CSC 440

Design and Analysis of Algorithms

Prof. Noah M. Daniels

ndaniels@cs.uri.edu
`http://www.cs.uri.edu/~ndaniels/`
Tyler 250

Changes to course policies or schedule may occur in response to unforeseen circumstances. I will notify the class of any changes immediately.

Course Description: Algorithm design and analysis, advanced data structures, computational complexity. Sorting, searching including hashing and balanced trees, string pattern matching, polynomial and matrix calculations, graph and network algorithms, NP-completeness and intractability. Algorithmic techniques including dynamic programming, divide and conquer, and greedy algorithms.
Prerequisite(s): CSC 340.
Text: *Algorithms*
Authors: Dasgupta, Papadimitriou, & Vazirani; **ISBN-13:** 978-0073523408

Course Objectives:

At the completion of this course, students will be able to:

1. Choose appropriate algorithms to solve common problems
2. Design new algorithms to solve new problems
3. Determine the theoretical efficiency of those algorithms
4. Implement those algorithms in the programming language of their choice
5. Benchmark and present the actual efficiency of those algorithms

Grade Distribution:

Class Participation	10%
Projects	70%
Final Exam	20%

Letter Grade Distribution:

>= 93.00	A	80.00 - 82.99	B-	67.00 - 69.99	D+
90.00 - 92.99	A-	77.00 - 79.99	C+	63.00 - 66.99	D
87.00 - 89.99	B+	73.00 - 76.99	C	60.00 - 62.99	D-
83.00 - 86.99	B	70.00 - 72.99	C-	<= 59.99	F

Course Policies:

- **Attendance**
  - You are expected to attend class. I do not take attendance *per se*, but in-class exercises will not be announced ahead of time.
- **Grades**
  - Grades in the **C** range represent performance that **meets expectations**; Grades in the **B** range represent performance that is **substantially better** than the expectations; Grades in the **A** range represent work that is **excellent**.
  - Grades will be maintained on Gradescope (numerically). Students are responsible for tracking their progress by referring to the online gradebook.
  - **Class participation** is very specific: there will be frequent in-class exercises, done in groups of 4-5 students. Participation in these exercises is expected. If you participate in at least **75%** of these exercises, you will receive 100% of the Class Participation grade. Otherwise you will receive **zero** for Class Participation.
- **Assignments**
  - Programming assignments should be done in **pairs**.
  - Written assignments (e.g. proofs and problem sets) must be done **individually**.
  - If you are ever unclear about whether an assignment should be done by yourself or with a partner, please ask!
  - All assignments will be submitted and graded through Gradescope. More information on Gradescope will be included with Assignment 0.
- **Exams**
  - There is a non-traditional one-on-one final exam. You will be given a list of questions in advance, and expected to prepare answers for all of them. You get to pick one question to answer, and I get to pick one question for you to answer; you can reject my choice once (you have to answer my second choice). This will be conducted over Zoom, and will take approximately 10 minutes.

Students with Disabilities

Any student with a documented disability is welcome to contact me as early in the semester as possible so that we may arrange reasonable accommodations. As part of this process, please be in touch with Disability Services for Students Office at 302 Memorial Union, Phone 401-874-2098.

Academic Honesty Policy:

All submitted work must be your own. If you consult other sources (class readings, articles or books from the library, articles available through internet databases, or websites) these MUST be properly documented, or you will be charged with plagiarism and will receive an F for the paper. In some cases, this may result in a failure of the course as well. In addition, the charge of academic dishonesty will go on your record in the Office of Student Life. If you have any doubt about what constitutes plagiarism, visit the following websites: the URI Student Handbook, and Sections 8.27.10 – 8.27.21 of the University Manual (web.uri.edu/manual/).

Some programming assignments will be done in pairs. For the purposes of pair programming assignments, "your own" means that the work is the product of you and your partner, together. Which assignments are pair assignments will be clearly spelled out in each assignment handout.

Attendance

Students are expected to attend class and classroom activities. Occasionally, students may miss class activities due to illness, severe weather, or sanctioned University events. If ill, students should not attend class and should seek medical attention especially if they have a communicable disease such as influenza (flu). Students should not attend class when the University announces classes are cancelled due to severe weather. Also, it is the policy of the University of Rhode Island to accord students, on an individual basis, the opportunity to observe their traditional religious holidays. Students desiring to observe a holiday of special importance must inform each instructor and discuss options for missed classes or examinations. See Sections 8.51.11 – 8.51.14 of the University Manual for policy regarding make-up of missed class or examinations.

I do not specifically take attendance. However, 10% of your grade is determined by class participation, which includes participating in group exercises. If you do not participate in at least 75% of group exercises, you will receive a zero for class participation.

Pair Programming:

Professional engineering and computer science are collaborative endeavors. And yet, while you are a student, you earn an individual grade. To balance these competing concerns, we support

- **Wide but shallow collaboration when discussing ideas and problems**
- **Deep but narrow collaboration when creating and debugging computer programs**

Wide but shallow collaboration means that while you are striving to understand a problem and discover possible paths to its solution, you are encouraged to discuss the problem and your ideas with friends and colleagues—you will do much better in the course, and at URI, if you find people with whom you regularly discuss problems. When the time comes to write code, however, group discussions are no longer appropriate.

