Homework 5

Due Friday, June 18 at 11:59 PM ET

Fun Sea Creature Fact: Sea otters have secret pockets near their armpits to store food!

Installing and Handing In

Accept the GitHub assignment **here**. Homework is handed in through Gradescope. For written portion, submit a PDF and match the questions accordingly. For the coding portion, submit ALL of the .py files.

1 Written Problems

Problem 5.1

Binary perfection

Purpose: Understanding tree structure and reviewing induction.
Resources: Induction lecture (May 25) and Tree lectures (June 3, June 8)

In class, we proved that the total number of *nodes* in a perfect binary tree of height h is $2^{h+1} - 1$. For this problem, you are going to prove that a perfect binary tree has a certain number of *leaves*.

Our recursive definition of a perfect binary tree is a binary tree T where one of the following two properties holds:

1. The root of T is a leaf, meaning the tree only has one node
2. T is considered to have height $h + 1$. The root of T has left and right subtrees, and each subtree is a perfect binary tree of height h.

Now, **prove** by induction that a perfect binary tree of height h has 2^h leaves, for all $h \geq 0$.

Note: In your proof, you may only make assumptions based on the definition provided above. There is one exception to this: you can assume that a tree of height zero has one node. Other assumptions, including the ones made in class, will not earn you full credit.

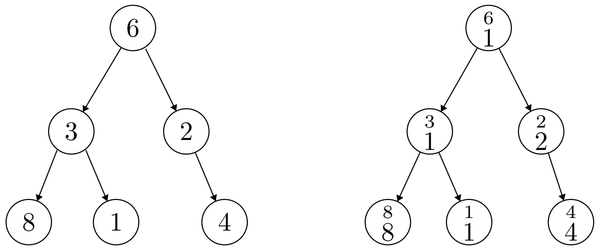
Problem 5.2

Learning from our children

Purpose: More practice with trees and decorations, which will be useful for heap.
Resources: Tree lecture (June 3), Heap lecture (June 8), and BST lecture (June 10).

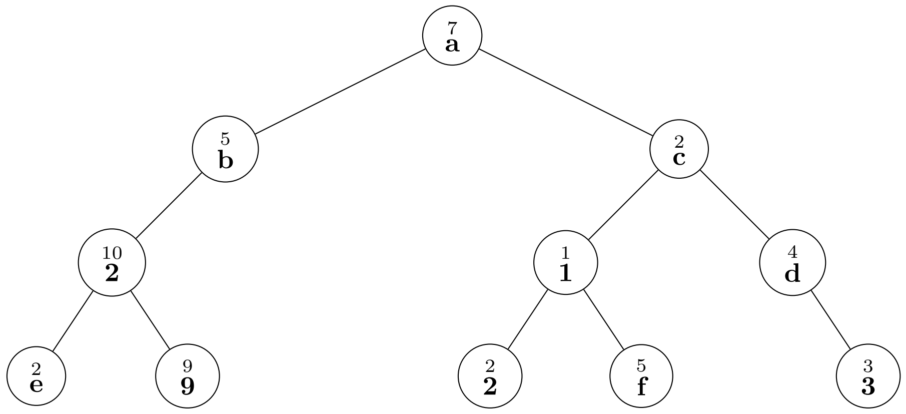
Consider an n-node binary tree T where each node N has an associated integer value, $N.u$. You will design an algorithm that decorates each node in T with another value, $N.z$. The value $N.z$ is the minimum of the u-values among all of N's descendants, where the node N is counted as one of its own descendants[1]. Thus for a leaf (which has only one descendant, itself), $N.z = N.u$. For the root, $N.z$ is the minimum of all the u-values in the whole tree.

A small before-and-after example is shown below. On the left is the original tree. On the right is the decorated tree, with u-values on the top and z-values on the bottom.



(a) The below figure shows a tree in which each node is labeled with its u-value. We've filled in a couple of z-values for you, in bold below the node's u-value. For each letter in the tree below, give the corresponding z-value.

[1]We generally don't count a node as one of its own descendants, but sometimes (as in this problem) it is convenient to do so.



(b) Write pseudocode for an algorithm to **decorate** each node in the tree with its z-value. Assume that each node has variables `u` and `z` that you can access by calling $N.z$ or $N.u$ and change as necessary by setting $N.z =$ an integer or $N.u =$ an integer. After the algorithm is run, every node's `z` value should have an integer assigned to it. Your algorithm should run in $O(n)$ time, where n is the number of nodes in the tree.

2 Python Problems

Problem 5.3

Binary Tree

Purpose: Code practice with Binary Trees.
Resources: Tree lectures (June 3 and June 8)

Overview

Fill in all of the methods in `bintree.py` to implement the linked binary tree data structure.

Input/Output

The valid input and output values for each function are specified in the `bintree.py` stencil. This stencil includes which exceptions should be thrown when.

You can assume your function will be tested on anything that fits within the stencil's definition for Input, but will **not** be tested on inputs that don't fit within the input definition.

Details

- Each method should meet the specifications outlined in the stencil code comments.

