CS410 Lecture 4
Neal Glew
HW1, Basic Structures

HW1 Q10 Solution
• Key maps directly into data array, but may not be initialised
• Need to know if entry initialised
• Keep stack of indices of initialised entries
• Need to search stack
• Keep another array to index into stack

HW1 Q10 Solution (cont.)

Basic Datastructures (CLR 11)
• Stacks
• Queues
• Linked Lists
• Sentinels
• Pointers as arrays
• Trees
Stacks

- Operations
  - Create empty stack
  - Is stack empty?
  - Push element
  - Pop element
- Stacks are LIFO, last element is called top of stack, first element is called bottom of stack

Stacks using Arrays

```java
class Stack {
    private int top;
    private Object items[];
    public Stack(int cap) {
        top = 0;
        items = new Object[cap];
    }
    public boolean empty() {
        return top <= 0;
    }
    public void push(Object item) {
        if (top >= items.length)
            throw new Full();
        items[top++] = item;
    }
    public Object pop() {
        if (empty())
            throw new Empty();
        return items[--top];
    }
}
```

Queues

- Operations
  - Create empty queue
  - Is queue empty?
  - Enqueue item
  - Dequeue item
- Queues are FIFO, first element is called front of queue, last element is called back of queue

Queues using Arrays

```java
class Queue {
    private int head, tail;
    private Object items[];
    public Queue(int cap) {
        head = tail = 0;
        items = new Object[cap];
    }
    public void enqueue(Object item) {
        int nt = (tail+1)%items.length;
        if (nt == head)
            throw new Full();
        items[tail] = item; tail = nt;
    }
    public boolean empty() {
        return head == tail;
    }
    public Object dequeue() {
        if (empty())
            throw new Empty();
        Object item = items[head];
        head = (head+1)%items.length;
        if (nt == head)
            throw new Empty();
        items[nt] = item; tail = nt;
    }
}
```

Linked Lists

- Sequence of nodes that are linked
- Singly linked and doubly linked - next and prev
- Head pointer, head and tail pointer, circular list
- List can be ordered or unordered

Linked Lists Pictorially
Linked List Code

class SinglyLN {
    Object data;
    SinglyLinkedNode next;
}

class DoublyLN {
    Object data;
    DoublyLN prev;
    DoublyLN next;
}

public class SLinkedList {
    private SinglyLN head;
    public Object find(Object item) {
        SinglyLN current = head;
        while (current!=null && item!=current.data)
            current=current.next;
        if (current!=null)
            return current.data;
        else
            throw new Absent();
    }
    public void insert(Object item) {
        SinglyLN nn = new SinglyLN();
        nn.data=item;
        nn.next=head;
        head=nn;
    }
    public void delete(Object item) {
        SinglyLN prev = null;
        SinglyLN current = head;
        while (current!=null && item!=current.data) {
            prev=current;
            current=current.next;
        }
        if (current==null) return;
        if (prev==null)
            head=current.next;
        else
            prev.next=current.next;
    }
}

Linked List Code

public class DLinkedList {
    private DoublyLN head;
    // Find same as SLinkedList
    public void insert(Object item) {
        DoublyLN nn = new DoublyLN();
        nn.data=item;
        nn.next=head.next;
        if (head!=null) head.prev=nn;
        nn.prev=null;
        head.next=nn;
    }
    public void delete(Object item) {
        DoublyLN current = head.next;
        while (current!=head && item!=current.data)
            current=current.next;
        if (current==head) return;
        current.prev.next=current.next;
        current.next.prev=current.prev;
    }
    DLinkedList() {
        head = new DoublyLN();
        head.next=head.prev=head;
    }
}

Sentinels

• Avoid the special cases
• Use a sentinel - a dummy node with no data that eliminates the special case

Doubly Linked Lists in One Field

h xor (h xor b) = b
a xor (a xor c) = c
b xor (b xor d) = d
d xor (b xor d) = b
c xor (a xor c) = a
a xor (a xor h) = h

Does not work in Java
Pointers as Arrays

- Old languages do not have dynamic memory
- O/S kernel might not either
- Implement your own dynamic memory with large array
  - Pointers are indices into array
  - Free list of available array entries

Trees

- Linked structure with more than one "next"
- If balanced then logarithmic
  - e.g. binary tree of depth $k$ holds $2^k$ items
- Can have parent pointers like doubly linked lists
- Binary trees have 2 children
- $k$-ary trees have $k$ children

Tree Code

class TreeNode {
    Object data;
    TreeNode left, right;
}
TreeNode find(TreeNode tree, Object key) {
    while (tree!=null)
        if (tree.data==key) return tree;
        else if (tree.data<key) tree=tree.right;
        else tree=tree.left;
    return null;
}

Variable Breadth Trees

- Nonfixed number of children
- If fixed at node creation time use array
- Otherwise use a Vector or a linked list

class TreeNode {
    Object data;
    TreeNode leftChild, rightSibling;
    TreeNode parent, leftSibling; // For doubly linked
}

Trees with Linked Children