COS 226: Introduction to Data Structures

Sudarshan S. Chawathe

University of Maine

Fall 2016

Data structures are schemes that organize data to permit efficient access in certain modes. The desired modes of access (different kinds of look-ups and modifications), and their relative importance in an application, typically guide the choice of existing data structures and the design of new ones. A judicious choice of data structures often results in very significant improvements in the running time of a program. In order to make such decisions, as well as to design new data structures, we need to understand existing data structures, their access modes, and performance characteristics. In this course, we study data structures from several perspectives, including design, analysis, and application.

News and Reminders:

- Please read the newsgroup for timely announcements.
- Class newsgroup: Local group `umaine.cos226` on NNTP server `creak.um.maine.edu`. A simple Web interface is at `http://cs.umaine.edu/~chaw/news/`.
- The most recent version of this document may be found at `http://cs.umaine.edu/~chaw/cos226/`.
- Some sections below point to material in separate documents that are found on the class Web site, linked from the online version of this document.
- Please use the PDF version of this document for printing and reference: `cos226.pdf`

Goals and Outcomes

Goals

- Understand several interesting data structures and their properties.
- Learn how to use data structures and other tools to solve problems in various application areas.
- Gain experience in reading the relevant research literature and other publications used to disseminate knowledge in the field.
- Practice the appropriate and ethical use of existing material of different kinds, such as source code, services, and documentation.
- Gain experience in contributing to the body of knowledge.
- Learn how to analyze of the running times of programs using simple mathematical methods.
- Gain experience in conducting and documenting experimental studies of programs.
- Improve our programming skills, with attention to software engineering principles.
- Improve our communication skills, with particular emphasis on written communication and, further, well-written programs.
- Learn how to manage a self-directed project.

Student Learning Outcomes

Upon successful completion of this course, students are able to

- List commonly used data structures, and the advantages and drawbacks of each.

- Determine suitable data structures for solving a given problem.
- Effectively read suitable publications related to the topic.
- Use resources such as others' code and writing in an ethical and professional manner.
- Contribute to the body of knowledge at an undergraduate level.
- Analyze the running times of programs using simple methods.
- Perform simple experimental studies of programs.
- Program with attention to community standards and good practices.
- Communicate their programming work effectively.
- Meet Quantitative Literacy General Education requirements, such as being able to [following text is from U. Maine Gen. Ed. documents]:
  - Translate problems from everyday spoken and written language to appropriate quantitative questions.
  - Interpret quantitative information from formulas, graphs, tables, schematics, simulations, and visualizations, and draw inferences from that information.
  - Solve problems using arithmetical, algebraic, geometrical, statistical, or computational methods.
  - Analyze answers to quantitative problems in order to determine reasonableness. Suggest alternative approaches if necessary.
  - Represent quantitative information symbolically, visually, and numerically.
  - Present quantitative results in context using everyday spoken and written language as well as using formulas, graphs, tables, schematics, simulations, and visualizations.

Contact Information

Class meetings:

Time: Tuesdays & Thursdays, 12:30–1:45 p.m.
Location: Little Hall, Room 120.

Instructor: Sudarshan S. Chawathe

Office: Neville Hall, Room 224.
Office hours: (Please check for changes.)
Tuesdays and Thursdays: 3:15–4:30 p.m.
Phone: +1-207-581-3930.
Please avoid calling except for truly urgent matters.
Email: `sudarshan.chawathe@maine.edu`
Use email only for messages unsuitable for the newsgroup. (See below.) Please use only this email address and put the string *COS226* near the beginning of the Subject header of the message. *All other messages may be ignored.*
Web: `http://cs.umaine.edu/~chaw/`.

Teaching Assistant: Ezekiel Rhodes

Office: TBA
Office hours: TBA
Email: `Ezekiel_Rhodes@umit.maine.edu`

Online Resources

Class Web site:
`http://cs.umaine.edu/~chaw/cos226/`
We will use the class Web site for posting assignments, readings, notes, and other material. Please monitor it.

Class Newsgroup: We will use the local USENET newsgroup `umaine.cos226` on the NNTP (net news) server `creak.um.maine.edu` for electronic discussions. The Web interface at `http://cs.umaine.edu/~chaw/news/` provides convenient access. Some further, more general, information on USENET appears at `http://en.wikipedia.org/wiki/Usenet`. The newsgroup is the primary forum for electronic announcements and discussions, so please monitor it regularly, and post messages there as well. Unless there is a reason for not sharing a question or comment, please *use the newsgroup, not email*, for questions and comments related to this course.

