CSCI 345

Data Structures and Algorithms

Fall 2025   MFW 11:35–12:45pm   MEY 129
Th 11:15am–1:05pm

`http://cs.wheaton.edu/~tvandrun/cs345`

Thomas VanDrunen

☎630-752-5692   📞630-639-2255   ✉Thomas.VanDrunen@wheaton.edu

Office: MEY 163   Office hours: Drop-in: MWF 3:30–4:30pm; Or by appointment through Calendly

---

Contents

CATALOG DESCRIPTION. Formal and experimental approaches to verifying algorithms' correctness and analyzing their efficiency. Abstract data types and their implementations. Efficient implementations of maps using balanced binary search trees and hash tables. Graph algorithms. Dynamic programming. Prerequisites: CSCI 243 and CSCI 245.

TEXTBOOK.
Thomas VanDrunen. *Algorithmic Commonplaces*. Franklin, Beedle, 2023.

LAB SECTION.
This course has a lab section CSCI 345L that meets 11:15am–1:05pm Thursday in CSCI 154.

PURPOSE OF THE COURSE. This course is a central part of the computer science major. This is the place where students solidify their understanding of algorithmic problem-solving, data abstraction, and the trade-offs of data structuring strategies. Moreover, students add knowledge of crucial algorithmic techniques and data structures to their toolkit. Specifically, there are six big ideas. The first three are pursued throughout the course, the last three refer to specific units.

1. The correctness of an algorithm can be verified formally using loop invariants and other proof techniques and empirically using unit tests.

2. The efficiency of an algorithm can be measured formally using algorithmic analysis, big-oh categories, etc, and empirically by running experiments.

3. Abstract data types, especially list, stack, queue, set, bag, and map, are specified by how they are used; data structures, such as arrays, linked lists, binary trees, and hash tables, are implementation strategies, each with trade-offs.

4. Searching in an unordered data structure such as a map can be done in logarithmic time using a balanced binary search tree .

5. Searching in an unordered data structure can be done in constant time using a hash table.

6. Problems with overlapping subproblems and optimal substructure can be solved efficiently using dynamic programming.

COURSE OUTLINE. (For a schedule, see the course website.)

I. Prolegomena

> Studying data structures and algorithms requires clear expression of the basic principles. When studying algorithms, we have two dimensions. For a given algorithm, we want to determine whether it is correct or not and how efficient it is; both of these things can be studied both formally and empirically. For that end we consider formal techniques for reasoning about what an algorithm does and categorizing its running time. We also identify basic algorithms that are used as elements in more complicated algorithms, which builds our intuition for determining an algorithm's complexity.
>
> Moreover, we need a clear distinction between *abstract data types* (ADTs), which specify how a container or other composite type presents itself to the rest of the system, and *data structures*, which are strategies for implementing ADTs. We define a set of standard ("canonical") ADTs, as well as identify the basic data structures that are used as elements in more complicated data structures.

   A. Algorithms and correctness

   B. Algorithms and efficiency

   C. Abstract data types

   D. Data structures

II. Case studies

> Before hitting the meat of the course, we explore a few extended examples. Each of these serves to to reinforce and illuminate the principles laid out in the previous unit. Moreover, each is used as elements in certain later topics.

   A. Linear sorting

   B. Disjoint sets and union/find

   C. Heaps and priority queues

   D. Bit vectors

III. Graphs

> The graph, as a mathematical abstraction, is one of the most important tools for representing the structure of information and relationships within information, used in one form or another by many fields. Accordingly, efficient ways to store and process graphs are a crucial piece of the study of data structures and algorithms.
>
> This exploration of standard graph algorithms is both an exercise in applying the principles of the course and a primer in algorithmic elements whose variations can be applied to many specialized applications.

   A. Graph concepts and implementation

   B. Basic traversal algorithms: Depth-first and breath-first

   C. Minimum spanning trees

   D. Shortest paths

IV. Trees

> In this chapter we embark on a task that stays with us for the rest of the course: Developing efficient data structures for implementing unordered collections, such as the map ADT. The naïve implementations have linear-time operations. This chapter considers one family of solutions: the binary search tree abstraction and various tree data structures that afford map implementations with logarithmic-time operations.
>
> Specifically we consider several strategies for *self-balancing binary search trees* and compare their performance and other practical aspects.

