

CSE 3500 – Algorithms and Complexity Syllabus

Algorithmic thinking is a powerful way to view the field of computer science and algorithms themselves are at the core of scientific endeavours across many disciplines. For example, careful algorithmic design and implementation is what enabled researchers at Celera Genomics to assemble tens of millions of DNA reads into the sequence of the human genome [1]. This course will cover algorithmic design and analysis, advanced data structures, and computational complexity. Specific topics include divide-and-conquer, dynamic programming, greedy, and graphical algorithmic design techniques; seminal algorithms, e.g., sorting, hashing, network flow; advanced data structures, e.g., search trees, heaps; NP-completeness and intractability; and, time permitting, more advanced topics.

Prerequisites include CSE 2050 or 2100, and 2500. CSE 3500 is only open to students in the School of Engineering, Cognitive Science majors, and declared Computer Science minors.

Logistics

The **course instructor** is Prof. [Derek Aguiar](mailto:derek.aguiar@uconn.edu), derek.aguiar@uconn.edu; office is ITE 267. Samson Weiner is the **Graduate TA** mostly in charge of office hours and Sara Anwer is the **Undergraduate TA** mostly in charge of grading.

Class times are Tuesday and Thursday at 3:30PM-4:45PM in [CAST 212](#).

Professor office hours are Tuesday after class and Wed. 5:30pm-7pm. **Samson's office hours** are Tuesday 12:00pm-2:00pm, Wednesday 3:00pm-5:00pm, Thursday 11:00am-12:00pm in ITE 114. See the [CSE 3500 Information Spreadsheet](#) for the latest information.

The **course website** is hosted on [HuskyCT](#) which is used for course material distribution, and announcements.

Changes to the course syllabus, schedule, or homeworks will be announced immediately in HuskyCT and in class.

Course Objectives

The course is designed as an end-to-end treatment of algorithm design and analysis. After completion, you will be able to:

1. formalize a given problem and recognize the algorithmic challenges.
2. design algorithms to solve problems.
3. analyze the theoretical properties of algorithms.
4. demonstrate proficiency in algorithmic implementation.
5. evaluate and present results from your implementation.

Course requirements

- Programming Assignments/Problem Sets – 50%
- 2 Midterm Exams – 25%
- Final Exam – 25%

Grades are administered in HuskyCT. Students are responsible for tracking their progress through the HuskyCT grade center.

Assignments

Homework assignments will consist of problem sets and programming. Programming portions may be completed and submitted individually or in pairs; if done in pairs: (a) each programming assignment must be done with a new partner (you cannot repeat partners); (b) one individual will hand in the assignment with both names and you will receive a joint grade; (c) you will discuss and implement all parts of the assignment collaboratively (no dividing up the assignment); (d) you will abide by the collaboration policy (linked to on HuskyCT and also part of Homework 0). Please work together with integrity. The instructors reserve the right to remove the privilege of working in pairs if violations of this policy occur.

We strongly encourage high-level discussion of ideas and problems with peers (e.g. past or present partners), but problem sets portions of homework must be completed and submitted individually. This means that each student will hand in their own problem set solution. If there is ambiguity about what constitutes “high-level discussion”, the category of the assignment, or anything else regarding assignments, please ask an instructor.

Additionally,

- everyone can have bad days, so, the lowest homework assignment will be dropped. You will still be responsible for the material on the exams.
- everyone will be allotted 3 no-questions-asked late days to be used at any point unless explicitly prohibited. These are calendar days, not business days (Saturdays and Sundays count towards the 3 days). You do not need to inform us when you are taking the late days – we will automatically apply them. Inform the instructor if observance of religious holidays, illness, disability, or other valid reasons interferes with your work so we can try our best to accommodate you.
- homeworks should be handed into HuskyCT. We *strongly* encourage L^AT_EX for typesetting work. A L^AT_EX template for assignments is included in HuskyCT. www.overleaf.com is a great online L^AT_EX editor that can be used for problem sets or programming assignment collaboration. See the [online introduction](#) to L^AT_EX to get started.
- if you are having difficulty with writing, the [writing center](#) can be a great resource.

