ARIZONA STATE UNIVERSITY

CSE 310 SLN 70793 — **Data Structures and Algorithms** — Fall 2016

Session C: 08/18/2016-12/02/2016

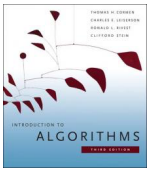
Instructor:	Dr. Violet R. Syrotiuk (e-mail: `syrotiuk@asu.edu`; office: BYENG 434)
Lectures:	MWF 10:45-11:35am, in COOR L1-20
Office Hours:	Mondays 8:30-9:30am, Tuesdays 9:00-10:00am, and by appointment only
TA:	TBA on Blackboard
Recitation Leader:	TBA on Blackboard

Course Description

CSE 310 studies how data structures impact the computational complexity (time and space) of algorithms. It is assumed that you are familiar with the data structures of stacks, queues, linked lists, and binary search trees (BSTs). This course covers the data structures of heaps, hash tables, balanced variants of BSTs (for example, red-black, 2-3, or splay trees), and graphs (undirected and directed, weighted and unweighted). Algorithmic techniques covered include divide-and-conquer, dynamic programming, and greedy. While most algorithms are analyzed in the worst or average case, also of interest are lower bounds, and amortized and probabilistic analyses.

Prerequisites: For CS/CSE students, grades of at least C in CSE 220 or CSE 240 and at least D in MAT 243. For CMS students, grades of at least C in CSE 210 and at least D in MAT 243 or MAT 300. Programming experience in `C` or `C++` is expected. Experience using Linux is helpful.

Required Textbook



Introduction to Algorithms, Third Edition, by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein. © 2009 by Massachusetts Institute of Technology. ISBN 978-0-262-03384-8.

Evaluation Procedure

Assignments	10%	4 of equal weight
Projects	30%	3 of equal weight, in `C/C++` required to run on `general.asu.edu` (a Linux machine)
Midterm Exams:	30%	3 of equal weight, in class on F 09/16, F 10/14, and W 11/09 (closed book)
Recitations:	5%	based on a weekly quiz in your recitation session
Final Exam:	25%	on W 12/07, 9:50-11:40am (closed book, comprehensive)
	100%	

Assignments	**Out**	**Due**	**Projects**	**Out**	**Milestone**	**Due**	**Midterms**	**Scheduled**
A1	M 08/22	F 09/09	P1	M 08/22	W 09/07	M 09/26	M1	F 09/16
A2	M 09/19	M 10/03	P2	M 09/26	W 10/12	M 10/31	M2	F 10/14
A3	M 10/17	W 11/02	P3	M 10/31	M 11/14	M 11/28	M3	W 11/09
A4	M 11/14	W 11/30						

Course Schedule

Foundations: (approximately 2 weeks)

- The Role of Algorithms in Computing (Chapter 1): What is an algorithm?
- Getting Started (Chapter 2): Insertion sort. Analyzing and designing algorithms. Mergesort.
- Growth of Functions (Chapter 3): Asymptotic notation. Standard notation and common functions.
- Divide-and Conquer (Chapter 4): Selected problems solved using divide-and-conquer. The substitution, recursion tree, and master method for solving recurrences.

Sorting and Order Statistics: (approximately 2 weeks)

- Heapsort (Chapter 6): Heaps and maintaining the heap property. The heapsort algorithm.
- Quicksort (Chapter 7): The quicksort algorithm and its analysis. Randomized quicksort.
- Medians and Order Statistics (Chapter 9): Minimum and maximum. The selection problem.

Data Structures: (approximately 2 weeks)

- Review of Elementary Data Structures (Chapter 10): Stacks and queues. Pointers and objects. Implementing lists and rooted trees.
- Hash Tables (Chapter 11): Direct-address tables. Hash tables and functions. Open addressing.
- Review of Binary Search Trees (Chapter 12): What is a binary search tree? Queries, insertion, and deletion.
- Red-Black Trees (Chapter 13): Properties of red-black trees. Rotation, insertion, and deletion.

