Compsci 201

Linked-List Code and APTs

Susan Rodger
February 26, 2020
M is for …

- **Markov, Maps**
  - Assignment you are working on
- **Method**
  - A function by any other name
- **Memory**
  - New Node, New ArrayList, …
Announcements

• Exam 1 – Ask for Regrade in Gradescope
  • I do all the regrades
  • Ask for Regrades in Gradescope by Sunday, March 1

• Assignment P3 due tomorrow

• Assignment P4 out Friday with a Part1 and Part2
  • Part 1 due March 5, Part 2 due March 19

• APT 4 due Tuesday!
Plan for Today and Week

• Review Linked List APTs
  • ListNode class and using it in a project
  • Creating your own main for testing

• Changing linked lists in methods
  • Invariants, pass-and-return,

• Visualize, reason, think, code
Markov 2: Efficiency

- Idea related to machine learning
  - Given a training text, use it to create a model
  - Using the model, generate random text
- Infinite Monkey Theorem?
  - Don't type at random
  - Use letter frequencies!!
Naïve, Brute Force Idea

• Given training text "the theatre through that helps"
  • Generate random text based on frequencies
• For a model-2 Markov process: start with "th"
  • Characters after "th": {"e","e","r","a"}
  • Choose one at random, say "e": generate!
    • Now use with "he", since "th" + "e" = "he"
    • Following "he": {" ", "a", "l"}
• Why naïve? Re-scan text every time for follows
Finding Follow Characters

• Scan entire text looking for key
  • https://coursework.cs.duke.edu/201spring20/p3-markovpart2-sp20

• Loop $O(T)$ for `myText` with $T$ characters
  • Again?

```java
@override
public ArrayList<String> getFollows(String key){
    ArrayList<String> follows = new ArrayList<String>();

    int pos = 0; // location where search for key in text starts
    while (pos < myText.length()){  
        int start = myText.indexOf(key, pos);
        if (start == -1){
            //System.out.println("didn't find "+key);
            break;
        }

        if (start + key.length() >= myText.length()){  
            //System.out.println("found end with "+key);
            follows.add(PSEUDO_EOS);
            break;
        }  

        // next line is string equivalent of myText.charAt(start+key.length())
        String next = myText.substring(start+key.length(), start+key.length()+1);
        follows.add(next);
        pos = start+1; // search continues after this occurrence
    }

    return follows;
}
```
Don't Scan N times, Scan Once

- We generate N random characters
  - Get follows N times, each O(T), total is O(NT)
- Suppose we find all N-grams, e.g., 2-grams
  - "th" -> {"e", "e", "r", "a"}
  - "he" -> {" ", "a", "l"}
  - ...
- Map of 2-gram to ArrayList of following chars
  - Create in O(T) time. Get follows is O(1)
    - So total is O(N + T)
Markov Big Picture

- Use BaseMarkov as a start, create EfficientMarkov
  - Make constructors work, create map
  - @Override getFollows to be O(1) not O(T)
  - Benchmark these programs

- Use WordGram rather than String
  - Generate word-based random text, not char
  - String is collection of characters, WordGram is collection of Strings
  - Use same idea for map, but use WordGram
From Last Time WOTO (go over)

What’s in a Node?

• Some information
• Place to snap another node

• In Java we’ll see
  • String reference: info
  • Node reference: next
Linked list with one Node

First

Bo
1) Add a Node to the front

First

Bo
1) Add a Node to the front

N nodes in linked list
Running time to add one Node to front?
2) Add a Node to the end

First

Fa

Bo
4) Again - Add a Node to the end

First

Fa

Bo

Me
4) Again - Add a Node to the end

N nodes in linked list
Running time to add one node to end?
5) First and Last
Add a Node to the end

First

Fa

Bo

Me

So

Last
5) First and Last
Add a Node to the end

N nodes in linked list
Running time to add one node to end?
Where does Node go for APT?

