in Figure 15-19 a binary search tree? Explain.
9. Write preconditions and postconditions for the ADT binary search tree operations.
10. Beginning with an empty binary search tree, what binary search tree is formed when you add the following values in the order given?
    a. W, T, N, J, E, B, A
    b. W, T, N, A, B, E, J
    c. A, B, W, J, N, T, E
    d. B, T, E, A, N, W, J

FIGURE 15-18 A tree for Exercises 1 through 6



CPSC 250

FIGURE 15-19 A tree for Exercises 7 and 8

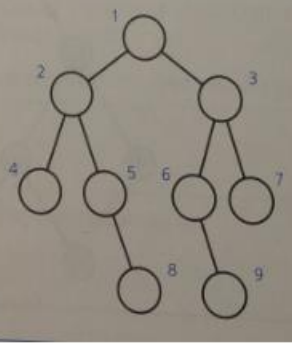


11. Consider the binary search tree in Figure 15-20. The numbers simply label the nodes so that you c reference them; they do not indicate the contents of the nodes.
   a. Without performing an inorder traversal, which node must contain the value that comes imme diately after the value in the root? Explain.
   b. In what order will an inorder traversal visit the nodes of this tree? Indicate this order by listing the labels of the nodes in the order that they are visited.

12. Consider a method `isLeaf` that returns true if a binary tree is a one-node tree—that is, if it consist only a leaf—and returns `false` otherwise.
   a. Specify the method `isLeaf`.
   b. If `isLeaf` were not a method of a class of binary trees, would a client of the class be able to implement `isLeaf`? Explain.

13. If duplicates are allowed in a binary search tree, it is important to have a convention that determine relationship between the duplicates. Items that duplicate the root of a tree should either all be in th subtree or all be in the right subtree, and, of course, this property must hold for every subtree. Why i convention critical to the effective use of the binary search tree?

FIGURE 15-20 A binary search tree for Exercise 11

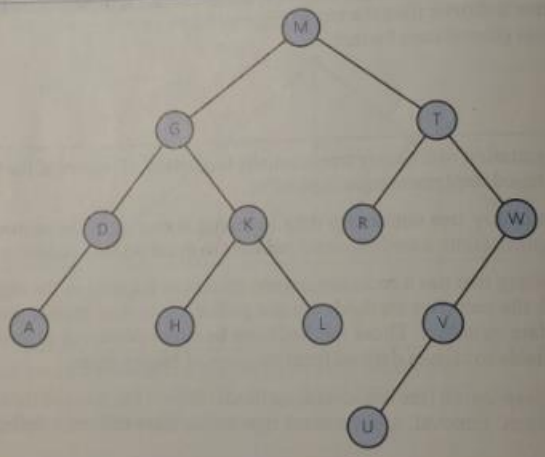


CPSC 250

EXERCISES

1. Consider the binary search tree in Figure 15-20 of Chapter 15. The numbers simply label the nodes so that you can reference them; they do not indicate the contents of the nodes.
   a. Which node must contain the inorder successor of the value in the root? Explain.
   b. Which node must contain the inorder predecessor of the value in the root? Explain.
2. Arrange nodes that contain the letters *A*, *C*, *E*, *F*, *L*, *V*, and *Z* into two binary search trees: one that has maximum height and one that has minimum height.
3. Consider the binary search tree in Figure 15-18a of Chapter 15.
   a. What tree results after you add the entries 80, 65, 75, 45, 35, and 25, in that order?
   b. After you add the nodes mentioned in part *a*, what tree results when you remove the entries 35 and 20?
4. Consider the binary search tree in Figure 16-21. What does the tree look like after you remove *M*, *D*, and *T*, in that order?

FIGURE 16-21 A binary search tree for Exercise 4



5. If you remove a data item from a binary search tree and then add it back to the tree, will you ever change the shape of the tree?
6. Suppose that the ADT binary tree has the operation

```
replace(item: ItemType, replacementItem: ItemType): boolean
```

It locates, if possible, the node in a binary tree that contains `item` and replaces `item` with `replacementItem`.

   a. Add the operation `replace` to the link-based implementation of the ADT binary tree given in this chapter. The operation should replace an item without altering the tree structure.
   b. Add the operation `replace` to the link-based implementation of the ADT binary search tree. Be sure that the tree remains a binary search tree. Do not use any public methods of the binary search tree.
   c. Implement a method `replace` within a client of `BinarySearchTree`.