CS 187: Programming with Data Structures (Spring 2010)

Lecture 11: More on Linked List

Rui Wang
Summary of Linked List

- **Links** are fundamental elements in linked list.
- Each **Link** contains data and a reference to another **Link** object.
- The **LinkedList** object keeps the reference to the first link in the list.
- We can dynamically creates many links and insert them to the list.
- Insertion and deletion are fast because nothing needs to be moved.
Summary of Linked List

• No simple indexing
• Inserting and deleting links at the beginning of the list are easy.
• Inserting and deleting links at the end of the list require traversing the entire list.
Today

- Double-ended Linked List
- Sorted Linked List
- Doubly Linked List
- Iterators
- Debugging
- Assignment 3 questions
Double-ended Linked List

- Similar to an ordinary linked list, but in addition to keep ‘first’, it also keeps a reference to the ‘last’ element in the list.
Double-ended Linked List

• What happens when the list is empty? Has only one element?
Double-ended Linked List

- Code
- What about `deleteLast()`?
Double-ended Linked List

• Code
• What about `deleteLast()`?
  – Unfortunately, having a reference to the ‘last’ element does not help deleting it. We need the next-to-last link.

• A trick question: you are given the reference to a link, and you know it’s somewhere in the middle of the list. How do you delete it in O(1) time?
Using Linked List

• Since Linked List is interchangeable with array in many cases, we can re-implement Stacks and Queues using Linked List.

• Example of Stack
  – Note that the fact it uses a linked list instead of an array is hidden.
  – The stack interface methods are exactly the same with before.
Using Linked List

• Since Linked List is interchangeable with array in many cases, we can re-implement Stacks and Queues using Linked List.

• Example of Queue
  – Can be easily implemented using a double-ended linked list.
Using Linked List

• Abstraction

Description of a data structure without regard to its detailed implementation.

1. data elements are hidden
2. interface methods are public

• Some more familiar examples
  – Memory allocation
  – File copy
  – Parsing arithmetic expressions
Sorted Linked List

• Elements are sorted in the list.
• Workshop demo
• This is similar to a sorted array.
• Pros
  – Insertion is fast (as no element needs to be moved)
  – Deleting the smallest element is fast
• Cons
  – Somewhat more complicated to implement
  – No benefit in searching (binary search is unavailable)
Sorted Linked List

• Code
• What about searching?
• **Cost analysis**
• Insertion?
• Deletion?
  – What about deleting the smallest element?
• Searching?
Sorted Linked List

• Code
• What about searching?
• **Cost analysis**
  • Insertion \( \rightarrow O(N) \)
  • Deletion \( \rightarrow O(N) \)
    – What about deleting the smallest element \( \rightarrow O(1) \)
• Searching \( \rightarrow O(N) \)
Sorted Linked List – Applications

- **Insertion Sort**
  - Still $O(N^2)$ comparison cost
  - But the copy cost is much lower

- **Implementing Priority Queue**
Doubly Linked List

• A doubly linked list has bidirectional references, one pointing to the next link, and one pointing to the previous link.
Doubly Linked List

class Link {
    public Record data;
    public Link next;
    public Link prev;
}

• **Pros**: flexibility
• **Cons**: complexity, memory consumption
• For clarity, we often call the ordinary linked list explicitly as **singly linked list**.
Doubly Linked List

- **Note**: Do not confuse Doubly Linked List with Double-ended List!

  Doubly Linked List $\neq$ Doubled-ended List

- **Traversal**:
  - displayForward()
  - displayBackward()
Doubly Linked List

• **Insertion:**
  – insertFirst()
  – insertLast()
  – insertAfter()

Make sure you understand code such as
```java
current.next.previous = newLink;
```
Doubly Linked List

• Deletion:
  – deleteFirst()
  – deleteLast()
  – deleteKey()
Iterators

• We need a convenient way to iterate through all elements in the list.
• For an array, we can use a loop over indices.
• For Linked List, the `display()` method does the iteration, but we need to wrap the relevant code to a class to allow for reuse.

• `ListIterator`
  – More on it during the discussion session
Debugging

• Eclipse Debugger
  – Step Over
  – Step In
  – Watch Variables
  – Breakpoint
  – Conditional breakpoint

• Online video tutorial

• println debugging
Assignment 3

- Demonstrate the output
- Idea of the guideline (next slide)
  - Each outer loop works on one row at a time
  - If this is a new row, you start from position 0
  - If this is an old row you’ve come back to, you start from the next position of where you were
set current position to 0
while (true) {
    loop from current position to n { // current row
        test conflict;
        if conflict free {
            push current position to stack;
            if this is the last row {
                print solution;
            } else { // prepare for the next row
                reset position to 0;
                break from loop;
            }
        }
    }
    if at the end of the row // no valid position
        current position = stack.pop() + 1;
}
Assignment 3

- You can keep a variable named ‘current_row’, but it’s not really necessary as stack.size() tells you how many queens you’ve already placed.

- Some common questions:
  Q: How to check conflicts with existing queens?
  A: Iterate through existing queens stored in the stack, and apply the formula discussed in class (see Queues lecture)
Assignment 3

• Some common questions:
  
  Q: How to iterate through existing queens in the stack? Isn’t that I have to pop out elements?
  
  A: 1) Yes you can, but inefficient
      2) Use stack’s `get(int index)` method
      3) Use a stack Iterator
Assignment 3

• Some common questions:
  Q: How do I find all solutions?
  A: When you’ve found a solution, print it out (call printSolution method). Then just pretend that you haven’t found a solution yet

Q: When should I quit the loop?
A: When you try to pop but the stack is empty
Announcements

• Assignment 3 is due on tomorrow (Friday).

• Extended TA Office Hours:
  – 2:00—5:00pm today
  – 2:30—4:00pm tomorrow (Friday)

• Assignment 2 grades will be released in SPARK by Friday