Algorithms and Data Structures for Data Science
Trees 3

CS 277
Brad Solomon

October 18, 2021
Informal Early Feedback

The instructor is well-prepared for each class / recording

I feel that I can actively participate in lecture.

I feel that I can actively participate in class in general.

I receive helpful and complete answers to my questions.
Informal Early Feedback

How helpful have the lectures been for learning material?

How helpful have the assignments been for learning material?
Informal Early Feedback

Too fast

Just right

Too slow
What can make CS 277 better?

“requirements and prerequisites for the class can be defined better for a more united audience”

“Share some links at the end of the lecture so that we can explore topics in more detail”
Final Project Proposal Due Oct 20th

Proposal Deadline: October 20th

Submission: Commit proposal to GitHub.

final_project/development/proposal.

Reminder: Next mp not released until October 20th!
Exam 2 October 29th - October 31st

Exam will be combination multiple choice and coding

Covers all content up until October 20th

Sign up for registration as early as October 14th

Coding question will use new workspace setup.
Learning Objectives

- Review binary search trees (and dictionaries)
- Conceptualize BST insertion
- Implement BST remove
Binary Search Tree (BST)

A BST is a binary tree $T = \text{treeNode}(\text{val}, T_L, T_r)$ such that:

$\forall n \in T_L, \ n.\text{val} < T.\text{val}$

$\forall n \in T_R, \ n.\text{val} > T.\text{val}$
class bstNode:
    def __init__(self, key, val, left=None, right=None):
        self.key = key
        self.val = val
        self.left = left
        self.right = right

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

```python
def find(self, key):
    n = self.find_helper(self.root, key)
    if n:
        return n.val
    else:
        return None

def find_helper(self, node, key):
    nkey = node.key
    if nkey > key:
        if node.left:
            return self.find_helper(node.left, key)
        else:
            return None
    elif nkey < key:
        if node.right:
            return self.find_helper(node.right, key)
        else:
            return None
    else:
        return node
```

![BST Diagram](image)

- Node 5 has children nodes 3 and 6.
- Node 3 has children nodes 1 and 4.
- Node 6 has children nodes 4 and 7.
- Node 1 is the root node.
Dictionary ADT

Data is often organized into key/value pairs:

Word ➔ Definition
Course Number ➔ Lecture/Lab Schedule
Node ➔ Edges
Flight Number ➔ Arrival Information
URL ➔ HTML Page
<table>
<thead>
<tr>
<th>geneID</th>
<th>length</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>367</td>
<td>529</td>
<td>4.5</td>
</tr>
<tr>
<td>163</td>
<td>785</td>
<td>4.8</td>
</tr>
<tr>
<td>18</td>
<td>428</td>
<td>6.6</td>
</tr>
<tr>
<td>737</td>
<td>799</td>
<td>1.6</td>
</tr>
<tr>
<td>502</td>
<td>278</td>
<td>4.9</td>
</tr>
<tr>
<td>941</td>
<td>484</td>
<td>47.7</td>
</tr>
<tr>
<td>219</td>
<td>456</td>
<td>3.0</td>
</tr>
<tr>
<td>491</td>
<td>473</td>
<td>4.6</td>
</tr>
<tr>
<td>953</td>
<td>704</td>
<td>30.6</td>
</tr>
<tr>
<td>739</td>
<td>819</td>
<td>30.2</td>
</tr>
<tr>
<td>361</td>
<td>298</td>
<td>82.2</td>
</tr>
<tr>
<td>524</td>
<td>736</td>
<td>6.8</td>
</tr>
<tr>
<td>702</td>
<td>570</td>
<td>66.9</td>
</tr>
<tr>
<td>466</td>
<td>414</td>
<td>9.9</td>
</tr>
<tr>
<td>820</td>
<td>383</td>
<td>53.8</td>
</tr>
<tr>
<td>546</td>
<td>588</td>
<td>33.0</td>
</tr>
<tr>
<td>643</td>
<td>961</td>
<td>3.8</td>
</tr>
<tr>
<td>59</td>
<td>248</td>
<td>6.7</td>
</tr>
</tbody>
</table>
Dictionaries in Python

```python
pyDictionary = {}
pyDictionary[3] = "B"
pyDictionary[6] = "C"
pyDictionary[1] = "D"

if 6 in pyDictionary:
    print(pyDictionary[6])
if 8 in pyDictionary:
    print(pyDictionary[8])
```
Dictionaries in Python

```python
bt = bst()
bst.insert(5, "A")
bst.insert(3, "B")
bst.insert(6, "C")
bst.insert(1, "D")
bst.insert(4, "E")
bst.find(6)
bst.find(8)
```
BST Insert

insert(33)
BST Insert

Base Case

Recursive Step

Combining
BST Insert

What binary would be formed by inserting the following sequence of integers: $[3,\ 7,\ 2,\ 1,\ 4,\ 8,\ 0]$
BST Remove

```
remove(40)
```
BST Remove

remove(25)
BST Remove

remove(13)
def findIOP(self, node):
    if node.left == None:
        return None
    curr = node.left
    while curr.right:
        curr = curr.right
    return curr
BST Remove

remove(51)
In-Order Successor

def findIOS(self, node):
    if __________ == None:
        return None
    curr = ______________
    while ________________:
        ________________
    return curr
def remove_helper(self, node, key):
    if node == None:
        return None
    if node.key > key:
        node.left = self.remove_helper(node.left, key)
    elif node.key < key:
        node.right = self.remove_helper(node.right, key)
    else:
        if node.left == None:
            temp = node.right
            return temp
        if node.right == None:
            temp = node.left
            return temp
        iop = self.findIOP(node)
        node.key = iop.key
        node.val = iop.val
        node.left = self.remove_helper(node.left, iop.key)
    return node

def remove(self, key):
    self.root = self.remove_helper(self.root, key)
BST Remove

What will the tree structure look like if we remove node 16 using IOS?
## BST Analysis – Running Time

<table>
<thead>
<tr>
<th>Operation</th>
<th>BST Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>find</td>
<td></td>
</tr>
<tr>
<td>insert</td>
<td></td>
</tr>
<tr>
<td>delete</td>
<td></td>
</tr>
<tr>
<td>traverse</td>
<td></td>
</tr>
</tbody>
</table>