Advanced Design and Analysis Techniques: (approximately 2 weeks)

- Dynamic Programming (Chapter 15): Selected problems solved using dynamic programming.
- Greedy Algorithms (Chapter 16): Selected problems solved using the greedy approach.
- Amortized Analysis (Chapter 17 and 21): Disjoint-set operations and their amortized analysis.

Graph Algorithms: (approximately 3 weeks)

- Elementary Graph Algorithms (Chapter 22): Representations of graphs. BFS and DFS.
- Minimum Spanning Trees (Chapter 23): The algorithms of Kruskal and Prim.
- Single-Source Shortest Paths (Chapter 24): The Bellman-Ford algorithm and Dijkstra's algorithm.

Use of Blackboard

Blackboard will be used in CSE 310. It is accessible from a browser at `http://my.asu.edu`. It contains the syllabus among other material. All homework assignments and programming projects will be posted on **Blackboard**. Hard copies of solutions to ssignments are due at the start of class (10:45am) on their due date. Solutions to projects must be submitted electronically on **Blackboard**. Use the discussion group to ask questions about anything related to CSE 310; you are expected to check it often and to participate.

Recitation Sessions

Starting Fall 2016, students in CSE 310 must also enroll in one recitation session (there is one available on each week day). Recitations provide another opportunity for more problem solving and clarification of lecture materials in a smaller setting. Each recitation will include a short quiz. Quizzes will be graded on an integer scale out of 5: zero (nothing done), 1 (incorrect), 2 (mostly incorrect), 3 (average), 4 (mostly correct), 5 (correct). The average of your top 10 scores in recitations will form your grade for recitations.

Class Policies

Electronic and hard copies of ssignments are due at the start of class (10:45am) on their due date. Projects are due electronically at *midnight* on their due date. Late assignments and projects will **not** be accepted, except on a documented emergency basis. For consideration of an extension you must contact me *prior* to the due date. Midterms will not be rescheduled, except on a documented emergency basis. The final exam date is scheduled by ASU and cannot be changed.

Whenever a new grade for work is available it will be posted on **Blackboard**. If you wish to appeal the grade, you must do so *in writing* within one week of the grade availability. The request together with the work itself must be submitted to the TA for assignments and projects, or the instructor for exams. The appeal must clearly indicate the work to review and provide a basis for the appeal. No appeal will be considered for an assignment if its electronic copy was not submitted. Your right to appeal is waived one week after the grade is posted.

It is imperative that you make a legitimate attempt to do all assigned work. Any scaling of grades at the end of the course takes into account the effort invested.

ASU Code of Conduct and Academic Integrity Policy

Plagiarism or any form of cheating in assignments, projects, or exams is subject to serious academic penalty; this may range from a grade of zero for the work to failure of the course. To understand your responsibilities as a student at ASU read:

- Student Code of Conduct and Student Disciplinary Procedures: `http://students.asu.edu/srr/code`
- Student Academic Integrity Page: `http://provost.asu.edu/academicintegrity`
- See also the content under the "Collaboration: What is and is not permitted" folder under the *Syllabus & Course Information* tab on Blackboard.

CSE 310 Course Learning Outcomes

Students who complete CSE 310 will be able to:

1. Define data structures such as heaps, balanced trees, and hash tables.
2. Explain how to use a specific data structure in modelling a given problem.
3. Identify, construct, and clearly define a data structure that is useful for modelling a given problem.
4. State some fundamental algorithms such as merge sort, topological sort, Prim's and Kruskal's algorithm, and algorithmic techniques such as dynamic programming and greedy algorithms.
5. Use a specific algorithmic technique in solving a given problem.
6. Design an algorithm to solve a given problem.
7. Define the notions of worst-, best-, and average-case running times of algorithms.
8. Analyze and compare different asymptotic running times of algorithms.
9. Analyze a given algorithm and determine its asymptotic running time.
10. Combine fundamental data structures and algorithmic techniques in building a complete algorithmic solution to a given problem.
11. Create several algorithmic solutions to a given problem and choose the best one among them according to given requirements on time and space complexity.