CS 209
Data Structures and Mathematical Foundations

03 / 04 / 2024

Instructor: Michael Eckmann

Today's Topics

- Questions/Comments?
- Binary Search Trees
  - implementation

Implementation of binary trees

- Similar to how we implemented a linked list of nodes, we can implement a binary tree of nodes where each node contains a reference to a left child and a right child (each of which could be None.)

```
class BTNode:
        # instance variables
        # data – the data in the BTNode
        # left – the BTNode to the left
        # right – the BTNode to the right

        # constructor that sets left and right to None
        def __init__(self, d):
            self.data = d
            self.left = None
            self.right = None
```

Implementation of binary trees

```
class BinaryTree:
        # instance variable
        # root – a BTNode that is the root of the tree

        # constructor that sets root to None
        def __init__(self):

            self.root = None


        # plenty more methods here to insert nodes, etc...
```

Binary Search Trees

- Binary Search Trees (BSTs)
  - A *tree* that is both a *binary tree* and a *search tree*.
  - we know the definition of a tree and a binary tree so:
  - We just need to define search tree.
    - A *search tree* is a tree where
      - every subtree of a node has data (aka keys) less than any other subtree of the node to its right.
      - the keys in a node are conceptually between subtrees and are greater than any keys in subtrees to its left and less than or equal to any keys in subtrees to its right.

Binary Search Trees

- A *binary search tree* is a tree that is a binary tree and is a search tree. In other words it is a tree that has
  - at most two children for each node and
  - every node's left subtree has keys less than the node's key, and every right subtree has keys greater than or equal to the node's key.

  - Definitions taken from http://www.nist.gov/ (The National Institute of Standards and Technology)

A few operations on binary trees

- getLeftmost node
  - Starting at root, follow the left until you hit a node whose left is None. That node is the leftmost node.
- getRightmost node
  - Starting at root, follow the right until you hit a node whose right is None. That node is the rightmost node.
- According to these definitions, will the leftmost and rightmost nodes always be leaves?

A few operations on binary trees

- The leftmost node can have a right child and the rightmost node can have a left child, so the leftmost and rightmost nodes in a binary tree aren't necessarily leaves.

- What can you say about the data that will be in the leftmost node in a binary search tree?
- What about the data in the rightmost node of a binary search tree?

Binary Search Tree implementation

- Implement code for a Binary Search Tree using Nodes
  - Insert
  - Get leftmost
  - Get rightmost