CSCE 350 - Data Structures and Algorithms

Credit Hours

3 hours

Contact Hours

3 lecture hours, 14-week format, about 150-minutes per week from two 75 minutes lectures per week in regular face-to-face lecture.

Class Times

This class is listed as “Face-to-Face”, i.e. is a regular face-to-face lecture format in class.

Section	Class Time	Room
350-001	Monday/Wednesday 3:55-5:10PM	Swearingen 2A27
350-002	Tuesday/Thursday 1:15PM-2:30PM	Innovation 1400

- Important dates this semester:
  - MLK Day: Jan. 16th, Monday
  - Test 1: February 13th /14th
  - Midpoint of Semester: March 2nd
  - Spring Break March 5th-12th
  - Test 2: March 22nd /23rd
  - Last day to drop a course or withdraw without a grade of "WF" being recorded: March 27th
  - Final Exam: See following table.

Final Exam Times (official times, according to Registrar)

Section	Final Exam Time
350-001	April 28th, Friday, 4:00 p.m. (third day of exams)
350-002	April 27th, Thursday, 12:30 p.m. (second day of exams)

Instructor

James O’Reilly (oreillyj@cse.sc.edu)

Teaching Assistant

Qinyang Li (qinyang@email.sc.edu)

- Specific questions related to the grading or submission of a particular assignment go to the TA, though please copy me. All others go to me.

Office Hours

All office hours are posted here: https://cse.sc.edu/~oreillyj , or more directly https://james-oreilly.youcanbook.me. If you need to meet outside these times, please email me at oreillyj@cse.sc.edu. We can meet F2F or on Teams but posted office hours will generally be for those physically at my office.

Regular business hours are preferred but not mandatory if these times do not work. **Please give me a window of times to pick from when requesting to talk outside of regular office hours to avoid another round of emails.**

Bulletin Description

Techniques for representing and processing information, including the use of lists, trees, and graphs; analysis of algorithms; sorting, searching, and hashing techniques.

Prerequisites

CSCE 240; MATH 174 or MATH 374 or MATH 574

Learning Outcomes

Students will be able to:

- Analyze algorithms using formal analysis measures.
- Describe the relevance of abstraction to problem solving.
- Analyze and use lists, trees, and graphs.
- Apply common algorithm design techniques such as brute force, divide-and-conquer, decrease-and-conquer, transform-and-conquer, dynamic programming, and the greedy technique.
- Use appropriate data structures in programming and otherwise.

Required Materials and Software

Required Textbooks:

Anany V. Levitin, *Introduction to the Design and Analysis of Algorithms*, 3rd Edition, Pearson, 2011. 978-0132316811

The required textbook is not particularly useful as a general reference (consider renting) but is necessary. I desirable for you to read the book on your own and there will be problems from the book.

We will also use GitHub for programming assignment submission. The college provides licenses for VMware, which will be optional, or lab machines for those without enough hard drive space on their machines. Students may remote in over VPN to the department lab machines. Instructions will be given early in the semester and are, at the moment, subject to change.

All readings/materials comply with copyright/fair use policies.

Course Overview

Topics covered (Time spent is Approximate)

1. Structured programming, stacks, queues, lists (3 hours)
2. Determining the Running Time of Programs, Order of Magnitude Analysis (6 hours)
3. Brute force (3 hours)
4. Divide-and-Conquer (4 hours)
5. Dynamic Programming (6 hours)
6. Transform-and-Conquer (4 hours)
7. The Greedy Technique (3 hours)
8. Decrease-and-Conquer (3 hours)
9. Graphs (3 hours)
10. Reviews and exams (4 hours)
11. More as time permits (NP-completeness, optimization)

Schedule of Topics (Approximate)

