Algorithms

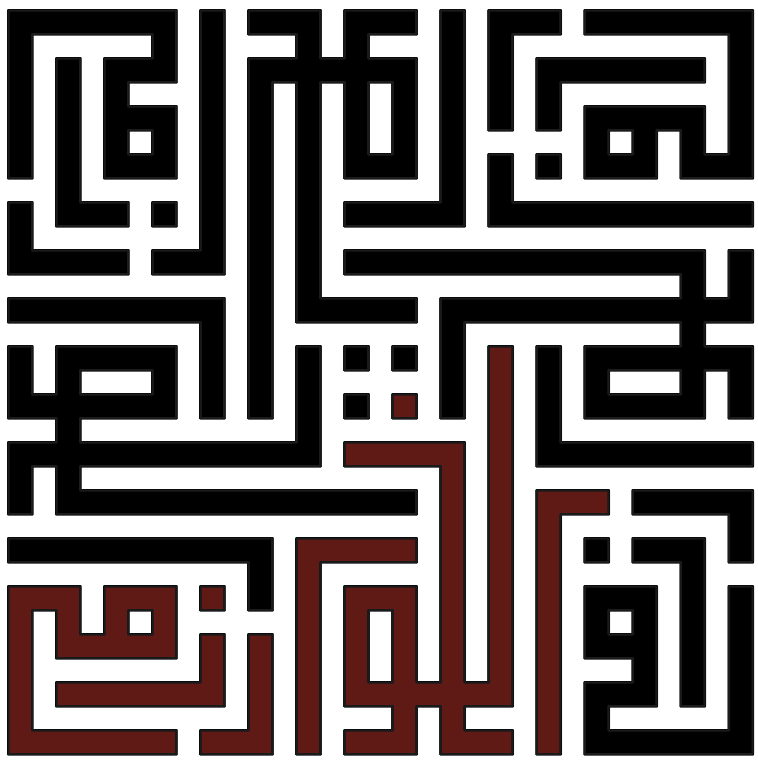
Department of Computer Science
University of Illinois at Urbana-Champaign

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
- **Fall 2008:** Reza Zamani-Nasab
- **Spring 2009:** Alina Ene, Ben Moseley, and Amir Nayyeri
- **Spring 2010:** David Morrison, Kyle Fox, and Rachit Agarwal
- **Fall 2010:** Alina Ene

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



Shall I tell you, my friend, how you will come to understand it?
Go and write a book on it.

— Henry Home, Lord Kames (1696–1782), to Sir Gilbert Elliot

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, Fall 2006, Spring 2007, Fall 2008, Spring 2009, Spring 2010, and Fall 2010. 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. 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.

Most (but not all) of the exercises at the end of each lecture note have been used at least once in a homework assignment, discussion section, or exam. You can also find a near-complete collection of homeworks and exams from past semesters of my class online at http://www.cs.illinois.edu/~jeffe/teaching/algorithms/. A large fraction of these exercises were contributed by some amazing teaching assistants:

> Aditya Ramani, Alina Ene, Amir Nayyeri, Asha Seetharam, Ben Moseley, Brian Ensink, Chris Neihengen, Dan Bullok, Dan Cranston, David Morrison, Johnathon Fischer, Ekta Manaktala, Erin Wolf Chambers, Igor Gammer, Gio Kao, Kevin Milans, Kevin Small, Kyle Fox, Lan Chen, Michael Bond, Mitch Harris, Nick Hurlburt, Nitish Korula, Rachit Agarwal, Reza Zamani-Nasab, Rishi Talreja, Rob McCann, Shripad Thite, and Yasu Furakawa.

*Stars indicate more challenging problems; many of these appeared on qualifying exams for the algorithms PhD students at UIUC. A small number of *really* hard problems are marked with a ★larger star; one or two *open* problems are indicated by ★enormous stars.

Please do not ask me for solutions to the exercises. If you're a student, seeing the solution will rob you of the experience of solving the problem yourself, which is the only way to learn the material. If you're an instructor, you shouldn't assign problems that you can't solve yourself! (I do not always follow my own advice; some of these problems have serious bugs.)

