CSCI 4104: Data Structures & Algorithms II
Spring 2022 Syllabus (Hybrid)

Instructor: Dr. Vincent A. Cicirello **Office:** G116
E-mail (course related): Use course Blackboard Course Messages tool. **Phone (office):** x3526
E-mail (other than this course): See campus directory for address.

Hybrid Course: The course is offered in the **Hybrid** modality. This means that the course is partially face-to-face, and partially online.

- **Online Portion of Course:** The online portion of the course is actually the "lecture" portion of the course, and consists of videos on all of the course topics. You are expected to view these in time to complete the associated assignments, and exams; and for preparation for the in person portion of the course. One of the benefits of conducting the lectures asynchronously online is that you can choose to rewatch parts as needed for review. An additional benefit is that if you are sick or otherwise have an emergency that requires your absence, you won't really miss anything of importance since you can view the lecture at the time of your choice.
- **Face-to-Face Portion of Course:** During the weekly, in-person portion of the course, we will: (a) answer the questions that you have about the course topics; (b) get additional assistance on the programming assignments; (c) review for exams; and (d) take exams.
- **IMPORTANT:** Although the course meets only 1.25 hours per week in person, **this is a 4-credit course**. In addition to time outside of class completing homework assignments, reviewing for exams, etc., you have the additional 2.75 hours per week for viewing the video lectures (see first bullet-point above). If you treat this as a 1.25-credit course and skip the video lectures, you will almost certainly fail.

Course Time and Location: Attendance during the face-to-face portion of the course is **required**, and it meets the following days/times and in the following locations:

- **Section 091:** Wednesdays, 11:20am-12:35pm, D019.

Office Hours:

- **Virtual:** Thursdays (via Zoom), 11:30am-1:30pm (Zoom link in Blackboard): You will end up in a "waiting room" when first connecting to avoid anyone accidentally entering while any students are discussing grades, etc with me. I will admit students into the actual Zoom room in the order they enter the waiting room.

Course Description: In this course, students deepen their knowledge of the design and analysis of computer algorithms. Advanced topics in algorithms and algorithm analysis covered in the course include graphs and graph algorithms, string matching, multi-threaded algorithms, and NP-completeness.

Prerequisites:

- CSCI 3103: Data Structures & Algorithms I (C or better),
- CSCI 2226: Foundations of Computer Science (C or better), and
- MATH 2215: Calculus I (C or better).

This course is a Q2 (Quantitative Reasoning Across the Disciplines): Among the math concepts from prior courses that you will use in this course are the following: (a) discrete math topics including set theory, logic, graphs, trees, and several other topics covered in MATH 2225/CSCI 2226; and (b) calculus topics including limits and derivatives (you might want to dust off your calculus textbook if you still have it).

Required Textbooks/Readings:

- Introduction to Algorithms, 3rd Edition, 2009, by T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. ISBN: 9780262033848.

Useful/Suggested Resources:

- Your calculus textbook (if you still have it).
- Your discrete math textbook (if you still have it).

Other Required Resources:

- Python 3 (current version is 3.10.1, but any version that starts with 3 is sufficient): https://www.python.org/
- **Python 3 will be used for all programming in this course.** [Note: Python 2 is still actively used but it is no longer maintained, and more importantly it has differences in syntax and different libraries than Python 3. Make sure you use Python 3. If you submit programming assignments that require Python 2, you will get a 0 for those assignments. Python 2 has been permanently frozen since April 2020.]
- Useful python related links:
    - Tutorials: https://docs.python.org/3/tutorial/
    - Language and API documentation: https://docs.python.org/3/

Course IDEA Objectives: The IDEA objectives of this course (paraphrased within course context) are:

- IDEA Objective 1: Gaining knowledge of the terminology, methods, trends of data structures and algorithms, including the fundamental principles and theories of algorithm analysis, such as asymptotic analysis, and complexity classes.
- IDEA Objective 3: Learning to apply data structures and algorithms to solving real-world problems.
- IDEA Objective 4: Developing professionally relevant skills, such as programming in the Python language, parallel programming concepts, among others.
- IDEA Objective 11: Learning to analyze and critically evaluate alternative data structures/algorithms.

Stockton Computer Science Student Learning Outcomes: This course supports the following CSCI Student Learning Outcomes:

- 1. An ability to analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.
    - a. Students will analyze a complex computing problem.
    - b. Students will apply principles of computing and other relevant disciplines to identify solutions to a complex computing problem.
- 2. An ability to design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.
    - a. Students will design a computing-based solution to meet a given set of computing requirements.
    - b. Students will implement a computing-based solution to meet a given set of computing requirements.
    - c. Students will evaluate a computing-based solution to meet a given set of computing requirements.
- 3. An ability to communicate effectively in a variety of professional contexts.
    - a. Students will write technical documentation of a computer-based system, process, component, or program.
- 6. An ability to apply computer science theory and software development fundamentals to produce computing-based solutions.
    - a. Students will apply computer science theory to produce computing-based solutions.
    - c. Students will evaluate the effects of alternative data representations and algorithms on the performance of computing-based solutions.

