Computer Science 343
Data Structures
Fall 2013

Dr. Stephen Bloch
office 213 Post Hall
phone 877-4483
email sbloch@adelphi.edu
Web page http://www.adelphi.edu/sbloch/
Class Web page http://www.adelphi.edu/sbloch/class/343/
office hours MW 2:30-4:00, T 10:00-4:00, or by appointment

August 23, 2013

1 Course Description

Expand on topics learned in CSC 172. Examine, implement, and analyze common data structures such as stacks, queues, lists, trees, heaps, and graphs. Understand how to choose an appropriate data structure for a real-world problem and use it in solving such problems.

2 Subject Matter

This is a course for anybody who's ever complained about a computer being slow. When faced with a computer program that takes unacceptably long to solve the problem at hand, one option is to buy faster, more expensive computer hardware. But this option has its limits: perhaps your money will run out, or perhaps you'll upgrade to the fastest computer on the market and still be unsatisfied. Often a better option is to find a more efficient approach to the problem — to re-think the *algorithm* so that it makes *better use* of the hardware. A $1,000 personal computer running a good algorithm can often outperform a $1,000,000 supercomputer running a bad (though correct) algorithm.

In 1976, Niklaus Wirth (inventor of the Pascal language) wrote a classic computer science textbook entitled *Algorithms + Data Structures = Programs*. The statement was somewhat oversimplified, but almost forty years later it still carries a lot of truth: a great deal of the work of writing a correct and efficient program consists of choosing algorithms and data

structures. Indeed, although Adelphi offers two separate courses CSC 343 ("Data Structures") and CSC 344 ("Algorithms and Complexity"), the subjects really can't be separated so easily: many important algorithms only work well with a particular data structure, and many important data structures are motivated by particular algorithms. The first semester will lean slightly in the "data structures" direction, and the second semester slightly in the "algorithms" direction, but we'll address both algorithms and data structures as they naturally come up.

In this course we'll learn how to measure the efficiency of an algorithm, independent of language, operating system, or hardware. We'll survey a variety of techniques for designing efficient algorithms and data structures. Many of these techniques will help you program correctly even when you're not worried about efficiency.

In the second semester, we'll learn some additional important algorithms and data structures. We'll learn how to prove that a given algorithm is "as good as it can get," in the sense that *no* algorithm, no matter how clever, will *ever* be better than this one. Finally, we'll study problems believed to have *no* efficient solution by computer program, and even some problems which have *no computational solution at all*, and how we deal with such problems in reality.

3 Course Learning Goals

By the end of this semester, you should be able to

- write and critique proofs, including those involving mathematical induction
- read and write algorithms in pseudocode
- present solutions to technical problems to your colleagues and answer their questions
- offer constructive critiques of other students' algorithms, data structures, and proofs
- respond constructively to such critiques of your work
- make good asymptotic estimates (in big-O notation) of the running time and memory consumption of a given program
- implement and use linked lists, array-based lists, stacks, queues, binary and n-ary trees, self-balancing trees, heaps, and hash tables, including the provision of access and traversal methods for these data structures
- discuss advantages and disadvantages of various algorithms and data structures to solve a given problem

4 Textbooks

I've chosen two textbooks to use throughout the two-semester sequence: they cover a lot of the same topics, but sometimes I like one author's treatment better than the other's.

One is the "bible" of algorithms and data structures, *Introduction to Algorithms* by Cormen, Leiserson, Rivest and Stein (MIT Press 2009, ISBN 0262033844). (The "Rivest" author is the R in the RSA cryptosystem, by the way.) It's a big, thick book, but relatively inexpensive, because there's not much second-hand market for it: once you've bought it, you'll probably refer to it for the rest of your programming career. If you really don't want to buy the big hardback, the second edition (and several hundred other books) is *free on-line* to ACM Student Members. I highly recommend that all CS majors become members of the ACM: my own membership costs $99/year, so at $19/year, student membership is a bargain. The second edition is about 90% the same as the current third edition; I'll try to have a copy of the third edition on reserve at the library.

The other textbook is Cliff Shaffer's *Data Structures and Algorithm Analysis* (Dover 2012, ISBN 0486485811). You can buy a printed copy, but it's also available free on-line from `http://people.cs.vt.edu/~shaffer/Book/` (in either a Java or a C++ edition).

I may also give out other reading assignments by email, on the Web, or in journals.

This is a 3-credit class, which means you should budget 6 hours per week outside class for reading and homework. In particular, you'll need to read about 35–40 pages per week, on average. *Make time in your weekly schedule for this!*

5 Prerequisites

I assume that everyone in the class has passed CSC-MTH 156 (Discrete Structures) or an equivalent course covering Boolean logic and algebra, graphs and trees, and perhaps recurrence relations. I assume that you've passed pre-calculus: we'll seldom need derivatives or integrals, but we'll need logs and exponentials *all the time*, so refresh your memory of them. I also assume that everyone in the class has passed at least a year of programming courses. I don't care much about the language, as long as you have written and debugged a number of programs and are familiar with the notions of algorithm, recursion, loop, array, linked list, *etc.* I'll put up examples in Java or Scheme syntax, whichever seems more natural for the problem at hand, and at least one assignment will have to be done in C++ (because it's about issues that Java and Scheme hide from you).

