12.1 - Week 12 - Applied - Theory

Objectives of this Applied Session

Objectives of this Applied Session

- To understand Binary Tress and Binary Search Trees
- To be able to analyse their complexity
- To understand why balancing a binary tree is useful

Some Binary Tree Basics

Let's double check your knowledge of Binary Trees with the following questions:

Question 1 *Submitted Oct 18th 2022 at 8:11:30 am*

What is the maximum possible height of a Binary Tree with N nodes?

What would this tree look like?

N-1

Question 2 *Submitted Oct 18th 2022 at 8:12:46 am*

For the following expression tree, write out the postorder traversal, inorder traversal, and then write the expression in unambiguous mathematical notation:

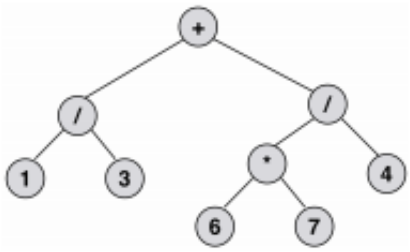


Figure 1: Expression tree

Postorder --> `13/67*4/+`

Inorder --> `1/3+6*7/4`

Mathematical --> `(1/3)+((6*7)/4)`

Height Calculation

Consider the below Binary Tree implementation to which we want to add a method:

```
class TreeNode(Generic[K, I]):
    def __init__(self , key: K, item: I = None) -> None:
        self.key = key
        self.item = item
        self.left = None
        self.right = None

class BinaryTree(Generic[K, I]):
    def __init__(self) -> None:
        self.root = None
```

`get_height(self) -> int` which returns the height of the binary tree.

Hint:

▶ Expand

More Recursion

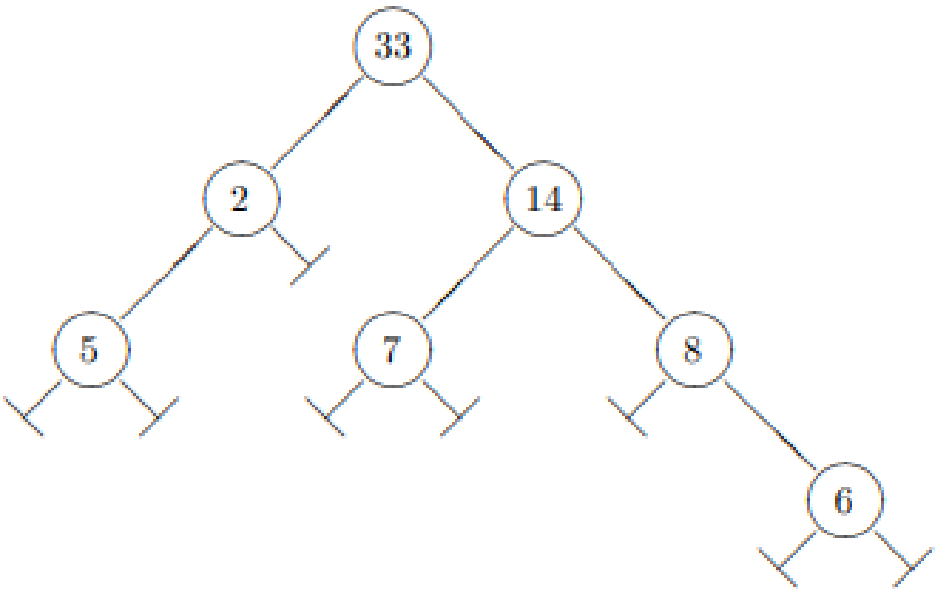
Yet again, with the same scaffold:

```
class TreeNode(Generic[K, I]):
    def __init__(self , key: K, item: I = None) -> None:
        self.key = key
        self.item = item
        self.left = None
        self.right = None

class BinaryTree(Generic[K, I]):
    def __init__(self) -> None:
        self.root = None
```

We want to add a new method to the `BinaryTree`, `sum_leaves(self) -> I`, which returns the total sum of leaf item values in the tree.

For example, in the tree below:



`tree.sum_leaves()` should return `5+7+6=18`.

Hint:

▶ Expand

Complexity

Now that we've written some code let's analyse its complexity.

Question 1 *Submitted Oct 18th 2022 at 8:50:58 am*

What is the complexity of your previous two functions. Why?

O(N)

Question 2 *Submitted Oct 18th 2022 at 8:51:26 am*

Bonus: How much space does these programs use? It is not **O(1)**.

