The linked list

EECS 214, Fall 2017
A problem with vectors

What if we want to add 6 between 5 and 7?
A problem with vectors

What if we want to add 6 between 5 and 7?
Books on a string

- The Art of Computer Programming
  - Volume 1: Fundamental Algorithms
  - Third Edition
  - Donald E. Knuth

- Surreal Numbers
  - D. E. Knuth

- Concrete Mathematics
  - A Foundation for Computer Science
  - Second Edition
  - Graham, Knuth, and Patashnik
Books on a string

The Art of Computer Programming
VOLUME 1
Fundamental Algorithms
Third Edition
DONALD E. KNUTH

SURREAL NUMBERS
D. E. KNUTH

The Art of Computer Programming
VOLUME 2
Seminumerical Algorithms
Third Edition
DONALD E. KNUTH

CONCRETE MATHEMATICS
A FOUNDATION FOR COMPUTER SCIENCE
GRAHAM KNUTH PATASHNIK
SECOND EDITION
Nodes and pointers

2 -> 4 -> 6 -> 8
Nodes and pointers

data 2
next

data 4
next

data 6
next

data 8
next

data 5
next
Nodes and pointers

```plaintext
<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
```
Nodes and pointers
Nodes and pointers

2 → 4 → 6 → 8

7 → 5 → 3
Nodes and pointers

```
data 2
next  
```

```
data 4
next  
```

```
data 6
next  
```

```
data 7
next  
```

```
data 5
next  
```

```
data 3
next  
```

```
data 8
next  
```

5
Inserting at the beginning
Inserting at the beginning
Inserting at the beginning
Inserting at the beginning

lst

<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
<th>data</th>
<th>next</th>
<th>data</th>
<th>next</th>
<th>data</th>
<th>next</th>
<th>data</th>
<th>next</th>
<th>data</th>
<th>next</th>
<th>data</th>
<th>next</th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
<td>4</td>
<td></td>
<td>5</td>
<td></td>
<td>6</td>
<td></td>
<td>7</td>
<td></td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>


Inserting at the beginning
Indirection
Indirection
Now in DSSL2
Linked lists in DSSL2

# List is ll { head: Link }
defstruct ll(head)

# Link is one of:
# - node { data: Number, next: Link }
# - nil()
defstruct node(data, next)
defstruct nil()
Linked lists in DSSL2

# List is ll { head: Link }
defstruct ll(head)

# Link is one of:
# - node { data: Number, next: Link }
# - nil()
defstruct node(data, next)
defstruct nil()

# new_list : -> List
def new_list():
    ll { head: nil() }

# insert_front : Number List ->
def insert_front(n, lst):
    lst.head = node { data: n, next: lst.head }
List operations in DSSL2

# get_front : List -> Number
def get_front(lst):
    if node?(lst.head): lst.head.data
    else: error('get_front: got empty list')
List operations in DSSL2

# get_front : List -> Number

```python
def get_front(lst):
    if node?(lst.head):
        lst.head.data
    else: error('get_front: got empty list')
```

# get_nth : List Natural -> Number

```python
def get_nth(lst, n0):
    def loop(link, n):
        if nil?(link):
            error('get_nth: list too short')
        elif n == 0:
            return link.data
        else:
            return loop(link.next, n - 1)
    loop(lst.head, n0)
```
More DSSL2 list operations

# find_nth_node : Link Natural → Link
def find_nth_node(link, n):
    if nil?(link): error('find_nth_node: too short')
    elif n == 0: link
    else: find_nth_node(link.next, n - 1)

# get_nth : List Natural → Number
def get_nth(lst, i):
    find_nth_node(lst.head, i).data

# set_nth! : List Natural Number →
def set_nth!(lst, i, val):
    find_nth_node(lst.head, i).data = val
What else might we want to do?
What else might we want to do?

- Insert or remove at the given position or the end.
- Split a list in two or splice two into one.
- Know how long the list is without counting.
Keeping the length

How can we make sure the len field is always right?
Keeping the length

How can we make sure the `len` field is always right?
Quick access to the tail

Which operations are simple now? Which are still more work?
Quick access to the tail

Which operations are simple now? Which are still more work?
Doubly-linked
Circular, doubly-linked with sentinel

sentinel

len 6

data 1
prev
next

data 2
prev
next

data 3
prev
next

data 4
prev
next

data 5
prev
next

data 6
prev
next

len 6
Empty (circular, doubly-linked w/sentinel)
Next time: asymptotic complexity