Week	Topics	Assignment(Released)
Week 1	Intro, Chapter 1 (Review of background material)	HW1 (50)
Week 2	Chapter 2 (Order of Magnitude Analysis, Running time of Algorithms)	HW2, Program 1
Week 3	Chapter 2 cont'd (More analysis, Recurrences, Sums for Iterative Algs., Limits)	HW3
Week 4	Chapter 3 (Brute Force) Exhaustive Search, Generate and Test	HW4*, Program 2*
Week 5	Chapter 4 (Decrease and Conquer) DFS, BFS	Sample Quiz 1 (25) Test1(chs1-4) (250)
Week 6	Chapter 5 (Divide and Conquer) Closest Pair, Convex Hull, Quick Sort, Merge Sort, DFS for Topological Sorting	Program 3
Week 7	Chapter 6 (Transform and Conquer) Presorting, AVL and Self-Balancing Trees, 2-3 Trees, Heaps	Program 4
Week 8	Chapter 7 (Space and Time Tradeoffs) Counting Sorts, Boyer-Moore, Hash Tables, **Spring Break**	HW5 (50)
Week 9	Chapter 7, cont'd B-Trees, time permitting	Program 5*
Week 10	Chapter 8 (Dynamic Programming), Basics, Warshall's, Floyd's	Sample Quiz 2 (25) , Test2(chs5-7) (250)
Week 11	Chapter 9 (Greedy Algs.) MST/Prim's, Dijkstra's	Program 6
Week 12	Chapter 9, cont'd	HW 6 (50)
Week 13	Chapter 11 (Limitations of Alg. Power)	Program 7*
Week 14	Special Topics, Review, "Flex" Time	
Final Exam	See table (link)	Final Exam (250/375/500)

Notes: A Week corresponds to (2) lectures and may not always line up with Monday/Wednesday Tuesday/Thursday of the same week. HW# is Homework numbered #. All assignments 100 points unless followed by a "(#)", where # is the number of points. See final exam description for how one or more (~25%) portions of the final exam may be skipped. *=see extra credit/drop section.

Student-to-Instructor (S2I), Student-to-Student (S2S), and Student-to-Content (S2C) Interactions

This course will be delivered **fully face-to-face**. I am going to attempt to record lectures, as the rooms assigned are well laid out for this, but the quality will not be sufficient for more than occasional use.

• Student-to-Instructor (S2I) Interaction: Students will attend regular lectures and may ask questions there, by email, or in office hours.

- Discussions in virtual office hours may be recorded, *assuming the student agrees*, and uploaded to Blackboard if the discussions are like in-class lecture questions and not regular "office hour" questions, e.g., troubleshooting programming assignments. Recorded lecture questions may also be uploaded.

• Students-to-Student (S2S) Interaction: Students will engage in discussions through in class discussion or collaboration on homework.

• Student-to-Content (S2C) Interaction: Students will engage with course content by completing homework/quizzes to follow the lecture, and other assessments.

Communication and Feedback

The instructor will reply to all feedback in a reasonable amount of time; the same is expected of the students. Specifically:

• Communication: Responses to email communication and questions will be provided within one business day. Questions arriving on the weekend may be answered the following Monday.

• Assignment Grading: Grades for assignments (quizzes, programming assignments, tests) will be returned within one business week (5 business days) of the due date.

If I fail to respond within these timeframes, feel free to send another email. Sometimes there's a spam filter (or an accidental swipe on my cellphone…).

Technology

Specific Technologies, Programming Languages, Programs

The students will install and/or use C++ with STL, GTest, *basic* git (through GitHub), and Ubuntu Linux.

Minimal Student Technical Requirements

Students should be familiar with C++ and have basic Linux Operating System, Ubuntu preferably, skills. The prerequisites cover these sufficiently.

Assignments

- Homework and Sample Quizzes, ~20-25%, 400-500 points:
    - I am going to do just homework, no real quizzes this semester. They will be submitted online, excepting the sample quizzes.
    - There will be two "sample quizzes", given ahead of time before class and gone over in class, at quarter weight (25 points), but will be "effort-graded", if there's work shown. The purpose of these quizzes is to ensure you see a problem type before a test and have an opportunity to see the solution, discussed in class and a "take-home quiz". You should scan/copy your solution before submission if you want a copy to study with.