Hint:

▶ Expand

O(N)

Time for "The Dark Souls" of Binary Trees

This meme could go on for a few more frames, but we don't want to scare you too much :)

Fix our buggy code

We want to extend the **BinarySearchTree** class defined in the lectures by adding a method **find_min**, which returns the minimum key in the tree, or None if the tree is empty. In doing so, it does not modify the tree. After trying twice, we've given up:

```
def find_min_1(self) -> K:
    return self.find_min_aux_1(self.root)

def find_min_aux_1(self, current: BinarySearchTreeNode[K, I]) -> K :
    if current is not None:
        return self.find_min_aux_1(current.left)
    else:
        return current

def find_min_2(self) -> K:
    if self.root is None:
        return self.root
    else:
        return self.find_min_aux_2(self.root)

def find_min_aux_2(self, current: BinarySearchTreeNode[K, I]) -> K:
    if current.left is not None:
        return current
    else:
        return self.find_min_aux_2(current.left)
```

Without executing the code, spot what errors the two solutions had, and implement your own in the scaffold across:

Complex Recursive Calls

Consider a usual BST.

In `Ex5.py`, create a function `sorted_splice(self, a: K, b:K) -> list[K]` that returns a sorted list with all keys between a and b. The idea is to do this without visiting all elements, if possible. Analyse the complexity of your approach.

Motivating AVL Trees

For your assessments you'll be tackling the concept of AVL Trees. Let's play around with some of the concepts regarding AVL Trees.

Question 1 *Submitted Oct 18th 2022 at 9:46:08 am*

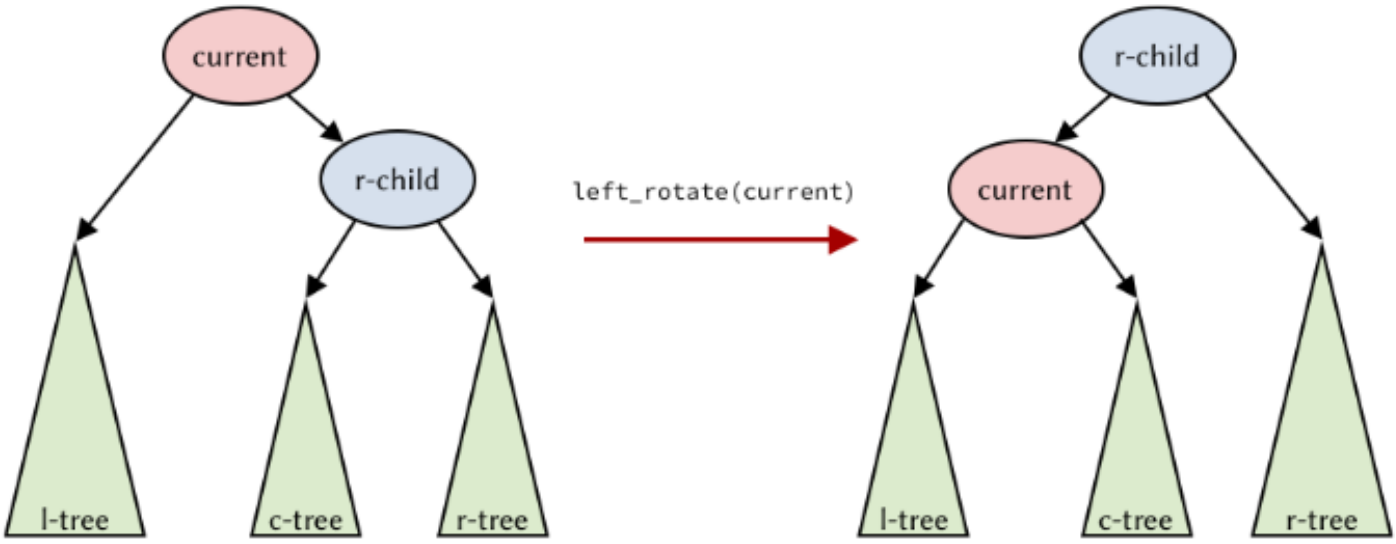
Would our code solutions from before have better complexities if the height of the tree was bounded by something smaller than **N**?

Yes

Question 2 *Submitted Oct 18th 2022 at 9:47:42 am*

Suppose we have a Binary Search Tree, and we apply a left rotation.

Explain, in your own words, why the tree still satisfies the main invariant of Binary Search Trees.



Be sure to mention the invariant, applied at both `r-child` and `current`.

:D

Question 3 *Submitted Oct 18th 2022 at 9:49:55 am*

Bonus. Prove/Convince yourself that if every subtree has balance factor -1, 0 or 1, then every subtree has at least $\mathbf{2^{h/2}}$ nodes, where **h** is the height of the subtree.

Hint:

▶ Expand

No response

Congrats on making it to the end!

Hope everyone is enjoying Assignment 3 so far 😵