Linked Lists

2-14-2011

Opening Discussion

- Do you have any questions about the assignment?
- Minute Essays
  - Stacks of queues and queues of stacks.
  - How do you use lists for this?
  - What is left to learn after this class?
- What is a list? What are the things that you can do with a list?

The List/Sequence ADT

- The next step up the ADT ladder is the list or sequence. Basically a list provides general access so you can add, remove, or search for things at random locations in the list.
- This type of functionality is provided without mutation by a Seq in Scala. With mutation it is the Buffer.

An Array Based List

- So how could we implement our list interface using an array?
- What methods of that implementation would be “fast”? Which ones would be “slow”?
- What do the terms fast and slow mean here in O terms and what operations are being considered for that?

Linked Lists

- There is an alternate method of implementing the list interface called the linked list. It is strong where an array list is weak, but weak where an array list is strong.
- A linked list is made of nodes and each node knows about one or two of its neighbors (has pointers to them).
- We move around linked lists by “walking” from node to node.
- Adding and removing can be very fast and always require very few memory writes.

Types of Linked Lists

- Linked lists can be implemented in many ways. The basic characteristic is that we only keep a reference to one node and nodes then link to one another.
- The linking can be single or double. A doubly linked list has nodes that know about both the next and the previous elements.
- Linked lists can also be circular. In a circular linked list, the first element links around to the back one.
- For optimization purposes, lists can keep track of a head and a tail, but that isn't required.

Implementing a Singly Linked List

- Let's work together to build an implementation of a singly linked list.
- We will implement the Buffer trait. It has eight abstract methods that we will be forced to implement to get everything else.

Sentinels

- A sentinel is an extra node in the list the represents the “end” of the list and doesn't store data.
- The purpose of the sentinel is to remove special cases. The next of the sentinel is what we have called head.
- They are most useful in a doubly linked list where the previous of the sentinel is tail.

Implementing a Doubly Linked List

- Now let's implement java.util.List with a doubly linked list with a sentinel. The list will also be circular.
- You should notice that this implementation never has to check for null because no references in the list should ever be null. This simplifies the code significantly. We also implicitly get a head and a tail with no extra work. If you don't have a sentinel you will write a lot of extra checks for nulls and even more to include a tail.

Minute Essay

- Any questions?
- Quiz #3 will be next class.