   A. Binary search trees

   B. Balanced binary search trees

      1. AVL trees

      2. Red-black trees

    3. Left-leaning red-black trees

    4. 2-3 trees

    5. B-trees

V. Dynamic programming

> Dynamic programming is a technique that uses tables to store intermediate results of recursive algorithms for certain optimization problems with overlapping subproblems.
>
> Our exploration is organized around an introduction of several problems suited to dynamic programming solutions, the explanation of the principles of dynamic programming, the application of those principles to the given problems, and, finally, the working out of a larger dynamic programming solution to a more complicated problem.
>
> That more complicated problem is a continuation of our quest for improved implementation of the map ADT: constructing a binary search tree not to achieve balance but optimality based on the look-up probabilities of the keys.

  A. Three problems

  B. Principles of dynamic programming

  C. Three dynamic programming solutions

  D. Optimal binary search trees

VI. Hashing

> Having achieved logarithmic-time operations for a map using a binary search tree, we pursue the quest further with hash tables, a family of data structures that support constant-time map operations. We consider several specific approaches and compare their real performance.

  A. Hashing general concepts

  B. Separate chaining

  C. Open addressing with linear probing

  D. Hash functions

  E. Perfect hashing

VII. String processing

> A lot of data used in modern applications is textual data. String processing plays a crucial role in many of the common uses of computation today. This includes considering a data structure for efficient map operations assuming string keys.

  A. Sorting strings

  B. Tries

  C. Regular expressions

LEARNING OUTCOMES. Corresponding to the six big ideas of the course listed earlier, this course's aim is that at the end of the semester, students are able to

1. State and demonstrate a loop invariant for an algorithm.

2. Determine the performance of an algorithm and identify a big-oh or big-theta category.

3. Differentiate between ADTs and data structures and articulate the trade-offs among data structures studied in class.

4. Articulate the factors in implementing a binary search tree and how they affect the performance of a BST implementation of a map.

5. Articulate the factors in implementing a hashtable and how they affect the performance of a hashtable implementation of a map.

6. Articulate the attributes of a problem that is suited for a dynamic programming solution and implement a given dynamic programming algorithm.

In addition to these, together we have the general objective of seeing algorithm design and implementation as a creative expression for God's glory and observing algorithmic solutions as part of God's creation.

Course procedures

HOW WE DO THIS COURSE. The rhythm of this course consists in students reading about new ideas in the text; the ideas being discussed in class and practiced in lab; and students working out the details of the ideas in projects. Most of students' work for this course is in projects in which they will implement the things we discuss in class. Students' self-directed study is enforced through practice problems and quizzes. On tests, students apply the things we have learned to new problems.

The course is organized into units (corresponding to chapters in the book, except that the first two chapters make one unit), with units divided into topics. Each topic corresponds to between one and three days in class. Each unit appears as a module on Canvas; topics within the module are indented together.

Each topic has a reading associated with it. In some cases the reading should be done before the material is presented in class, but for most topics the student can choose whether to do the reading before or after class, depending on how that student learns best. The reading is important as the "full version" of the information that is only highlighted in class. Most topics also have practice problems, a quiz, and a project.

QUIZZES. Quizzes are administered through Canvas, and students can take them when convenient before the indicated due date. These quizzes mainly enforce the reading and practice problems.

LABS. Lab exercises are done using pair programming. See schedule of lab activities below.

PROJECTS. As stated above, projects are where students will do most of their work and, presumably, most of their learning. To a certain extent, students may work on projects at their own pace. The due dates found on Canvas indicate when a student should be done with the individual projects in order to keep up with the course, but no penalty is given for projects later than the posted due date. All projects must be turned in by the last day of classes, May 2. (Official deadline is midnight between May 2 and May 3. Enforced deadline is when I wake up in the morning on May 3.) Projects are graded for correctness, verified by unit tests (and inspection for general conformity to the intent of the project). Students are supplied all unit tests used in grading (except that the right is reserved to modify the data used in the unit tests to make sure the code hasn't been hardwired to pass unit tests with specific data). See the *Policies etc* section of this syllabus for other details. Other information about projects, including practical suggestions, can be found on the course website.