    - You may work together in small (2-4 person groups, but submitting your work each separately) on Homework but submitting your homework implicitly states that you could do the work independently, i.e. by the time you submitted, you "got it". Copying is still plagiarism; if I have a collection of assignments that are conspicuous I may ask you to do similar problems to distinguish cheating from collaboration.
- Tests: Two, ~13% each, 250+250 points. Closed book, closed notes. Formula sheets for things like summation closed forms will be provided.
- Final Exam: ~14.23-25% (cumulative, see below) Material from the earlier tests may be required to solve/interpret problems for later material correctly, e.g. "Give an algorithm that runs in worst case O(n log n) time that does _____" would be fine for the final. You will be given the option of skipping portions of the final, subject to the following rules:
    - **You must have a B (79.5) or above on the corresponding test to skip a section.**
    - You must elect to do this before the final (details to follow). You won't get to see the problems before making your decision.
    - The final exam will be about 50% (250 points) post test 2 material (chapters 8,9,11) and **cannot be skipped**. This material is somewhat more advanced (Dynamic Programming, Greedy algorithms) so it likely that your score on this section will be lower than your second test score (the first test will be more challenging for those who have a mathematical weakness).
    - If you skip a section, your total points for the course will drop, in the denominator as well. This will not be reflected in Blackboard.
- **Late Policy for Assignments**: -15% of total points per day late, rounded to the nearest point, up to two days late. Not accepted after that. I may for an individual assignment restrict the maximum lateness, especially as we approach the end of the semester or a test (I will want to release solutions at least one day ahead of a test). Do not wait until the last minute. Please don't send us frantic emails if you have to submit a few minutes late, we will ignore the late penalty for assignments up to a little late, where "a little late" means about one hour late.
- Programming: ~30% 5-7*100 point programming assignments. We will use C++ with the STL. Rules:
    - Does not compile or goes into infinite loop/recursion/out of memory: 0% More if we can figure it out easily but less than 50% if not.
    - Please run your code one last time before submitting it.
    - **Important:** I use scoring scripts, which output a score. These are typically thin wrappers for unit tests which give you a *floor* for your grade if you satisfy the requirements. Notes on grading:
        1. Your grade may go *up* if there's significant work. You should not expect more than 50% of the points back on anything incomplete with the exception of monolithic assignments (where you solve a single problem), but that's contingent on your work, documentation, and possible unit tests you provide.
        2. Your grade may go *down* if you somehow pass a test and either fail to satisfy the requirements, such as "use dynamic programming to …" or "solve Problem X in

O(n) time" and do not. It is possible but too cumbersome to test this so we may just look at your code (and the unit tests have times as well).

3. Anything that can be construed as "cheating" the test will just get a zero for the *whole* assignment, e.g. remembering the expected outputs to a short list of inputs and just recording them. I may just swap out tests/input. It is a waste of the grader's time to deal with "sneakiness".

- Extra Credit and Drops: I will drop the lowest two assignments of HW4, Program 2, Program 5 and Program 7. If the second lowest is at least 60%, the points will be added back to the numerator (but its max points not added to the denominator – this is the extra credit). These assignments are likely to be available for longer periods.
- **Important: I will not necessarily tell you everything that is expected. If you have any doubt that your submission is adequate, please ask.** Notes:
  1. If I think it might not be fairly obvious, given the material we have covered so far, then I will tell you any time requirements, e.g. "Give code to tell if two large same length strings of characters using 'A','B','C',…,'Z' are anagrams of one **another in worst case O(n) time**, where n is the number of characters in one string. ".
  2. **Important:** We will assume you know all the material up to a certain point so using/implementing a quadratic time $O(n^2)$ sort, e.g. Selection Sort, when you are aware of O(n log n) sorts like Quicksort (or just using std::sort), suggests a lack of understanding *and may be penalized*.
  3. Combining the above, I can and will assume that you've absorbed all the information up to a certain point. An example: "Give an algorithm to find duplicates in a given array". If you use a double-nested loop to test pairwise or use selection/insertion sort, all $O(n^2)$, *after* we've talked about merge sort or quick sort, you would lose points as Mergesort and Quicksort are both O(nlogn). After we've covered hash tables you would lose substantial points (>50%) for selection sort and fewer for merge sort as you can get O(n) performance. If you aren't given a target efficiency or approach, then the goal is to figure out the *best* way, given what you should know.
  4. If you have code, please push to GitHub and send me a link to the repository, if you think I may need to look at it. This makes it much more likely that you will receive a quick response, explained below.
- We use GitHub and this is a *good* thing for all of us but I expect you to use it for backups, as well as submission:
  1. If you solve even *part* of a problem and take a break or whatever, then go ahead and push. These will be individual assignments, so you don't have to worry about breaking other people's code.
  2. You should, at a minimum, *look* at your final submission on GitHub. Cloning to a new directory and running all tests will probably be nearly as quick and less error prone once you are accustomed to working with git on the command line.

If you have not had CSCE 247, our usage is very basic and has had very few issues over the years I've done it this way.

3. "I lost my code" or "My hard drive died" is **not** an excuse for losing a lot of work. Backing things up is part of our job. (Please let us know if you encounter major technical difficulties at some point but this is mostly for students who get a zero because of a git mistake *early* in the semester.)

