Announcements, Coming up

- Wednesday, 2/21 (today)
  - Project 3: DNA (Linked List) available, due 3/4
  - APT Quiz 1 due tomorrow
    - LunchPlans, Follower, OverEnroll are on practice quiz!
- Monday 2/26
  - Nothing due
- Next Wednesday 2/28
  - APT 5 (linked list problems) due
    - Has mix of required, optional (full-value extra credit), and practice (half-value extra credit).

Today's agenda

1. Linked list problems/exercises

2. Testing & Debugging, Concepts & Tools

3. More DIYLinkedList
Person in CS: Barbara Liskov

- Turing Award Winner in 2008 for contributions to practical and theoretical foundations of programming language and system design, especially related to data abstraction, fault tolerance, and distributed computing.

- "The advice I give people in general is that you should figure out what you like to do, and what you can do well—and the two are not all that dissimilar, because you don’t typically like doing something if you don’t do it well. ... So you should instead watch—be aware of what you’re doing, and what the opportunities are, and step into what seems right, and see where it takes you."

ListNode Pointer Problems

Drawing Pictures

- Visualization is very important: Draw pictures!
- Try your algorithm/code one step at a time with:
  - 0 nodes
  - 1 node
  - 2 nodes
  - 3 nodes
- Check boundary conditions
- Is this pointing to what I think it’s pointing to? Check!
Append linked lists of ListNodes

• Append listB to listA using...
  • O(1) additional memory,
  • No copying values,
  • Just changing pointers in the input lists.

```
listA  2 ——> 0 ——> 1

listB  9 ——> 7
```

Append linked lists of ListNodes

• Conceptual algorithmic questions:
  • How to get a reference to the last node of listA?
  • How to update last node to point to the first node of listB?
  • What to return?

```
listA  2 ——> 0 ——> 1

listB  9 ——> 7
```

How to get a reference to the last node?

Starting with the standard list traversal idiom we know...

```
while (listA != null)
    listA = listA.next;
```

But after exiting this loop, listA is just null. Stop one node before...

```
while (listA.next != null)
    listA = listA.next;
```
How to update last node to point to the first node of listB?
Recall: Writing `node.next = otherNode;` makes node \(\rightarrow\) (point to) otherNode.

```
last.next = listB;
```

What to return?
If listB is appended to the end of listA, need to return a reference to the first node of listA.

```java
while (listA.next != null) {
    lista = listA.next;
    listA.next = listB;
    return lista;
}
```

Append linked lists of ListNodes:
Putting it all together

```java
public static ListNode append(ListNode listA, ListNode listB) {
    ListNode first = listA;
    while (listA.next != null) {
        listA = listA.next;
    }
    listA.next = listB;
    return first;
}
```

• Reminding again: Accomplished with O(1) additional memory and without copying any values.
• Not necessarily a lot of lines of code, but...
• easy to get lost without planning and visualization before/while coding.
Consider the mystery method shown in the code. Suppose listA has \( n \) nodes and listB has \( m \) nodes. How many new nodes are created in the call to `mystery(listA, listB)`?  

```java
class Mystery {
    public static ListNode mystery(ListNode listA, ListNode listB) {
        ListNode myList = new ListNode(listA.info);
        ListNode current = myList;
        ListNode next = listA;
        while (next != null) {
            current.next = new ListNode(next.info);
            current = current.next;
        }
        return myList;
    }
}
```

Answer: \( n + m \)

---

Same mystery method shown in the code. Suppose listA has \( n \) nodes and listB has \( m \) nodes. The runtime complexity of `mystery` is...  

```java
class Mystery {
    public static ListNode mystery(ListNode listA, ListNode listB) {
        ListNode myList = new ListNode(listA.info);
        ListNode current = myList;
        ListNode next = listA;
        while (next != null) {
            current.next = new ListNode(next.info);
            current = current.next;
        }
        return myList;
    }
}
```

Answer: \( O(n+m) \)

---

Same mystery method. Suppose listA is initially \([2, 0, 1]\) and listB is initially \([-1]\). What will `mylist` be when returned?  

```java
class Mystery {
    public static ListNode mystery(ListNode listA, ListNode listB) {
        ListNode myList = new ListNode(listA.info);
        ListNode current = myList;
        ListNode next = listA;
        while (next != null) {
            current.next = new ListNode(next.info);
            current = current.next;
        }
        return myList;
    }
}
```

Answer: \([2, 0, 1, -1]\)
Same mystery method. Suppose we have a program that invokes it as follows:

```java
ListNode listA = ... // creates a linked list
ListNode listB = ... // creates another linked list
ListNode newList = mystery(listA, listB)
```

// More code that uses listA and listB

In the additional code using listA and listB after the creation of newList, are listA and listB any different than before the method call?

