Final Review

Announcements

Environment Diagrams

Su24 Final Q2: Sweet Diadreams

https://cs61a.org/exam/su24/final/61a-su24-final.pdf#page=5

(Demo)

Tree Recursion

Su24 Final Q3: Movie Theater Seating

https://cs61a.org/exam/su24/final/61a-su24-final.pdf#page=7

(Demo)

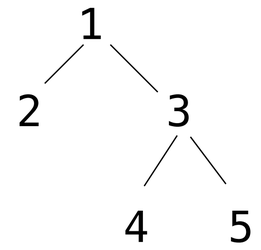
Trees

61A Fall 2015 Final Question 3 [Extended Remix]

Definition. A *path* through a Tree is a list of adjacent node labels that starts with the root label and ends with a leaf label.

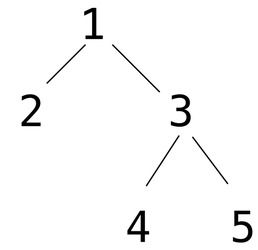
```
def count_big(t, n):
    """Return the number of paths in t that have a sum larger or equal to n.

    >>> t = Tree(1, [Tree(2), Tree(3, [Tree(4), Tree(5)])])
    >>> count_big(t, 3)
    3
    >>> count_big(t, 6)
    2
    >>> count_big(t, 9)
    1
    """
    if t.is_leaf():
        return one(_____t.label >= n_____)
    else:
        return ____sum([count_big(b, n-t.label) for b in t.branches])____
```

```
def one(b):
    if b:
        return 1
    else:
        return 0
```



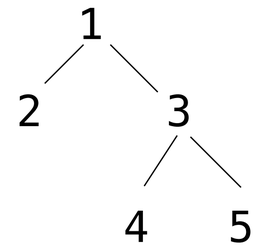
61A Fall 2015 Final Question 3 [Extended Remix]

```
def print_big(t, n):
    """Print the paths in t that have a sum larger or equal to n.

    >>> t = Tree(1, [Tree(2), Tree(3, [Tree(4), Tree(5)])])
    >>> print_big(t, 3)
    [1, 2]
    [1, 3, 4]
    [1, 3, 5]
    >>> print_big(t, 6)
    [1, 3, 4]
    [1, 3, 5]
    >>> print_big(t, 9)
    [1, 3, 5]
    """
    def helper(t, p):
        p = p + [t.label]
        if t.is_leaf():
            if sum(p) >= n:
                print(p)
        else:
            for b in t.branches:
                helper(b, p)
    helper(t, [])
```



61A Fall 2015 Final Question 3 [Extended Remix]

```
def big_links(t, n):
    """Yield the paths in t that have a sum larger or equal to n as linked lists.

    >>> t = Tree(1, [Tree(2), Tree(3, [Tree(4), Tree(5)])])
    >>> for p in big_links(t, 3):
    ...     print(p)
    <1 2>
    <1 3 4>
    <1 3 5>
    >>> for p in big_links(t, 6):
    ...     print(p)
    <1 3 4>
    <1 3 5>
    >>> for p in big_links(t, 9):
    ...     print(p)
    <1 3 5>
    """
    if     t.is_leaf() and t.label >= n     :
        yield       Link(t.label)
    for b in t.branches:
        for x in    big_links(b, n - t.label)   :
            yield       Link(t.label, x)
```



OOP + Linked Lists

Su24 Final Q5: CS 61A Web Browser

https://cs61a.org/exam/su24/final/61a-su24-final.pdf#page=12

(Demo)

Mutability + Iterators/Generators

Su24 Final Q1: Phrase Phonetics

https://cs61a.org/exam/su24/final/61a-su24-final.pdf#page=3

Su24 Final Q1: Phrase Phonetics

Assume the following code has been executed. No error occurs when executing this code block.

```
phrases = ['sweet', 'dreams', 'good', 'night', '!']

def crowdstrike():
    while phrases:
        yield phrases.pop()

i1 = iter(phrases)
i2 = iter(phrases[1:])
```

Hint: Draw it out!

(a) (1.0 pt)

```
>>> next(i1) + next(i2)
```

```
'sweetdreams'
```

(b) (1.0 pt)

```
>>> phrases.insert(1, 'question')
>>> next(i2) + next(i1)
```

```
'goodquestion'
```

Su24 Final Q1: Phrase Phonetics

Assume the following code has been executed. No error occurs when executing this code block.

```
phrases = ['sweet', 'dreams', 'good', 'night', '!']

def crowdstrike():
    while phrases:
        yield phrases.pop()

i1 = iter(phrases)
i2 = iter(phrases[1:])
```

Hint: Draw it out!

(a) (1.0 pt)

```
>>> next(i1) + next(i2)
```

```
'sweetdreams'
```

(b) (1.0 pt)

```
>>> phrases.insert(1, 'question')
>>> next(i2) + next(i1)
```

```
'goodquestion'
```

(c) (2.0 pt)

```
>>> c = crowdstrike()
>>> next(i2) + next(i1) + next(c)
```

```
'nightdreams!'
```

Su24 Final Q1: Phrase Phonetics

Assume the following code has been executed. No error occurs when executing this code block.

```
phrases = ['sweet', 'dreams', 'good', 'night', '!']

def crowdstrike():
    while phrases:
        yield phrases.pop()

i1 = iter(phrases)
i2 = iter(phrases[1:])
```

Hint: Draw it out!