- Programming Questions (by email to me oreillyj@cse.sc.edu): We will talk about programming early in the semester, when the first assignment is assigned.
    1. Please screenshot or at least copy the error with a small explanation.
    2. Please push to GitHub before sending the question. Include in your email a hyperlink to your repository. Some questions can be answered by just looking at your code (easy on a smartphone, if I'm away from my computer) or perhaps running it. Both are made easier by just sending me a link. Your link, easily copied from GitHub will generally have the form of

       https://github.com/csce350-spring23/program-<NUM>-<GH Username>

       - <NUM> is some number 1,2,…
       - <GH Username> is your GitHub username, which is often *not* the same as your UofSC username.

- GitHub Difficulties (email me at oreillyj@cse.sc.edu):
    1. Please screenshot or at least copy an error with a small explanation.
    2. Please give your GitHub Username and, if you are not using your UofSC email, your UofSC username.
    3. If it is an issue preventing submitting on time, I expect you to zip your submission and send to Qinyang (qinyang@email.sc.edu). Copy me at oreillyj@cse.sc.edu. We will talk more about programming early in the semester, when the first assignment is released.

- There may be *some* opportunities for a *significant (altogether about half a letter grade)* of extra credit throughout the semester.
- I am using total points (summing to about 2200) this semester.
    - 3*Smaller Homework will be 50 points each.
    - 3* Longer Homeworks will be 100 points each.
    - 2* Sample "quizzes" will be 25 points each.
    - Programming Assignments (5-7) will be 100 points each.
    - Tests (2) will be 250 points and the Final Exam will be 250-500 points each.
    - Looking at "Total" Points or "Weighted Total" in Bb may be helpful but not 100% accurate because of the extra assignments which might be dropped.
    - Rubrics and Solutions will be uploaded to Blackboard as they become available.

Test Security and Honor Code

All students will be expected to follow the Honor Code for all assignments and explicitly state that they have done so for the Tests and Final.

Grading Schema

Item	Weight, assuming full final
Homework	~ 25%
Tests (2)	~25% , TOTAL
Final Exam	14-25%
Programming Assignments	~25-35%%
Extra Credit	~5%

Grading Policy (rounding 0.5 up); Scale: A (90-100%), B+ (86-89%), B (80-85%), C+ (76-79%), C (70-75%), D+ (66-69%), D (60-65%), F (0-59%)

Important: <u>You are responsible for checking your grades in Blackboard regularly</u>. No grading issues for an assignment will be addressed more than two weeks after a grade has been entered, or less as the end of the semester comes, where I will send reminders for you to check. The grader will never leave something they have graded blank – there will be either a zero or some other numerical grade and some feedback explaining the score.

Side note regarding Blackboard grades: The grades on Blackboard are "raw". I will do the *minimum* manipulation there to preserve the data's integrity. I won't go back and do explicit drops, scale to 2200, or anything like that. I want you to see exactly what I will use to calculate your grade, "unfiltered", as errors are almost always per assignment. It's not hard to reduce the denominator by 50 points and then do the division to get a "current" grade.

Pass/Fail Option

This course is normally taught to majors and not typically offered as Pass/Fail.

Attendance Policy

"Attendance" means reviewing the material and keeping up with the course. Students are expected to be responsible and keep up with the material. If there is a sudden issue, e.g. from illness or injury, notifying me as soon as possible is best for all involved. I do not track attendance but presence in the classroom, activity in the forums, and logging into Blackboard and viewing material (which can be tracked) are things that can factor into an assessment of whether a student has done their part to keep up with the material.

Accommodating Disabilities

Students with disabilities should contact the Student Disability Resource Center. The contact information is below:

1705 College Street, Close-Hipp Suite 102,
Columbia, SC 29208

Phone: 803.777.6142 Fax: 803.777.6741
Email: sasds@mailbox.sc.edu
Web: https://sc.edu/about/offices_and_divisions/student_disability_resource_center/index.php

These services provide assistance with accessibility and other issues to help those with disabilities be more successful. Additionally, students with should review the information on the Disabilities Services website and communicate with the professor during the first week of class. Other academic support resources may help students be more successful in the course as well.

Library Services (http://www.sc.edu/study/libraries_and_collections)

Writing Center (http://www.cas.sc.edu/write)

Carolina Tech Zone (http://www.sc.edu/technology/techstudents.html)

Honor Code

The Honor Code is a set of principles established by the University to promote honesty and integrity in all aspects of the campus culture. It is the responsibility of every student at the University of South Carolina to adhere steadfastly to truthfulness and to avoid dishonesty in connection with any academic program. A student who violates, or assists another in violating the Honor Code, will be subject to University sanctions. (See http://www.sc.edu/policies/ppm/staf625.pdf)

All Cheating will result in a grade of zero for the assignment and a referral to the Office of Academic Integrity. Cheating on a test/exam or any combination of assignments totaling 10% or more of the final weighted grade is grounds for failure in the entire course, pending a final ruling by Academic Integrity.

