Lab 8: Mutable Trees **lab08.zip (lab08.zip)**

Due by 11:59pm on Wednesday, March 20.

Starter Files

Download lab08.zip (lab08.zip). Inside the archive, you will find starter files for the questions in this lab, along with a copy of the Ok (ok) autograder.

Topics

Consult this section if you need a refresher on the material for this lab. It's okay to skip directly to the questions and refer back here should you get stuck.

Mutable Trees

Required Questions

Getting Started Videos

Mutable Trees

Q1: WWPD: Trees

Read over the `Tree` class in `lab08.py`. Make sure you understand the doctests.

Use Ok to test your knowledge with the following "What Would Python Display?" questions:

```
python3 ok -q trees-wwpd -u
```

Enter `Function` if you believe the answer is `<function ...>`, `Error` if it errors, and `Nothing` if nothing is displayed. Recall that `Tree` instances will be displayed the same way they are constructed.

```
>>> t = Tree(1, Tree(2))
------

>>> t = Tree(1, [Tree(2)])
>>> t.label
------

>>> t.branches[0]
------

>>> t.branches[0].label
------

>>> t.label = t.branches[0].label
>>> t
------

>>> t.branches.append(Tree(4, [Tree(8)]))
>>> len(t.branches)
------

>>> t.branches[0]
------

>>> t.branches[1]
------
```

Q2: Cumulative Mul

Write a function `cumulative_mul` that mutates the Tree `t` so that each node's label is replaced by the product of its label and the labels of all its descendents.

> **Hint**: Be careful of the order in which you mutate the current node's label and process its subtrees; which one should come first?

```
def cumulative_mul(t):
    """Mutates t so that each node's label becomes the product of its own
    label and all labels in the corresponding subtree rooted at t.

    >>> t = Tree(1, [Tree(3, [Tree(5)]), Tree(7)])
    >>> cumulative_mul(t)
    >>> t
    Tree(105, [Tree(15, [Tree(5)]), Tree(7)])
    >>> otherTree = Tree(2, [Tree(1, [Tree(3), Tree(4), Tree(5)]), Tree(6, [Tree(7)])])
    >>> cumulative_mul(otherTree)
    >>> otherTree
    Tree(5040, [Tree(60, [Tree(3), Tree(4), Tree(5)]), Tree(42, [Tree(7)])])
    """
    "*** YOUR CODE HERE ***"
```

Use Ok to test your code:

```
python3 ok -q cumulative_mul
```

Q3: Prune Small

Removing some nodes from a tree is called *pruning* the tree.

Complete the function `prune_small` that takes in a `Tree t` and a number `n`. For each node with more than `n` branches, keep only the `n` branches with the smallest labels and remove (*prune*) the rest.

> *Hint*: The `max` function takes in an `iterable` as well as an optional `key` argument (which takes in a one-argument function). For example, `max([-7, 2, -1], key=abs)` would return `-7` since `abs(-7)` is greater than `abs(2)` and `abs(-1)`.

```
def prune_small(t, n):
    """Prune the tree mutatively, keeping only the n branches
    of each node with the smallest labels.

    >>> t1 = Tree(6)
    >>> prune_small(t1, 2)
    >>> t1
    Tree(6)
    >>> t2 = Tree(6, [Tree(3), Tree(4)])
    >>> prune_small(t2, 1)
    >>> t2
    Tree(6, [Tree(3)])
    >>> t3 = Tree(6, [Tree(1), Tree(3, [Tree(1), Tree(2), Tree(3)]), Tree(5, [Tree(3), Tre
    >>> prune_small(t3, 2)
    >>> t3
    Tree(6, [Tree(1), Tree(3, [Tree(1), Tree(2)])])
    """
    while _____________________________:
        largest = max(_________________, key=____________________)
        _________________________
    for __ in _____________:
        ___________________
```

Use Ok to test your code:

```
python3 ok -q prune_small
```

Check Your Score Locally

You can locally check your score on each question of this assignment by running

```
python3 ok --score
```

This does NOT submit the assignment! When you are satisfied with your score, submit the assignment to Gradescope to receive credit for it.

Submit

Submit this assignment by uploading any files you've edited **to the appropriate Gradescope assignment.** Lab 00 (https://cs61a.org/lab/lab00/#submit-with-gradescope) has detailed instructions.

In addition, all students who are **not** in the mega lab must complete this attendance form (https://go.cs61a.org/lab-att). Submit this form each week, whether you attend lab or missed it for a good reason. The attendance form is not required for mega section students.

Optional Questions

Q4: Delete

Implement `delete`, which takes a Tree `t` and removes all non-root nodes labeled `x`. The parent of each remaining node is its nearest ancestor that was not removed. The root node is never removed, even if its label is `x`.

```
def delete(t, x):
    """Remove all nodes labeled x below the root within Tree t. When a non-leaf
    node is deleted, the deleted node's children become children of its parent.

    The root node will never be removed.

    >>> t = Tree(3, [Tree(2, [Tree(2), Tree(2)]), Tree(2), Tree(2, [Tree(2, [Tree(2), Tree
    >>> delete(t, 2)
    >>> t
    Tree(3)
    >>> t = Tree(1, [Tree(2, [Tree(4, [Tree(2)]), Tree(5)]), Tree(3, [Tree(6), Tree(2)]),
    >>> delete(t, 2)
    >>> t
    Tree(1, [Tree(4), Tree(5), Tree(3, [Tree(6)]), Tree(4)])
    >>> t = Tree(1, [Tree(2, [Tree(4), Tree(5)]), Tree(3, [Tree(6), Tree(2)]), Tree(2, [Tr
    >>> delete(t, 2)
    >>> t
    Tree(1, [Tree(4), Tree(5), Tree(3, [Tree(6)]), Tree(6), Tree(7), Tree(8), Tree(4)])
    """
    new_branches = []
    for _________ in _________________:
        ______________________
        if b.label == x:
            ___________________________________
        else:
            ___________________________________
    t.branches = ____________________
```

Use Ok to test your code:

```
python3 ok -q delete
```