Lists

CSE 373
Data Structures
Lecture 3

List ADT

- What is a List?
  - Ordered sequence of elements A_1, A_2, ..., 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^6
- 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
List: Array Implementation

Insert Z

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>

MAX_SIZE-1

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>Z</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>

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

Pointer-Based Linked List

```
pl

null

node

Value

Next

null

node

Value

Next
```

Pointer-based Insert

Insert the value E after P

```
InsertAfter(p : node pointer, v : thing):
  x : node pointer;
  x := new node;
  x.value := v;
  x.next := p.next;
  p.next := x;
```

Insertion After
Linked List with Header Node

- Header node
- Value
- Next
- First actual list node
- NULL

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

Pointed 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)

Linked List Delete

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

348

X: node pointer

null

10^3 10^2 10^1 10^0 sign

null

Y: node pointer

null

10^3 10^2 10^1 10^0 sign

Zero

null

-1

Recursive Addition

Positive numbers (or negative numbers)

3427 +898

+8

7

342

+89

10

Recursive calls

-898

Recursive calls

Recursive Addition

Mixed numbers

3427 -898

-8

7

342

-89

-10

Recursive calls

Example

Mixed numbers

1000000

-999999

0

100000

-9

-99999

-10

Recursive calls

-1