Regarding programming assignments:

- Code will be run through Moss whenever possible.
- There are two central issues with coding assignments:
    - Answers or partial solutions to assignments are often available, even outside of "cheating" sites.
    - The Internet provides a deep bank of resources for many problems, libraries, etc. that you may find *legitimately* useful for learning some topic/technology or applying what you have learned to a new problem.
- Accordingly, a blanket ban on searching the Internet is unreasonable but then so is searching the Internet for an exact or even large partial solution to a given problem. Some situations are clearly cheating, others just using what resources you have at your disposal, and still more are somewhere in between. If you're implementing Quicksort, typing "C++ array" or "C++ std::vector" is perfectly fine but "C++ quicksort" or "C++ Hoare partition" is obviously not. Here are the guidelines we will use for this course:
    - All substantial code that is "paraphrased" from another source must be cited.
    - You should never need to cite more than a small amount of code, i.e., the next few following, obviously related lines; do not bother citing very basic general features you had to look up (e.g. pointers).

    - A citation for a large region of code that trivializes a problem is no protection against an allegation of violating the Honor Code. **A resource that offers too much of a solution to a given problem is too specific and should never be used or read**. Additionally, if we must fairly "subtract" the large region's code from your work, then you are certainly better off just not turning it in, leaving that part blank, or developing a partial solution that demonstrates what you *do* understand. It will be assumed that a student who submits a large block of code, even if cited, merely hoped we would not notice. Some assignments may be graded by script and it is expected that all students always follow the Honor Code, regardless of whether the professor or their teaching assistant is looking.
    - You must understand all code submitted. You may be called upon to explain it either your professor or the Teaching Assistant if we are concerned there may have been an Honor Code violation.
    - In general, smaller "snippets" are fine, assuming you understand how they work and so long as they do not trivialize a problem. A "snippet" should be a *small*, *cohesive* piece of code to solve a *single* programming task. Learning the ways other, likely more experienced, coders use a language or library is a good way to become fluent in a programming language and its idioms. You should not feel a need to obfuscate a single line of code if it matches what you need, given a citation, but if it is more than a very small number of lines, then that is much more suspect. You should never copy-paste code into your solutions.
    - Whether a region of code is evidence of cheating or not will generally be assessed based on:
        - Its size, i.e., whether it is a "snippet" or stolen *logic*, such as a whole (sub)algorithm.
        - Whether it trivializes a given problem, i.e., how much of the given problem is solved by your source. This is especially hazardous for *simpler* problems.
        - How similar it is to another programming solution, either a fellow student's or something online.
- Collaboration: It is perfectly fine to talk in general terms with your peers, i.e. talk about language features, the basics of an algorithm to be implemented, etc., but in general, students working on individual assignments should almost never look at one another's code. **One easy rule: A student who is not done with an assignment should never view another's source code for that assignment.** If one student decides to help another then referring to general resources on an algorithm, demonstrating on pen and paper or whiteboard, or referring to programming tutorials/StackOverflow is certainly more illuminating than the final answer. This statement on collaboration is largely paraphrasing the following link and you can see there https://integrity.mit.edu/handbook/writing-code#:~:text=Search-,Writing%20Code,inline%20comment%20in%20the%20code for more examples. One exception is when someone is stuck and the peer viewing their code cannot use the information, e.g. the helper is *already done* and all the helper is doing is, e.g., finding a small syntax error or helping with a small region of code, especially if the helper is not viewing *their own* solution while helping. The two submissions will almost certainly be very different if there was not too much

help given and the person being helped should then be able to fully reconstruct and explain their solution.

- If I feel any assignment, programming or otherwise, is an Honor Code violation and have any doubt, I will consult a colleague and see if they agree. If so, then the case will be forwarded to the Office of Academic Integrity at https://www.sc.edu/about/offices_and_divisions/student_conduct_and_academic_integrity/index.php .
- When in doubt: Ask us whether what you are considering doing is acceptable. A "no, try looking at the problem like…" is much better than later being found to have plagiarized another's work.
- On a somewhat lighter note, How to Cheat: https://github.com/genchang1234/How-to-cheat-in-computer-science-101