TESTS. There are four tests, currently scheduled as Test 1 on Wed, Oct 15; Test 2 on Thur, Nov 13 (during the lab section); and Tests 3 and 4 on Thursday, Dec 18, 10:30am–12:30pm. Note that Tests 3 and 4 together constitute the final exam for this course, but are treated separately for the purposes of this syllabus. Specifically

- Test 1 is on paper and has *conceptual problems* from the first third of the course (prolegomena, case studies, and graphs).

- Test 2 is at a computer and has *programming problems* from the first half of the course (prolegomena, case studies, graphs, and trees).

- Test 3 is on paper and has *conceptual problems* from the last two-thirds of the course (trees, dynamic programming, hash tables, and strings).

- Test 4 is at a computer and has *programming problems* from the second half of the course (dynamic programming, hash tables, and strings).

GRADING. In order to pass the course (that is, to receive a D grade or better), a student must (a) achieve a minimum score of 75% on the projects and (b) achieve an average score of at least 50% on the tests.

(Note that students are given all the test cases used in grading and may resubmit projects throughout the semester. There is no reason for a student who takes the course seriously and acquires basic competency not to get 100% on projects.)

For students who have met the minimum requirements, their semester score is calculated as the weighted average of the following scores (each as a percentage):

instrument	weight
Projects	30
Tests	60 (4 @ 15 each)
Other	10

. . . where *Other* comprises short assignments, lab activities, readings, quizzes, etc. Some of these may be graded for correctness, others may be marked as done or not. Also, students may get extra credit points applied to the "other" category for each mistake in the book they are the first to discover, at the instructor's discretion.

I use the "Gradebook" feature on Canvas only to communicate scores on individual assignments and tests. I do **not** use the Canvas gradebook for my official record-keeping for scores, for calculating semester scores, or determining letter grades. Please **ignore** any grade estimate that Canvas gives you for this course.

Policies etc

LAB RULES. The lab activities are subject to the following rules, which are consistent with other course policies for academic integrity and classroom behavior.

1. Cell phones and other personal devices should never be visible during the lab session. If a phone or similar device is ever visible during the lab session (including while the student is walking toward the door), the student will receive a 0 on that lab activity.

2. Students should never open their email or social media accounts during the lab session. Any student checking email or social media during lab will receive a 0 on that lab activity.

3. Students should never use StackOverflow or similar sites during the lab session. Any student who looks at websites other than official Java documentation for help during lab will receive a 0 on that lab activity.

4. Students should never use ChatGPT or similar tools during lab session (or any other time). Any student who has ChatGPT or similar tools open during lab will receive a 0 on that lab activity on a first offense and will fail the course on a second offense.

5. Students will receive a 0 for a missed lab without a reasonable excuse. Reasonable excuses include (a) college-related activities such as athletic participation, which the student has notified the instructor about ahead of time, and (b) illness. In the case of an excused miss, the student should make up the lab on his or her own time, sending a solution to the lab to the instructor by email, due by the Monday following the lab, unless otherwise approved.

ACADEMIC INTEGRITY. Students must not use ChatGPT, CodePilot or similar "Generative AI" tools for anything related to this course such as (but not limited to) help or solutions to the projects, labs, or practice problems. Any student who has been found to use Generative AI in any way relating to the project sequence in this course will fail the course.

Collaboration among students in the class is permitted on projects and most assignments. Using *code* for projects from any outside resources (print, electronic, or human) is not permitted. Using *ideas* from outside resources in projects is not recommended, but if a student does get use any outside resources for concepts used in projects, they must be cited using the same standards as would be used in a research paper. If you relied on an outside resource in any way for any project, submit a file with your project named CITATIONS giving citations for the resources and

an explanation of how and why you used them. On any assignment given from the textbook, no resources that specifically serve as solutions to exercises from the textbook may be used.

As much as possible, the projects are designed to minimize the opportunity for students to fool the automated grading, that is, to submit solutions that happen to pass the tests but fail to implement the intended approach to the problem. Students, however, are expected to abide by the intent of the projects and shall not submit solutions that "cheat" the unit tests if they discover a way to do so.