- Each of the methods you are writing should run in $O(1)$ worst-case time.
- The stencil contains 4 classes:
  1. `EmptyBinTreeException`: This class defines an exception that you should throw when methods are called on an empty binary tree. The docstrings provided in the stencil should indicate when this exception should be thrown.
  2. `InvalidInputException`: This class defines an exception that you should throw when methods are called with invalid parameters. The docstrings provided in the stencil should indicate when this exception should be thrown.
  3. `Node`: This class defines a `Node`. Your tree will contain many of these. Note that a Node can have a *value* of None. You are responsible for filling out any method in this class marked with `TODO`. *It is very important that you read the note below regarding the* `self` *argument passed to each method in this class.*
  4. `BinTree`: This class defines the Linked Binary Tree data structure. You are responsible for filling out any method in this class marked with `TODO`. *It is very important that you read the note below regarding the* `self` *argument passed to each method in this class.*
     – `height()` is one of the methods contained within the `BinTree` class. Remember that the height of an empty tree is undefined and the height of a one-node tree is zero.
     – You can assume all nodes passed into any of `BinTree`'s functions will be part of an existing tree.
- Do not name variables the same name as a method declaration. You will get a TypeError saying that the "object is not callable."
- **IMPORTANT note about the `self` argument:** Remember that when doing object-oriented programming in Python, there's an obligatory `self` argument for every function. This argument can be used to access the internal attributes of a particular class, since these variables cannot be accessed directly. We do this from within a method by calling `self.<attribute>` (without the angle brackets). Also remember, however, that the `self` argument is only used when *defining* a method (`def func(self, my_args)`). When *calling* the method, Python passes this argument for you, and you should leave it out (`func(my_args)`).
- We have written a `bt.popup()` function you can call to see a visualization of your tree! Using this isn't required, but if you do want to utilize it you must install graphviz (a helpful library for visualizing graphs and trees). The visualizer will only function properly on `bt.popup()` calls if each vertex in the binary tree has a unique name. Remember to take out any calls to `bt.popup()` in your tests when handing in.

- **Attribute Error** In python an attribute can be thought of as the actions objects can perform. For example, you can append to a variable of type list, but you cannot append to a variable of type int (if you try, you will get an Attribute Error).

  These errors will likely be of the form `Attribute Error:"CLASS X" object has no attribute of type "FUNCTION Y"` For example, you may get the attribute error saying `"str" object has no attribute "hasLeft"`. This means you created a string, let's call it `x`, and then called `x.hasLeft()`. However, strings don't have the attribute `hasLeft`, so `x` does not have the attribute `hasLeft()`. Instead `hasLeft()` can only be called on variables of type `Node`. In this case, you should think about if you initialized `x` to be a node and not just a value (which are strings)?

  The general approach to these errors is first understanding that you are calling a function on an object that doesn't have access to that function. Most likely your variable is initialized to the wrong type, and you should make sure you're initializing correctly (creating a Node not a string, or a Node not a tree, etc.)

Testing

You should write all of your test functions in the provided file `bt_test.py`. Don't forget to comment each test so it's clear what you're testing for.

3 Socially Responsible Computing Response

(Optional, for an extra late pass)

Problem 5.4

We've written up four responses which are representative of some of the main ideas we saw expressed to the socially responsible computing question from HW1: "How much ethical responsibility does a computer scientist/engineer/designer/etc. have for the technology that they create?"

The purpose of this assignment is to reflect on the complexity of identifying computer science/engineering responsibility and introduce alternative perspectives.

Choose two different viewpoints to focus on, and answer two of the questions below for each viewpoint. Both responses should be 4-5 sentences (8-10 sentences total). Please make it clear which viewpoints and questions you are responding to. **Late passes will only be granted to those with thoughtful answers.**

- What challenges would exist when implementing this framework in our current cultural/political context? Why?

- What did the response include that you had not considered?
- What do you think the response might lack in its viewpoint?

Viewpoints:

- **Viewpoint A** Computer scientists/engineers, not the users, have full ethical responsibility for the technology they produce. For example, some may blame the users who post hate speech on Facebook, Twitter, and other social media platforms, rather than the developers of these platforms. None of these actions would have been possible, however, had developers not built the features that enabled them. People building technology have immense power relative to users of it; the way a technology is built may promote certain uses or encourage users to do things that they might not have done, and users can be manipulated by technology. In short, the buck stops with the computer scientists/engineers. Should they find that their tech is being used in an unethical or illegal manner, it is their duty to prevent these uses. These misuses of a technology should be considered during the development process, in the same way that developers consider test cases and bugs.
- **Viewpoint B** I do believe that computer scientists have some ethical responsibility over their technology. However, it is important to think about this in the context of an individual. Not everybody has the economic means and/or available employment opportunities to consider the ethical implications of their job. Some people are unable to speak about potential ethical concerns in tech, because they need to have a decent living and job stability. This can be especially true for engineers whose visas are sponsored by the companies they work for, as both their livelihood and residency are dependent on their work situation. As a result, I don't think computer scientists/engineers, especially in entry level positions, should be faulted for developing potentially unethical technologies. The ethical responsibility for these projects, which are often developed on a large scale by multiple people, lies with the managers, executives, and lead designers on a project.
- **Viewpoint C** Concerns about ethical considerations from technologists dampen the speed of scientific progress. While ethical concerns are important, we should not let them get in the way of research. Similarly, a company's primary responsibility is to create a profit; ethical dilemmas should be left for governments and (by extension) the democratic process. Governments ultimately bear the ethical responsibility for negative impacts of a technology, because they need to regulate its uses. In addition, it is often impossible to assign responsibility to an individual entity for the development of a technology; much tech is built on the work of small collaborations from many contributors, so responsibility cannot be easily applied.

- **Viewpoint D** Programmers can only be held responsible to a limited extent for a technology's negative uses because it is impossible to determine every use. They also have the responsibility to meaningfully imagine and investigate potential negative uses of their technology and attempt to prevent them. For example, architects could reasonably expect a building in California to face multiple earthquakes; they are responsible for a building's collapse if it is not built with these conditions in mind. This said, some technologies can also only promote negative uses. In cases like these, some technology should not be released to the world; technologists should not release things that cause more problems than they solve.