CS 2302 Data Structures
Fall 2018

1. General Information

Instructor:

Diego Aguirre
Email: daguirre6@ utep.edu
Web: www.aguirrediego.com
Office hours: Tuesdays and Thursdays 3:10p.m-4:10p.m, or by appointment.
Office location: CCSB 3.1022
Gmail: diego4.aguirre@gmail.com

Teaching Assistant:

Anindita Nath
Email: anath@miners.utep.edu
Office hours: TBA
Office location: TBA

Lectures

TR 1:30p.m. – 2:50p.m. in LART 318

Class Website

https://aguirrediego.com/fall-2018-data-structures/

Book

Data Structures Essentials (Python) - zyBook

2. Objectives and Outcomes

This is the third and final course in the fundamental computer science sequence. Students will learn about fundamental data structures and analysis and design of algorithms.

Level 3: Synthesis and Evaluation:
Level 3 outcomes are those in which the student can apply the material in new situations. This is the highest level of mastery. On successful completion of this course, students will be able to

1. Given a problem, judge which data structures are required to solve it efficiently and justify the selection.
2. Given a non-recursive algorithm, examine its loop structure, assess its asymptotic running time, and express it using big-O notation.
3. Given a recursive algorithm, examine its structure, formulate and solve a recurrence equation defining its running time, and express it using big-O notation.
4. Design and implement solutions to computational problems based on iteration and recursion.
5. Trace the behavior of non-trivial methods and algorithms.

Level 2: Application and Analysis:
Level 2 outcomes are those in which the student can apply the material in familiar situations, e.g., can work a problem of familiar structure with minor changes in the details. Upon successful completion of this course, students will be able to:

1. Describe, implement, and use the following data structures:
    a) Heaps
    b) Hash tables
    c) Balanced trees
    d) Graphs
    e) Disjoint set forests
2. Describe, implement, and apply the following graph algorithms:
    a) Connected components
    b) Breadth-first search
    c) Depth-first search
    d) Topological sorting
    e) Minimum spanning trees (Kruskal's and Primm's)
    f) Single-source shortest paths (Dijkstra's algorithm)
3. Trace the behavior of recursive programs using activation records.
4. Reason about the running times of algorithms in relation to the size of their inputs.

Level 1: Knowledge and Comprehension
Level 1 outcomes are those in which the student has been exposed to the terms and concepts at a basic level and can supply basic definitions. On successful completion of this course, students will be able to:

1. Identify and explain the following algorithm design techniques:
    a) Greedy algorithms
    b) Divide and conquer
    c) Dynamic programming
    d) Backtracking
2. Explain the concept of NP completeness.
3. Explain the utility of randomized algorithms.

3. Policies and Other Information

Prerequisites: Minimum "C" grade in CS2401 and MATH 2300.

Textbook: Reading and laboratory assignments will be drawn from the Data Structures Essentials (Python) zyBook. You are required to obtain this zyBook for use in this course.

Grading: Final grades will be based on a combination of lab projects, homework assignments, in-class attendance and performance, three partial exams, and a final exam. The approximate weights are as follows:

- 25% - Lab projects
- 10% - Homework assignments, in-class exercises, and quizzes
- 39% - Partial Exams (3 exams, 13% each)
- 26% - Final Comprehensive Exam

The nominal percentage-score-to-letter-grade conversion is as follows:

- 90% or higher is an A
- 80-89% is a B

- 70-79% is a C
- 60-69% is a D
- below 60% is an F

Additionally, any one of the following **will result on a final grade of F**, even if the overall average is greater than 60%.

- Obtaining an average of less than 60% on the lab projects
- Obtaining a grade of less than 50% on the final exam
- Obtaining an average of less than 50% on the partial exams
- Not submitting ALL lab projects by the end of the semester, even if they are too late to receive credit

We reserve the right to adjust these criteria downward, e.g., so that 88% or higher results in an "A", based on overall class performance. The criteria will not be adjusted upward, however. You must earn a "C" or better to be able to register for upper division computer science courses.

Late homework submission: Homework up to a day late will receive up to 80% of full credit, and it will not be accepted after that.

Collaboration: Collaboration among students is strongly encouraged.
It is OK to:

- Talk with other students about approaches and ideas.
- Get ideas and extra information from the internet, books, etc.

However, it is not OK to:

- Share code with another student (if a piece of code is submitted by two or more students, both students are guilty of cheating, regardless of who wrote the original code).
- Use code acquired from an outside source (the internet, a friend, etc.)
- Look at another student's code
- Debug another student's code

We will use software to detect plagiarized programs and take appropriate disciplinary actions if necessary.

Cellular telephones are prohibited during lecture and lab sessions. Students are required to turn off their cellular telephones before entering the classroom.

Attendance policy: Students are expected to attend all lectures. Students arriving more that five minutes after the start of a lecture won't be allowed to enter the classroom.