A project or assignment on which a student violates these policies will be rejected without option to resubmit. Repeated offenses will be handled through the college's official disciplinary procedures.

LATE ASSIGNMENTS. Projects have no due date except that all projects must be turned in by the last day of classes. No credit will be given for late work.

PROJECT GRADING. My intention is to grade projects solely on correctness as demonstrated by unit tests, but I reserve the right to inspect code for conformance to specifications that are not easily verified automatically. You will be given all the unit tests used in grading, except that the raw data used by the unit tests during grading may be different from that provided to students, to prevent students from simply writing code that apes the responses the specific unit tests check for. **Extra credit** shall be given to students who discover cases not covered by the given unit tests; in that case, the student must write a JUnit test case and provide a would-be solution to the project that passes the current set of unit tests but fails the new one. **Bonus extra credit** shall be awarded if the instructor's own solution fails the new test. Moreover, **and this is important**, that new test case shall be distributed to the class and added to the test cases used in grading. Any student who has already submitted a solution that does not pass the new test case should resubmit a corrected solution.

TESTS USING COMPUTERS. For Tests 2 and 4, the computer science lab (Meyer 154) shall be reserved for our use. The lab machines shall be logged into special accounts for the purpose of test-taking. Internet access shall be turned off. Students shall be given a few JUnit tests to clarify the problem but **not** the JUnit tests to be used in grading.

ATTENDANCE. Students are expected to attend all class periods, in person whenever possible. It is courtesy to inform the instructor when a class must be missed.

EXAMINATIONS. Students are expected to take all tests, quizzes, and exams as scheduled. In the case where a test must be missed because of legitimate travel or other activities, a student should notify the instructor no later than one week ahead of time and request an alternate time to take the test. In the case of illness or other emergencies preventing a student from taking a test as scheduled, the student should notify the instructor as soon as possible, and the instructor will make a reasonable accommodation for the student. The instructor is under no obligation to give any credit to students for tests to which they fail to show up without prior arrangement or notification in non-emergency situations. The final exam block, when Tests 3 and 4 are held, is Thursday, Dec 18, 10:30am–12:30pm. I do not allow students to take finals early (which is also the college's policy), so make appropriate travel arrangements.

ACCOMMODATIONS. If you have a documented need for accommodations, I will have received a letter on your behalf from the Learning and Accessibility Services Office. But *please talk to me* about what accommodations are most useful to you. In particular, if you desire accommodations for test-taking, talk to me a reasonable amount time in advance (say, at least two class periods) so arrangements can be made. (See also the College's statement below.)

OFFICE HOURS. My *drop-in* office hours this semester are MWF 3:30–4:30pm. You can make an appointment through Calendly; I'm available most of the day on Thursday and sometimes on other days.

ELECTRONIC DEVICES. My intent is for my courses to be electroinic-device-free zones. Unless you have made special arrangements with me, please keep all laptops, tablets, phones, etc silenced and put away. If you need to check your phone for something, please discreetly step out into the hall. In particular, **NO TEXTING OR USING SOCIAL MEDIA DURING CLASS MEETINGS.**

All this, the Lord willing.

College syllabus statements

THE COLLEGE REQUIRES THAT THE FOLLOWING STATEMENTS BE INCLUDED IN ALL SYLLABI.

The "Academic Information" website referred to below is found and `https://catalog.wheaton.edu/undergraduate/academic-policies-information/academic-information/`

ACADEMIC INTEGRITY. (See "Integrity of Scholarship" on "Academic Information" website.)
The Wheaton College Community Covenant, which all members of our academic community affirm, states that, "According to the Scriptures, followers of Jesus Christ will...be people of integrity whose word can be fully trusted (Psalm 15:4; Matt. 5:33-37)." It is expected that Wheaton College students, faculty and staff understand and subscribe to the ideal of academic integrity and take full personal responsibility and accountability for their work. Wheaton College considers violations of academic integrity a serious offense against the basic meaning of an academic community and against the standards of excellence, integrity, and behavior expected of members of our academic community. Violations of academic integrity break the trust that exists among members of the learning community at Wheaton and degrade the College's educational and research mission.