- Where does the class Node live? Use ListNode
  - In same package/folder as class, e.g., APT
Where does Node go for P4: next assignment

• Where does the class Node live?
  • Nested/inner class, e.g., in LinkedStrand.java

```java
private class Node{
    String info;
    Node next;
    public Node(String key, Node link){
        this.info = key;
        this.next = link;
    }
}
```
Visualizing Running Code – Java Tutor

• Simple node demo:
• New nodes added at front: 3->2->1->0
Code - create and main

```java
public Node create(int n) {
    Node list = null;
    for (int k=0; k<n; k++) {
        list = new Node(k, list);
    }
    return list;
}

public static void main(String[] args) {
    ListDemo ld = new ListDemo();
    Node list = ld.create(4);
    int size = ld.count(list);
    System.out.printf("size is %d", size);
}
```
public class ListDemo {
    private class Node {
        int info;
        Node next;
        Node(int val, Node follow) {
            this.info = val;
            this.next = follow;
        }
    }
    public int count(Node list) {
        int total = 0;
        while (list != null) {
            total += 1;
            list = list.next;
        }
        return total;
    }
}
Alternatives to “plain” linked lists

- Doubly-linked lists, node has .prev and .next
  - Facilitates iterating from front and back
  - Some code easier don’t need “peek ahead”
Alternatives to “plain” linked lists

- Lists with “header nodes”
  - Create “dummy” node, not part of list
  - First now always points to a “dummy” node
  - Last always points to a node
    - Google "dummy header node" for details
Alternatives to “plain” linked lists

• Lists with “header nodes”
  • Create “dummy” node, not part of list
  • First now always points to a “dummy” node
  • Last always points to a node
    • Google "dummy header node" for details

Here is a list with 1 node
Source code in the wild

  - OpenJDK Java 8
  - LinkedList<String> means Node<String>
    - LinkedList<Integer> means Node<Integer>

```java
private static class Node<E> {
    E item;
    Node<E> next;
    Node<E> prev;

    Node(Node<E> prev, E element, Node<E> next) {
        this.item = element;
        this.next = next;
        this.prev = prev;
    }
}
```
Class Invariant

- Invariant: always true after each method executes
  - Maintain first and last as pointers to the first and last nodes. What does this mean?

- Note ||

```java
89  /**
90  * Pointer to first node.
91  * Invariant: (first == null && last == null) ||
92  * (first.prev == null && first.item != null)
93  */
94  transient Node<E> first;
95
96  /**
97  * Pointer to last node.
98  * Invariant: (first == null && last == null) ||
99  * (last.next == null && last.item != null)
100  */
101  transient Node<E> last;
```
Maintaining Invariant

• How do we remove the first node?
  • If there is a node, re-assign first, notice assert!
• What is GC?
• Lines 177-178 …
  • What is first?
• Line 182?

```java
/**
 * Unlinks non-null first node f.
 */
private E unlinkFirst(Node<E> f) {
    // assert f == first && f != null;
    final E element = f.item;
    final Node<E> next = f.next;
    f.item = null;
    f.next = null; // help GC
    first = next;
    if (next == null)
        last = null;
    else
        next.prev = null;
    size--;
    modCount++;
    return element;
}
```
/**

* Unlinks non-null first node f.
*/

private E unlinkFirst(Node<E> f) {
    // assert f == first && f != null;
    final E element = f.item;
    final Node<E> next = f.next;
    f.item = null;
    f.next = null; // help GC
    first = next;
    if (next == null)
        last = null;
    else
        next.prev = null;
    size--;
    modCount++;
    return element;
}
ListCount - ListNode APT

• Must use the ListNode class in same package
  • Not an inner class, used by all APT programs
  • Create project, create class, copy from APT
• [https://www2.cs.duke.edu/csed/newapt/listcount.html](https://www2.cs.duke.edu/csed/newapt/listcount.html)
Similarities and Summary

• Code typically uses `while (list != null)`
  • Move `list = list.next` in body of loop

• If you want to stop on last node rather than after?
  • Must make sure that list cannot be initially null

```java
private Node getLast(Node list) {
    while (list.next != null) {
        list = list.next;
    }
    return list;
}
```
WOTO