(a) (1.0 pt)

```
>>> next(i1) + next(i2)
```

```
'sweetdreams'
```

(b) (1.0 pt)

```
>>> phrases.insert(1, 'question')
>>> next(i2) + next(i1)
```

```
'goodquestion'
```

(d) (2.0 pt)

```
>>> list(c)
```

```
['night', 'good', 'dreams', 'question', 'sweet']
```

(2.0 pt)

```
>>> c = crowdstrike()
>>> next(i2) + next(i1) + next(c)
```

```
'nightdreams!'
```

Su24 Final Q1: Phrase Phonetics

Assume the following code has been executed. No error occurs when executing this code block.

```
phrases = ['sweet', 'dreams', 'good', 'night', '!']

def crowdstrike():
    while phrases:
        yield phrases.pop()

i1 = iter(phrases)
i2 = iter(phrases[1:])
```

Hint: Draw it out!

(a) (1.0 pt)

```
>>> next(i1) + next(i2)
```

```
'sweetdreams'
```

(b) (1.0 pt)

```
>>> phrases.insert(1, 'question')
>>> next(i2) + next(i1)
```

```
'goodquestion'
```

(2.0 pt)

```
>>> c = crowdstrike()
>>> next(i2) + next(i1) + next(c)
```

```
'nightdreams!'
```

(d) (2.0 pt)

```
>>> list(c)
```

```
['night', 'good', 'dreams', 'question', 'sweet']
```

(e) (2.0 pt)

```
print(next(i2)) or print(next(i1))
```

```
!
StopIteration
```

Scheme Lists

Su24 Final Q6b: zip

(5.0 points) zip

Implement `zip` which takes in two lists, `s0` and `s1`, and returns a list of lists where the nested list at index `i` contains exactly two elements: the element at index `i` in `s0` and the element at index `i` in `s1`. If `s0` and `s1` have different lengths, only zip together the first `k` elements where `k` is the length of the shorter list.

```
; doctests
scm> (zip '(1 2 3) '(-1 -2 -3))
((1 -1) (2 -2) (3 -3))
scm> (zip '(1 2 3) '(-1 -2 -3 -4 -5))
((1 -1) (2 -2) (3 -3))


(define (zip s0 s1)
    (if __________
            (c)
        nil
        (cons __________ (zip __________))
                 (d)             (e)
    )
)
```

i. (1.0 pt) Fill in blank (c).

ii. (1.0 pt) Fill in blank (d).

iii. (1.0 pt) Fill in blank (e).

iv. (2.0 pt) Is `zip` tail recursive?

- ◯ Yes, it is tail recursive.
- ◯ No, it is not tail recursive.

(Demo)

Tail Recursion

Su24 Final Q6a: all

(2.0 points) all

`all` takes in a list, `s`, and returns `#t` if all the elements of the list are truth-y or if `s` has no elements. Otherwise, it returns `#f`.

```
; doctests
scm> (all (list 0 1 2 3 4 5))
#t
scm> (all (list 0 1 2 3 (< 4 2) 5))
#f
scm> (all '())
#t
```

```
(define (all s)
    (if (null? s)
        #t
        (and (car s) (all (cdr s)))
    )
)
```

i. (2.0 pt) Is `all` tail recursive?

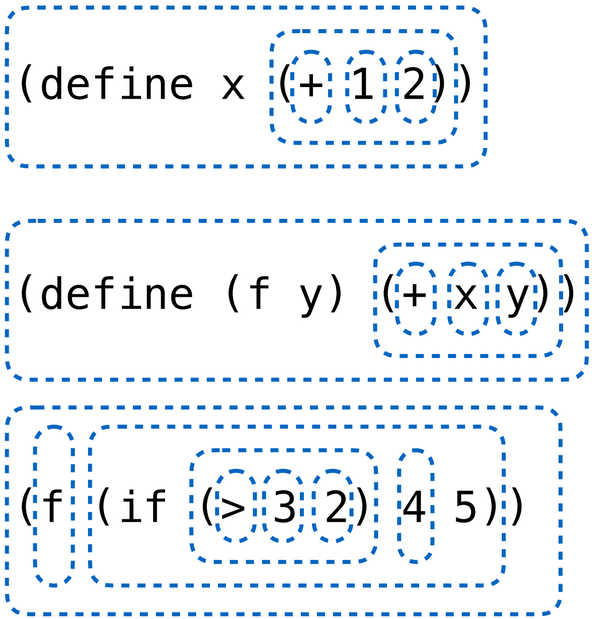
- ◯ Yes, it is tail recursive.
- ◯ No, it is not tail recursive.

(Demo)

Interpreters

Interpreter Analysis

What expressions are passed to scheme_eval when evaluating the following expressions?



SQL

Su24 Final Q8: Team USA Basketball

https://cs61a.org/exam/su24/final/61a-su24-final.pdf#page=27

(Demo)

Questions?