4.3: Linked Structures

Linked list of strings.
- A recursive data structure.
- A string plus a pointer to another linked list (or empty list).
- Unwind recursion: linked list is a sequence of strings.

Linked Lists

public class List {   private String name;   private List next;}

AlicenullBobCarol

List Demo

Linked vs. Sequential Allocation

Goal: process a collection of objects.

Sequential allocation: put object one after another.
- TOY: consecutive memory cells.
- Java: array of objects.

Linked allocation: include in each object a link to the next one.
- TOY: link is memory address of next object.
- Java: link is reference to next object.

Key distinctions.
- Sequential allocation: random access, fixed size.
- Linked allocation: sequential access, variable size.
Traversing a List

Paradigm for traversing a null-terminated linked list.

```
for (List x = c; x != null; x = x.next) {
    System.out.println(x.name);
}
```
Paradigm for traversing a null-terminated linked list.

```
for (List x = c; x != null; x = x.next) {
    System.out.println(x.name);
}
```

4.4: Stacks and Queues
Stack and Queue ADTs

Fundamental data type.
- Set of operations (add, remove, test if empty) on generic data.
- Intent is clear when we insert.
- Which object do we remove?

Stack.
- Remove the object **most recently added**. ("last in first out")
- Analogy: cafeteria trays, surfing Web.

Queue.
- Remove the object **least recently added**. ("first in first out")
- Analogy: Registrar’s line.

Multiset.
- Remove any object.
- Law professor calls on arbitrary student.

Queue operations.
- enqueue: Insert a new object onto queue.
- dequeue: Delete and return the object least recently added.
- isEmpty: Is the queue empty?

```java
public static void main(String[] args) {
    Queue q = new Queue();
    q.enqueue("Vertigo");
    q.enqueue("Just Lose It");
    q.enqueue("Pieces of Me");
    q.enqueue("Pieces of Me");
    System.out.println(q.dequeue());
    while(!q.isEmpty())
        System.out.println(q.dequeue());
}
```

A sample queue client

More Applications of Queues

Some applications.
- iTunes playlist.
- Breadth first search.
- Data buffers (iPod, TiVo).
- Graph processing (stay tuned).
- Asynchronous data transfer (file IO, pipes, sockets).
- Dispensing requests on a shared resource (printer, processor).

Simulations of the real world.
- Traffic analysis of Lincoln tunnel.
- Waiting times of customers in McDonalds.
- Determining number of cashiers to have at a supermarket.

Queue: Linked List Implementation

Linked list implementation.
- Maintain linked list of elements.
- Let `first` be reference to first node on list.
- Let `last` be reference last node on list.
Queue: Linked List Implementation

Linked list implementation.
- Maintain linked list of elements.
- Let first be reference to first node on list.
- Let last be reference last node on list.

Insert.

```java
Node x = new Node();
x.item = "for";
last.next = x;
last = x;
```

Queue: Linked List Implementation

Linked list implementation.
- Maintain linked list of elements.
- Let first be reference to first node on list.
- Let last be reference last node on list.

Insert.

```java
Node x = new Node();
x.item = "for";
last.next = x;
last = x;
```
Queue: Linked List Implementation

Linked list implementation.
- Maintain linked list of elements.
- Let first be reference to first node on list.
- Let last be reference last node on list.

Delete.

```java
String val = first.item;
first = first.next;
return val;
```

```
public class Queue {
    private Node first;
    private Node last;
    private class Node {
        String item;
        Node next;
    }
    public boolean isEmpty() { return first == null; }
    public void enqueue(String anItem) {
        Node x = new Node();
        x.item = anItem;
        x.next = null;
        if (isEmpty()) { first = x; last = x; }
        else { last.next = x; last = x; }
    }
    public String dequeue() {
        String val = first.item;
        first = first.next;
        return val;
    }
}
```
Binary Trees

- Organize homogeneous collection of values (all same type).
- Associate two pointers with each value.
- Use pointers to access each branch of the tree.

Java implementation of a binary tree of strings is:

```java
public class Tree {
    private String s;
    private Tree left;
    private Tree right;
}
```

Parse Trees

- Parse tree. Representation of the syntactic structure of a statement, sentence, or expression.

Ex: arithmetic expressions.

- 10 * 12 + 7
  - (10 * 12) + 7?
  - 10 * (12 + 7)?
Parse Tree Construction
How do we read it back in and create the tree?
• Read string from standard input.
• If + or * operator, construct left and right subtrees recursively.

```
public class ParseTree {
    private String s; ← represent data as a string, e.g., "+" or "1234"
    private ParseTree left; ← left subtree
    private ParseTree right; ← right subtree

    public int eval() {
        if (s.equals("+")) return left.eval() + right.eval();
        else if (s.equals("*")) return left.eval() * right.eval();
        else return Integer.parseInt(s); ← convert from string to integer
    }
}
```

Preorder Traversal
How do we print out the information?
• Print string.
• Print left subtree recursively.
• Print right subtree recursively.

No parentheses!

```
public String toString() {
    if (s.equals("+")) return s + left.toString() + right.toString();
    else return s;
}
```

Other Types of Trees
Other types of trees.
• Family tree.
• Parse tree.
• Unix file hierarchy.
Other Types of Trees

Other types of trees.
- Family tree.
- Parse tree.
- Unix file hierarchy.
- Phylogeny tree.

Binary Search Tree

Other Types of Trees

Other types of trees.
- Family tree.
- Parse tree.
- Unix file hierarchy.
- Phylogeny tree.
- GUI containment hierarchy.


Other Types of Trees

Other types of trees.
- Family tree.
- Parse tree.
- Unix file hierarchy.
- Phylogeny tree.
- GUI containment hierarchy.
- Binary search trees.
- NCAA basketball tournament.
- Barnes-Hut tree for fast N-body simulation.
Linked Structures Overview

**Linked structures.** Simple abstraction for customized access to data.

**Singly linked structures.**
- Linked list.
- Circular linked list.
- Parent-link tree.

**Doubly linked structures.**
- Binary tree.
- Patricia tries.
- Doubly linked circular list.

Conclusions

**Sequential allocation:** supports indexing, fixed size.
**Linked allocation:** variable size, supports sequential access.

Linked structures are a central programming abstraction.
- Linked lists.
- Binary trees.
- Graphs.
- Sparse matrices.

Announcements

Thinking about majoring in Computer Science?

Or doing the Certificate in Applications of Computing?

Then: visit the all-new "Life in the Computer Science Department: A Guide for the Humble Undergraduate":
- a handy FAQ that answers many many questions

And/Or: Come talk to me and/or Kevin

AND CERTAINLY attend at least one of:
- C.S. open house for BSE freshmen **TODAY**, Friend Convocation Room, 5:45: tours, demos, pizza (AB's welcome)
- C.S. open house for AB sophomores also **TODAY**, C.S. Tea Room, 4 PM (but no pizza, and maybe fewer demos) (BSE's welcome)