CS 146 (Sections 3, 4, 5, 6): Data Structures and Algorithms, Spring 2019

David Scot Taylor
Associate Professor
Dept. of Computer Science
San Jose State University
212 MacQuarrie Hall
Phone: (408) 924-5124 (email works better)
Email: david.taylor "at" sjsu.edu

My office hours for Spring 2019

- Tuesday, 9:00-11:00 (only 9-9:30 on Feb 5, but I will have hours 9:30-11 on Thursday, Feb 7)
- Other times available, set up an appointment by email.

Class Meetings:

- Section 3: SCI 311, Mon/Wed 7:30-8:45
- Section 4: SCI 311, Mon/Wed 9:00-10:15
- Section 5: SCI 311, Mon/Wed 10:30-11:45
- Section 6: SCI 311, Mon/Wed noon-1:15

Prerequisite Courses

You must show me that your prerequisite courses have been satisfied. **If you do not submit prerequisites by Monday, February 4 (Tuesday is drop date), you might be dropped from the course, if other students are waiting for your space. Note, there are office hours, but no class meeting on Tuesday.** Further, I will not give out any add codes without first seeing prerequisite proof.

You should show me grades for CS46B, Math30, and Math42, or their equivalents on a San Jose departmental course equivalence form. You must have a C- or better in each course. You should also have CS49J if you took CS46B elsewhere in a language other than java.

Prerequisite courses and what they covered:
CS46A: Introduction to programming, conditionals, iteration, and recursion.
CS46B: Stacks and queues, lists, dynamic arrays, binary search trees. Iteration over collections. Hashing. Seaching, elementary sorting. Big-O notation. Standard collection classes.
CS49J: Java programming language, if your CS46A/B were in a different language.
Math30: Ideally, you would have sequences and series (Math 31), but proof of Math 30 will suffice.
Math42: Sets, logic, proofs, induction, combinatorics, probability, and equivalence classes.

Peer Instructors

We several Peer Educators this semester:

- Supplemental Instruction Leaders will be in the class
  - Shivanku Mahna, email shivanku.mahna at sjsu.edu.
  - Hassib Ashouri, email hassib.ashouri at sjsu.edu

  Each week, they will hold sessions outside of regular class meetings in order to cover class topics or supplemental material.

- Peer Mentor will be available for tutoring
  - Dev Kapupara

Course Format

Much of this course will be taught in "flipped" format: rather than using classtime for lectures, we will be using class to introduce ideas through problem solving, work on difficult problems, and in a few cases, code review. Reading, videos, rote homework problems, and programming will take place at home.

Course Website

The main course website will through SJSU's Canvas website at https://sjsu.instructure.com/. Basic course information, including a link to this greensheet, is at http://www.cs.sjsu.edu/faculty/~taylor/term/fall18/CS146/.

Course Description

Implementations of advanced tree structures, priority queues, heaps, directed and undirected graphs. Advanced searching and sorting (radix sort, heapsort, mergesort, and quicksort). Design and analysis of data structures and algorithms. Divide-and-conquer, greedy, and dynamic programming algorithm design techniques.

Course Goal

To examine various ways to represent data used by programs and to compare these representations in terms of their memory requirements and the resulting program execution times.

Time will be spent learning algorithms and data structures, mathematical tools and techniques (recursion, recurrence relations) useful for their design and analysis, and seeing some examples of when they are needed.

Course Objectives

- To ensure that students are familiar with ways to implement elementary data structures and their associated algorithms.
- To introduce students to the implementation of more complex data structures and their associated algorithms.
- To acquaint students with advanced sorting techniques.
- To teach students how to determine the time complexity of algorithms.
- To introduce students to algorithm design techniques.

