COS 226, SPRING 2012

Algorithms and Data Structures

Kevin Wayne

http://www.princeton.edu/~cos226

Course Overview

- outline
- why study algorithms?
- usual suspects
- coursework
- resources

What is COS 226?

- Intermediate-level survey course.
- Programming and problem solving, with applications.
- Algorithm: method for solving a problem.
- Data structure: method to store information.

topic	data structures and algorithms
data types	stack, queue, bag, union-find, priority queue
sorting	quicksort, mergesort, heapsort, radix sorts
searching	BST, red-black BST, hash table
graphs	BFS, DFS, Prim, Kruskal, Dijkstra
strings	KMP, regular expressions, TST, Huffman, LZW
advanced	B-tree, suffix array, maxflow, simplex

Why study algorithms?

Their impact is broad and far-reaching.

Internet. Web search, packet routing, distributed file sharing, ...

Biology. Human genome project, protein folding, ...

Computers. Circuit layout, file system, compilers, ...

Computer graphics. Movies, video games, virtual reality, ...

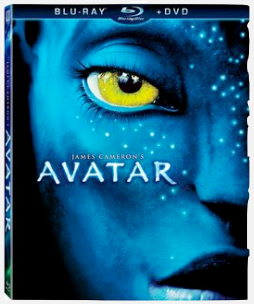
Security. Cell phones, e-commerce, voting machines, ...

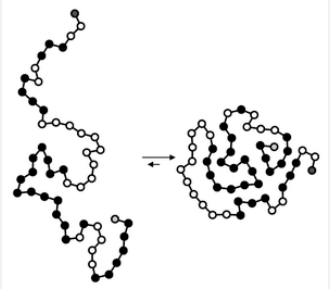
Multimedia. MP3, JPG, DivX, HDTV, face recognition, ...

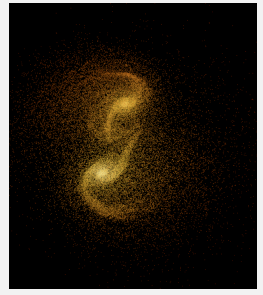
Social networks. Recommendations, news feeds, advertisements, ...

Physics. N-body simulation, particle collision simulation, ...

⋮



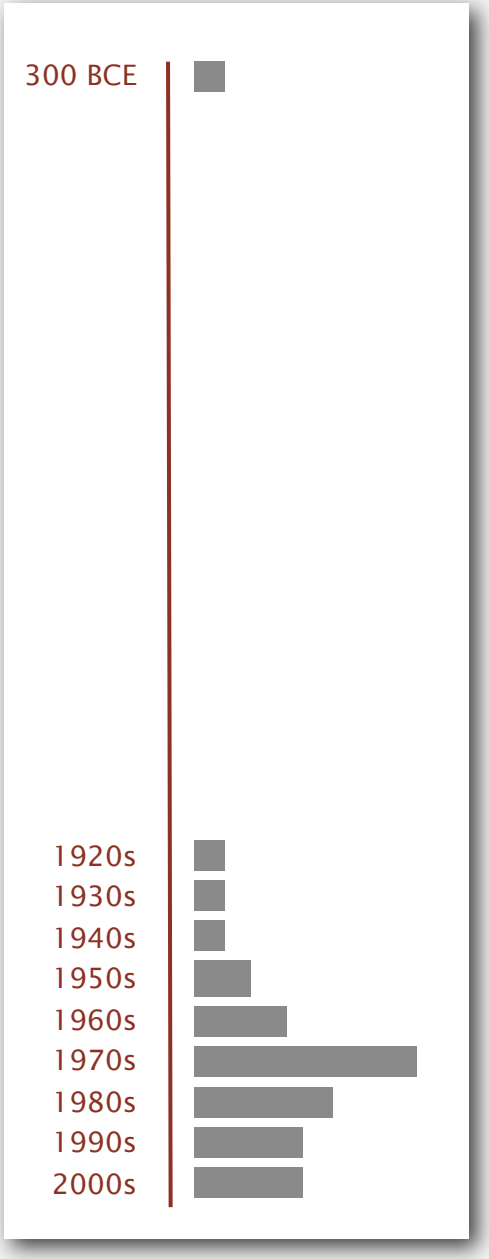




Why study algorithms?

Old roots, new opportunities.