Class mailing list: *Please make sure you are on the class mailing list.* The mailing list will use the email address for each student as recorded in the official university records (*Maine Street* system). We will use this mailing list only for urgent messages because all other messages will go on the class newsgroup. I anticipate fewer than a dozen messages on this list over the semester.

Grading Scheme

Grade components: *Students are expected to complete and submit all assigned coursework in good faith; those who fail to do so will earn a failing grade, regardless of overall numerical score.*

component	% of grade
class participation and groupwork	10
classroom exercises	5
homeworks (about 4)	20
two quizzes (short exams)	10
two midterm exams	20
final exam	20
term project	15

Class participation: Students are expected to contribute to learning by asking questions and making relevant comments in class and on the class newsgroup. Quality is more important than quantity. Disruptive activity contributes negatively. See policies below.

Classroom exercises: Our work in the classroom will include a number of short group exercises, meant to solidify understanding of the concepts being discussed. One or more such exercises are likely to be part of most class meetings. The exercises will be graded primarily for effort, group work, and other contributions, and less so for simple correctness. Since attendance is not mandatory (cf. policies), some low-scoring exercises will be dropped for each student. Please ask for clarifications if there are concerns about the interaction of this component and the attendance policy.

Homeworks: Homeworks include programming and non-programming ones, often mixed. No collaboration is permitted. Everyone is encouraged to discuss the problems and solution strategies *at a high level*, but the final solution and details must be individual work. If the boundary between permissible and non-permissible interactions is unclear, please ask for clarifications.

Exams and Quizzes: All exams and quizzes are *open book, open notes*. You are free to bring with you any resources that you find useful. However, no communications are permitted other than between students and me. The use of computers during exams is strongly discouraged, but brief use may be permitted provided it does not cause a disturbance, at the discretion of the proctor. You may use the Internet, but only as a library to look up material you may find useful. Ask for clarifications in case of any doubt. The exams are designed to require no equipment other than a pen and paper, along with the textbook and assigned readings.

Midterm exams will be held during regular class meetings, and will be roughly an hour long. Each quiz is a short exam, roughly half an hour long, held during part of a class meeting. The final exam follows the usual university schedule, and is thus held outside of regular class meeting times, and often in a different location.

Term Project: In addition to the programming and other homeworks, this course features a term project. Group term projects are strongly encouraged, with groups of two to four students being typical, but individual term projects are also permitted. The details of the project are fairly flexible, and all students are encouraged to propose project that excites them. A few project ideas will also be provided in class for use as term projects, perhaps with some of modifications. The main requirement for the project is that it demonstrate the ability to work independently and apply the concepts studied in the course to an application. Projects will be graded based on a project submission that includes a project report, complete and well-documented source code and build instructions, and a script for a demonstration. *These materials will be due two weeks before finals week.* Further details will be announced in class.

Policies

Due dates: All due dates and times, as announced in class, are strict, to the second. If you believe your work was delayed by truly exceptional circumstances, let me know as soon as those circumstances are known to you and I will try to make a fair allowance. However, *the default is that you get a zero if you don't turn in the work on time*, and fail the class if you don't turn it in at all (cf. Grade Components above).

Attendance: Although I expect students to attend all class meetings, I will not be taking attendance. *If you miss a class meeting, you are responsible for catching up on the lost material, including any important announcements made in class, on your own.* If you have a valid reason for missing a class, let me know early and I will try to help you make up the class. There will be no make-up exams or quizzes. A missed test earns zero credit. If you have a valid reason for missing a test, let me know as early as that reason is known to you and I will make a fair allowance but there will be no make-up tests in any case.

Classroom activities: This course is based on an active learning format, so effective classroom activities are critical to its success. Students are expected to contribute to their own learning and that of their classmates, and to devote 100% of their attention to these activities while in class. On a similar note, all electronic and other distractions (computers, phones, assorted gizmos, etc.) must be completely silenced and put away for the entire duration of the class. (Students who need any such devices for disability accommodations should follow the guidelines outlined below. Others who need any accommodation in this regard due to special circumstances should make advance arrangements with the instructor.) No food or drink is allowed in class, other than water in a spill-proof container. Students who violate these rules or otherwise cause distractions in class will be asked to leave with *no warning*; habitual violators will face disciplinary action.