Deep but narrow collaboration means pair programming. In pair programming, you work with a partner under the following constraints:

- **When work is being done on the program, both partners are present at the computer. One partner holds the keyboard; the other watches the screen. Both partners talk, and the keyboard should change hands occasionally.**

- **You submit a single program (or design) under both your names. That work gets one grade, which you both receive.**

The following policies are essential:

- ***Every source of assistance must be acknowledged in writing.*** **This rule applies to discussions with classmates or course staff as well as assistance you might find in the library or on the web. There is never a penalty for seeking help with a problem, but help must be acknowledged.**

- **If circumstances, such as scheduling difficulties, make it impossible for you to work as part of a pair, you may ask the course staff for permission to divide an assignment into parts and to do some parts as a member of a pair and other parts as an individual. Such parts must appear in different files, and** ***each file must be clearly identified as the work of an individual or the work of a pair.*** **Work done jointly by the pair should be submitted by both members of the pair.** ***Files containing joint work must be identical.*** **If you as an individual modify a file containing joint work, and you submit the modified file, that act will be considered a violation of academic integrity.**

- **It is** ***never acceptable*** **to divide an assignment into parts and have some parts done by one partner and other parts done by the other. Submitting work done by someone else as your own will be considered an** ***egregious violation of academic integrity.*** **Submitting individual work as the product of pair programming is also a violation of academic integrity.**

- **If your partner disappears in mid-project, the correct procedure is submit the work done in partnership at that point, even if it is incomplete or broken. You may then follow up with an additional submission of whatever you complete on your own.**

- **Unless you are working with another student as part of a programming pair, it is** ***not acceptable*** **to permit that student to see any part of your program, and it is** ***not acceptable*** **to permit yourself to see any part of that other student's program. In particular, you may not test or debug another student's code, nor may you have another student test or debug your code. (If you can't get code to work, consult a TA or the instructor.) Using another's code in any form or writing code for use by another will be considered a violation of academic integrity.**

- ***Never, ever*** **share your Linux account password (or private key) with another individual (whether student, faculty, or staff). Sharing your password with another student will be considered a violation of academic integrity. If you wish to transfer code between accounts, sending a git bundle via scp or email is one good approach. You are encouraged to submit** ***general programming questions*** **to online forums such as Stackoverflow. Questions about particular homework problems must** ***never*** **be posted online—send mail to the course staff.**

Finally, be aware that *pair programming is a privilege, not a right.* If you foul up and don't fix it, I may revoke your pair-programming privileges. Fouling up consists of any of the following unacceptable behavior:

- **Repeatedly failing to keep appointments with your partner**
- **Lying to your partner about what you have done**
- **Violating academic integrity**
- **Other similarly egregious offenses**

If I find it necessary to revoke your pair-programming privileges and you believe I have done so unfairly, you may appeal to the department chair.

Tentative Course Outline:

The weekly coverage might change as it depends on the progress of the class. However, you must keep up with the reading assignments. The reading assignment listed for a given class period should be completed by that class period (hence, no reading due for the first lecture).

Date	Content
Jan 23	• Administrivia; syllabus; models of computation • **Assignment 0 out**
Jan 25	• Assignment 0 due • **Assignment 1 (perfect marriage) out** • Invariants, Bipartite Matching • Reading assignment: DPV Chapter 1
Jan 30	• Sorting 1 • Big O notation, Analysis
Feb 1	• A1 due • Sorting 2, Divide and Conquer 1 • Reading: DPV Ch 2
Feb 6	• Divide-and-conquer 2: median-finding, convex hull • Assignment 1 due • **Assignment 2 (Convex Hull) out**
Feb 8	• Divide-and-conquer 3 • DFT, FFT, polys
Feb 13	• Divide and conquer 4 • van Emde Boas trees
Feb 15	• Graphs 1 • Reading: DPV Ch. 3 • Assignment 2 due • **Assignment 3 (Rubik's Cube) out**
Feb 20	• Graphs 2 • Rubik's cube solving • Reading: DPV Ch. 4
Feb 22	• Graphs 3 • Dijkstra •
Feb 27	• Graphs 4: Bellman-Ford • Assignment 3 due
Feb 29	• Greedy Algorithms 1: Exact Change • **Assignment 4 (Compression) out** • Reading: DPV Ch. 5
Mar 5	• Greedy Algorithms 2: Huffman Coding
Mar 7	• Assignment 4 due • Greedy Algorithms 3: Minimum Spanning Trees
Mar 12	• Spring Break!
Mar 14	• Spring Break!
Mar 19	• Dynamic Programming 1 • Fibonacci

Week	Content
Mar 21	• Dynamic Programming 2: Knapsack, Shortest Paths • Reading: DPV Ch. 6 • **Assignment 5 (Seam Carving) out**
Mar 26	• Dynamic Programming 3: Sequence Alignment
Mar 28	• Network Flow 1
Apr 2	• Network Flow 2 • Assignment 5 due • **Assignment 6 (Network Flow) out**
Apr 4	• Network Flow 3
Apr 11	• Complexity 1 • Halting Problem, Undecidability, Computability, Polynomial Time Hierarchy • Reading assignment: Scooping the Loop Snooper • Reading assignment: DPV Ch. 8 • Assignment 6 due
Apr 16	• Complexity 2 • Hardness, Completeness, Reductions, Convex Hull • Reading assignment: DPV Ch. 9 • Assignment 7 out
Apr 18	• Complexity 3 • 3-D matching, compiler optimization
Apr 23	• Complexity 4
Apr 25	• Randomized Algorithms • Quantum Computing • Last day of class • Assignment 7 due