CSC207.01 2014S, Class 31: Doubly-Linked Lists

Overview

- Preliminaries.
  - Admin.
  - About Exam 2.
- Review of singly-linked lists.
- Insertion and deletion in singly-linked lists.
- Doubly-linked lists.
- Circularly-linked lists.

Preliminaries

Admin

- No writeup for today.
- Reading for Friday: Java Tutorial on Generics and Anonymous Inner Classes
  - We will unpack it on Friday.
- Seniors: A reminder to pledge for the yearbook.
- Second years: A reminder to declare majors. I’d love to see many/most of you declare CS majors.
- For today’s class, we’re doing to work together to figure out the details of doubly-linked lists. Do you prefer Rebelsky’s round-robin recitation, small group discussion, or a combination?
- Extra credit:
  - CS table Friday: TBD

About Exam 2

- Exam 2 distributed in draft form.
- The code should be ready tonight.
- The unit tests should be ready on Friday.

Review of singly-linked lists

- Small pieces, loosely joined
- Nodes
  - Data
  - Next
- The list class includes a pointer to the front of the list
- Iterators for the list traditionally keep track of the element right before the element that next returns
  - Node cursor
• Issue: What’s the element before the front?
• Add: Redirect the next to a new node

```java
public class SinglyLinkedListIterator implements ListIterator {
  public Node cursor;

  public void add(T value) { this.cursor.next = new Node(value, this.cursor.next); this.cursor =
    this.cursor.next; }

  public void remove() { } // remove()

} // class SinglyLinkedListIterator
```

Sam confuses his students and then tries to clarify

• If we’re building a class that implements Iterator, there are only three methods
  • next
  • hasNext
  • remove (optional can throw UnsupportedOperationException)
• If we’re building a class that implements ListIterator, there are a host of other methods
  • add
  • set
  • previous
  • hasPrevious
  • nextIndex
  • previousIndex
  • ...

How do we remove an element?

• We need to remove the link in the chain that goes to the element we just returned
• Whoops! We have to get to the previous node.
  • One strategy: Go to the front and walk forward. O(n)
  • We’d like removal to be O(1)

**Doubly-linked lists**

• Add a backwards link!

• Remove should be easy

```java
this.cursor.prev.next = this.next; this.cursor.next.prev = this.prev;
```

• Corner cases?
  • Likely to crash at the end of the list
if (this.cursor.prev != null) this.cursor.prev.next = this.next; if (this.cursor.next != null) this.cursor.next.prev = this.prev;

Circularly-linked lists

- If we add a dummy node at the end/beginning of the list, we solve this problem! No special cases (or fewer special cases)

Copyright (c) 2013-14 Samuel A. Rebelsky.

This work is licensed under a Creative Commons Attribution 3.0 Unported License To view a copy of this license, visit http://creativecommons.org/licenses/by/3.0/ or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.