CompSci 404.1 – Data Structures and Algorithms

Faraaz Sareshwala

October 9, 2019

Welcome to CompSci 404.1 (Data Structures and Algorithms). This course will give you an overview and hands-on experience with some of the more popular data structures, algorithms, design techniques, and tools used in the industry today. The course focuses on the fundamental data structures knowledge that every software engineer must have along with the algorithms associated with working with them. In addition, we will cover applications of these data structures and algorithms in the context of various real-world situations that you may find yourself in while on the job. While working through problem sets and assignments, you you will also gain hands on experience with the tools and best practices that software engineers use to work with each other.

This will be a challenging course. You should be prepared to devote many hours every week to your homework and programming assignments. At times, you will feel very frustrated while writing and debugging your programs. That is normal and a part of the learning process. I am here to help guide you through the material. Should you have any questions during the semester, please always feel free to reach out to me either in class, through canvas, through email, or any other method you feel comfortable with.

Contents

1 Course Information

1.1 Instructor

Name	Faraaz Sareshwala
Email Address	fsareshwala@berkeley.edu

1.2 Course Details

Class Time	Wednesday, 6:30 pm to 9:30 pm
Class Location	Classroom 211
Textbook(s)	Algorithms (4th Edition) by Robert Sedgewick and Kevin Wayne
	Cracking the Coding Interview (6th Edition) by Gayle Laakman McDowell
Prerequisites	Proficiency in at least one imperative programming langauge
	Basic familiarity with the unix command line environment
	Experience with basic data structures and algorithms

1.3 Course Objectives

This course will teach you to implement and apply knowledge of data structures and algorithms. Specifically, by the end of this course, you will be able to:

- Implement basic data structures and algorithms from scratch, understand their time and space complexities, design trade-offs, and when to use which data structure or algorithm
- Apply systems knowledge to design software and select appropriate trade offs based on any set of constraints and requirements
- Work with modern software engineering tools and understand their best practices in order to be able to hit the ground running on day one of a new job

2 Grading

Course grades are broken down into the following categories.

Homework assignments	35%
Programming assignments	50%
Final Project	15%

2.1 Grade Distributions

Your final letter grade will be determined using the following percentage breakdown.

- 93% or higher: **A**
- 90% or higher: **A-**
- 87% or higher: **B+**
- 83% or higher: **B**
- 80% or higher: **B-**
- 77% or higher: **C+**
- 73% or higher: **C**
- 70% or higher: **C-**
- 67% or higher: **D+**
- 63% or higher: **D**
- 60% or higher: **D-**
- 59.99% or lower: **F**

2.2 Homework Assignments

Homework will typically consist of answering simple questions focusing on the concepts of what we are learning in class. Make sure to show all of your work when responding to questions.

Homework assignments will be available as PDF files online for download. When completing your homework, please use a PDF editor to directly input your answers into the homework PDF file itself. You can find instructions on how to edit PDF files online. For example, if you use a Mac, Preview comes built in with PDF editing capabilities.

All homework assignments are submitted online. The last day to submit any homework assignment is the last day of instruction.

2.3 Programming Assignments

Programming assignments are larger, more complete programs which will utilize the concepts we learn in class in a more practical setting. Programming assignments will ask you to solve real problems that you might be tasked to solve by a real company. They are complex and you might find yourself taking many hours to complete a single task. Sometimes, you may get very frustrated with why something doesn't work the way you expect it to. Other times, you will be searching for a bug that just doesn't make sense. Programming takes time and it isn't always immediately clear how much time a task will take before actually setting out to complete it. That is why it is imperative to start your programming assignments early. If you get stuck, reach out to me and I will help you.

Programming assignments will be graded on correctness, code organization, efficiency, style, and documentation, in that order of priority.

All programming assignments must be submitted online. I will not accept printed code. When submitting your programs, make sure to include all files (e.g. implementation files, header files, makefiles, etc) necessary to compile and execute your code. Missing files may cause your program to not compile or work properly and result in your final grade being deducted as defined in the late work policy in section 2.5 on page 3.

The last day to submit any programming assignment is the last day of instruction.

2.4 Assignment Due Dates

The schedule for homework and programming assignment due dates is available on the online calendar.

2.5 Late Submissions

I strongly encourage you to keep up with the pace of the class. You risk putting yourself at a distinct disadvantage for learning when you get into the habit of submitting work after the due date. However, I do recognize that unforeseen events happen in life.

I will accept late homework and programming assignments up to one week late. For each day late, you will lose 20% of your final grade on that assignment. For example, if you miss submitting an assignment on its original due date and submit it the next day, the maximum possible grade you can get on that particular assignment is 80%.

The last day to turn in any late work is the last day of instruction.

2.6 Attendance

I strongly suggest you to attend and participate in all lectures. Attending lecture allows you the opportunity to ask questions and get immediate feedback should you not understand something in the material. Moreover, your presence ensures that you keep up with the pace of the class and are aware of any announcements that I make throughout the course. In short, your attendance is critical to your success in this course.