Since we need to release homework solutions, homework grades will be reduced by 20% per late day until the solutions are released (and then homework submissions are no longer accepted).

Grade Scale

Conversion from numerical grade to letter grade follows the scale below.

Grade	Letter Grade	GPA
93-100	A	4
90-92	A-	3.7
87-89	B+	3.3
83-86	B	3
80-82	B-	2.7
77-79	C+	2.3
73-76	C	2
70-72	C-	1.7
67-69	D+	1.3
63-66	D	1
60-62	D-	0.7
<60	F	0

Textbooks

- Algorithms, Dasgupta and Papadimitriou, 2006. **ISBN-13:** 978-0073523408, (\$36). There is no *required* textbook for the course, although much of the course will follow this book. The cheapest of the recommended books with succinct coverage of a majority of the material covered in class [2]. See the [CSE 3500 Information Spreadsheet Schedule](#) for more details.
- Algorithm Design, Kleinberg and Tardos, 2005. **ISBN-13:** 978-0321295354, (\$154). A treatment of algorithms that is more design focused than CLRS [3].
- Introduction to Algorithms, CLRS, 2009. **ISBN-13:** 978-0262033848, (\$65). Introduction to Algorithms by Cormen, Leiserson, Rivest, and Stein (CLRS) [4] is a good reference book. The Homer Babbidge Library has three copies of CLRS available (2 for short 3-hour loan periods).

Schedule

A tentative ordering of topics covered is included below. A current schedule with readings is available in [HuskyCT](#).

1. Preliminaries: asymptotic notations, solving summations, recurrences, invariants, proof techniques.
2. Dynamic programming algorithms: edit distance, seam-carving, Bellman-Ford, Floyd-Warshall
3. Divide-and-conquer algorithms: multiplication, binary search, merge-sort, proof of Master theorem, the fast-Fourier transform
4. Data structures: priority queues, search trees, suffix trees, suffix arrays

5. Probabilistic algorithms: hashing, selection
6. Competitive analysis: ski problem, multiplicative weights update algorithm
7. Greedy algorithms: minimum spanning trees, clustering, Huffman encoding, Lempel-Ziv, JPEG compression, Burrows-Wheeler transform
8. Graph algorithms: search algorithms, topological sorting, matching, maximum flow/minimum cut
9. Intractable problems: NP-completeness, reductions, satisfiability, graph theoretic problems
10. Advanced topics: parallel algorithms, approximation algorithms, optimization algorithms

Policies

Academic Integrity

Formally, we follow the university policy on academic integrity to discourage and penalize academic misconduct.

Academic misconduct is dishonest or unethical academic behavior that includes, but is not limited to, misrepresenting mastery in an academic area (e.g., cheating), failing to properly credit information, research, or ideas to their rightful originators or representing such information, research, or ideas as your own (e.g., plagiarism). – UConn Community Standards, Academic Misconduct

Students found in violation of academic integrity may be subject to failing the assignment, the course, and/or review by the academic integrity hearing board. See the [student code](#) and references therein for more information.

Informally, all academic work you submit must be your own or in collaboration with explicitly specified peers (where permitted). Discuss classwork, exercises, and problems with peers in a manner that helps all parties understand a problem or possible paths towards its solution. Do not copy solutions from other students or from any other resource. You will put your future self in the best position to succeed in academia or industry if you dedicate yourself to learning and retaining the material.

Disabilities

The University offers many services to its students with disabilities through the [Center for Students with Disabilities](#). Eligibility for these services is determined individually based on documented need. If you have a diagnosed disability (physical, learning, or psychological) that will make it difficult for you to carry out the course work as outlined, or that requires accommodations such as recruiting note-takers, readers, or extended time on exams or assignments, please advise the instructor during the first two weeks of the course so that we may review possible arrangements for reasonable accommodations.

References

- [1] H. Chial, "Dna sequencing technologies key to the human genome project," *Nature Education*, vol. 1, no. 1, p. 219, 2008. [Online]. Available: <https://www.nature.com/scitable/topicpage/dna-sequencing-technologies-key-to-the-human-828>.
- [2] C. Dasgupta and U. V. Vazirani, *Algorithms*. 2006.
- [3] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson Education India, 2005.
- [4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.