List of Course Topics:

- Introduction to algorithms
- Python programming (prior Python background is not necessary)
- Algorithm analysis (both asymptotic analysis and amortized analysis)
- Graphs (data structures and basic algorithms)
- Minimum spanning tree algorithms
- Single source shortest paths algorithms
- All pairs shortest paths algorithms
- Multithreaded (or Parallel) Algorithms
- Multithreaded and multiprocess programming in Python
- String matching algorithms
- NP Completeness

Course Lecture Content: All video lecture content is within Blackboard. The **"Topics (Videos and Notes)"** page is where you will find the lecture materials for the course topics. For each topic, I have recorded several videos, as well as have provided in note form any PowerPoint slides I used during the videos. You can also find Python source code used in the videos there as well. If you don't watch the videos, then you will almost certainly fail the course.

Grading:	Written Problem Sets (6)	20%
	Programming Assignments (5)	35%
	Exams (3)	45% (15% each)

Grading Scale:

A: at least 90.00	A-: at least 89.00	B+: at least 88.00
B: at least 80.00	B-: at least 79.00	C+: at least 78.00
C: at least 70.00	C-: at least 69.00	D+: at least 68.00
D: at least 60.00	D-: at least 59.00	F: less than 59.00

I reserve the right to adjust the scale at the end of the semester. Such adjustments are rare, but will only be in your favor; and are highly unlikely to occur at the D-/F boundary. Note the 2 decimal places in the chart above (i.e., I do not round to the nearest whole number): e.g., unless I adjust the grade scale, an 89.99 is an A-, etc. If I adjust the scale, it is done using a semi-automated approach involving clustering (i.e., "automated" == a program I wrote suggests a new scale based on all of the grades of the class; "semi-" == if that program's output is crazy, I ignore it and leave the scale alone; and "clustering" == a statistical technique). I never simply add a constant number of points to everyone's overall course score. What is clustering? For a non-technical explanation, consider the hypothetical question, "Are the grades of this student, who is currently in the B range, more like the grades of the A students than the rest of the B students, or somewhere in between the two?" Clustering may take a student in the B range from the scale above (at least 80.00 but less than 88.00), and either keep them in the B range if their grades are more like the rest of the B students, or bump them all the way up to the A range if their grades are more like the A students, or bump them partially up to either B+ or A- if they are somewhere in between.

Other Grade Related Info: If you are in the B.S. in Computer Science, then this course is a core required course and must be passed with at least a C to count toward your degree requirements. If you are in the old B.S. in CSIS (CS concentration), then this course is just a CS elective and must simply be passed (any passing grade).

Exams: The exams are not explicitly cumulative, although many concepts covered in the course build on earlier topics. **You are allowed to use one page (standard letter-size paper, 8.5in by 11in), double-sided, paper notes only during the exams.** You are not allowed to use any other resources during exams (no electronic devices other than the lab computer on which you are taking the exam). During the exams, the lab computers will be restricted so that the only thing you can access is Blackboard via a web browser. Other devices (including calculators) are not allowed. Textbooks, and other books, are also not allowed. You are, however, allowed two blank letter-sized paper pages for scrap paper during the exams, in case you need to work through something by hand. There will be no Python syntax or programming on the exams. However, there may be problems on the exams involving reading and/or writing algorithms in pseudocode.

Make-Up Exams: Make-up exams will not be given (i.e., missed exam = 0), with the following exceptions:

1. Medical excuse: Provide documentation, preferably to the Wellness Center who will then contact all of the instructors of your courses.
2. Based on University policy (https://stockton.edu/policy-procedure/documents/procedures/2030.pdf), if you are to be absent for a religious holiday on the date of an exam, you must notify me of that planned absence during the first 10 business days of the semester.

Programming Assignments: There will be 5 programming assignments. The time allotted to each will generally be 2 weeks, but may vary depending upon the amount of work required. You will be required to work independently. Programming assignments will involve implementing data structures and algorithms covered within the course, and either applying them to a given problem, or in some cases performing an experimental comparison of alternative algorithms for a problem. Programming assignments will be implemented in Python 3. Any assignments submitted that require Python 2.7 will receive a grade of 0. Details of all programming assignments, including deadlines are available in Blackboard from the start of the semester. You are free to work ahead if you wish.