- Yes, different values and different node objects
- Different values but the same node objects
- Different node objects but the same values
- No, same values and same node objects

We looked at this append method in lecture. If listA has n nodes and listB has m nodes, the runtime complexity of append is...°

```java
public static ListNode append(ListNode listA, ListNode listB) {
    ListNode first = listA;
    while (listA.next != null) {
        listA = listA.next;
    }
    listA.next = listB;
    return first;
}
```

- O(1)
- O(n)
- O(m)
- O(n+m)
- O(n+m)

Same append method. Suppose we have a program that invokes it as follows:

```java
ListNode listA = ... // creates a linked list
ListNode listB = ... // creates another linked list
ListNode newList = append(listA, listB)
```

// More code that uses listA and listB

In the additional code using listA and listB after the creation of newList, what is the relationship between listA and newList?

- listA refers to the same object as newList
- listA and newList have the same values but refer to different objects
- No relationship, different values and different objects
Canonical Linked List Problem

• How do we reverse nodes in a linked list (without creating a new list)?
  • Go from A->B->C to C->B->A
  • Typical interview style question
    • https://leetcode.com/problems/reverse-linked-list/
    • https://www.hackerrank.com/challenges/reverse-a-linked-list

Methodical Development

• Turn list = ['A', 'B', 'C'] into rev = ['C', 'B', 'A']
  • Move one node at a time, no new nodes!
    • Iterative/loop solution with invariant
      • High-level idea: As we iterate, insert front of list to front of rev

rev = [ ] list = A -> B -> C

Loop Invariants

• An invariant is something that is true immediately before loop condition (top of the loop)
  • May become false part way through loop
  • Always re-established before guard check

• Iterate over list once, create reversed as we go
  • Invariant: rev points to reversed part of list iterated over so far
    • Before first condition? rev = null
    • Then at the end? Return rev
One node at a time, assume invariant!

- Before first iteration: rev is empty (null)
- After one iteration: rev is list reversed so far
  - list has moved to represent [B,C]
  - So rev represents [A]
- How to move B to front?
  - Why temp needed?
    - Don’t lose C-node!

\[
\begin{align*}
\text{rev} = [A], \text{list} = [B, C], \\
\text{after iteration: } [B, A], [C]
\end{align*}
\]

- Pictures and code
  1. \( \text{temp} = \text{list.next} \) (so we don’t lose [C])
  2. \( \text{list.next} = \text{rev} \) (add to front point to [A])
  3. \( \text{rev} = \text{list} \) (reestablish invariant)
  4. \( \text{list} = \text{temp} \) (list updated)

Working code, check invariant

- Initialization, rev?
- Update
- Check loop

```java
public ListNode reverse(ListNode front) {
    ListNode rev = null;
    ListNode list = front;
    while (list != null) {
        ListNode temp = list.next;
        list.next = rev;
        rev = list;
        list = temp;
    }
    return rev;  // like front = rev
}
```
Testing and Debugging

An Algorithmic Problem-Solving Process: UPIC

- Understand: Understand the problem you're trying to solve. Read carefully, do examples.
- Plan: Generalize insights, develop an algorithm.
- Implement: Translate the algorithm into code.
- Correctness: Test and debug to verify and fix the code.

Not really a linear process

So, something is not correct. Could be...
1. The plan (algorithm) did not match the understanding.
2. The implementation does not match the plan.
3. The understanding was not correct.
First approach to correctness

- Natural temptation to rely on reading source code to verify correctness.
- Like editing an essay for a class, read and check that it makes sense, look for typos.
- But...

Code is complex and interrelated

Miss something in your essay? The rest of the essay may still make sense?

One thing wrong in the code? Could prevent the whole program from functioning. And code gets complicated!

A tale of two programmers...

Too confident
"I’m amazing at programming, I don’t need to test my code because I know it’s correct."

Low confidence
"My code doesn’t work, that must be because I’m personally bad at this. There is no way I could figure this out myself."

The beginning of a security vulnerability, broken app, …

Mistaken expectations, feeling helpless, not sure what to do.
What is testing?

Verifying that an implementation behaves as expected.

• What is behavior is expected?
• Given an input, what output is expected?

Can test at multiple levels: single method (unit), class (integration), whole project(integration/functionality)...

