COS 226: Introduction to Data Structures

Sudarshan S. Chawathe

University of Maine

Fall 2009

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 `news.cs.umaine.edu`. Web interface to get started: `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

- 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.

Contact Information

Class meetings:

Time: Tuesdays & Thursdays, 2:00–3:15 p.m.

Location: Neville Hall, Room 227.

Instructor: Sudarshan S. Chawathe

Office: Neville Hall, Room 224.
Office hours: (Please check for changes.)
Tuesdays & Thursdays: 8:00–8:30 a.m., 10:30–11:00 a.m., 1:45–2:00 p.m., 3:15–3:30 p.m.
Phone: +1-207-581-3930.
Please avoid calling except for truly urgent matters.
Email: `chaw@cs.umaine.edu`
Use email only for messages unsuitable for the newsgroup. (See below.) Please put the string *COS226* near the beginning of the Subject header of your messages to me.
Web: `http://cs.umaine.edu/~chaw/`.

Teaching Assistant: Mark Royer

Office: East Annex, Room 229.
Office hours: (Please check for changes.)
Mondays & Wednesdays: 1:00–4:00 p.m.
Phone: +1-207-581-2005.
Email: `mroyer@cs.umaine.edu`

Online Resources

Class Web site:
`http://cs.umaine.edu/~chaw/cos226/`
We will use the class Web site for posting homework assignments, hints, solutions, etc. Please monitor it.

Class Newsgroup: We will use the local USENET newsgroup `umaine.cos226` on the NNTP (net news) server `news.cs.umaine.edu` for electronic discussions. If you are unfamiliar with USENET, you may find the Web interface at `http://cs.umaine.edu/~chaw/news/` useful as a quick way to get started. You may find further information on USENET 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 your 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.* A sign-up sheet is circulated at the first class meeting. If you miss it, please contact me to get on the list. 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:

class participation	5 %
classroom exercises	5 %
homeworks (about 5)	25 %
two quizzes (short exams)	10 %
two midterm exams	20 %
final exam	20 %
independent 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. *Please make sure all disruptive devices are disabled while in class.* If you have a good reason for wanting to be disturbed in class, please contact me to make the appropriate arrangements.

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 for correctness as well as effort and group work. Since attendance is not mandatory (see policies below), a significant number of low-scoring exercises will be dropped for each student. Please see me if you have 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. You are encouraged to discuss the problems and solution strategies *at a high level*, but the final solution and details must be your individual work. If you are unclear on the boundary between permissible and non-permissible interactions in this regard, please ask me.

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 is permitted *provided it does not cause a disturbance*. You may use the Internet, but only as a library to look up material you may find useful. As above, check with me if you are unclear on what is permitted. 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 meetings.

Project: In addition to the programming and other homeworks, this course features a semester-long independent project. You may work either individually or in groups, although I encourage the latter. The details of the project are fairly flexible, and you are encouraged to propose a project that excites you. I will also propose a few projects that you could use, perhaps with some of your own 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 the week before finals week*, but may be submitted earlier—there is no penalty for early submissions. Further details will follow.

Policies

Due dates: All due dates (and times) are strict, as announced in class. 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.*

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. 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 exam in any case).

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.

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 Java *source* files.

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 result should be a source-file package that works on all platforms supported by Java J2SE. 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 course so that we can go over 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.

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 required (see above), it is a good idea for everyone to have accounts on both our main Unix host (gandalf) and the cluster of PCs. Class accounts will be generated based on the forms distributed at the first class meeting. If you miss them, please get in touch with me. You should be able to access your gandalf account from anywhere on the Internet (including the labs in Neville Hall and elsewhere on campus) by using *ssh* to connect to `cs.umaine.edu`. On most Unix hosts, the command `ssh -l` *`username`* `cs.umaine.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 sections of the textbook covered in each class and those due at the next class. 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, 3rd edition, 2006. The university bookstore carries this book, which is a "required textbook" for this course. The edition is important.

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.

Other Readings: All the following are recommended, but not all are required. Further details and additional readings will appear here.

1. 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.

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

   This tutorial is optional reading but I strongly suggest that everyone read it. 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).

3. 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.

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

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

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

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

8. 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/200809/cos226/`.

Tuesday	Thursday
September 1st C1 Introduction; trees §18.0–18.3.	3rd C2 Traversals; binary search trees; order statistics; §18.4–18.end.,19.0–19.2.
8th C3 Analysis of algorithms; maximum contiguous subsequence; §5.0–5.3.	10th C4 Static searching; further analysis; §5.4–5.end.
15th C5 BST analysis, AVL trees; §19.3–19.4.	17th C6 ★ **Quiz 1**, regular class time & place.
22nd C7 Red-black trees; §19.5.	24th C8 AA-trees; §19.6.
29th C9 AA-trees; §19.6.	**October** 1st C10 B-trees; disk data structures; §19.7–19.end.
6th C11 ★ **Midterm Exam 1**, regular class time & place.	8th C12 Special tutorial.
13th ×*No class.* Fall break Oct. 10th–13th.	15th C13 AA-trees; B-trees; §19.6,19.8.
20th C14 B-trees; binary heaps; §19.8,21.1–21.3.	22nd C15 Splay trees; §22.1–22.2.
27th C16 Splay trees; §22.3–22.4.	29th C17 ★ **Quiz 2**, regular class time & place.
November 3rd C18 Skew heaps §23.1.	5th C19 Pairing heap; §23.2.
10th C20 Hashing; §20.1–20.4.	12th C21 Hashing; §20.5–20.7.
17th C22 ★ **Midterm Exam 2**, regular class time & place.	19th C23 Graphs; shortest paths; §14.1–14.3.
24th C24 Graphs; shortest paths; §14.4–14.5.	26th ×*No class.* Thanksgiving break Nov. 25th–29th.
December 1st C25 Sorting; §8.1–8.4.	3rd C26 Sorting; selection; §8.5–8.8.
8th C27 Review.	10th C28 Review.
15th ×*No class.* Finals week Dec.14th–18th.	17th ×*No class.* Finals week Dec.14th–18th. ★ **Final exam will be as scheduled by the University.**

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

- 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`.
  - Class Exercise 9: `hwq/ce09.pdf`.
  - Class Exercise 10: `hwq/ce10.pdf`.
  - Class Exercise 11: `hwq/ce11.pdf`.
  - Class Exercise 12: `hwq/ce12.pdf`.
  - Class Exercise 13: `hwq/ce13.pdf`.
  - Class Exercise 14: `hwq/ce14.pdf`.
  - Class Exercise 15: `hwq/ce15.pdf`.
  - Class Exercise 16: `hwq/ce16.pdf`.
  - Class Exercise 17: `hwq/ce17.pdf`.
  - Class Exercise 18: `hwq/ce18.pdf`.
  - Class Exercise 19: `hwq/ce19.pdf`.
- Homework assignments:
  - Homework 1: `hwq/hw01.pdf`.
  - Homework 2: `hwq/hw02.pdf`.
  - Homework 3: `hwq/hw03.pdf`.
  - Homework 4: `hwq/hw04.pdf`.
  - Homework 5: `hwq/hw05.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`;
- A few ideas for term projects: `notes/projideas.pdf`

Homework Submission

Handwritten answers to on 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 only the upload interface at `http://cs.umaine.edu/~chaw/u/`.

Sample Code

Sample code and other files will appear here.

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

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