Office hours: All students are encouraged to make use of both the instructor's and TA's office hours to further their learning, obtain assistance on homework assignments, obtain feedback on their class performance, etc. However, office hours are not to be used as a substitute for attending and participating in class meetings (see above). Similarly, assistance with homework assignments will be limited to what is appropriate based on fairness to all; students are expected to demonstrate substantial effort on the assignment before seeking assistance.

Make-up classes: I may have to reschedule a few classes due to my other professional commitments. I will make every attempt to minimize the number of such occurrences and to reschedule for a time that works for most students. Further, I will make sure no student is penalized by such occurrences.

Academic honesty (standard university wording): Academic dishonesty includes cheating, plagiarism and all forms of misrepresentation in academic work, and is unacceptable at The University of Maine. As stated in the University of Maine's online undergraduate Student Handbook, plagiarism (the submission of another's work without appropriate attribution) and cheating are violations of The University of Maine Student Conduct Code. An instructor who has probable cause or reason to believe a student has cheated may act upon such evidence, and should report the case to the supervising faculty member or the Department Chair for appropriate action.

Disabilities (standard university wording): If you have a disability for which you may be requesting an accommodation, please contact Ann Smith, Director of Disabilities Services, 121 East Annex, 581-2319, as early as possible in the term.

Special circumstances (standard university wording): In the event of an extended disruption of normal classroom activities, the format for this course may be modified to enable its completion within its programmed time frame. In that event, you will be provided an addendum to the syllabus that will supersede this version.

Programming

The focus of this course is on data structures, algorithms, algorithm analysis, and problem solving techniques in Computer Science, and not on programming, much less programming in a particular language. Programming is, however, a valuable part of the course as it helps us solidify the abstract concepts we study. We will use Java as the primary programming language. Submissions will be in the form of packaged, well documented, Java *source* files. Proper documentation and packaging of source code and other material is a crucial component of assigned work and submissions failing in this regard will receive no credit.

Programming Environment and Tools: You are free to choose details such as operating system, development environment, and editor based on your preferences. However, no matter what you use, the submission should be a *source*-code package that works on a standard Java SE platform. In particular, submissions should work on any operating system and hardware supported by Java SE. Further details on the packaging, submission, and testing procedure will be provided in class and on the newsgroup.

Other Languages: If you prefer to use a language other than Java, please contact me. I am quite open to the idea, and encourage interested students to explore it further. However, please check with me very early in the semester so that we can determine the specifics to make sure your submissions can be tested and graded fairly. You should avail of this option only if you are confident enough of your programming skills to not require any programming help, and are prepared to take on additional work. *This option is designed for students who are proficient in Java and wish to use this opportunity to master another language, not for students weak in Java who wish to avoid it.* Anyone granted this option will still be responsible for all Java-related material in the course.

Literate Programming: All submitted work must use a *literate programming style*: Your programs must be designed with a human as the intended reader, although they must also compile and run correctly. *Programs that do not meet this requirement are likely to receive a zero score with no further consideration.* Details will be discussed in class. The use of any specific literate-programming or documentation tool is neither necessary nor sufficient for this requirement.

Class Accounts: Although the use of official class accounts, on department computers, is not strictly required, it is a good idea for everyone to have accounts on both our main Unix host (`aturing`) and the cluster of PCs. Among other uses, these accounts will permit testing that code submissions work correctly in a reference environment. Class accounts will be generated based on the forms distributed at the first class meeting. You should be able to access your `aturing` account from anywhere on the Internet, including the labs in Neville Hall and elsewhere on campus, by using *ssh* to connect to `aturing.umcs.maine.edu`. On most Unix hosts, the command `ssh -l` *`username`* `aturing.umcs.maine.edu` should suffice. For Windows hosts, the freely available *Putty* program works well: `http://www.chiark.greenend.org.uk/~sgtatham/putty/`. *Do not use unencrypted telnet sessions to connect to your account.*

Schedule

At the beginning and end of each class, I typically announce the topics and textbook sections covered in that class and those due at the next class. It is important that students read the material *before* the class in which it is discussed and, in general, keep up with readings and studies. An approximate schedule appears in Figure 1. Please use it only as a rough guide to plan your studies. *Do not use it to schedule travel or other events.* If you need a definite answer on when something will or will not occur, you should check with me.

