Lists

CSE 373
Data Structures
Lecture 3
Readings

• Reading
  › Sections 3.1 - 3.2
List ADT

• What is a List?
  › Ordered sequence of elements $A_1, A_2, \ldots, A_N$
• Elements may be of arbitrary type, but all are the same type
• Common List operations are
  › Insert, Find, Delete, IsEmpty, IsLast, FindPrevious, First, Kth, Last
Simple Examples of List Use

- Polynomials
  - $25 + 4x^2 + 75x^{85}$
- Unbounded Integers
  - 4576809099383658390187457649494578
- Text
  - “This is an example of text”
List Implementations

• Two types of implementation:
  › Array-Based
  › Pointer-Based
List: Array Implementation

• Basic Idea:
  › Pre-allocate a big array of size MAX_SIZE
  › Keep track of current size using a variable count
  › Shift elements when you have to insert or delete

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>...</th>
<th>count-1</th>
<th>MAX_SIZE-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A₁</td>
<td>A₂</td>
<td>A₃</td>
<td>A₄</td>
<td>...</td>
<td>Aₙ</td>
<td></td>
</tr>
</tbody>
</table>
List: Array Implementation

Insert Z

0 1 2 3 4 5 MAX_SIZE-1
A B C D E F

A B Z C D E F MAX_SIZE-1
Array List Insert Running Time

• Running time for N elements?
• On average, must move half the elements to make room – assuming insertions at positions are equally likely
• Worst case is insert at position 0. Must move all N items one position before the insert
• This is O(N) running time. Probably too slow.
List: Pointer Implementation

• Basic Idea:
  › Allocate little blocks of memory (nodes) as elements are added to the list
  › Keep track of list by linking the nodes together
  › Change links when you want to insert or delete

![Diagram of list nodes with pointers to Value and Next fields]

10/4/02  Lists - Lecture 3  9
Pointer-Based Linked List

$pL$ node

Value  Next

Value  Next

node

NULL
Pointer-based Insert

Insert the value E after P
Insertion After

InsertAfter(p : node pointer, v : thing): {
  x : node pointer;
  x := new node;
  x.value := v;
  x.next := p.next;
  p.next := x;
}
Linked List with Header Node

Advantage: “insert after” and “delete after” can be done at the beginning of the list.
**Pointer Implementation Issues**

- Whenever you break a list, your code should fix the list up as soon as possible
  - Draw pictures of the list to visualize what needs to be done
- Pay special attention to boundary conditions:
  - Empty list
  - Single item – same item is both first and last
  - Two items – first, last, but no middle items
  - Three or more items – first, last, and middle items
Pointer List Insert Running Time

- Running time for N elements?
- Insert takes constant time (O(1))
- Does not depend on input size
- Compare to array based list which is O(N)
To delete the node pointed to by P, need a pointer to the previous node.
Doubly Linked Lists

- FindPrev (and hence Delete) is slow because we cannot go directly to previous node
- Solution: Keep a "previous" pointer at each node
Double Link Pros and Cons

• Advantage
  › Delete (not DeleteAfter) and FindPrev are fast

• Disadvantages:
  › More space used up (double the number of pointers at each node)
  › More book-keeping for updating the two pointers at each node
Unbounded Integers Base 10

-4572

\[\begin{array}{cccc}
10^3 & 10^2 & 10^1 & 10^0 \\
4 & 5 & 7 & 2 \\
null & & & -1 \\
\end{array}\]

X : node pointer

348

\[\begin{array}{cccc}
10^2 & 10^1 & 10^0 & \text{sign} \\
3 & 4 & 8 & 1 \\
null & & & \end{array}\]

Y : node pointer
Zero

```
null ← -1
null ← 1
```

10/4/02 Lists - Lecture 3
Recursive Addition

- Positive numbers (or negative numbers)

```
3427 + 898
  __________
  3425

7 + 8
  __________
  15

342 + 89
  __________
  431

Recursive calls
```
Recursive Addition

• Mixed numbers

\[
\begin{align*}
3427 & \quad 7 \\
-898 & \quad -8 \\
\hline
\quad & \quad 9 \\
\quad & \quad -10 \\
\quad & \quad -1
\end{align*}
\]
Example

• Mixed numbers

\[
\begin{array}{c}
1000000 \\
-999999
\end{array}
\quad \begin{array}{c}
0 \\
-9
\end{array}
\quad \begin{array}{c}
1000000 \\
-999999
\end{array}
\quad \begin{array}{c}
1 \\
-1
\end{array}
\quad \begin{array}{c}
-10 \\
-1
\end{array}
\quad \begin{array}{c}
0
\end{array}
\]

Recursive calls