Course Learning Outcomes

Upon successful completion of this course, students should be able to:

- Understand the implementation of lists, stacks, queues, search trees, heaps, union-find ADT, and graphs and be able to use these data structures in programs they design
- Prove basic properties of trees and graphs
- Perform breadth-first search and depth-first search on directed as well as undirected graphs
- Use advanced sorting techniques (heapsort, mergesort, quicksort)
- Determine the running time of an algorithm in terms of asymptotic notation
- Solve recurrence relations representing the running time of an algorithm designed using a divide-and-

conquer strategy
- Understand the basic concept of NP-completeness and realize that they may not be able to efficiently solve all problems they encounter in their careers
- Understand algorithms designed using greedy, divide-and-conquer, and dynamic programming techniques

Required Texts/Readings

Textbooks

This textbook is very widely used, and I hope it will come in handy beyond this course. The 3rd edition, for the material we cover, is quite similar to the 2nd edition. (The 2nd edition managed to obfuscate a few issues from the 1st edition while clarifying others.) I think the majority of changes from the 2nd to the 3rd edition are in sections we don't cover, though some of the exercises and readings have changed. When possible, I will post assignments for both the 2nd and 3rd editions of the book.

Introduction to Algorithms, 3rd Edition
Cormen, Leiserson, Rivest, and Stein
ISBN-10: 0262033844
ISBN-13: 978-0262033848
MIT Press, 2009

You can find errata (bug reports) for the book http://www.cs.dartmouth.edu/~thc/clrs-bugs/bugs-3e.php, for whichever printing of the book you get.

Other Technology Requirements

Students are expected to have wireless laptops, with Java and a Java IDE installed.

Other Readings

I will make any additional reading material as needed, either by a link or by hardcopy.

Course Requirements and Assignments

Workload

The following will be regularly assigned for time outside of class:

- Video lectures
- Rote homework problems or video quizzes given in Canvas.
- Readings from textbook or handouts
- Written homework problems, such as textbook exercises
- 4 Programming assignments
- 2 Practice Exams

During the introduction of new material, homework is our chance to learn by making mistakes. It is expected that you will make an effort in all of the above for the sake of learning, and to give yourself feedback about your understanding the material.

The purpose of the rote homework is to give you enough practice working on problems to either understand how to solve the problems, or at least to learn from solutions to those problems.

I encourage you to watch each video closely and carefully, pausing and taking notes if needed, before

attempting homework on that video. If you don't do well on your first attempt at the homework, it indicates that you have not watched the video closely enough. This should give you some immediate feedback on whether or not you are watching the videos closely enough.

Canvas homework can (usually) be taken up to 5 times, but it is really your first attempt that will give you the most feedback on how well you understand the material. Using what would be a terrible homework assignment to give an example of the difference between the first attempt and later attempts: imagine that you are told to study the first 15 digits of pi. After doing that, if you were asked "What digit is in the is the 10 millionths position?" and you could answer that, it would give some indication that you had memorized the requested digits. On the other hand, if you got it wrong, and then looked at the digits of pi before taking a second attempt to answer the question correctly after memorizing the single digit requested, it would give little confidence that you knew any other digits of pi. Hopefully, your homework will be much more interesting than that example.

If, even **after a second attempt** at a Canvas homework, you still cannot get the correct answers, you can ask a classmate or somebody else to help you. **You may not simply ask them for the answers. (That counts as academic dishonesty.)** You can, on the other hand, have them walk you through the problem, step by step, until you get to an understanding of the answer. (For this course, that counts as diligence. You should understand your submitted answers, otherwise you need to spend more time on them.) Because the homework has a time limit for submission, you might want to take your 3rd attempt, if needed, to record the questions, so that you can do them with more time on your 4th attempt. You really shouldn't need a 5th attempt. The reason attempts are limited is to stop anybody from making blind submissions until they happen to get the correct answers.

For written homework problems, they will not be graded on correctness, but on whether or not it looks like an honest effort attempt was made to answer that problem. **You may discuss problems with others, but anything other than superficial comments must be documented.** You should not simply copy solutions, nor look for solutions (on the web or elsewhere), but if needed you can have somebody explain a problem to you in full, until you understand the solution. I might only return solutions for those problems for which you turn in evidence of putting in enough effort.

For both Canvas and written homework, you should do each homework, unless you are positive that you understand the topic so well that doing the homework would not be a good use of your time. And for those students? They should be the ones helping classmates to understand the material, as outlined in the two preceding paragraphs.

