INFO6205

Program Structure and Algorithms

Syllabus

Prof. Robin Hillyard, Boston

Spring 2020

r.hillyard@neu.edu

This course covers the fundamentals of designing data structures and the algorithms which manipulate them. This is an important class for any aspiring developer as data structures and algorithms are at the core of every application. My goal is not only to teach you the fundamentals of the subject, but also to give you an understanding of *why?*

If you're in any doubt whether to take this class, then you should take it.

Prerequisites: INFO 5100 or permission of instructor.

Introduction: In almost every aspect of programming, you will find an understanding of data structures, their accompanying algorithms, the invariants that bind them together and, of course, order of complexity, to be essential. We cover such basic structures as bags, sets, lists, stacks, queues, priority queues, trees, symbol tables (including hash maps), and graphs. Significant emphasis is placed on data-structures/algorithms for sorting and searching as these expend considerable resources in many applications. The course emphasizes the importance of *reduction* in the domain of problem-solving and, from this, we derive a number of standard problem-solving techniques: brute force, divide-and-conquer, space/time tradeoffs, dynamic programming. We also briefly cover data compression, greedy algorithms, iterative improvement, non-deterministic and other algorithms. Underpinning this whole subject, we also cover the fundamental concepts of (Shannon) entropy and complexity, as well as theoretical and experimental measurements of performance. The course will also illustrate the various design techniques with problems in graph theory, especially as it applies to social networking paradigms.

Data structures, algorithms and invariants are the three fundamental pillars of programming. It makes no sense to have one without the other two (although, given two of them, you should usually be able to reconstruct the third). The class will be detail-oriented—at least for the major structures/algorithms—and will provide an essential component for anyone contemplating a career as a software developer. Although the subject could be studied using almost any language, the language of this class is *Java*. We will be using some aspects of Java 8 so you should try to familiarize yourself with the functional aspects of Java. Since INFO 5100 is a pre-requisite, you are assumed to have a *good* grasp of programming in Java. If this is not the case, then you will need to do some significant brush-up *before* class begins. Don't wait until the first quiz to discover you don't really know Java.

I also will spend some time giving you a short preparation for the "coding interview."

The course is only 14 weeks—and part of that time you will be working on a significant project—so there will be some details of algorithms for which we simply do not have time. These are easy to assimilate from the class textbook or from the internet. But we will cover all the most important topics in sufficient detail for you to be able to understand them—and reproduce them on demand.

Teaching style: If there is one difference between my lectures and other similar lectures, it is an emphasis on the fundamentals. You will learn the most important standard technique: how to reduce an **O**(N) problem to an **O**(log N) problem. If you take only one thing away with you, it should be this.

Although I use presentation slides to cover the detail of the course, I also spend quite a lot of time interacting with the class on the blackboard in order to get the fundamental concepts across.

Each week (or at least most weeks), we will typically have a quiz (probably on HackerRank) which will take 30 minutes or so. This is to a) help you to inwardly digest the material from that week and b) give you some practice with coding exercises. I do the quizzes because students asked for them. In the past, I have made the mistake of granting your wish that you have at least a few days to study the topic of the quiz. Unfortunately, this has an overall deleterious effect—so you should expect to do the quiz on a topic you have *just* learned. *You must also be present in class in order to do a quiz.*

Office Hours: Many past students have been reluctant to meet with me (or the TAs) during office hours. We are a resource for you to use, especially if you are finding the work difficult. My office hours are posted at my home page (http://www1.coe.neu.edu/~rhillyard/).

Labs: I may set some labs for you to get some interactive programming experience. We particularly emphasize testing, source control (github) and good coding practice.

Discussion forum: If you have general questions about the assignments, lectures, textbook, or other course materials, please post on our Slack channel (see Blackboard).

Grading: Your grade for the course will be based on the following components (see Blackboard for details—please note that this may vary slightly):

1. Programming assignments (17%);
2. In class quizzes—typically executed in HackerRank (18%);
3. Mid term exam (17%);
4. Term project (usually in teams of two) (22%);
5. Final exam (22%);
6. Class participation via TopHat (4%)

Textbook: The following textbook is *required*. It contains a wealth of information beyond what we can cover in lectures; it is certain to enhance your understanding of data structures and algorithms.

> *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne, Addison-Wesley Professional, 201x, ISBN 0-321-57351-X.

Programming assignments: The programming assignments involve applying the material from the lectures to solve problems in science, engineering, and commerce. The assignments emphasize the practical aspects of performance tuning (including, of course, design-for-performance) measured typically by experiment (i.e. benchmarking). Most programming assignments (and quizzes) will be based on the notion of TDD (test-driven development). We strongly emphasize the importance of unit tests, as well as source control (such as GitHub).

Quizzes: The quizzes will be available on HackerRank. They consist of short multiple-choice questions on the material in the lectures and readings, usually accompanied by one simple coding problem.

Exams: The schedule for the in-class midterm exam will be announced. The final exam is scheduled by the Registrar during the finals week. The midterm exam will be closed-book. The final exam may be (partially) open-book. Please see my Guidelines to Exams (on Blackboard).

Academic Integrity: We take honesty, avoiding plagiarism, and academic integrity in general very seriously in this class. All of the work you will be doing has been done by someone else and is freely available on the internet. But you learn nothing by copying code from the internet. This all makes it very difficult to develop quizzes and exams that are challenging. So, we must have a strict policy of no copying during our quizzes and exams. We use various tools to try to ensure this but there are usually ways to cheat. Don't do it. You risk getting in F—or worse. You must familiarize yourself with the University's policy at http://www.northeastern.edu/osccr/academic-integrity-policy/. You may collaborate in assignments (and of course the group project) but, for assignments, the actual submission must be your own.

Computers: You may develop your programs on any machine that you like: we encourage you to use your own equipment. I recommend IntelliJ IDEA as an IDE (it's free).

TopHat: We will utilize TopHat as an aid to encourage participation in class. For this you must subscribe to the App.

Course Outline. The following topics will be covered, depending on time:

Approximate Schedule of INFO6205

1	Introduction; Intro to Reduction.	Programming Model; Abstract Data Types; Practical Thinking
2	ADT contd	Bags, Stacks, Queues; Hash-coding.
3	Analysis	Entropy, Induction, Numbers.
4	Union Find	UnionFind contd.
5	Elementary sorts	Mergesort
6	Quicksort	QuickSelect
7	Heapsort	Elementary Symbol Tables
8	Mid-term exam	Binary Search Trees
9	Balanced Search Trees	Hash Tables
10	Hash Tables contd.	How to ace your technical interview.
11	Project kickoff.	Undirected Graphs
12	Directed Graphs	Minimum Spanning Trees
13	Minimum Spanning Trees continued	String Sorts
14	Regex, P/NP, etc., Finite Automata	Software Engineering