Josh Bloch

- Led design of Java Collections Framework
- Formerly Java Chief Architect at Google
- Professor of the Practice CMU

**APIs should be easy to use and hard to misuse.** It should be easy to do simple things; possible to do complex things; and impossible, or at least difficult, to do wrong things.
APIs should be easy to use and hard to misuse. It should be easy to do simple things; possible to do complex things; and impossible, or at least difficult, to do wrong things.

```java
public class LinkedList<E>
    extends AbstractSequentialList<E>
    implements List<E>, Deque<E>, Cloneable, java.io.Serializable
```
Katherine Johnson

- Died Monday, Feb 24
- “Hidden Figures” – book/movie
- NASA Mathematician – “computer”
- Calculated by hand, trajectories for space flights.
- One of first African-American women to work as NASA Scientist
- “In the early days of NASA women were not allowed to put their names on the reports – no woman in my division had had her name on a report. I was working with Ted Skopinski and he wanted to leave and go to Houston ... but Henry Pearson, our supervisor – he was not a fan of women – kept pushing him to finish the report we were working on. Finally, Ted told him, "Katherine should finish the report, she's done most of the work anyway." So Ted left Pearson with no choice; I finished the report and my name went on it, and that was the first time a woman in our division had her name on something”
Modify and Return linked list

- If we pass a pointer to first node and ..
  - Want to "remove first"
  - We must return a pointer to modified list
- void change(ListNode first)
  - Call change(list)
    - first = first.next
  - list not changed after call
Removing first node?

• We’d have to re-assign to first
  • Straight-forward with instance variables

• Alternative: one method that changes list
  • Idiom: pass and return
    • cannot “change” a parameter \textit{via} assignment
  • In Java parameters are copies, pass-by-value
  • Assigning to parameter with = has no effect!
Example: remove all X’s from list

- Singly linked list, we need to link around
  - Check `list.next.info` remove `deadNode`
  - Remove: `list.next = list.next.next`
    - Should we write `deadNode.next = null` too?
deleteAll(list, target)

• Reason about invariant and special cases
  • First node, Last Node, No Nodes, All Nodes
• See `LowLevelLinkDemo.java`
• See [LowLevelLinkDemo.java](https://coursework.cs.duke.edu/classcode/)

```java
44  // invariant: list != null
45  while (list.next != null) {
46      if (list.next.info.equals(target)) {
47          list.next = list.next.next;
48      }
49      else {
50          list = list.next;
51      }
52  }
```
private Node deleteAll(Node list, String target) {
    Node first = list;
    if (first == null) return null;

    // invariant: list != null
    while (list.next != null) {
        if (list.next.info.equals(target)) {
            list.next = list.next.next;
        }
        else {
            list = list.next;
        }
    }

    // all done except first node
    if (first.info.equals(target)) {
        return first.next;
    }

    return first;
}
What does pass-by-value mean?

• Pass a copy of the variable

```python
list1 = ld.deleteAll(list1, "corn");
```

```python
list1 = ld.deleteAll(list1, "squash");
```
RemoveMin APT

- [https://www2.cs.duke.edu/csed/newapt/removemin.html](https://www2.cs.duke.edu/csed/newapt/removemin.html)
- This method cannot be void
  - Can the first node be minimal node?

- **Usage:** `list = removeMin(list)`
  - Pass-and-return: call, modify, return, assign
  - Works when list = 5->8->6->9 and 5 removed
Idiom: pass-and-return

• Change the list passed in, return the list.
  • Assign in the call, e.g. \( x = \text{changeUp}(x) \)

```java
Thing xx = new Thing();
change(xx);
// can xx be different after call?
// can write xx.mutate()
// cannot assign to xx in change
xx = changeUp(xx);
```
Invariants

• Class level: true before each method executes
  • Established at construction
  • Re-established by each method

• Loop level: true before each loop guard evaluation
  • Established before first iteration of loop
  • Re-established after each loop iteration

• Reason formally and informally about code
WOTO