6 Grading

There will probably be 5 homework assignments, each worth 10% of the semester grade, and a final exam worth 15%. The remaining 35% of your semester grade is class participation, especially your in-class presentations of homework solutions (see below) and your constructive participation while other students are presenting.

The final exam must be taken at the scheduled time, unless arranged in advance or prevented by a documented medical or family emergency. If you have three or more exams scheduled on the same date, or a religious holiday that conflicts with an exam or assignment due date, please notify me in writing within the first two weeks of the semester in order to receive due consideration. Exams not taken without one of the above excuses will be recorded with a grade of 0.

7 Written homework

There will be several kinds of homework in this class. At one extreme are the analysis and "thought" problems on paper, resembling problems in a math class. At the other extreme are programming assignments, which may generally be written in any language that you and I both know and that runs at Adelphi (*e.g.* Scheme, Prolog, Java, C, C++). (There will probably be at least one programming problem that *must* be done in C++, because it deals with issues that Scheme and Java take care of for you.) In between are *pseudocode* assignments: these need to be precise descriptions of an algorithm, like a computer program, but they don't need to meet the syntactic requirements of a compiler (only a human being will read them) and you can ignore details that aren't relevant to the problem at hand. For example, in a problem that wasn't primarily about sorting, you might say "sort table A in increasing order by value" as one line of the algorithm. On the other hand, if the assignment *were* about sorting, I would expect you to give the details of your sorting algorithm.

8 Revising homework

For each homework assignment except the last, you have the option to *rewrite and resubmit* the assignment, up to a week after you've gotten it back graded. If you choose to do this, I'll grade the revised version, multiply the grade by 80%, and keep the larger of this number or the original grade. Which means if you had above about a 75% on the original assignment, there's not much point in resubmitting because you're unlikely to improve your grade.

Homework assignments turned in late (up to a week after it's returned) will be treated as resubmissions, worth at best 80%. Any homework assignment turned in more than a week after the assignment is returned graded will get a zero. Any homework assignment turned in after Dec 11 (the last day of class) will get a zero. (This is so I have, perhaps, time to grade them before the final exam.)

9 In-class presentations

A substantial part of your grade comes from in-class presentations of your solutions to homework problems. (This is one reason I don't want you turning in homework late: I want to see *your* solution, rather than have you copy down a classmate's presentation and turn it in.) For each of the 5 homework sets, you are to choose a problem that you want

to present, contact me in advance for a time slot (and to make sure nobody else is doing the same problem), then present your solution at the board and answer technical questions from me and the other students. When it's your day to present a problem, try to get to class a few minutes early and start writing it up on the board. If you're not sure of your solution or your presentation, please discuss it with me in my office before the day you want to present it in class. Your grade for in-class presentations will be my assessment of how well you presented the solution and answered questions about it. I'll also pay attention to who asks those questions.

10 Attendance policy

I expect you to be in the classroom at 12:00, and to still be in the classroom at 12:50; if you're not present, and miss some important material, it's your problem. If you must miss a class, arrive late, or leave early, please let me know as far in advance as possible. I don't explicitly grade on attendance, but consistent lateness or absence will impact the "class participation" part of your grade.

11 Students with Disabilities

If you have a disability that may impact your ability to carry out assigned course work, and are not enrolled in the Learning Disabilities Program, please contact the staff in the Disability Support Services Office (DSS), University Center, Room 310, (516) 877-3145. DSS@adelphi.edu. DSS will review your concerns and determine, with you, appropriate and necessary accommodations. All information and documentation of disability is confidential.

In particular, you may choose whether, and in how much detail, to discuss your disability with the instructor.

12 Ethics

The Adelphi University Code of Ethics applies to this course; look it up on the Web at `http://academics.adelphi.edu/policies/ethics.php`.

Assignments in this class are to be done individually, or in teams of of two by prior permission; in the latter case, you *may not do multiple homeworks with the same partner.* You may *discuss general approaches* to a problem with classmates, but you *may not copy* large pieces of programs or homework solutions. If you do, *all* the students involved will be penalized (*e.g.* I'll grade the assignment once and divide the points equally among the several people who turned it in).

All work on an exam must be entirely the work of the one person whose name is at the top of the page. If I have evidence that one student copied from another on an exam, *both* students will be penalized; see above.

13 Student course evaluations

During the last two weeks of the class, you will be informed, via email and eCampus, that the University's online course evaluation form is available for your input. It will no longer be available after the beginning of final-exam week, so make sure you fill it out before then. They don't show me any of the results until after I've turned in semester grades. We really do take this stuff seriously in deciding what to do differently in future courses, so please give detailed feedback (something more specific and useful than "this course sucks!" or "this course rocks!").

14 Schedule

This class meets every Monday, Wednesday, and Friday from 12:00-12:50 PM in Hagedorn 111, unless we agree to change that. The schedule of topics, reading assignments, and homework assignments will be maintained on the Web at
`http://www.adelphi.edu/sbloch/class/343/calendar.html`
The dates are subject to change depending on how classroom discussions actually go. I expect you to have read the reading assignments *before* the lecture that deals with that topic; this way I can concentrate my time on answering questions and clarifying subtle or difficult points in the textbook, rather than on reading the textbook to you, which will bore both of us. **Please read ahead!**