

CS 473: Algorithms

Department of Computer Science
University of Illinois at Urbana-Champaign

Instructor: Jeff Erickson

Teaching Assistants:

- **Spring 1999:** Mitch Harris and Shripad Thite
- **Summer 1999 (IMCS):** Mitch Harris
- **Summer 2000 (IMCS):** Mitch Harris
- **Fall 2000:** Chris Neihengen, Ekta Manaktala, and Nick Hurlburt
- **Spring 2001:** Brian Ensink, Chris Neihengen, and Nick Hurlburt
- **Summer 2001 (I2CS):** Asha Seetharam and Dan Bullok
- **Fall 2002:** Erin Wolf, Gio Kao, Kevin Small, Michael Bond, Rishi Talreja, Rob McCann, and Yasutaka Furakawa
- **Spring 2004:** Dan Cranston, Johnathon Fischer, Kevin Milans, and Lan Chen
- **Fall 2005:** Erin Chambers, Igor Gammer, and Aditya Ramani
- **Fall 2006:** Dan Cranston, Nitish Korula, and Kevin Milans
- **Spring 2007:** Kevin Milans

© Copyright 1999–2007 Jeff Erickson. Last update January 12, 2007.

This work may be freely copied and distributed, either electronically or on paper.
It may not be sold for more than the actual cost of reproduction.

This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 2.5 License.
For license details, see <http://creativecommons.org/licenses/by-nc-sa/2.5/>.

For the most recent edition of this work, see <http://www.uiuc.edu/~jeffe/teaching/algorithms/>.

*You know, I could write a book.
And this book would be thick enough to stun an ox.*
— Laurie Anderson, “Let $X=X$ ”, *Big Science* (1982)

*I'm writing a book. I've got the page numbers done,
so now I just have to fill in the rest.*
— Stephen Wright

About These Notes

This course packet includes lecture notes, homework questions, and exam questions from algorithms courses I taught at the University of Illinois at Urbana-Champaign in Spring 1999, Fall 2000, Spring 2001, Fall 2002, Spring 2004, Fall 2005, and Fall 2006. These lecture notes and my videotaped lectures were also offered over the web in Summer 1999, Summer 2000, Summer 2001, Fall 2002, and Fall 2005 as part of the UIUC computer science department's online master's program.

I wrote a plurality of the lecture notes in Spring 1999; I revise them and add a few new notes every time I teach the course. The recurrences “pre-lecture” is based on a handout written by Ari Trachtenberg, based on a paper by George Lueker, from an earlier semester taught by Ed Reingold, but I have heavily revised and amended it. With the exception of Homework Zero, which is always my fault, most of the homework problems and their solutions were written by teaching assistants: Aditya Ramani, Asha Seetharam, Brian Ensink, Chris Neihengen, Dan Bullok, Dan Cranston, Johnathon Fischer, Ekta Manaktala, Erin Wolf Chambers, Igor Gammer, Gio Kao, Kevin Milans, Kevin Small, Lan Chen, Michael Bond, Mitch Harris, Nick Hurlburt, Nitish Korula, Rishi Talreja, Rob McCann, Shripad Thite, and Yasu Furakawa. “Johnny's” multi-colored crayon homework was found under the TA office door among the other Fall 2000 Homework 1 submissions.

Lecture notes were posted to the course web site a few days (on average) after each lecture. Homeworks, exams, and solutions were also distributed over the web. I have deliberately excluded homework and exam solutions from this course packet.

Prerequisites

For the most part, these notes assume that the reader has mastered the material covered in the first two years of a typical undergraduate computer science curriculum. (Mastery is not the same thing as ‘exposure’ or ‘a good grade’; hence, Homework Zero.) Specific prerequisites include:

- Proof techniques: direct proof, indirect proof, proof by contradiction, combinatorial proof, and induction (including its “strong”, “structural”, and “recursive” forms). Lecture 0 requires induction, and whenever Lecture $n - 1$ requires induction, so does Lecture n .
- Discrete mathematics: Boolean algebra, predicate logic, sets, functions, relations, recursive definitions, trees (as abstract objects, not just data structures), graphs
- Elementary discrete probability: expectation, linearity of expectation, independence
- Iterative programming concepts: variables, conditionals, iteration, subroutines, indirection (addresses/pointers/references), recursion. Programming experience in any language that supports pointers and recursion is a plus.