Textbook and Readings

Textbook: Mark Allen Weiss. *Data Structures and Problem Solving Using Java*. Addison-Wesley, 4th edition, 2010. The university bookstore carries this book, which is a required textbook for this course. You may be able to get by with the 3rd edition, at your risk, but do not use an even earlier version as the changes are substantial.

The core topics for this course are found mainly in Chapters 18 and beyond; a few earlier chapters, such as 5, 8, and 14 are also relevant. Detailed coverage information will be announced as we progress in the semester. Most chapters in roughly the first third of the textbook, as well as some later chapters, discuss topics that are covered in the prerequisite course, COS 225. We will not be covering these topics in this course but they are important for successful completion of homeworks and tests, so it is advisable to brush up on them.

Other Readings: All the following are recommended, but not all are required. Further details and additional readings will be announced in class and may appear here as well.

1. Arne Andersson. Balanced search trees made simple. In *Proceedings of the Workshop on Algorithms and Data Structures*, pages 60–71, Montreal, Canada, August 1993.

   This paper introduces AA-trees and includes very nice examples and figures.

2. Sanjeev Saxena. Dominance made simple. *Information Processing Letters*, 109(9):419–421, April 2009.

   This short paper is a good example of how some of the basic concepts studied in this course may be used as building blocks to solve more complex problems.

3. Gilad Bracha. Generics in the Java programming language. Tutorial. `http://java.sun.com/`, July 2004.

   The concepts explained here are essential for making good use of generics in Java and it is very painful to learn them the hard way (e.g., while debugging your code).

4. Sudarshan S. Chawathe. Segment-based map matching. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 1190–1197, Istanbul, Turkey, June 2007.

   The main purpose of this paper, for this course, is providing a concrete example how data structures and related concepts find use in current research and applications. Sections I, II, and III are required reading. The rest of the paper is optional reading.

5. Derrick Coetzee. An efficient implementation of Blum, Floyd, Pratt, Rivest, and Tarjan's worst-case linear selection algorithm. `http://moonflare.com/`, January 2004.

6. Jon Bentley and Don Knuth. Programming pearls: Literate programming. *Communications of the ACM*, 29(5):364–369, May 1986.

7. Paul E. Black. Dictionary of algorithms and data structures. `http://www.nist.gov/dads/`, September 1998.

8. Lloyd Allison. Suffix trees. `http://www.allisons.org/ll/AlgDS/Tree/Suffix/`, 2008.

Tuesday	Thursday
August 30th C1 Introduction; trees §18.0–18.3.	**September** 1st C2 Traversals; binary search trees; order statistics; §18.4–18.end.,19.0–19.2.
6th C3 Analysis of algorithms; maximum contiguous subsequence; §5.0–5.3.	8th C4 Static searching; further analysis; §5.4–5.end.
13th C5 BST analysis, AVL trees; §19.3–19.4.	15th C6 ★ **Quiz 1**, regular class time & place.
20th C7 Red-black trees; §19.5.	22nd C8 AA-trees; §19.6.
27th C9 B-trees; disk data structures; §19.7–19.end.	29th C10 AA-trees; B-trees; §19.6,19.8.
October 4th C11 Catch-up; review.	6th C12 ★ **Midterm Exam 1**, regular class time & place.
11th ×*No class.* Fall break Oct. 10th–11th.	13th C13 B-trees; binary heaps; §19.8,21.1–21.3.
18th C14 Splay trees; §22.1–22.2.	20th C15 Splay trees; §22.3–22.4.
25th C16 Skew heaps §23.1.	27th C17 ★ **Quiz 2**, regular class time & place.
November 1st C18 Splay trees.	3rd C19 Pairing heap; §23.2.
8th C20 Sorting; §8.1–8.4.	10th C21 Sorting; selection; §8.5–8.8.
15th C22 Catch-up; review.	17th C23 ★ **Midterm Exam 2**, regular class time & place.
22nd C24 Graphs; shortest paths; §14.4–14.5.	24th ×*No class.* Thanksgiving break Nov. 23th–Nov. 27th.
29th C25 Graphs; shortest paths; §14.1–14.3. **Term project submissions due.**	**December** 1st C26 Synthesis.
6th C27 Catch-up; review. *CS education week Dec. 5–11.* **Term Projects Exhibition**	8th C28 *CS education week.*
13th ×*No class.* Finals week Dec. 12th–16th. ★ **Final Exam per Univ. schedule.**	15th ×*No class.* Finals week Dec. 12th–16th. ★ **Final Exam per Univ. schedule.**

