Announcements

Guest Lecturers next week:

• Monday: Vibhor Rastogi
• Wednesday: Sean Shih-Yen Liu

(I will be giving a keynote lecture at a conference in Brazil, about my research on probabilistic databases)
Today’s Outline

• Lists
• Stacks and queues
• Begin trees

Remember:
• Reading assignment for Monday: Ch. 1-3
• For today: 3.1, 3.2, 3.3 (did you read them?)
The List ADT

• List of items, e.g. 7, 3, 9, 5
• Insert, delete, find, replace, ...
• Implementations: linked list or array list

![Diagram of a linked list with nodes 7, 3, 9, 5]
Linked List Implementation

First, need to define a single node:

```java
private static class Node<AnyType> {
    public Node( AnyType d, Node<AnyType> n ) {
        data = d;   next = n;
    }

    public AnyType data;
    public Node<AnyType> next;
}
```
Simple Linked v.s. Double Linked

What are the tradeoffs?
Double Linked List Implementation

```java
private static class Node<AnyType>
{
    public Node( AnyType d, Node<AnyType> p, Node<AnyType> n )
    {
        data = d; prev = p; next = n;
    }

    public AnyType data;
    public Node<AnyType> prev; /* now we can go back */
    public Node<AnyType> next;
}
```
Linked List Implementation

```java
public class MyLinkedList<AnyType> implements Iterable<AnyType> {

    /* all methods go here (see book, code examples) */

    private int theSize;
    private Node<AnyType> beginMarker;
    private Node<AnyType> endMarker;
}
```

![Diagram of linked list with nodes containing 7, 3, 9, and 5, and markers labeled beginMarker and endMarker.]
Operations on the Linked List

Discuss in class the following operations:
1. Find(9); Find(2);
2. Insert(11); at the beginning, at the end, in the middle;
3. Remove(9);
Array Implementation

Discuss in class the following operations:
1. Find(9); Find(2);
2. Insert(11); at the beginning, at the end, in the middle;
3. Remove(9);
**DEFINITION**: The Big-O notation

\[ T(n) = O(f(n)) \] if there exist constants \( c \) and \( n' \) such that: \( T(n) \leq c f(n) \) for all \( n \geq n' \)
Summary: Some Complexities

<table>
<thead>
<tr>
<th>Function</th>
<th>Linked list</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AnyType get( int idx)</code></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td><code>int find(AnyType x)</code></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td><code>void add(x,idx)</code></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td><code>/* before/after */</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>void remove(int idx)</code></td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td><code>InsertAnywhere(x)</code></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
### Summary: Some Complexities

<table>
<thead>
<tr>
<th>Operation</th>
<th>Linked list</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnyType get( int idx)</td>
<td>$O(k)$ (\text{where } k = \text{idx})</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>int find(AnyType x)</td>
<td>$O(n)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>void add(x,idx)</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>/* before/after */</td>
<td></td>
<td></td>
</tr>
<tr>
<td>void remove(int idx)</td>
<td>$O(1)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>InsertAnywhere(x)</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>
Iterators

• Review questions from Weiss, Chapt. 3.3

• What is an Iterator?

• What is an Iterable?

• What is a Collection?
public interface Iterator<AnyType>
{
    boolean hasNext();
    AnyType next();
    void remove();
}

An Iterable provides a method named iterator that returns and object of type Iterator. It can be used in a for loop (see book...).

public interface Collection<AnyType>
    extends Iterable<AnyType>
{
    ... 
    boolean add( AnyType x);
    boolean remove( AnyType x );
    ... 
}
Application: Polynomial ADT

Attempt 1:
- linked list implementation:
- $A_i$ is the coefficient of the $x^{i-1}$ term:

\[
\begin{align*}
5 + 2x + 3x^2 & \quad (5 \ 2 \ 3) \\
7 + 8x & \quad (7 \ 8) \\
3 + x^2 & \quad (3 \ 0 \ 1)
\end{align*}
\]

Problem?
$4 + 3x^{2001}$
Sparse Vector Data Structure:
$4 + 3x^{2001}$

$(<4 \ 0> \ <3 \ 2001>)$
Addition of Two Polynomials?