ACCOMMODATIONS. (See "Learning and Accessibility Services" on the "Academic Information" website).
Wheaton College believes that disability is an indispensable part of the diversity of God's Kingdom. We work to provide equal access to College programs and activities as well as spaces of belonging for students with disabilities. Students are encouraged to discuss with their professors if they foresee any disability-related barriers in a course. Students who need accommodations in order to fully access this course's content or any part of the learning experience should connect with Learning and Accessibility Services (LAS) as soon as possible to request accommodations `http://wheaton.edu/las` (Student Services Building - Suite 209, las@wheaton.edu, phone 630.752.5615). The accommodations process is dynamic, interactive, and completely free and confidential. Do not hesitate to reach out or ask any questions.

STUDENT WELLNESS. Wheaton College professors are often among the first to witness, observe, or become aware of a matter that a student is navigating, which has disrupted their communication pattern, course workflow, and concerning behaviors. If you need support, use the student resource and request form found on the student portal through the Wheaton Gateway. If your faculty believes you need support and care, they will submit a CARES referral form for us to reach out to you. We offer coordinated care, advocacy, resources, encourage engagement, and ongoing support for acute matters you may navigate. Aligning with the attendance policy as outlined in the student handbook, Student Wellness can provide notices of absences when applicable. Likewise, we are here to work with you through circumstances beyond your control. We welcome the opportunity to meet and come alongside you to share information that will keep you well-informed about campus and community resources to aid your holistic health, retention, and academic student success. We are located in the Student Service Building – Suite 218. You may contact us via email at student.wellness@wheaton.edu or by phone at 630-72-5941. We are also available by walk-ins or scheduled appointments with either the Resource and Support Specialist, Kedisha Kelly, the Associate Dean, Dr. Carrie Williams, or the Dean, Dr. Toussaint Whetstone.

BEHAVIOR POLICY (See "Classroom Demeanor" on the "Academic Information" website).
Appropriate classroom demeanor is expected of all students. A faculty member may remove any student from a class if the student exhibits uncivil conduct, which includes behavior that is disinterested, disengaged, disrespectful, disruptive, defiant, or disturbing.

GENDER INCLUSIVE LANGUAGE POLICY (See "Gender Inclusive Language" on the "Academic Information" website).
Please be aware of Wheaton College's policy on inclusive language, "For academic discourse, spoken and written, the faculty expects students to use gender inclusive language for human beings."

EQUITY AND TITLE IX: Wheaton College instructors help create a safe learning environment on our campus. The College requires employees to report incidents of discrimination, harassment, and sexual misconduct to the Title IX Coordinators/Equity Officers. When they learn of an incident that may be a crime or may be a violation of the College Nondiscrimination Policies, instructors at the college have a duty to report and are required to share all relevant information with the College. Confidential resources available to students include Confidential Advisors, the Counseling Center, Student Health Services, and the Chaplain's Office. More information on these resources and College Policies is available `http://www.wheaton.edu/equityandtitleIX`.

WRITING CENTER: See `https://www.wheaton.edu/academics/services/writing-center/`. The Writing Center is a free resource that equips undergraduate and graduate students across the disciplines to develop effective writing skills and processes. This academic year, the Writing Center is offering in-person consultations in our Center in Buswell Library, as well as synchronous video consultations online. Make a one-on-one appointment with a writing consultant here. `https://wheaton.mywconline.com/`.

MANDATORY REPORTING: All employees of the College are mandatory reporters of suspected child abuse and neglect as required by the State of Illinois. Mandated reporters are required to call the Department of Children and Family Services Hotline (1-800-25-abuse) when they have reasonable cause to believe that a child known to them in their professional or official capacity may be an abused or neglected child. Mandated reports should call the Hotline if the alleged victim is a child under the age of 18; the alleged perpetrator is a parent, guardian, relative, caregiver, or any person who lives resides in the same home or who came to know the child through an official capacity or position of trust (ie: teacher, coach or healthcare provider) and; there must be an incident of harm or set of circumstances that would lead a reasonable person to suspect that child was abused or neglected.