In some programs where this course is taught, attendance is mandatory. If that is the case for the program you are in, then you must attend lecture. Not only is it critical to your success in this course, but also your enrollment in the program: your scholarship and stipend depend on it. To be considered "present", you must be present for the entire class period. If you arrive late, or leave early, you risk being marked absent for the

day (unless prior arrangements have been made with the instructor). Frequently absent or tardy students will be sent to the program coordinator for academic review.

2.7 Academic Honesty

All work submitted is to be completed individually and is to be the sole product of your own efforts. You may work together to solve homework and programming assignments. In fact, I encourage you to do so. You may also use the textbook and various online resources (e.g. Stack Overflow) as a reference. However, all solutions and source code you submit must be your own work. Should you use code you found online or in the textbook in your programs, you must properly attribute that piece of code with a comment indicating the source. For example, source code you find online on Stack Overflow must be commented with the link to the borrowed source code. For source code you use from the textbook, indicate the page number the source code appeared on.

All members of the UC Berkeley Extension community are expected to act with honesty, integrity, and respect for others. The university does not tolerate any form of academic dishonesty. There are strict consequences, including expulsion from the program, for cheating on or plagiarising any part of the homework assignments, programming assignments, or exams. For further information, please refer to:

- Tips for Maintaining My Academic Integrity
- UC Berkeley Extension Code of Student Conduct

3 External Tools

Throughout this course, we will be making use of various external tools in order to enrich the student and instructor experience.

3.1 Campuswire

Campuswire is an academic forum and realtime chat application which you will use to post questions about your assignments while outside the classroom. For example, if you are working on your homework assignments or programming assignments and need some help, you can use campuswire to quickly reach others in your class and ask your question. I will be monitoring campuswire as well and will be offering help whenever I can.

You should receive an invitation for campuswire in your email inbox before the first day of class. If you haven't received an invite yet, please follow up with me and we will get you signed up.

3.2 GitHub

GitHub is an online source code hosting tool based on the `git` version control system. We will use GitHub to collaborate on projects and turn in assignments. You do not need prior knowledge of `git` in order to take this class. We will cover enough of the basics of working with `git` in lecture for you to be able to complete your assignments.

A GitHub account is free. Please sign up and set up SSH access with your account. Further instructions to do so are available here.

4 Class Schedule

The following table contains a tentative schedule of topics covered in the course.

Session	Topic
March 25, 2020	Introduction, `git` concepts, Java overview, Complexity
April 1, 2020	Elementary data structures (linked lists, stacks, queues)
April 8, 2020	Sorting (elementary sorts, merge sort, quick sort)
April 15, 2020	Union Find, Searching (binary heaps, binary search trees)
April 22, 2020	Searching (hash tables), Graphs (BFS, DFS)
April 29, 2020	Graphs (connected components, topological sort, strong components)
May 6, 2020	Graphs (spanning trees, shortest paths)
May 13, 2020	Strings (sorting, tries)
May 20, 2020	Strings (searching, compression)
May 27, 2020	Probabalistic Data Structures (bloom filters), Dynamic Programming

5 Tips and Philosophies for Success

The following list is a set of tips for your success in this course. If you do the following things, I can almost guarantee you that you will be happy with your results.

- Come to class and pay attention without any distractions. Put your cell phone on silent and close your laptop unless instructed to use it.
- Ask questions immediately when you are unsure of something. Topics frequently build on top of one another. It does no one any good if you didn't understand something 30 minutes ago and the class has already moved on to the next topic.
- Do all homework assignments and programming assignments on time and keep up with the pace of the class.
- Start your assignments early and don't procrastinate. Programming usually takes more time than you initially anticipate.
- Read the textbook chapter to be discussed before attending the class it will be discussed in. Seeing the material again will help you learn it faster and more concretely.
- Form study groups with the rest of your classmates so that you can have a support system and get immediate feedback when completing the homework and programming assignments.
- Practice a lot! As they say, practice makes perfect. The best way to get better at programming is to actually program. That is why this is a programming intensive course.

6 About the Instructor

I graduated in December 2009 from the University of California, Davis with a Bachelor's of Science in Computer Science. Since then, I have been working in the industry at a variety of companies.

I am currently a software engineer at Twitter where I work in Twitter's realtime storage group. The realtime storage group is responsible for designing and supporting a custom NoSQL database (Manhattan) and custom caching solutions (Twemcache, Nighthawk) for Twitter's user facing applications. Previously, I used to work on the Traffic team which was responsible for Twitter's Layer 7 software load balancer and router. There, I lead research and development efforts for Layer 7 DDoS mitigation at Twitter's edge.

Before Twitter, I worked at Quantcast. There, I built and scaled their bidding infrastructure from 10,000 transactions per second to over two million transactions per second. I lead the design and implementation of various infrastructure components such as their realtime bidding engine, distributed hash table, distributed file system, and custom map reduce implementation.

At the UC Berkeley Extension, I teach various courses on computer science including Object Oriented Programming using `C++` and Data Structures and Algorithms.

If you are interested in what any of the above terms mean, ask me. I am very happy to talk about my work. I am also happy to help guide you if you are interested in pursuing a career in computer science or software engineering.