CS 310: Array-y vs Listy Stuff

Chris Kauffman

Week 4-2
Logistics

HW 2 Looming
Up around Friday - had to redesign for sanity

Reading
- Weiss Ch 16 Stacks/Queues
- Weiss Ch 17 Linked Lists

Goals Today
- ArrayQueue
- Linked Lists
- Inside Classes
- Maybe iterators

Practice problems
- Write your own doubly linked list with List operations add(x) add(i,x) remove(i) get(i) set(i,x)
- 17.9: Implement LinkedList.removeAll(T x), get rid of all equals(x)
- 17.20: Augment ListIterator to have add(T x) for positional adding
ArrayQueue Demo

- In weiss/nonstandard/ArrayQueue.java
- Slightly modified demo version in today's code pack
- Uses plain java arrays, not ArrayList
- Array doubling in size done manually

Note interesting functions in Weiss's version

```java
private int increment(int x){...}
private void doubleQueue() { ... }
```
Links Aren’t Just for Queues

- Recall ListQueue - built on Node
- Slight tweaks enable general List data structure with get(i)/set(i,x)/add(x) capabilities
- Node fields head (front) and tail (back)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Big-O Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(x)</td>
<td>Add x to the end</td>
<td></td>
</tr>
<tr>
<td>add(i,x)</td>
<td>Add x at position i</td>
<td></td>
</tr>
<tr>
<td>remove(i)</td>
<td>Remove element i</td>
<td></td>
</tr>
<tr>
<td>get(i)</td>
<td>Retrieve element i</td>
<td></td>
</tr>
<tr>
<td>set(i,x)</td>
<td>Set element i to x</td>
<td></td>
</tr>
</tbody>
</table>

- How are these operations implemented? Pseudocode or Java
- What are the runtime complexities of each?
- How are the operation complexities of LinkedList different from ArrayList?
Print elements front to back

```java
class ArrayList/LinkedList{
    public void printAll(){...}
}
```

- ArrayList implementation
- SinglyLinkedList implementation
- Make both $O(N)$

Print elements back to front

```java
class ArrayList/LinkedList{
    public void printAllReverse(){...}
}
```

- ArrayList implementation
- SinglyLinkedList implementation (!)
- Can both be $O(N)$?
Double Your Fun

- Singly linked nodes: only next
  - Node n = new Node(data, next);
- Doubly linked also has previous
  - Node n = new Node(data, previous, next);

How about printAllReverse() now

---

1 Source: David H. Hovemeyer’s notes
To Header or Not to Header

- May be able to simplify using extra space
- Auxiliary 'header' and 'tailer' nodes
- Draw pictures to understand these
- Weiss uses header/tailer nodes
- **Consider** code below for \texttt{add(x)} to the back of a linked list

**No Header**

```java
public void add(T x){
    if(empty()){
        head =
            new Node(x,null,null);
        tail = head;
    } else{
        tail.next =
            new Node(x, tail, null);
        tail = tail.next;
    }
}
```

**With Header/Tailer**

```java
public void add(T x){
    Node n =
        new Node(x, tail.previous, tail);
    Node old = tail.previous;
    old.next = n;
    tail = n;
}
```
Part of HW 2 will involve implementing a basic doubly linked list
▶ Constraint: no use of java.util.LinkedList
▶ Some functionality will require more control than standard class
▶ Will provide version of Weiss’s doubly-linked list as a starting point; you must modify and complete it
A Problem

Recall

- ArrayList.get(i): \(O(1)\)
- LinkedList.get(i): \(O(n)\)

Trouble

List<Integers> l = ...;
int sum = 0;
for(int i=0; i<l.size(); i++){
  sum += l.get(i);
}

What is the complexity of the loop?
Peeking Inside with Iterators

Arrays are simple
  ▶ get/set anything
  ▶ add/remove is obvious
  ▶ Very clear how data is laid out
Just about every other data structure is less so
  ▶ Getting/setting nontrivial
  ▶ Must preserve some internal structure - control access
  ▶ Element-by-element needs to be done carefully
These qualities give rise to iterators
  ▶ A view of a data structure
  ▶ But first a detour...
Nested Namespaces

Java restriction: 1 public class per file
Sometimes this is bothersome

1. Want many small classes, one-offs
   - add, max, hasZero in HW1

2. Group of related classes
   - One external public class
   - Uses some internal classes
   - Internal classes not for public consumption
   - List, Node, Iterator

In case 2
   - Endow internal classes with access to containing class
   - Makes programming more convenient
Nested and Inner Classes

Straight from the official docs

class OuterClass {
    ...
    static class StaticNestedClass {
        ...
    }
    class InnerClass {
        ...
    }
}

Both nested/inner classes

- Put multiple classes in a single file
- Give access to namespace of OuterClass
- Have access to private methods of OuterClass

---

2 Courtesy of Oracle
For Next Week

Practice problems

- Write your own doubly linked list with List operations add(x) add(i,x) remove(i) get(i) set(i,x)
- 17.9: Implement LinkedList.removeAll(T x), get rid of all equals(x)
- 17.20: Augment ListIterator to have add(T x) for positional adding

Reading

- Chapter 17 (lists)
- Chapter 20 (Hash Tables)