Acknowledgments

The lecture notes and exercises draw heavily on the creativity, wisdom, and experience of thousands of algorithms students, teachers, and researchers. In particular, I am immensely grateful to the almost 1400 Illinois students who have used these notes as a primary reference, offered useful (if sometimes painful) criticism, and suffered through some truly awful first drafts. I'm also grateful for the contributions and feedback from teaching assistants, all listed above.

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 tremendously by many discussions with faculty colleagues at UIUC—Edgar Ramos, Herbert Edelsbrunner, Jason Zych, Lenny Pitt, Mahesh Viswanathan, Margaret Fleck, Shang-Hua Teng, Steve LaValle, and especially Chandra Chekuri, Ed Reingold, and Sariel Har-Peled. I stole the first iteration of the overall course structure, and the idea to write up my own lecture notes, from Herbert Edelsbrunner.

Finally, "Johnny's" multi-colored crayon homework was found under the TA office door among the other Fall 2000 Homework 1 submissions. The square Kufi rendition of the name "al-Khwārizmī" on the back of the cover page is mine.

Additional References

I strongly encourage my students (and other readers) *not* to restrict themselves to a single textual reference. Authors and readers bring their own perspectives to the material; no instructor 'clicks' with every student, or even every very strong student. Finding the author that most effectively gets their intuition into *your* head take some effort, but that effort pays off handsomely in the long run.

The following references have been particularly valuable sources of inspiration, intuition, examples, and problems. This list is incomplete!

- Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974. (I used this textbook as an undergrad at Rice, and again as a masters student at UC Irvine.)
- Thomas Cormen, Charles Leiserson, Ron Rivest, and Cliff Stein. *Introduction to Algorithms*, third edition. MIT Press/McGraw-Hill, 2009. (The second edition was my recommended textbook until 2005. I also used the first edition as a teaching assistant at Berkeley.)
- Sanjoy Dasgupta, Christos H. Papadimitriou, and Umesh V. Vazirani. *Algorithms*. McGraw-Hill, 2006. (This is the current recommended textbook for my undergraduate classes.)
- Jeff Edmonds. *How to Think about Algorithms*. Cambridge University Press, 2008.
- 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.
- Jon Kleinberg and Éva Tardos. *Algorithm Design*. Addison-Wesley, 2005. (This is the current recommended textbook for my graduate algorithms classes.)
- Donald Knuth. *The Art of Computer Programming*, volumes 1–3. Addison-Wesley, 1997. (My parents gave me these for Christmas when I was 14. I didn't actually read them until *much* later.)

- 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 Sedgewick. *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.
- Lecture notes, slides, homeworks, exams, and video lectures posted by innumerable colleagues around the world.
- The Source of All Knowledge (Google) and The Source of All Lies (Wikipedia).

Prerequisites

For the most part, these notes assume the reader has mastered the material covered in the first two years of a strong undergraduate computer science curriculum, and has the intellectual maturity to recognize and repair any remaining gaps in their mastery. (Mastery is not the same thing as 'exposure' or 'a good grade'; this is why I start every semester with 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: High-school algebra, naive set theory, Boolean algebra, first-order predicate logic, sets, functions, relations, modular arithmetic, recursive definitions, trees (as abstract objects, not data structures), graphs.
- Elementary discrete probability: uniform vs non-uniform probability distributions, 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, linked lists, binary search trees, at least one balanced search tree (such as AVL trees, red-black trees, B-trees, skip lists, splay trees, or treaps), binary heaps.
- Fundamental abstract data types: dictionaries, stacks, queues, priority queues; the difference between this list and the previous list.
- Fundamental algorithms: elementary arithmetic, linear search, binary search, sorting (selection, insertion, merge-, heap-, quick-, radix, anything but bubble-), pre-/post-/inorder tree traversal.

- Basic algorithm analysis: Asymptotic notation (o, O, Θ, Ω, ω), translating loops into sums and recursive calls into recurrences, evaluating simple sums and 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.

Some of this prerequisite material is covered briefly in these notes, but more as a reminder than a good introduction.

Caveat Lector!

With few exceptions, each of these notes contains far 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 graduate 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 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)