COMP 250

Lecture 5

doubly linked lists
Java LinkedList

Sept. 16, 2016
Doubly linked lists

Each node has a reference to the next node and to the previous node.
class DNode< E > {

    DNode< E >  next;
    DNode< E >  prev;
    E          element;

    // constructor

    DNode( E e ) {
        element = e;
        prev = null;
        next = null;
    }

}
Motivation:
recall removeLast( ) for singly linked lists

The only way to access the element before the tail was to loop through all elements from the head. This took time $O(N)$ where $N$ is the size of the list.
For a doubly linked list, removing the last element is much faster.

```java
removeLast(){
    tail = tail.prev
    tail.next = null
    size = size - 1
}
```
## Time Complexity  \[(N = \text{list size})\]

<table>
<thead>
<tr>
<th></th>
<th>array list</th>
<th>SLinkedList</th>
<th>DLinkedList</th>
</tr>
</thead>
<tbody>
<tr>
<td>addFirst</td>
<td>O( N )</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>removeFirst</td>
<td>O( N )</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>addLast</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>removeLast</td>
<td>O( 1 )</td>
<td>O( N )</td>
<td>O( 1 )</td>
</tr>
</tbody>
</table>
List Operations

get(i)
set(i,e)
add(i,e)
remove(i)

For a linked list, many operations require access to node i.

The “edge cases” (i = 0, i = size – 1) usually require extra code, which can lead to coding errors.
Common linked list trick: avoid edge cases with “dummy nodes”
class DLinkedList<E>{

    DNode<E> dummyHead;
    DNode<E> dummyTail;

    int size;

    // constructor
    DLinkedList<E>(){
        dummyHead = new DNode<E>();
        dummyTail = new DNode<E>();
        dummyHead.next = dummyTail;
        dummyTail.prev = dummyHead;
        size = 0;
    }
}
Q: How many objects in total in this figure?

A: \[ 1 + 6 + 4 = 11 \]
remove(i) {  // recall end of lecture 3 on arrays

    node = getNode(i)  // next slide
    node.next.prev = node.prev
    node.prev.next = node.next
    size = size - 1
}

BEFORE

i - 1

node

i

i + 1

AFTER

next  prev  element

next  prev  element
getNode(i) {

    // check that 0 <= i < size (omitted)

    node = dummyHead.next
    for (k = 0; k < i; k++)
        node = node.next
    return node
}
More efficient (half the time)...

getNode( i ) {

    if ( i < size/2 ){
        // iterate from head
        node = dummyHead.next
        for (k = 0; k < i; k ++)
            node = node.next;
    }

    else{
        // iterate from tail
        node = dummyTail.prev
        for (k = size-1; k > i; k --)
            node = node.prev
    }

    return node
}
<table>
<thead>
<tr>
<th>Operation</th>
<th>array list</th>
<th>SLinkedList</th>
<th>DLinkedList</th>
</tr>
</thead>
<tbody>
<tr>
<td>addFirst</td>
<td>O( N )</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>removeFirst</td>
<td>O( N )</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>addLast</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>removeLast</td>
<td>O( 1 )</td>
<td>O( N )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>remove( i )</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
## Time Complexity  \((N = \text{list size})\)

<table>
<thead>
<tr>
<th>Operation</th>
<th>array list</th>
<th>SLinkededList</th>
<th>DLinkedList</th>
</tr>
</thead>
<tbody>
<tr>
<td>addFirst</td>
<td>O( N )</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>removeFirst</td>
<td>O( N )</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>addLast</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>removeLast</td>
<td>O( 1 )</td>
<td>O( N )</td>
<td>O( 1 )</td>
</tr>
<tr>
<td>remove( i )</td>
<td>O(N)</td>
<td>O( N )</td>
<td>O( N )</td>
</tr>
</tbody>
</table>

As I will discuss that later, “O( )” ignores constant factors.
Java LinkedList class

https://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html

It uses a *doubly linked list* as the underlying data structure.

It has some methods that ArrayList doesn’t have e.g.:

- addFirst()
- removeFirst()
- addLast()
- removeLast()
Q: What is the time complexity of the following?

```java
LinkedList<E> list = new LinkedList<E>();

for (k = 0; k < N; k++) // N is some constant
    list.addFirst(new E(...)); // or addLast(..)

for (k = 0; k < k < list.size(); k++)
    list.get(k);
```
Q: What is the time complexity of the following?

```java
LinkedList<E> list = new LinkedList<E>();

for (k = 0; k < N; k ++)
    // N is some constant
    list.addFirst( new E( .... ) );  // or addLast(..)

A: \[ 1 + 1 + 1 + \ldots + 1 = N \]

\[ O(N) \]

for (k = 0; k < k < list.size(); k ++)
    list.get( k );
```

I am omitting what I would do with this element since that’s not the point here e.g. I could print it.
Q: What is the time complexity of the following?

```java
LinkedList<E> list = new LinkedList<E>();
for (k = 0; k < N; k++) // N is some constant
    list.addFirst(new E( .... )); // or addLast(..)
```

A: \[ 1 + 1 + 1 + \ldots + 1 = N \quad O(N) \]

```java
for (k = 0; k < list.size(); k++) // size == N
    list.get(k);
```

A: \[ 1 + 2 + 3 + \ldots + N = \frac{N(N+1)}{2} \quad O(N^2) \]
ASIDE: Java ‘enhanced for loop’

```java
for (k = 0; k < list.size(); k++)
    .......
```

A more efficient way to iterate through elements in a `LinkedList` object is to use:

```java
for (E e : list)
    // ‘list’ references the LinkedList<E> object
    // Do something with each element e in list
```
What about “Space Complexity”? 

All three data structures use space $O(N)$ for a list of $N$ elements.
Java terminology
(time permitting)

• method “signature”
  • name
  • number and type of parameters,
  • return type

• method “overloading”
  • add( int index, E element)
  • add( E element )

• remove(E element)
• remove(int i)
Java terminology

What is method “overloading” vs. “overriding”?  

Classes can “inherit” methods from other classes.

Sometimes you do not want a class to inherit the method, however, and so you “override” it by writing a more suitable one.

We will learn about inheritance formally at the end of the course.....
Announcements

• Assignment 1 posted (due in ~2 weeks)

• Exercises for singly linked lists (practice coding)

• Eclipse (IDE) tutorials next week.
  Goodbye DrJava! Hello Eclipse!
I asked the TA’s which IDE they use:

- Eclipse or sublime
- jetbrains, IntelliJ
- Eclipse
- Eclipse, also IntelliJ
- Eclipse
- Simple text editor / vim + command line. (However I had to use Eclipse and DrJava in the past.)
- Eclipse, but I like using text editor + command line for small things.
- Eclipse (netbeans for GUIs)