Figure 1: ***Approximate*** *schedule, likely to change.*

9. Samuel W. Reynolds. A generalized polyphase merge algorithm. *Communications of the ACM*, 4(8):347–349, 1961.

   This paper provides a succinct and readable description of polyphase merging. It is a very useful supplement to the description in the textbook, which is missing many important details.

Exercises, Homeworks, Tests, and Notes

Material will appear here as we move along the semester. It may be useful to refer to the homeworks and tests from the previous session: `http://cs.umaine.edu/~chaw/201509/cos226/`.

- Class exercises:
  - Class Exercise 1: `hwq/ce01.pdf`.
  - Class Exercise 2: `hwq/ce02.pdf`.
  - Class Exercise 3: `hwq/ce03.pdf`.
  - Class Exercise 4: `hwq/ce04.pdf`.
  - Class Exercise 5: `hwq/ce05.pdf`.
  - Class Exercise 6: `hwq/ce06.pdf`.
  - Class Exercise 7: `hwq/ce07.pdf`.
  - Class Exercise 8: `hwq/ce08.pdf`; sample solution: `hwq/ce08s.pdf`.
  - Class Exercise 9: `hwq/ce09.pdf`.
  - Class Exercise 10: `hwq/ce10.pdf`.
  - Class Exercise 11: `hwq/ce11.pdf`.
- Homework assignments:
  - Homework 1: `hwq/hw01.pdf`.
  - Homework 2: `hwq/hw02.pdf`.
  - Homework 3: `hwq/hw03.pdf`.
  - Homework 4: `hwq/hw04.pdf`.
- Quizzes and Exams:
  - Quiz 1: `hwq/q01.pdf`.
  - Midterm Exam 1: `hwq/mt01.pdf`.
  - Quiz 2: `hwq/q02.pdf`.
  - Midterm Exam 2: `hwq/mt02.pdf`; sample solution: `p/mt02s.pdf`.
- Notes:
  - An extended example of skew-heap insertions: `notes/skew-heap-50.pdf`.
- A few ideas for term projects:
  `notes/projideas.pdf`

Homework and Project Submissions

Handwritten answers to non-programming problems should be submitted in class on the due date, at the beginning of class, unless prior alternate arrangements are made. If you prefer to type your answers, please make sure the result uses the proper symbolic notation for mathematical constructs. *Illegible, hard to read, or otherwise messy submissions, whether handwritten on typed, are likely to be returned without grading, for zero credit.* Answers to programming problems should be submitted electronically, using the packaging and submission procedure that will be described in class and on the class newsgroup.

All electronic submissions must be made using the upload interface at `http://cs.umaine.edu/~chaw/u/`. Electronic submissions in **all other forms**, such as email or physical media, will be **discarded and receive no credit.**

If your upload is successful, you will be presented with a confirmation Web page similar to the following sample. You should record the reported MD5 checksum and timestamp.

```
SUCCESS: Please note the following for your records.

Successfully saved cos226-hw01-aardvark-alice-4233.tgz.
MD5 checksum: 09ee098b83d94c7c046d6b55ebe84ae1
Timestamp: 2016-09-10 03:37:42
```

If you do not see something very similar then your submission is unsuccessful.

If (and only if) there are unexpected problems and you are unable to submit your work as above, then you should save your file on your own computer (with some backups), compute its MD5 checksum using the md5sum utility on Unix-like systems (or other similar tools), and submit the file name, time stamp, and MD5 checksum (only, not the file itself) by email with a suitable Subject header.

Sample Code

Sample code and other files may appear here.

Additional material, such as recently added files as announced in class, may be found in the code subdirectory:

`http://cs.umaine.edu/~chaw/cos226/code/`