- Fundamental data structures: arrays/vectors, linked lists, search trees, heaps
- Fundamental abstract data types: dictionaries, stacks, queues, priority queues; the difference between this list and the previous list.
- Fundamental algorithms: searching, sorting (selection, insertion, merge, heap, quick, anything but bubble), tree traversal, depth/breadth first search
- Basic algorithm analysis: Asymptotic notation (o , O , Θ , Ω , ω), translating loops into sums and recursive calls into recurrences, evaluating simple sums and (linear and divide-and-conquer) recurrences.
- Mathematical maturity: facility with abstraction, formal (especially recursive) definitions, and (especially inductive) proofs; following mathematical arguments; recognizing syntactic, semantic, and/or logical nonsense; writing the former rather than the latter

In the future, I hope to write review notes that cover more of these topics. Alas, the future has not yet arrived.

Acknowledgments

The lecture notes, homeworks, and exams draw heavily on the following sources, all of which I can recommend as good references.

- Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974. (This was the textbook for the algorithms classes I took as an undergrad at Rice and as a masters student at UC Irvine.)
- Sara Baase and Allen Van Gelder. *Computer Algorithms: Introduction to Design and Analysis*. Addison-Wesley, 2000.
- Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 1997.
- Thomas Cormen, Charles Leiserson, Ron Rivest, and Cliff Stein. *Introduction to Algorithms*, second edition. MIT Press/McGraw-Hill, 2000. (I used the first edition of this book (without Stein) as a teaching assistant at Berkeley.)
- Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh V. Vazirani. *Algorithms*. McGraw-Hill, 2006.
- Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- Michael T. Goodrich and Roberto Tamassia. *Algorithm Design: Foundations, Analysis, and Internet Examples*. John Wiley & Sons, 2002.
- Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Molecular Biology*. Cambridge University Press, 1997.

- Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley, 2005. (This is my current recommended textbook.)
- Udi Manber. *Introduction to Algorithms: A Creative Approach*. Addison-Wesley, 1989. (I used this textbook as a teaching assistant at Berkeley.)
- Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- Ian Parberry. *Problems on Algorithms*. Prentice-Hall, 1995. (This book is out of print, but it can be downloaded for ‘free’ from <http://www.eng.unt.edu/ian/books/free/license.html> .)
- Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.
- Robert Sedgwick. *Algorithms*. Addison-Wesley, 1988. (This book and its sequels have by far the best algorithm *illustrations* I’ve seen anywhere.)
- Robert Endre Tarjan. *Data Structures and Network Algorithms*. SIAM, 1983.
- Robert J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 2001.
- Class notes from my own algorithms classes at Berkeley, especially those taught by Dick Karp and Raimund Seidel.
- Various journal papers, conference papers, surveys, and lecture notes from other algorithms researchers (cited in the notes).
- The Source of All Knowledge (Google) and The Source of All Lies (Wikipedia).

Naturally, these notes owe a great deal to the people who taught me this algorithms stuff in the first place: Bob Bixby and Michael Perlman at Rice; David Eppstein, Dan Hirshberg, and George Lueker at UC Irvine; and Abhiram Ranade, Dick Karp, Manuel Blum, Mike Luby, and Raimund Seidel at UC Berkeley. I’ve also been helped immensely by many discussions with faculty colleagues at UIUC—Chandra Chekuri, Edgar Ramos, Herbert Edelsbrunner, Jason Zych, Lenny Pitt, Mahesh Viswanathan, Shang-Hua Teng, Steve LaValle, and especially Ed Reingold and Sariel Har-Peled—as well as voluminous feedback from students and teaching assistants. I stole the overall course structure (and the idea to write up my own lecture notes) from Herbert Edelsbrunner.

Caveat Lector!

With a few exceptions, each of these notes contains way too much material to cover in a single lecture. In a typical 75-minute lecture, I tend to cover 4 to 5 pages of material—a bit more if I’m lecturing to grad students than to undergraduates. Your mileage may vary! (Arguably, that means that as I continue to add material, the label “lecture notes” becomes less and less accurate.)

Despite several rounds of revision, these notes and problems still contain lots of mistakes, errors, bugs, gaffes, omissions, snafus, kludges, typos, mathos, grammaros, thinkos, brain farts, nonsense, garbage, cruft, junk, and outright lies, all of which are entirely Steve Skiena’s fault. I revise and update these notes every time I teach the course, so please let me know if you find a bug. (Steve is unlikely to care.)

Whenever I teach the algorithms class, I award extra credit points to the first student to post an explanation and correction of any error in the lecture notes to the course newsgroup. Obviously, the number of extra credit points depends on the severity of the error and the quality of the correction. If I'm not teaching the course, encourage your instructor to set up a similar extra-credit scheme, and forward the bug reports to Steve me!

Of course, any other feedback is also welcome!

Enjoy!

— Jeff

It is traditional for the author to magnanimously accept the blame for whatever deficiencies remain. I don't. Any errors, deficiencies, or problems in this book are somebody else's fault, but I would appreciate knowing about them so as to determine who is to blame.

— Steven S. Skiena, *The Algorithm Design Manual* (1997)