Problem Sets: Problem sets will include problems, exercises, and review questions related to the course topics. The written homework exercises must be done individually. You must show your work for full credit. A simple answer without showing the work that lead to it will receive no credit in most cases. Problem sets will also be collected via Blackboard. Details of all problem sets, including deadlines are available in Blackboard from the start of the semester. You are free to work ahead if you wish.

- **Paper and then scan**: I strongly suggest that problems involving math are done on paper and then scanned to submit in Blackboard. It will probably take you less time to do it that way, since you need to show all of your work, then to try to format math in Word or some other word processing software. There are labs on campus that have scanners.
- **Phone PDF scanner apps**: If you don't have a scanner, and the labs on campus with scanners are occupied by classes, then there are some good (and free) apps available for phones that use your phone camera to scan a document as a pdf (including multipage pdf documents). One such app that I have personally used on Android is "Office Lens" from Microsoft. It is free and even has no ads (some of the other free apps have ads). Microsoft's "Office Lens" app works well, and does a good job of finding edges of the page, and even orienting the image well if you are not holding your phone straight.
- Yes, I am aware that there are websites containing the textbook solutions. A few graduate students at a school I won't name worked through all of the problems and posted their solutions online. I have a copy of this, so I will know if you use them. See the academic honesty policy in the syllabus. Also, approx. 33% of their solutions are either totally or partially incorrect; and 33% are incomplete (e.g., an answer without work, etc). So even if I don't notice, you won't likely get a good grade using those solutions.

Due Dates: Assignments are due in Blackboard by 11:59pm on the dates due (dates found in Blackboard on the assignment pages). Late assignments are penalized as follows: (a) 25% off if late by no more than 24 hours, (b) 50% off if late by no more than 48 hours, (c) 75% off if late by no more than 72 hours, and (d) a grade of 0 if late by more than 72 hours. The first time an assignment of each type is late (within 72

hours), the late penalty is waived—i.e., you can be late with one programming assignment (within 72 hours of deadline) AND one written problem set (within 72 hours of deadline).

Academic Honesty: Please familiarize yourself with Stockton's policy on academic honesty. Each violation is penalized by a 0 on the relevant assignment/exam/etc, plus a 10 point penalty on your overall course grade. For example, if you have one violation, you'll have a 0 on that assignment or exam plus 10 points off your overall average, but if you have two violations, you'll have grades of 0 on the two assignments/exams/etc and 20 points off your overall average. Example violations include, but are not limited to: (a) any form of cheating on an exam or assignment, (b) passing off the work of another as your own (including other students, former students, code or problem solutions found on the Internet written by someone else, etc), (c) assisting someone in violating the academic honesty policy, (d) asking someone to assist you in cheating or other academic honesty violations (even if they refuse to help you cheat), etc. [Yes, I encountered that last one once in a General Studies course.]

Incomplete Policy: In general, no grades of incomplete will be given. The only exception to this rule is an institutionally documented medical emergency that necessitates your complete absence from Stockton for at least two continuous semester weeks. Additionally, you must be caught up on all work up to the point where your medical emergency began and currently in the "C" range or better overall at the point where the emergency began.

Timeline: The following chart indicates approximately where you should be in terms of viewing the video lectures throughout the semester. You should use this as a plan for when to complete viewing of the video lectures. The 3rd column also lists when assignments are due (11:59pm on the deadlines).

Date	**Topic (Use as Guide to Plan Video Lecture Viewing)**	**Assignments Due**
January 19	Course overview / Introduction to algorithms	
January 26	Algorithm analysis	
February 2	Algorithm analysis	Problem Set 1
February 9	**Exam 1:** Intro to Algorithms and Algorithm Analysis	Program 1 (Feb 11)
February 16	Graphs, and graph data structures & algorithms	
February 23	Minimum Spanning Trees	Problem Set 2
March 2	Shortest path algorithms (single-source and all-pairs)	Problem Set 3
March 9	**Exam 2:** Graph data structures and algorithms, MST, shortest path problems.	Program 2 (March 11)
March 16	**NO CLASS: Spring Break**	
March 23	Multithreaded algorithms	Program 3 (March 28)
March 30	String matching algorithms	Problem Set 4
April 6	**NO CLASS:** Advising Day	Problem Set 5
April 13	NP Completeness	Program 4 (April 15)
April 20	NP Completeness	Problem Set 6
April 27	**Exam 3 (last in-person meeting):** Multithreaded algorithms, String Matching Algorithms, NP-Completeness	
	No course meeting during finals week, but last programming assignment is due on May 3 (must be on time, no late waivers, late=0 for this one).	Program 5 (May 3)