You are expected to code your own programs, but can get help from others. Talking is good. Sharing code is not, and this includes reading their code and retyping it, or having them dictate it to you. Do not look for premade solutions. If you get help from others, it should be documented in comments. **Do not copy code.** You should understand what your code does. If I ask you what something does in your code, and you don't understand why it is in your code or what it does? That is unacceptable, as it indicates that you are submitting work which is not your own. If you can get somebody to explain something to you in detail, to the point that you can understand and code it, that is okay. **Your code will be checked for correctness, but graded on your ability to answer questions about it. It is possible to get credit for code that doesn't work. It is possible to not get credit for working code if you don't seem to understand what it does. This latter case may also be deemed academic dishonesty.**

Success in this course is based on the expectation that students will spend, for each unit of credit, a minimum of 45 hours over the length of the course (normally three hours per unit per week) for instruction, preparation/studying, or course related activities, including but not limited to internships, labs, and clinical practica. Other course structures will have equivalent workload expectations as described in the syllabus.

Class Participation

Class partiticipation and feedback are very important to keep the course interesting. *If I am covering material too slowly or quickly, or if I am not clearly explaining things, you must let me know.* I prefer an interactive learning environment. If you disagree with something I say, speak up. Argue with me in front of

the class. It will make the class better, and right or wrong, constructive interaction will not not hurt your grade. If you are correct, clearly my mistake should be corrected. If you are incorrect, probably I have not explained something clearly anyway, and at least half of the class is confused by it. Point it out right then and there. In cases of exceptional participation that seem to benefit the class as a whole, I reserve the right to improve a student's grade by up to 1/3 grade.

University Policies

Per University Policy S16-9, university-wide policy information relevant to all courses, such as academic integrity, accommodations, etc. will be available on Office of Graduate and Undergraduate Programs' Syllabus Information web page at http://www.sjsu.edu/gup/syllabusinfo/.

Drop and Add Dates

Note that for this semester, the last day to drop without consequence is *Tuesday, February 5?*, and the last day to add is *Tuesday, February 12?*. After these dates it becomes very difficult to drop or add a class, so be sure you are where you want to be before these dates arrive!

Grading

Grading will likely be different than anything you have ever seen or heard of before.

Homework:

Homework is for learning material, and for giving you feedback. Grading homework in the traditional sense seems to encourage students to worry about grades more than learning the material, which is not considered to be the best way to learn. Because of this, individual homework scores will not have a direct effect on the grade of that individual, they will only have a small, indirect effect, as described below. Of course, proper effort on homework by an individual will likely result in higher exam scores for that individual.

Instead, homework scores will be slightly used to set the curve for the exams: if the class as a whole is doing its homework, I would expect to assign a higher letter grade to the median test score than if the class, as a whole, is not doing its homework.

I do **not** believe in assigning homework for the sake of assigning homework. If you understand the material without doing the homework, and the homework seems like a waste of time to you? In that case, your advanced understanding should be reflected by your performance on the rote tests (described below), with the expectation that you should do well on those tests. If you are in the top half of the class on those exams, your homework score will be ignored when setting the curve for the class, so your lack of homework will not hurt anybody. You are encouraged to help your classmates to learn the material, teaching it to them will help you to understand it even more.

Note, **this policy should highly discourage students from cheating on homework:** the (admittedly small) risk of being caught is balanced by the fact that your individual homework score will not help you individually, it will only play a very small role in setting the curve for the exams. Because your homework is only one of many homeworks used to set that curve, in cheating, you are taking all of the individual risk, while not reaping any individual reward. Additionally, you won't actually learn the material.

Due to the way that Canvas homework (multiple attempts) and written homework (graded on effort) are graded, the students making a real effort on homework should, as a whole, expect to get credit for a high percentage (90%?) of homework problems.

Rote Tests: There will be two in-class tests in the final weeks of the semester, that will test "rote"

knowledge. You will be given a template for each exam a week in advance. These two exams combined will be scored on a (curved) scale from F to B-. The curve will be based on both test performance, and general class homework performance.

