CS 61A
Spring 2021

Review!

Lost 8: April 9, 2021

1 Learning Goals

- Get exam-level practice with various topics covered thus far

2 Recursion! Take the leap!

"Plaudite, amici, comedia finita est" (Applaud, my friends, the comedy is over).– Beethoven [Said on his deathbed]

This quote is not really relevant to the question, but I (Murtz) found it amusing, even though I did not write the question. Let us write a symphony!

Assume you are given a function, `judge`, that takes in a list and returns a real number. The list is a representation of a symphony and the number is how good the symphony is. Given this function, use tree recursion to write a function that generates the best symphony of a predetermined length.

```
def best_symphony(symphony_so_far, possible_notes, length):
    """
    Takes in an incomplete list symphony_so_far, a list of
    possible notes to add to it, and a maximum length for
    the symphony. Recursively computes the best possible
    symphony of that length that can be composed by
2.1 adding notes from possible_notes to symphony_so_far.
    """

    if ____________________________________________________:

        ____________________________________________________

    symphonies = []

    for ____________________________________________________:

        symphony = ____________________________________________________________________________

        symphonies += [(symphony, judge(symphony))]

    return max(_______________________________, key=_____________________________)[0]
```

3 Iterators/Generators! Linked Lists! Trees! Next! Yield! Woohoo!

3.1 To make the `Link` class iterable, implement the `LinkIterator` class.

```
class Link:
    empty = ()
    def __init__(self, first, rest=empty):
        self.first = first
        self.rest = rest
    def __iter__(self):
        return LinkIterator(self)


class LinkIterator:
    def __init__(self, link):


    def __iter__(self):


    def __next__(self):
```

3.2 Implement `sum_paths_gen`, which takes in a tree `t` and and returns a generator which yields the sum of all the nodes from a path from the root of a tree to a leaf.

You may yield the sums in any order.

```
def sum_paths_gen(t):
    """
    >>> t1 = tree(5)
    >>> next(sum_paths_gen(t1))
    5
    >>> t2 = tree(1, [tree(2, [tree(3), tree(4)]), tree(9)])
    >>> sorted(sum_paths_gen(t2))
    [6, 7, 10]
    """

    if ____________________________:

        yield ____________________

    for ____________________________:

        for ____________________________:

            yield ____________________
```

3.3 Implement `long_paths`, which returns a list of all *paths* in a tree with length at least `n`. A path in a tree is a linked list of node values that starts with the root and ends at a leaf. Each subsequent element must be from a child of the previous value's node. The *length* of a path is the number of edges in the path (i.e. one less than the number of nodes in the path). Paths are listed in order from left to right. See the doctests for some examples.

```
def long_paths(tree, n):
    """Return a list of all paths in tree with length at least n.

    >>> t = Tree(3, [Tree(4), Tree(4), Tree(5)])
    >>> left = Tree(1, [Tree(2), t])
    >>> mid = Tree(6, [Tree(7, [Tree(8)]), Tree(9)])
    >>> right = Tree(11, [Tree(12, [Tree(13, [Tree(14)])])])
    >>> whole = Tree(0, [left, Tree(13), mid, right])
    >>> for path in long_paths(whole, 2):
    ...     print(path)
    ...
    <0 1 2>
    <0 1 3 4>
    <0 1 3 4>
    <0 1 3 5>
    <0 6 7 8>
    <0 6 9>
    <0 11 12 13 14>
    >>> for path in long_paths(whole, 3):
    ...     print(path)
    ...
    <0 1 3 4>
    <0 1 3 4>
    <0 1 3 5>
    <0 6 7 8>
    <0 11 12 13 14>
    >>> long_paths(whole, 4)
    [Link(0, Link(11, Link(12, Link(13, Link(14)))))]
    """
```

4 Scheme! Languages! Parentheses!

4.1 Fall 2020 Final, Question 5

Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.