CSCI-201: Principles of Software Development

Spring 2022

Lab03: Binary Search Trees

Introduction:

Binary Search Trees (BST) are a very powerful (but not perfect) data structure to that every developer should have in their programming tool belt.

Binary search trees are ordered (each node has a Comparable key) — the left child is always smaller than the parent node and the right child is always larger than the parent node. Given that we are able to traverse our BST in a sorted order. When inserting or searching for an element in a BST, the key of each visited node has to be compared with the key of the element to be inserted or found. A BST does not support duplicates.

Implementation:

We define an inner private class to define nodes in BST. Each node contains a key, a left link, and a right link. The left link points to a BST for items with smaller keys, and the right link points to a BST for items with larger keys.

Search. A recursive algorithm to search for a key in a BST. If the tree is empty, we return false. if the search key is equal to the key at the root, we return true. Otherwise, we search (recursively) in the appropriate subtree. We recommend a recursive implementation.

Insert. A recursive algorithm to insert a key in a BST. The implementation is similar to search. Again, we recommend a recursive implementation.

Clone. A recursive algorithm for cloning a BST. The clone is shallow – it does not clone internal data. Please consult lecture 4 for cloning. Try to implement cloning recursively.

Iterator. An iterative implementation (this is a requirement-!) of the in-order traversal algorithm. The iterator is implemented as a private inner class. Please consult lecture 4 for the iterator details. Hint: you will need a stack to support an iterative implementation.

Testing:

Expected output (you must match it exactly):

```
1,2,3,4,5,6,8,9,201,
1,2,3,4,5,6,8,9,
the tree height is 5
true
false
the dictionary size is 175538
the dictionary height is 39
```

Grading Criteria:

Labs are graded based on your understanding of the course material. To receive full credit, you will need to:

1) complete the lab following the instructions above
2) show your understanding of the lab material by answering questions upon check-off

Instructors, to ensure consistency across all lab sections, please strictly stick to the following criteria:

Binary Search Tree: **(9 Points)**

a) Iterator is complete and working (3 pts)
b) Clone is complete and working (3 pts)
c) Insert and search complete and working (3 pts)

Check-off Questions: **(1 Point)**

Please randomly select one question.

Question 1: What is the worst runtime of inserting an element to a BST?
Question 2: What is the worst runtime of inserting an element to a balanced BST?
Question 3: Explain the difference between shallow copy and deep copy.
Question 4: What are generics and why they are important?
Question 5: What is an iterator and why are iterators necessary?