Final Exam:

- For Section 3, the final exam will be Tuesday, May 21, at 7:15-9:30
- For Section 4, the final exam will be Thursday, May 16, at 7:15-9:30
- For Section 5, the final exam will be Wednesday, May 15, at 9:45-12:00
- For Section 6, the final exam will be Friday, May 17, at 9:45-12:00

Your final exam will be a mixture of rote and advanced questions. I will give you some practice problems (during the semester) to give you some idea of what that means, but you will **not** get a template for the exam. It will be graded on a curve, from F to A+.

Programs:

You are expected to work on all 4 programs, to submit your programs, and to be able to explain your programs to me. While most programming will be done outside of the classroom, for the first two programs, there will be some time spent during class for you to program, ask questions, and explain your programs to me. Different programs are worth different numbers of points (2, 4, 3, and 2 respectively), for 11 points total. For each program, if you are able to program enough of the work, and explain it to me, you can get credit for the program, even if it does not answer every test case. A program that works perfectly that you cannot explain will not get credit, and may violate academic honesty policies. If you get under 2/11 program points, you will lose an entire letter grade for the course. At least 2 points, but fewer than 5, you will lose 2/3 of a letter grade. At least 5 points, but fewer than 8, will lose 1/3 of a letter grade.

Course grade

Your course grade will be the maximum of your Rote Tests grade and your Final Exam grade, as modified by your Programs.

Recording Lectures or Sharing Course Materials

You can make audio recordings of class for your own personal use, but they should not be reproduced or distributed. If, for some reason, you want video, please come discuss it with me.

Course material developed by the instructor is the intellectual property of the instructor and cannot be shared publicly without his/her approval. You may not publicly share or upload instructor generated material for this course such as exam questions, lecture notes, or homework solutionswithout instructor consent.

Tentative Class Schedule

A precise schedule will be online in the school's Canvas system. Below is just a rough outline, by week rather than date. (A few special dates, for in class labs and exams, are added.) Although we usually index from 0 in CS, week 1.1 the first (and only) meeting of the first week of class. Week 2.2 refers to the second meeting of the second week.

Approximate Week **Subject to change**	Topics Covered
Week 1.1	Introductions, Administrivia, Warm-Up
Week 1.2	Finish Warm-Up, Review, Application

Week 2.1	Defining Problems and Loop Invariants January 30, bring computers, we might use them.
Week 2.2	Feb 4, Heap Lab: bring computers
Week 3.1	Asymptotic Notation
Week 3.2	Recurrence Relations, Recursion Exercise
Week 4.1	Master Theorem, Sorting, Exercises
Week 4.2	Quicksort, Select
Week 5.1	23/234 Tree, Problem Lower Bounds, exercises
Week 5.2: February 27	23-Tree Code Review: bring computers
Week 6.1	External Memory and exercises
Week 6.2: March 6	23-Tree Code Review: bring computers
Week 7.1	Exercises, Review
Week 7.2	Start Graphs, TopSort
Week 8.1	Graphs
Week 8.2	Graphs
Week 9.1	Graphs
Week 9.2	Graphs
Week 10.1	Graphs
Week 10.2	Review, Dynamic Programming
Week 11.1	Dynamic Programming
Week 11.2	Knapsack, Intro to NP
Week 12.1	NP
Week 12.2	NP decision vs. Optimization
Week 13.1 April 29	Review and Advanced Topics
Week 13.2 May 1	Review and Advanced Topics
Week 14.1 May 6	Rote exam 1: through B-Trees
Week 14.2 May 8	Rote exam 2: after B-Trees
Week 15.1 May 13	Exams returned, Review for Final.

- For Section 3, the final exam will be Tuesday, May 21, at 7:15-9:30
- For Section 4, the final exam will be Thursday, May 16, at 7:15-9:30
- For Section 5, the final exam will be Wednesday, May 15, at 9:45-12:00
- For Section 6, the final exam will be Friday, May 17, at 9:45-12:00