15 + 10x^{50} + 3x^{1200}

5 + 30x^{50} + 4x^{100}

Complexity?
Addition of Two Polynomials

One pass down each list: $O(m+n)$

$15 + 10x^{50} + 3x^{1200}$

$p \rightarrow \begin{array}{c} 15 \\ 0 \end{array} \rightarrow \begin{array}{c} 10 \\ 50 \end{array} \rightarrow \begin{array}{c} 3 \\ 1200 \end{array}$

$5 + 30x^{50} + 4x^{100}$

$q \rightarrow \begin{array}{c} 5 \\ 0 \end{array} \rightarrow \begin{array}{c} 30 \\ 50 \end{array} \rightarrow \begin{array}{c} 4 \\ 100 \end{array}$

$r \rightarrow \begin{array}{c} 20 \\ 0 \end{array} \rightarrow \begin{array}{c} 40 \\ 50 \end{array} \rightarrow \begin{array}{c} 4 \\ 100 \end{array} \rightarrow \begin{array}{c} 3 \\ 1200 \end{array}$
Sparse Matrices

\[
\begin{bmatrix}
18 & 0 & 33 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 99 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 27 \\
\end{bmatrix}
\]

( <row ( <column data> <column data> ... ) > 
  <row ( <column data> <column data> ... ) >
  ... )

(  <1 ( <1 18> <3 33> ) > ,  <4 ( <4 99> ) >  <6 ( <6 27> ) >  

( <row column data> <row column data> ... )

(  <1, 1, 18>  <1, 3, 33>  <4, 4, 99>  <6, 6, 27> )
First Example: Stack ADT

- LIFO: Last In First Out
- Stack operations
  - create
  - destroy
  - push
  - pop
  - top
  - is_empty
Stacks in Practice

- Function call stack
- Removing recursion
- Balancing symbols (parentheses)
- Evaluating Postfix Notation

Infix: \(7 + 8 \times 4 - 30\)

Postfix: \(7 \ 8 \ 4 \ \ast \ + \ 30 \ -\)
Infix to Postfix with a Stack

\[ 7 + 8 * 4 - 30 \]

Infix to Postfix with a Stack

7 + 8 * 4 – 30 #

Stack:

Output:

7
Infix to Postfix with a Stack

7 + 8 * 4 − 30 #

Stack:

Output:

7 8
Infix to Postfix with a Stack

\[ 7 + 8 \times 4 - 30 \]

Stack:

\[
\begin{array}{c}
* \\
+ \\
\end{array}
\]

Output:

7 8
Infix to Postfix with a Stack

7 + 8 * 4 − 30 #

Stack:

Output:

Stack:

+ *

7 8 4
Infix to Postfix with a Stack

\[ 7 + 8 \times 4 - 30 \]

Stack:

Output:

\[ 7 8 4 * \]
Infix to Postfix with a Stack

7 + 8 * 4 – 30 #

(empty stack)

Stack:

Output:

7 8 4 * +
Infix to Postfix with a Stack

\[ 7 + 8 \times 4 - 30 \# \]

Stack:

Output:

\[ 7 \ 8 \ 4 \ \times \ + \]
Infix to Postfix with a Stack

7 + 8 * 4 – 30 #

Stack:

Output:

7 8 4 * + 30
Infix to Postfix with a Stack

7 + 8 * 4 – 30 #

Stack:

Output:

7 8 4 * + 30 -
Array vs. Linked List

• Too much space, or not enough space
• Not as complex
• Could make array more robust

• Kth element accessed “easily”
• Insert requires shift

• Can grow as needed
• More memory per item in the queue
• Linked list code more complex

• Kth element access requires linear scan??
Second Example: Queue ADT

- FIFO: First In First Out
- Queue operations
  - create
  - destroy
  - enqueue
  - dequeue
  - is_empty

We briefly mentioned it in Lecture 01...

Circular Array Queue Data Structure

enqueue(Object x) {
    Q[back] = x ;
    back = (back + 1) % size
}

dequeue() {
    x = Q[front] ;
    front = (front + 1) % size;
    return x ;
}

How test for empty list?
How to find K-th element in the queue?
What is complexity of these operations?
Limitations of this structure?
Linked List Queue Data Structure

```
void enqueue(Object x) {
    if (is_empty())
        front = back = new Node(x)
    else
        back.next = new Node(x)
        back = back.next
}

Object dequeue() {
    assert(!is_empty())
    return_data = front.data
    temp = front
    front = front.next
    delete temp
    return return_data
}

bool is_empty() {
    return front == null
}
```

Circular Array vs. Linked List

- Too much space, or not enough space
- Not as complex
- Could make array more robust

- Can grow as needed
- More memory per item in the queue
- Linked list code more complex
Applications of Queues

- Tree traversals, graph traversals
  - Will see in coming lectures
- Wherever fairness is needed:
  - Printer queues
  - Packets in a network
  - http requests at a web server
  - Etc.