COMP 554: Analysis of Algorithms
Fall 2022

THIS IS A DRAFT SYLLABUS WHICH MAY BE UPDATED THROUGHOUT THE COURSE
Last updated: August 18, 2022

Instructor

Michael Soltys
michael.soltys@csuci.edu
http://www.msoltys.com
Sierra Hall 3327
Office hours: Mondays 4:00-6:00

Course Information

COMP 554
Lecture time & place: Mondays 7:15-10:15, location: del Norte 1535
Course web page: http://prof.msoltys.com/?page_id=6532

Course Description

This course is an advanced review of the analysis of algorithms. The material covered consists of five parts: (i) basics of proving correctness of algorithms using pre/post-conditions, loop invariants and termination. (ii) Three classical algorithm design techniques: greedy, divide-and-conquer and dynamic programming (most of the course is dedicated to this part). (iii) Analysis of performance using worst-case complexity. (iv) Implementation issues (we will implement algorithms in Python 3). (v) We will also cover NP-hardness, and four more advanced types of algorithms: approximation, online, randomized and parallel.

Prerequisite: We will cover the necessary background, but some *Discrete Math* and some *Programming Experience* will be helpful.

Student Learning Outcomes (SLOs)

Upon a successful completion of the course you will be able to:

1. Design algorithms to solve problems according to standard design principles (greedy, divide-and-conquer and dynamic programming);
2. Measure the performance of an algorithm in terms of worst-case complexity and Big-Oh notation, and indicate trade-offs (speed versus memory usage, etc.);
3. Prove the correctness of a given algorithm, i.e., that is solves correctly a given problem.

Course Outline

Topics in order:

1. Correctness: pre/post-conditions, examples of division and Euclid's algorithm. [2 weeks]
2. Ranking algorithms: PageRank, Stable Marriage, Pairwise Comparisons. [2 week]
3. Greedy algorithms: Minimum-cost spanning trees, especially the idea of a promising solution; job scheduling with profits; other examples. [4 weeks]
4. Divide and conquer algorithms: Mergesort, multiplication of binary numbers, Savitch's algorithm for reachability in little space (Savitch's algorithm is an example of performance where little space requires a long time); other examples. [2 weeks]
5. Dynamic Programming Algorithms: Longest monotone subsequence problem, All pairs shortest path problem, variants of the Knapsack problem and introduction to NP-completeness and approximation algorithms. Illustration of the "overwriting" technique in implementation. Activity selection with profits. [4 weeks]
6. Online, randomized and parallel algorithms. [2 weeks]

Textbook

3rd edition of *An Introduction to the Analysis of Algorithms*, by Michael Soltys, published by World Scientific (ISBN: 978-981-3235-90-8). See web page: http://www.msoltys.com/book, for additional material related to the textbook: slides, GitHub repository with solutions to programming problems and an errata sheet for the 3rd edition.

Grading

Five assignments worth 20% each, to be completed in teams of three students. There are three expectations regarding the assignment:

1. First, it is part of solution development to have a back and forth between the instructor and the students, in order to understand fully the requirements and specifications. Thus, you should ask in class if anything about the assignment is not clear, as usually there are many implicit assumptions that must be made explicit.
2. Second, a programming problem solution will consist of Python code, well documented with comments, as well as a PDF submission explaining your solution and providing background. The PDF should contain at the top the names of the group members, and the assignment number and/or title. Please submit the two files separately (do not zip), so that they can be viewed directly in Canvas.

3. Third, working successfully in a group is part of the assignment; team members bring different talents, abilities and styles. It is normal to have frictions; learn to communicate with each other in order to resolve those frictions. This aspect of the assignment will not be graded directly, but it will be graded indirectly, in that if your team works well together, the final product will be better.

Grade determination

From	To	Letter Grade	From	To	Letter Grade
97	100	A +	77	79.99	C+
94	96.99	A	74	76.99	C
90	93.99	A-	70	73.99	C-
87	89.99	B+	67	69.99	D+
84	86.99	B	64	66.99	D
80	83.99	B-	60	63.99	D-
			0	59.99	F

Policies

1. **Academic Dishonesty:** By enrolling at CSU Channel Islands, students are responsible for upholding the University's policies and the Student Conduct Code. Academic integrity and scholarship are values of the institution that ensure respect for the academic reputation of the University, students, faculty, and staff. Cheating, plagiarism, unauthorized collaboration with another student, knowingly furnishing false information to the University, buying, selling or stealing any material for an examination, or substituting for another person may be considered violations of the Student Conduct Code (located at http://www.csuci.edu/campuslife/student-conduct/academic-dishonesty.htm). If a student is found responsible for committing an act of academic dishonesty in this course, the student may receive academic penalties including a failing grade on an assignment or in the course, and a disciplinary referral will be made and submitted to the Dean of Students office. For additional information, please see the faculty (located at https://senate.csuci.edu/policies/2013-2014/sp-13-06-policy-on-academic-dishonesty-rev-oct2016.pdf), also in the CI Catalog.

   The assignments will be written in groups. Each group has to work independently of the other groups; verbal discussions of problems among groups are allowed, but you should not show written notes, and you should not leave such discussions with written notes. Please speak to the instructor if these expectations are not clear.

2. **Disability Statement:** If you are a student with a disability requesting reasonable accommodations in this course, please visit Disability Accommodations and Support Services (DASS) located on the second floor of Arroyo Hall, or call 805-437-3331. All requests for reasonable accommodations require registration with DASS in advance of need: https://www.csuci.edu/dass/students/apply-for-services.htm. Faculty,

students and DASS will work together regarding classroom accommodations. You are encouraged to discuss approved accommodations with your faculty.

3. **Course Policies Subject to Change:** It is the student's responsibility to check the course's web page frequently to stay abreast of the course, and for corrections or updates to the syllabus. Any changes will be posted there.

Course Assessment

Computer Science Student Learning Outcome (SLO) "1." states:

Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.

Here is the rubric for this outcome:

Performance Indicator	**Unsatisfactory**	**Developing**	**Satisfactory**	**Exemplary**
1. **Algorithmic design:** ***principle of computing***	no understanding of problem, no solution	problem understood, but solution wrong	problem understood and a solution given	problem understood and best solution given
2. **Performance analysis:** ***computational complexity***	no understanding of what is requested	understanding of worst-case but no Big-O estimate	worst-case analysis and a Big-O estimate given	worst-case analysis resulting in tight Big-O estimate
3. **Proof of correctness:** ***Mathematics as other discipline that helps identify solution***	no understanding of how to approach the proof	providing general direction but no details	an outline of the proof given and aspects of framework	a complete proof, with framework of pre/post-condition and invariants

The threshold will be 80%, that is, at least 80% of students must meet the "satisfactory" or "exemplary" level. All three rows will be measured by the corresponding question on the final exam:

A Design Question: A problem is posed, and the students must choose one of the three basic algorithm design techniques to solve it, and present the solution in clear and correct pseudo-code.

A Performance Question: An algorithm is posed, and the student must evaluate its time and/or space complexity in terms of worst-case performance expressed in Big-O notation, and trade-offs, e.g., optimization versus speed, or time versus space resources.

Proof of correctness Question: The student will be given a problem, and an algorithmic solution will be requested, together with the proof of correctness of the algorithm; the student will be required to tie the algorithmic solution to the problem, and to show that the algorithm solves that problem.