Black box testing (access to program but not source code) and white box testing (access to source code).

SandwichBar APT Example

Given:
• String[] available, a list of ingredients the sandwich bar can use, and
• String[] orders, the types of sandwiches I like, in order of preference (most preferred first)

return the 0-based index of the sandwich I will buy. If the bar can make no sandwiches I like, return -1.

Example:
• available: { "ham", "cheese", "mustard" }
• orders: { "ham cheese" }
• Should return: 0

The first test: the compiler

• Compiler performs static analysis; check for errors detectable in the source code before running.
  • Often type errors (e.g., trying to assign a String to an int, trying to treat an Array as a list, ...)
Manual test

• Given an input, what is the expected output?
• Run program with expected input. What do you get?

```
public static void main(String[] args) {
    String[] testAvailable = {"ham", "cheese", "mustard"};
    String[] testOrders = {"ham cheese"};
    SandwichOrder testInstance = new SandwichOrder(testAvailable, testOrders);
    int testResult = testInstance.whichOrder(testAvailable, testOrders);
    System.out.println(testResult);
}
```

I expect the code to return 0, example from before. And it does! My solution must work!

How many tests are enough?

```
public static void main(String[] args) {
    String[] testAvailable = {"cheese", "mustard", "lettuce"};
    String[] testOrders = {"cheese ham", "cheese mustard lettuce", "hamburger", "bun"};
    SandwichOrder testInstance = new SandwichOrder(testAvailable, testOrders);
    int testResult = testInstance.whichOrder(testAvailable, testOrders);
    System.out.println(testResult);
}
```

That's not right, I can't make a ham and cheese sandwich without ham…

• Can never have enough tests to guarantee correctness, but...
• More and more diverse tests can help increase confidence.

Automated testing?

For when you want to run many tests without doing it manually one at a time...automate it!

Ways you use automated testing in 201:
• Junit tests – Junit is a popular external library, no built-in standard library unit testing in Java.
• Gradescope autograder
• APT server

In professional software development? You also write the tests!
Test early, test small, test often

- **Unit testing**: Term for tests conducted on the smallest units of code that take inputs and produce outputs.
  - In Java, typically methods, preferably short ones (10-20 lines). Test as soon as you write, don’t wait!
  - Method getting too complex? Helper method!

```
public void testSize() {
    String str = "Hello, World!"
    int size = str.length();
    System.out.println("Size of string is: "+size);
}
```

Expected output: 13

```
What your size() method returns
```

Debugging

**Debugging loop**:  
1. Detect unexpected behavior through testing.  
2. Isolate cause of unexpected behavior.  
3. Change implementation.  
4. Test again.

**How to isolate the cause of unexpected behavior**

- Want to identify the first point of divergence from expected behavior.
  - May have started long before your test result!

- Try to answer the question:
  - What is the first line of code in which method of which class that first did something different than I expected?  
  - Never fixate on line 30 if you’re not sure lines 1-29 are working.
Debugging Methods

- Three common methods:
  - Examine code and small examples by hand
  - Add print statements to code
  - Use a debugger tool

- We have already seen the basic debugger tool built into an extension on your visual studio code. Will review in detail today.

Debugger tool

- Instead of run? Choose debug!
- Walk through execution of program line by line.
- See current state of all variables line by line.

Set a breakpoint

- Start by setting a **breakpoint** in your code.
- Says "run the program until the first time this line executes, then pause to step line by line."
- If you want to go line by line from the beginning? Set to first line in **main**.
Debug options

Will see a menu like this:

- **Continue**: Go to next breakpoint
- **Step over**: Execute line, go to next. Run whole methods.
- **Step into**: Same as over unless method call. Steps into methods, jumping to first line of method code.
- **Step out**: Break out of method back to where called
- **Restart**: Start over again at first breakpoint
- **Stop**: Stop debugging session

---

State of program

Can see all values of all local variables while executing at highlighted line.

Can step through to determine first time values diverge from expectations.

---

Debugging linked list?

- Appears as a nested "list" of object references.
- Expand one node at a time.
Want something more visual?

[link to pythontutor.com/java.html]

Can use if you need to visualize stepping through some pointer code.

---

**Debugging reflection**

Goal is to become a more active and empowered tester and debugger.

- Build confidence as you develop.
- Take active steps to isolate the problem.
- Test, use the debugger, gather data, reason about it.
- Less time staring at the code, feeling frustrated.

---

**More DIYLinkedList**

**Live Coding**

- Writing unit tests?
- Add to arbitrary index?
- Efficient iterator?