# CSCI 4104: Data Structures & Algorithms II
# Fall 2024 Syllabus

**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.

**Course Time and Location:** Attendance is **required**. Class meets as follows:
- **Section 001:** Tuesdays, 10:30am-12:20pm, F210
- **Section 002:** Tuesdays, 2:30pm-4:20pm, D019

**Modality:** This class is a hybrid, which meets face-to-face half-time, and asynchronously the other half of the time. Online video lectures must be watched on time for the covered topic. If you skip the video lectures, it is like skipping 1/2 of a course, and you will almost certainly fail in that case.

**Office Hours:**
- Tuesdays 12:30pm – 2:00pm (G116)
- Other days/times (both virtual and on campus) by appointment.
- You can also just drop by my office, and if I'm there I'm happy to assist you.

**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, 4th Edition, 2022, by T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. ISBN: 9780262046305.

**Useful/Suggested Resources:** Your calculus and discrete math textbooks (if you still have them).

**Course Lecture Content:** Course lectures are in the form of videos posted at the start of the semester within Blackboard. When we meet in the classroom, we'll use face-to-face class meetings to work through problems, review course concepts, consider further examples of Python programming concepts, review for exams, and take exams. 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.

**Other Required Resources:**

- Python 3 (current version is 3.12.5, 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/
- IDLE: IDLE is the standard development environment for Python, and it is part of the installation of the official distribution of Python. I will use IDLE during class examples, and Python 3 (including IDLE) is installed in all campus labs. In reality, it doesn't matter what development environment you use. So if you are already familiar with PyCharm or another Python development environment that you prefer, or even a more general programming-aware text editor like VSCode, feel free to use it when working on homework assignments. It really doesn't matter. You will be submitting the Python source code that you write (.py files) and how you wrote them doesn't matter as long as they work. Be aware, however, that the last programming assignment involves Python's multiprocessing module, and students in prior semesters discovered that PyCharm is incompatible with that Python module. So if you decide to use PyCharm, for the last assignment you will either need to switch to IDLE, or if you really want to use PyCharm, you'll need to run your code from the command line and not from within PyCharm.

| **Grading:** | Written Problem Sets (6) | 20% |
|---|---|---|
| | Programming Assignments (4) | 35% |
| | Exams (3) | 45% (15% each) |

**Other Grade Related Info:** This course is a core required course and must be passed with at least a C to count toward the B.S. in Computer Science.

**Grading Scale:**

| | | |
|---|---|---|
| A: at least 90.00 | A-: at least 89.50 | B+: at least 89.00 |
| B: at least 80.00 | B-: at least 79.50 | C+: at least 79.00 |
| C: at least 70.00 | D: at least 60.00 | F: less than 60.00 |

**Note:** The chart above deliberately does not include C-, D+, or D-. Those grades are not used in this course.
**Another note:** 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 C/D or D/F boundaries. 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, more like the grades of 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), 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.

**Incomplete Policy:** In general, no grades of incomplete will be given. The only exception 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 when your medical emergency began and currently in the "C" range or better overall when the emergency began.

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

- 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.
- Learning to apply data structures and algorithms to solving real-world problems.
- Developing professionally relevant skills, such as programming in the Python language, parallel programming concepts, among others.
- 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.

**Generative A.I. Prohibited:** Generative artificial intelligence (AI) programs, such as ChatGPT, GitHub Copilot, and other similar systems, may **not** be used for any work or assignments required in this course. Although you will undoubtedly need to use such tools in your future careers, it is important that you develop your programming skills and problem solving skills so that you can later effectively use such tools to enhance your productivity. Copilot, ChatGPT, etc often produce solutions that are buggy, only partially correct, among other issues that one must address with their own programming skills. The use of generative AI programs defeats the programming requirements and critical thinking skills that are vital to achieving our learning outcomes. Submission of partial or complete work from generative AI programs is not permitted and will be treated as an Academic Honesty violation, and handled in accordance with the course Academic Honesty procedure (see below) as well as Stockton's Student Academic Honesty Procedure and handled in accordance with these Procedures.

**Academic Honesty:** Please familiarize yourself with Stockton's Student Academic Honesty Procedure. 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, code generated by generative A.I. such as but not limited to ChatGPT, GitHub Copilot, 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.]

**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. I will be monitoring you from the podium computer, watching to make sure that the only page within Blackboard that you are on is the exam itself. 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 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.

**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).

**Programming Assignments:** There are 4 programming assignments. The time allotted to each varies depending upon the amount of work required. You are required to work independently. Programming assignments 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. You are only allowed to use the standard Python 3 library when working on programming assignments. You are not allowed to use any 3rd party libraries. There are many high-quality, very useful, 3rd party libraries available for Python, but many of these include things that I want you to do, which is why they are all off-limits. If running "pip install" is needed to use it, then it is not allowed for programming assignments. When I grade programming assignments, I do so in a "sandbox", on a Raspberry Pi with no access to the rest of my local network, and with only the base installation of the latest Python, without any 3rd party libraries, and without pip. So that 3rd party library that you want to use is not installed on my system, and I've disabled my ability to install it (pip is the utility used to install 3rd party Python libraries). **An assignment that attempts to use a 3rd party library will fail to run on my system, and may or may not receive any points, depending upon how extensively you relied upon the 3rd party library.**

**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 (at least for previous editions of the book). 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, from what I have seen of those "solutions", approximately 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.

**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** |
|---|---|---|
| September 10 | Course overview / Introduction to algorithms | |
| September 17 | Algorithm analysis | |
| September 24 | Algorithm analysis / Review for Exam | Problem Set 1 |
| October 1 | **Exam 1:** Intro to Algorithms and Algorithm Analysis | Program 1 (Oct 3) |
| October 8 | Graphs, and graph data structures & algorithms | |
| October 15 | Minimum Spanning Trees | Problem Set 2 |
| October 22 | Shortest path algorithms (single-source and all-pairs) | Problem Set 3 |
| October 29 | **NO CLASS: Advising Day** | Program 2 (Oct 31) |
| November 5 | Slack in schedule / Review for Exam | |
| November 12 | **Exam 2:** Graph data structures and algorithms, MST, shortest path problems | |
| November 19 | Multithreaded algorithms | Problem Set 4 (Nov 19); Program 3 (Nov 21) |
| November 26 | String matching algorithms | Problem Set 5 |
| December 3 | NP Completeness / Review for Exam | Problem Set 6 |
| December 10 | **Exam 3:** Multithreaded algorithms, String Matching Algorithms, NP-Completeness | |
| December 17 | Last programming assignment due 11:59pm (must be on time, no late waivers, late == 0) | Program 4 |