Syllabus

Course Meeting Times

Lectures: 2 sessions / week, 1 hour / session

Recitations: 2 sessions / week, 1 hour / session

Course Description

This course provides an introduction to mathematical modeling of computational problems. It covers the common algorithms, algorithmic paradigms, and data structures used to solve these problems. The course emphasizes the relationship between algorithms and programming, and introduces basic performance measures and analysis techniques for these problems.

Prerequisites

A firm grasp of Python and a solid background in discrete mathematics are necessary prerequisites to this course. You are expected to have mastered the material presented in *6.01 Introduction to EECS I* and *6.042J Mathematics for Computer Science*.

If you have not taken and been successful in each of these subjects, please speak with a TA or professor before enrolling. We do allow students who have equivalent, other experience with the material described above to enroll, but with the firm understanding that mastery of this material is assumed and that course staff will not feel obligated to cover it or to help students who are struggling with it.

6.006 is a 12-unit (4-0-8) subject and serves as a Foundational Computer Science subject under the new curriculum. It is a direct prerequisite for *6.046 Design and Analysis of Algorithms,* the theory header.

Textbooks

Required

Buy at Amazon Cormen, Thomas, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*. 3rd ed. MIT Press, 2009.

ISBN: 9780262033848.

For the student who finds books helpful, we also suggest:

Buy at Amazon Miller, Bradley, and David Ranum. *Problem Solving with Algorithms and Data Structures Using Python*. 2nd ed. Franklin, Beedle & Associates, 2011. ISBN: 9781590282571.

Software

6.006 programming environment setup

Lectures and Recitations

One-hour lectures are held twice a week. You are responsible for material presented in lectures, including oral comments made by the lecturer (or other information that may not be present in the notes).

One-hour recitations are held twice a week, one day after the lectures. You are responsible for the material presented in recitation, which may include new material not presented in lectures. Recitation attendance has been well-correlated with quiz performance in past semesters. Recitations also give you a more intimate opportunity to ask questions of and to interact with the course staff. Your recitation instructor is responsible for determining your final grade.

Problem Sets

We will assign seven problem sets during the course of the semester. Each problem set will consist of a programming assignment, to be completed in Python, and a theory assignment.

If you collaborate with others in any fashion, you must list their names as collaborators. For details, please see the section on our collaboration policy; we take this very seriously.

Late assignments will be severely penalized. (This penalty is currently a 1% deduction every six minutes or part thereof until the end of the tenth hour after the deadline, after which submissions will receive no credit.)

Quizzes

We will give two evening quizzes during the semester; these will each be two hours in duration. There will also be a final exam during finals week.

Grading Policy

Your final grade will be determined by the grades you receive on problem sets, on quizzes, and on the final. The particulars of this policy are subject to the discretion of the course staff.

ACTIVITIES	PERCENTAGES
Problem sets	30%

Quizzes	20% each
Final exam	30%

Coding Assignments

The code that you hand in will be graded based on its correctness, its quality, and the details of the algorithm that it implements.

Correctness

We will provide a set of public unit tests with each problem to help you test your work. However, when grading, we will use additional unit tests that will not be available to you; we reserve the right to test any behavior specified by or following from the problem statement. Submissions that run for excessive amounts of time may be scored as incorrect.

Theory

Code should represent an implementation of an appropriately designed algorithm. While we do not necessarily expect you to achieve any lower bounds that may exist for a particular problem, submissions should not be overly inefficient in either time or space.

Copying another student's code is considered cheating. We may use both manual and automated methods to detect cheating.

Written Assignments

We expect you to enter proofs using LaTeX math mode directly into Gradetacular. We have a two-step process for grading proofs. First, you'll enter your proof into Gradetacular before the time that the problem set is due. We will provide the solutions 10 hours after the problem set is due, which you will use to find any errors in the proof that you submitted. Your critique will usually be due by the following lecture. Your grade will be based on your solution and your critique.

The same late policy applies to the grading part of the assignment (1% off every six minutes that the problem set is late). Please note that if you require an extension, we will need to know in advance and you must have a good reason for needing it. In addition, we trust that you will not look at the posted solutions when completing the problem set under an extension. Looking at the solutions under these conditions constitutes a breach of the honor code, and is a serious offense.

The best responses will be concise, correct, and complete. Failing to answer part of the question, being overly verbose, missing special or edge cases, and answering mistakenly will each reduce your score.

When you are called upon to "give an algorithm," you must provide (1) a textual description of the algorithm, and, if helpful, pseudocode; (2) at least one worked example or diagram to illustrate how your algorithm works; (3) a proof (or other indication) of the correctness of the algorithm; and (4) an analysis of the time complexity (and, if relevant, the space complexity) of the algorithm.

Remember that, above all else, your goal is to communicate. After all, if a grader cannot understand your solution, they cannot give you any credit for it.

Collaboration Policy

The goal of homework is to give you practice in mastering the course material. Consequently, you are encouraged to collaborate on problem sets. In fact, students who form study groups generally do better on exams than do students who work alone. If you do work in a study group,

however, you owe it to yourself and your group to be prepared for your study group meeting. Specifically, you should spend at least 30–45 minutes trying to solve each problem beforehand.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You are asked on problem sets to identify your collaborators. If you did not work with anyone, you should write that you did not have collaborators. If you obtain a solution through research (e.g., on the web), acknowledge your source, but write up the solution in your own words. It is a violation of this policy to submit a problem solution that you cannot orally explain to a member of the course staff.

Code you submit must also be written by yourself. You may receive help from your classmates during debugging. Don't spend hours trying to debug a problem in your code before asking for help. However, regardless of who is helping you, only you are allowed to make changes to your code. Both manual and automatic mechanisms will be employed to detect plagiarism in code.

No other 6.006 student may use your solutions; this includes your writing, code, tests, documentation, etc. It is a violation of the 6.006 collaboration policy to permit anyone other than 6.006 staff and yourself to see your solutions to either theory or code questions.

Plagiarism and other anti-intellectual behavior cannot be tolerated in any academic environment that prides itself on individual accomplishment. If you have any questions about the collaboration policy, or if you feel that you may have violated the policy, please talk to one of the course staff. Although the course staff is obligated to deal with cheating appropriately, we often have the ability to be more understanding and lenient if we find out from the transgressor himself or herself rather than from a third party.