Universidade Federal de Ouro Preto
Lecture Notes
Graphs

Prof. Rodrigo Silva

April 12, 2023

1 Binary Search Trees

1.1 Python

```python
class Node:
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

class BST:
    def __init__(self):
        self.root = None

    def insert(self, data):
        if not self.root:
            self.root = Node(data)
        else:
            self._insert(data, self.root)

    def _insert(self, data, current_node):
        if data < current_node.data:
            if not current_node.left:
                current_node.left = Node(data)
            else:
                self._insert(data, current_node.left)
        elif data > current_node.data:
            if not current_node.right:
                current_node.right = Node(data)
            else:
                self._insert(data, current_node.right)
        else:
            print("Value already in tree.")

    def search(self, data):
        if self.root:
            found = self._search(data, self.root)
            if found:
                return True
            return False
        else:
            return None

    def _search(self, data, current_node):
        if data == current_node.data:
            return True
        elif data < current_node.data and current_node.left:
            return self._search(data, current_node.left)
        elif data > current_node.data and current_node.right:
            return self._search(data, current_node.right)
        return False
```

```
48
49      def inorder(self, node):
50          if node is not None:
51              self.inorder(node.left)
52              print(node.data)
53              self.inorder(node.right)
```

Listing 1: Binary Search Tree in Python

1.2 C++

```
1  class Node:
2      def __init__(self, data):
3          self.data = data
4          self.left = None
5          self.right = None
6
7  class BST:
8      def __init__(self):
9          self.root = None
10
11     def insert(self, data):
12         if not self.root:
13             self.root = Node(data)
14         else:
15             self._insert(data, self.root)
16
17     def _insert(self, data, current_node):
18         if data < current_node.data:
19             if not current_node.left:
20                 current_node.left = Node(data)
21             else:
22                 self._insert(data, current_node.left)
23         elif data > current_node.data:
24             if not current_node.right:
25                 current_node.right = Node(data)
26             else:
27                 self._insert(data, current_node.right)
28         else:
29             print("Value already in tree.")
30
31     def search(self, data):
32         if self.root:
33             found = self._search(data, self.root)
34             if found:
35                 return True
36             return False
37         else:
38             return None
39
40     def _search(self, data, current_node):
41         if data == current_node.data:
42             return True
43         elif data < current_node.data and current_node.left:
44             return self._search(data, current_node.left)
45         elif data > current_node.data and current_node.right:
46             return self._search(data, current_node.right)
47         return False
48
49     def inorder(self, node):
50         if node is not None:
51             self.inorder(node.left)
52             print(node.data)
```

```
53                 self.inorder(node.right)
```

Listing 2: Binary Search Tree in Python