Disabilities: If you feel that you may have a disability that requires accommodation, contact the The Center for Accommodations and Support Services (CASS) at 747-5148, go to Room 106E Union, or email cass@utep.edu

4. Lab Submission Guidelines

Lab assignments will be posted on-line. Each lab grade will be computed from the following three elements:

- Report (40% of grade)
- Source code (60% of grade)
- Demo session (pass/fail)
- Code review (pass/fail)

Report:
You must submit a printed report of every lab that includes the following items:

- Introduction – Description of the problem you are trying to solve
- Proposed solution design and implementation – How did you solve (or attempt to solve) the problem? Provide an informal, high-level description. Description of your code (not the actual code). Explain the design choices you made, including how you broke the program into modules, your user interface, input and output, etc.
- Experimental results – Describe the experiments you performed to test your program and show the output your program produced. The experiments must be described in a way that allows anybody to replicate them using your code. **Include sample runs that illustrate the outputs and running times of your program under different types of inputs. If results are not included in the report, we will assume that your program does not work.**
- Conclusions – Explain what you learned from the project.
- Appendix – Source code
- A signed academic honesty certification stating the following: "I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class."

Reports will be graded as follows:

- Completeness (12%)
  Does your report cover all required aspects in enough detail?
- Clarity (10%)
  Are those aspects clearly explained?
- Language (10%)
  Is the report written with proper grammar and spelling?
- Presentation (8%)
  Is the formatting appropriate?

Source Code:

Working programs must be uploaded to GitHub and the link sent by email to both your TA and instructor, using the e-mail addresses listed below. Labs not submitted this way will not be eligible for credit.

- Diego Aguirre: diego4.aguirre@gmail.com
- Manoj Saha: msaha@miners.utep.edu

Include at the beginning of the subject line the following characters: "CS2302"

Source code will be graded using the following guidelines:

- Correctness (36%)
  Does the program compile?
  Does the program run correctly?
- Design (6%)
  Are operations broken down into methods in a reasonable way?

- Style (6%)
  Is the program indented correctly and consistently?
  Do methods and variables have meaningful names?
- Robustness (6%)
  Does the program handle erroneous or unexpected input gracefully?
- Documentation (6%)
  Do all program files begin with a comment that identifies the course, author, assignment, instructor, T.A., date of last modification, and purpose of program?
  Are all methods clearly documented?
  Are all non-obvious code segments clearly explained?

Programs will be tested by the TA. Students are free to use any programming environment they want, but it's their responsibility to ensure that there are no compatibility problems.

Demo session:
After submitting your program and your report, you must schedule a one-on-one session with your TA in which you will explain how your code works and he/she will ask questions to test your understanding of the program being submitted. The TA will then assign a pass/fail grade for this session. A student receiving a failing grade in this session will receive a grade of zero for the whole lab; otherwise he/she will receive the grade corresponding to the combination of submitted report and source code. Demo sessions will last around five minutes and will normally be scheduled during the T.A.'s office hours. If it the student's responsibility to make an appointment with the T.A. for the demo session in a timely manner. **Failure to schedule or show up for a demo session will result in a failing grade for the corresponding lab.**

Code review:
Pull requests is how some teams review their members' source code. For each lab, you will be asked to provide constructive feedback to 1 or 2 classmates using GitHub's pull requests. **Failure to provide feedback to your classmate(s) will result in a failing grade for the corresponding lab.**

Policy on late projects:
Lab project grades will be reduced by a factor of 10% for each working day or fraction they are late.

Official turn-in dates:
For grading purposes, the official turn-in date for labs is when all three parts are finished. Thus, a lab will be considered to be late if ANY of the three parts is late. There will be a two-working-day grace period for reports and a three-day grace period for demo sessions. For example, if a lab is due on Monday, the source code must be submitted on or before midnight on Monday, the report must be submitted on or before Wednesday and the demo must be shown on or before Thursday. You can't schedule a demo unless you have submitted the source code and the report.

Missing lab assignments:
All labs must be submitted by the end of the semester in order to pass the class. Additionally, a student who has submitted less than 75% of the labs due by the time a midterm exam is given won't be allowed to take that exam.

5. Standards of Conduct and Academic Dishonesty

You are expected to conduct yourself in a professional and courteous manner, as prescribed by the UTEP Standards of Conduct: http://admin.utep.edu/portals/68/Standards_of_Conduct_2013-2014.pdf

Academic dishonesty includes but is not limited to cheating, plagiarism and collusion. Cheating may involve copying from or providing information to another student, possessing unauthorized materials during a test, or falsifying data (for example program outputs) in laboratory reports. Plagiarism occurs when someone represents the work or ideas of another person as his/her own. Collusion involves collaborating with another person to commit an academically dishonest act.

Professors are required to - and will - report academic dishonesty and any other violation of the Standards of Conduct to the Dean of Students.