Binary Search Trees

Depth First Traversal

The `BinarySearchTree` Python class has a `.depth_first_traversal()` instance method that prints the in-order depth-first traversal of the tree. The output will always be in ascending order. It takes no variables and returns nothing.

```
  def depth_first_traversal(self):
    if (self.left is not None):
      self.left.depth_first_traversal()
    print(f'Depth={self.depth}, Value=
{self.value}')
    if (self.right is not None):
      self.right.depth_first_traversal()
```

Getting a Node by Value

The `BinarySearchTree` Python class has a `.get_node_by_value()` instance method that takes in a `value` and returns the corresponding `BinarySearchTree` node, or `None` if it doesn't exist. The method uses recursion to search through the sides of the tree. On an averagely balanced binary search tree with `N` nodes, the performance would be `O(logN)`, just like the Binary Search algorithm.

```
  def get_node_by_value(self, value):
    if (self.value == value):
      return self
    elif ((self.left is not None) and (value <
self.value)):
      return self.left.get_node_by_value(value)
    elif ((self.right is not None) and (value >=
self.value)):
      return self.right.get_node_by_value(value)
    else:
      return None
```

Insertion

The `BinarySearchTree` Python class has an `.insert()` method that takes in a `value` and uses recursion to add a new node to the tree while maintaining the binary tree property. The method returns nothing. On an averagely balanced binary search tree with `N` nodes, the performance would be `O(logN)`.

```
  def insert(self, value):
    if (value < self.value):
      if (self.left is None):
        self.left = BinarySearchTree(value,
self.depth + 1)
        print(f'Tree node {value} added to the left
of {self.value} at depth {self.depth + 1}')
      else:
        self.left.insert(value)
    else:
      if (self.right is None):
        self.right = BinarySearchTree(value,
self.depth + 1)
        print(f'Tree node {value} added to the
right of {self.value} at depth {self.depth + 1}')
      else:
        self.right.insert(value)
```

Constructor

The Python implementation of the `BinarySearchTree` class should contain `value` and `depth` instance variables, as well as `left` and `right` pointers. The constructor has the following parameters:

- `value`
- `depth`, which has a default value of `1`

The `left` and `right` pointers are set to `None` in the constructor.

```
def __init__(self, value, depth=1):
  self.value = value
  self.depth = depth
  self.left = None
  self.right = None
```

← Previous

Next →

Related Courses