DISCUSSION 09

Linked Lists, Efficiency, Mutable Trees

Mingxiao Wei
mingxiaowei@berkeley.edu

Mar 23, 2023

FROM LAST TIME... 👀

How are you feeling now?

scared for the midterm	Chill af	iffy on trees -- still a lil lost on them but i understand oop well
tired	good	Stressed for the midterm
hungry	sleepy	Okay
Great!	tired and behind on cs	Pretty tired
LIttle stressed for the midterm	tired :(	good
behind on content	tired~~~~	tired
sleepy	Mid	good good
tired	a little depressed	panic
Very tired	Good!	Please don't rain anymore :/
Good :)	Cold... The rain is heavy...	

LOGISTICS 

- ANTS  
  - The whole project due tomorrow 03/24
  - Submit by today 03/23 for one extra point!
- Come to OH! (schedule)
- Ask Us Anything lecture tomorrow - submit your questions to the professors here
  - Not recorded, but will be livestreamed
  - It'll be fun I promise

ABOUT THE 2ND MIDTERM 😮

TL;DR: There's going to be another midterm.
Don't be too surprised on Apr 7th :)

- Fri 4/7, 7-9 pm
- Logistics - Ed post #2069
  - If you need ANY alterations (left-handed desk, remote, other accommodations), fill out this form by Mon 04/03!!
- Preparations
  - Familiarize yourself with the topics in scope
  - Review sessions - see Ed for more info
  - Do past exams!
    - Quality > quantity
    - Post on exam threads on Ed for help
    - Walkthrough videos/guide are your friend!

LINKED LIST 

LINKED LIST

- Sequence = ordered collection of elements
- Lists are an implementation of sequence
  - E.g., `[6, 1, 'a']`
- Linked list:

  - a node = a rectangular above = a `Link` object
  - A linked list knows its `first` value and its `rest` - another linked list

LINKED LIST - IMPLEMENTATION

```
class Link:
    empty = ()
    def __init__(self, first, rest=empty):
        assert rest is Link.empty or isinstance(rest, Link)
        # rest must be either a linked list or Link.empty
        self.first = first
        self.rest = rest
```

```
>>> lnk = Link(1, Link(2))
>>> lnk.first
1
>>> lnk.rest
Link(2)
>>> lnk.rest.first
2
```

CONSTRUCTING A LINKED LIST


```
>>> lnk3 = Link('a')
>>> lnk2 = Link(1, lnk3)
>>> lnk1 = Link(6, lnk2)
>>> lnk1
Link(6, Link(1, Link('a')))
>>> lnk1.rest
Link(1, Link('a'))
>>> lnk1.rest is lnk2
True
```

EMPTY LINKED LIST

```
class Link:
    empty = ()
    def __init__(self, first, rest=empty):
        assert rest is Link.empty or isinstance(rest, Link)
        # rest must be either a linked list or Link.empty
        self.first = first
        self.rest = rest
```

- `Link.empty` - *the* empty linked list
  - Can be implemented as anything
- To check if a linked list `lnk` is empty: `lnk is Link.empty`

```
>>> lnk = Link(1, Link(2))
>>> lnk.rest.rest is Link.empty
True
```

LINKED LIST - REPR

```
class Link:
    empty = ()
    ...
    def __repr__(self):
        if self.rest:
            rest_repr = ',' + repr(self.rest)
        else:
            rest_repr = ''
        return 'Link(' + repr(self.first) + rest_repr + ')'
```

- `__repr__` returns a string, that, when evaluated, returns a `Link` object containing the same values

```
>>> a = 2
>>> lnk = Link(1, Link(a))
>>> repr(lnk)
'Link(1, Link(2))'
```

LINKED LIST - STR

```
class Link:
    empty = ()
    ...
    def __str__(self):
        string = '<'
        while self.rest is not Link.empty:
            string += str(self.first) + ''
            self = self.rest
        return string + str(self.first) + '>'
```

```
>>> lnk = Link(1, Link(2))
>>> str(lnk)
'<1 2>'
>>> lnk2 = Link(lnk, Link(3))
>>> print(lnk2) # same as print(str(lnk2))
<<1 2> 3>
```

LINKED LIST - PROBLEM SOLVING STRATEGIES

- Pay attention to whether it's **mutation** or **constructing a new linked list**
  - mutation problems - often returns `None`
  - returning a new linked list - in what order to construct?
- To mutate a linked list, reassign its instance attributes
  - `link.first = ...`
  - `link.rest = ...`
- Make sure that a linked list is not `Link.empty` before accessing any instance attributes!

WORKSHEET

ATTENDANCE! 🤠

go.cs61a.org/mingxiao-att

- The attendance form and slides are both linked on our section website!
- Please leave any anonymous feedback here go.cs61a.org/mingxiao-anon
- Please do remember to fill out the form by midnight today!!