raw results — A1

CSC148 winter 2014

linked structures

week 8

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~heap/148/W14/
416-978-5899

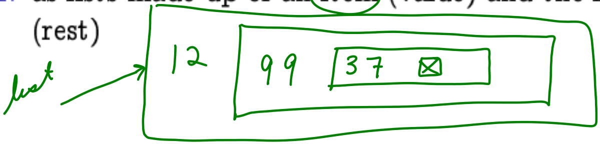
March 4, 2014

Outline

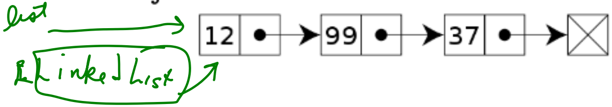
linked lists, two concepts

There are two useful, but different, ways of thinking of linked list structures

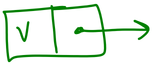
1. as lists made up of an $\overset{\text{head}}{\text{item}}$ (value) and the remaining list (rest)



2. as objects (nodes) with a value and a reference to other similar objects



a node class



```
class LListNode:
    """Node to be used in linked list"""

    def __init__(self: 'LListNode', value: object,
                 nxt: 'LListNode' =None) -> None:
        """Create a new LListNode containing value
        referring to next node nxt

        nxt --- None if and only if we are on the last node
        value --- always a Python object, there are no empty nodes
        """
        self.value, self.nxt = value, nxt
```

a wrapper class for list

The list class keeps track of information about the entire list — such as its front.

```
class LinkedList:
    """Collection of LListNodes"""

    def __init__(self: 'LinkedList') -> None:
        """Create an empty LinkedList"""
        self.front = None
        self.size = 0
```

special node, front of list

insertion

```
def insert(self: 'LinkedList', value: object) -> None:
    """Insert LListNode with value at front of self

    >>> lnk = LinkedList()
    >>> lnk.insert(0)
    >>> lnk.insert(1)
    >>> lnk.insert(2)
    >>> str(lnk.front)
    '2 -> 1 -> 0 -> None'
    >>> lnk.size
    3
    """
```

$$\begin{tikzcd}\boxed{LL} \arrow[r] & \boxed{X}\end{tikzcd}$$

$$\begin{tikzcd}\boxed{LL} \arrow[r] & \boxed{0} \arrow[r] & \boxed{X}\end{tikzcd}$$

$$\begin{tikzcd}\boxed{LL} \arrow[r] & \boxed{1} \arrow[r] & \boxed{0} \arrow[r] & \boxed{X}\end{tikzcd}$$

$$\begin{tikzcd}\boxed{LL} \arrow[r] & \boxed{2} \arrow[r] & \boxed{1} \arrow[r] & \boxed{0} \arrow[r] & \boxed{X}\end{tikzcd}$$

```
self.front = LListNode(value, self.front)
self.size += 1
```

deletion

```
def delete_front(self: LinkedList) -> None:
    """Delete front LListNode from self

    self must not be None

    >>> lnk = LinkedList()
    >>> lnk.insert(0)
    >>> lnk.insert(1)
    >>> lnk.insert(2)
    >>> lnk.delete_front()
    >>> str(lnk.front)
    '1 -> 0 -> None'
    >>> lnk.size
    2
    """
```

$$\begin{tikzcd}\boxed{LL} \arrow[r] \arrow[rr, bend left] & \boxed{2} \arrow[r] & \boxed{1} \arrow[r] & \boxed{0} \arrow[r] & \text{None}\end{tikzcd}$$

size -= 1

Computer Science
UNIVERSITY OF TORONTO

reversing

```
def reverse(ln: LListNode) -> LListNode:
    """Return the linked list starting
    at ln in reverse order

    ln is not None

    >>> ln = LListNode(0)
    >>> ln1 = LListNode(1, ln)
    >>> ln2 = LListNode(2, ln1)
    >>> ln3 = LListNode(3, ln2)
    >>> lnr = reverse(ln3)
    >>> str(lnr)
    '0 -> 1 -> 2 -> 3 -> None'
    """
```