CompSci 201, L12: Debugging and Testing
Logistics, Coming up

• Project 3: DNA (Linked List) available, due 10/17

• Wednesday, 10/5 (today)
  • APT 5 Due

• Friday, 10/7
  • Discussion, linked list

• Monday 10/10 – Tuesday 10/11
  • Fall break, no class meeting, no helper/office hours

• Wednesday 10/12
  • APT6 (linked list) due
Today’s agenda

1. Wrapping up linked list problems
   • WOTO2 from 10/3
   • Reverse in-place

2. Testing

3. Debugging
Reviewing WOTO 2 from last time

duke.is/zstttj

Don’t need to complete again, just reviewing answers.
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://leetcode.com/problems/reverse-linked-list/)
  • [https://www.hackerrank.com/challenges/reverse-a-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

```
rev ← → list
```

```
A ← → B ← → C
```
Invariant to help reason about code

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

- rev points to list reversed so far
  - before loop iterates at all? rev = null
  - Then at the end we just return rev
one node at a time, assume invariant!

• 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!
rev = [A], list = [B,C], change: [B,A], [C]

- Pictures and code
  1. `temp = list.next`   (so we don't lose ['C'])
  2. `list.next = rev`   (add to front point to [A])
  3. `rev = list`        (reestablish invariant)
  4. `list = 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.

Only doing a lot of actual programming in this step!
Not really a linear process

So, something is not correct. Could be...

1. My plan (algorithm) did not match my understanding.
2. My implementation does not do what I wanted my algorithm to do.
3. I did not fully understand the problem.
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...

```java
// This is an example of a single line comment using two slashes
/**
 * This is an example of a multiple line comment using the slash and asterisk.
 * This type of comment can be used to hold a lot of information or deactivate
 * code, but it is very important to remember to close the comment.
 */
package fibsandlies;
import java.util.Map;
import java.util.HashMap;

/**
 * This is an example of a Javadoc comment; Javadoc can compile documentation
 * from this text. Javadoc comments must immediately precede the class, method,
 * or field being documented.
 * @author Wikipedia Volunteers
 */
public class FibCalculator extends Fibonacci implements Calculator {
    private static Map<Integer, Integer> memoized = new HashMap<>();

    /**
     * The main method written as follows is used by the JVM as a starting point
     * for the program.
     */
    public static void main(String[] args) {
        memoized.put(1, 1);
        memoized.put(2, 1);
        System.out.println(fibonacci(12)); // Get the 12th Fibonacci number and print to console
    }

    /**
     * An example of a method written in Java, wrapped in a class.
     * Given a non-negative number FIBINDEX, returns
     * the Nth Fibonacci number, where N equals FIBINDEX.
     * @param fibIndex The index of the Fibonacci number
     * @return the Fibonacci number
     */
    public static int fibonacci(int fibIndex) {
        if (memoized.containsKey(fibIndex)) {
            return memoized.get(fibIndex);
        }

        int answer = fibonacci(fibIndex - 1) + fibonacci(fibIndex - 2);
        memoized.put(fibIndex, answer);
        return answer;
    }
}
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!

Working C code from 1998 contest, see Wikipedia
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.”

The beginning of a security vulnerability, broken app, ...

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.”

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

Verifying that an implementation functions as expected.

- What is functionality 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* (can run program, can’t see 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, ...)

```java
public int whichOrder(String[] available, String[] orders){
    for (int i=0; i<orders.length; i++) {
        if (canMake(available, orders[i])) {
            return orders[i];
        }
    }
}
```

- **Type mismatch: cannot convert from String to int** Java(16777235) [9, 24]
Manual test

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

```
25    public static void main(String[] args) {
26        String[] testAvailable = { "ham", "cheese", "mustard" };  
27        String[] testOrders = { "ham cheese" }; 
28        SandwichBar testInstance = new SandwichBar();
29        int testResult = testInstance.whichOrder(testAvailable, testOrders);
30        System.out.println(testResult);
31    }
```

I expect the code to return 0, example from before. And it does! My solution must work!
How many tests are enough?

- 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!

You mostly *use* automated testing in 201 rather than building it yourself:

- JUnit tests
- Gradescope autograder
- APT server

Could learn this if you want, JUnit is a general-purpose Java testing library, not just 201. Examples in projects.
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!

```java
public void testSize() {
    for (String s : strs) {
        final IDnaStrand strand = setTimeout(Duration.ofMillis(10000), () -> {
            IDnaStrand str = getNewStrand(s);
            return str;
        });
        assertEquals(s.length(), strand.size(), "This test checks if .size() returns the correct value" + " for basic cases. Your code did not return the correct .size() for strand " + s);
    }

    expected output
    What your size() method returns
```
Debugging loop:

1. Detect unexpected behavior through testing.
2. Isolate \textit{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

Good start, might get complicated

To step through execution line by line

Allows you to see the state of the program while running. Tip: Can add print statements for APTs, will show up on server!

• Today we will look at the basic debugger tool built into an extension on your visual studio code.
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
Can see all values of all local variables while executing at highlighted line.

Can step through to determine *first* time values diverge from expectations.
Testing & Debugging SandwichBar

Live coding
Debugging linked list?

- Appears as a nested “list” of object references.

- Expand one node at a time.

```java
public static void main(String[] args) {
    int[] myNums = {2, 0, 1};
    ListNode myNumsList = listFromArray(myNums);
    System.out.println(getVal(myNumsList, 1));
}
```
Want something more visual?

[link](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 evidence the code 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
• Gain confidence, gain independence
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.”