CMPSCI 187: Programming With Data Structures

Review for Second Midterm
David Mix Barrington
7 November 2012

Review for Second Midterm

- Format Changes, What is Covered
- Vocabulary
- Software Engineering
- Tracing Code
- Finding Errors
- Timing Analysis
- Code Questions

Format Changes

- There are only five vocabulary questions for 10 total points.
- There are now three SE questions for 15 total points.
- There are now three timing questions for 15 total points.
- The code base is now on page 13, separate from the score page, so you may tear it out if you like. Don’t hand it in unless you have put work on it.
- You’ll probably need the overflow page for Question 7, the long coding problem. Remember that you also have the backs of each page to write on.

What is Covered?

- As the syllabus says, primarily Chapters 4 (recursion), 5 (queues), and 6 (lists) of DJW and thus all material in the lecture slides through #25 (last Monday). We covered Chapter 7 (variants of lists) briefly on Monday so it is fair game but not emphasized.
- In Fall 2011 the course was divided into quarters rather than thirds. The second and third midterms from that offering thus correspond more or less to the material of this midterm, except that sorting was on the third midterm in Fall 2011 and we are doing it later this term.
- I will expect fairly intimate knowledge of the programming projects, and some familiarity with the case studies in DJW. Code examples and coding questions will generally use DJW's interfaces for queues and lists, but these will be on the code base given with the exam. The method names for the `java.util` list and queue interfaces might be on the vocabulary section.

Vocabulary Questions

- I grade these 2 for “mostly right”, 1 for “kind of right”, 0 for “mostly wrong”.
- I want you to address both the terms that are given. Defining one of them correctly is usually good for one point.
- Here are some examples from the prior tests. Remember that there are both a practice test and a real test for both the second and third Fall 2011 midterms. Some of those questions refer to things in last year’s book only.

this (constructor) and this (object)
deque and dequeue
enqueue (DJW) and add (java.util.Queue)
linear node and linear list
iterator and Iterator
stack and queue
remove( ) and remove(element)
LLNode<T> and LLNode<SledDog>

Software Engineering Questions

- These normally ask you in general how you would implement a change to some piece of code in one of the projects or one of the case studies.
- The questions from last fall refer to the projects and case studies from that term, so it’s hard to go into them in any detail.
- A complete answer will go into the indirect effects of one change you might make on other parts of the overall project or case study.
- Details of your answer may depend on choices you made in the implementation of a project -- please be sure to explain how you did something if it is relevant to the change you are explaining. I won’t necessarily have access to your specific code as I am grading the exams.

Code Tracing Questions

- The key question here is “what will happen” -- what will be the observable output of the code fragment or method?
- A correct answer doesn’t necessarily need much explanation, but if your exact answer is incorrect I will look at the work you show to decide how much partial credit to give, if any. Showing me your thinking process, in whatever format, is a good idea.

```
public static boolean fannyLikes (String input) {
    if (input.length( ) <= 1) return false;
    char first = input.charAt(0); // first letter
    String rest = input.substring(1); // all but first letter
    if (first == rest.charAt(0)) return true;
    return fannyLikes (rest);}

 String [ ] test = {"dogs", "puppies", "cats", "kittens", "horses",
                   "colts", "fillies"};
 for (int i = 0; i < 7; i++)
    System.out.println (fannyLikes (test[i]);
```

Finding Errors

- You are given a piece of bad code and I want to know *what will happen*. Identifying what is wrong may not be enough. Will it compile? Will it throw an exception given valid input? Will it do something clearly unintended?
- This last category assumes that you know what the code is intended to do. This may be clear from the name of the method, or from comments in the code. But these are toy examples, to prove a point, and may not have a clearly defined purpose.

```
public static boolean isOK (String w) {
      if (w.length( ) == 0) return false;
      char first = w.charAt(0); // first letter
      String rest = w.substring(1); // rest of string
      if (first  == rest.charAt(0)) return true;
      else return isOK (rest + first);}
```

Timing Analysis

- Remember that your answer should be a big-O function of N, like O(1), O(N), O(N log N), and so forth. It should be clearly defined what N is in each problem.
- Remember that O(1) means “bounded above by a constant” and not just “constant”. You are looking for the **worst-case** running time. Even very large numbers can be O(1) if they do not depend on N.
- Remember the rules of arithmetic for big-O. If f = O(g), then O(f) + O(g) = O(g). O(f) times O(g) is O(fg).
- Running times for loops multiply if the loops are **nested** but not otherwise.
- A good first step in one of these problems is to trace the code with N = 3 or so, so you can see what will happen for general N.

Short Coding Questions

- One question last year involved a Maze object, which was based on a two-d array like that of a MapGrid. The question was to determine whether there was a path from any cell to any other cell. This year we could ask, given a MapGrid, whether it has only one continent. You would need to write a method similar to `markBlobs` to do this.

- Given a bunch of DogTeams, and a method to feed a SledDog and return a boolean saying whether it needs to be fed again, another question was to arrange a feeding queue for the dogs with some kind of Queue object. The queue was to continue calling the feed method until all dogs were fed.

- The other two short coding questions involved sorting and priority queues which we haven't done yet.

Long Coding Questions

- Given a list of SledDogs, write code to arrange them into a series of lists, one for each age.

- The Hex game involved finding whether certain paths exist in a two-d array. i gave the rules for the game, told you that lots of methods were already available, and had you write a class that would enable the players to play one game of Hex. Some of the methods had you checking whether paths exist.

- The dishonest GuessANumber game from Discussion #8 was a long coding problem last year, where you were given the idea of the game and the overall method of cheating, and told to implement it.

- A subteam of a DogTeam t is a DogTeam with a subset of the elements of t, in the order that they occur in t. A question last year was to use recursion to construct, given t, an array containing all the possible subteams of t.