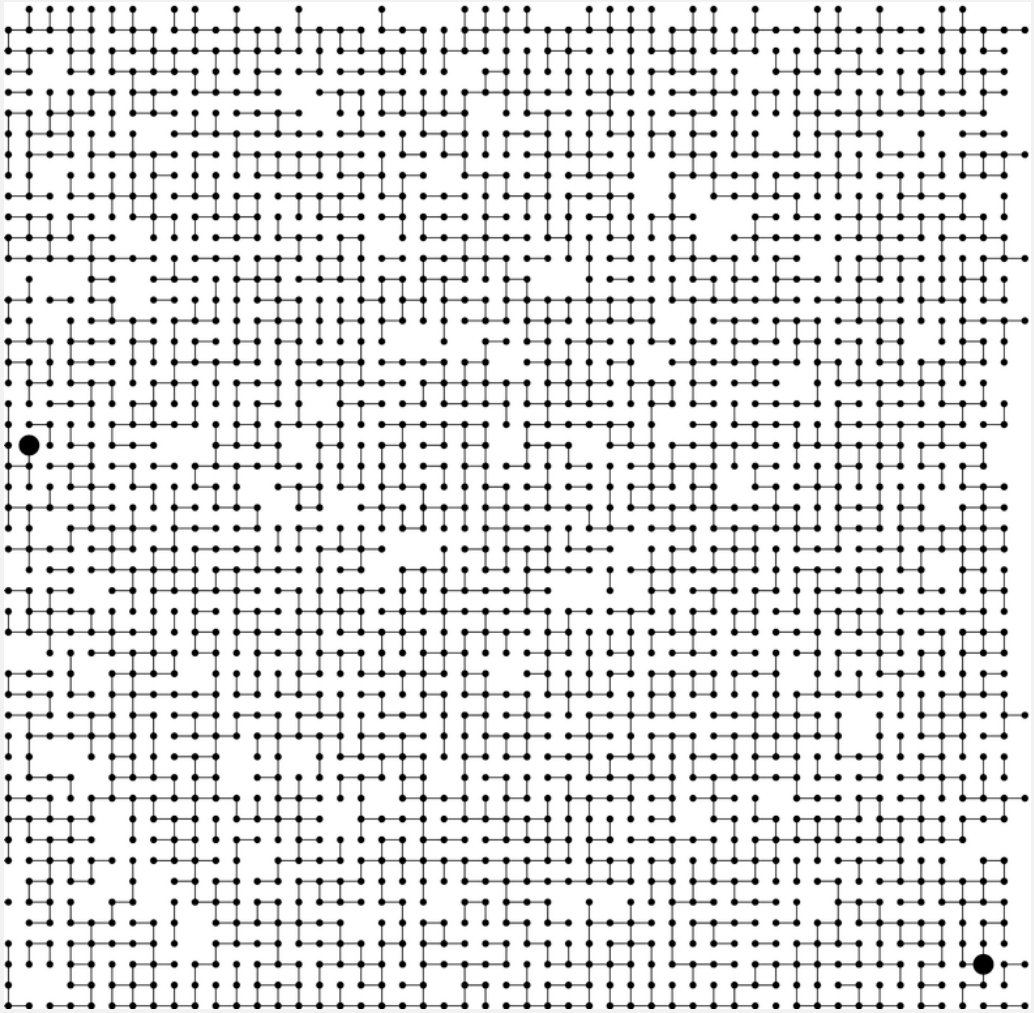
- Study of algorithms dates at least to Euclid.
- Formalized by Church and Turing in 1930s.
- Some important algorithms were discovered by undergraduates in a course like this!



Why study algorithms?

To solve problems that could not otherwise be addressed.

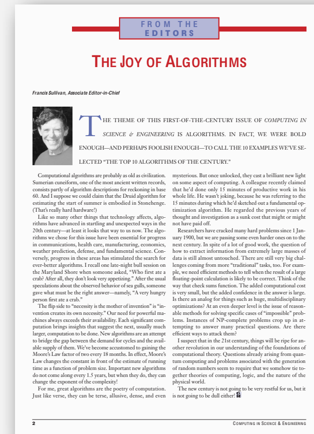
Ex. Network connectivity. [stay tuned]



Why study algorithms?

For intellectual stimulation.

> " For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing. " — Francis Sullivan



FROM THE EDITORS

THE JOY OF ALGORITHMS

Francis Sullivan, Associate Editor-in-Chief

THE THEME OF THIS FIRST-OF-THE-CENTURY ISSUE OF *COMPUTING IN SCIENCE & ENGINEERING* IS ALGORITHMS. IN FACT, WE WERE BOLD ENOUGH—AND PERHAPS FOOLISH ENOUGH—TO CALL THE 10 EXAMPLES WE'VE SELECTED "THE TOP 10 ALGORITHMS OF THE CENTURY."

Computational algorithms are probably as old as civilization. Sumerian cuneiform, one of the most ancient written records, consists partly of algorithm descriptions for reckoning in base 60. And I suppose we could claim that the Druid algorithm for estimating the start of summer is embodied in Stonehenge. (That's really hard hardware!)

Like so many other things that technology affects, algorithms have advanced in startling and unexpected ways in the 20th century—at least it looks that way to us now. The algorithms we chose for this issue have been essential for progress in communications, health care, manufacturing, economics, weather prediction, defense, and fundamental science. Conversely, progress in these areas has stimulated the search for ever-better algorithms. I recall one late-night bull session on the Maryland Shore when someone asked, "Who first ate a crab? After all, they don't look very appetizing." After the usual speculations about the observed behavior of sea gulls, someone gave what must be the right answer—namely, "A very hungry person first ate a crab."

The flip side to "necessity is the mother of invention" is "invention creates its own necessity." Our need for powerful machines always exceeds their availability. Each significant computation brings insights that suggest the next, usually much larger, computation to be done. New algorithms are an attempt to bridge the gap between the demand for cycles and the available supply of them. We've become accustomed to gaining the Moore's Law factor of two every 18 months. In effect, Moore's Law changes the constant in front of the estimate of running time as a function of problem size. Important new algorithms do not come along every 1.5 years, but when they do, they can change the exponent of the complexity!

For me, great algorithms are the poetry of computation. Just like verse, they can be terse, allusive, dense, and even mysterious. But once unlocked, they cast a brilliant new light on some aspect of computing. A colleague recently claimed that he'd done only 15 minutes of productive work in his whole life. He wasn't joking, because he was referring to the 15 minutes during which he'd sketched out a fundamental optimization algorithm. He regarded the previous years of thought and investigation as a sunk cost that might or might not have paid off.

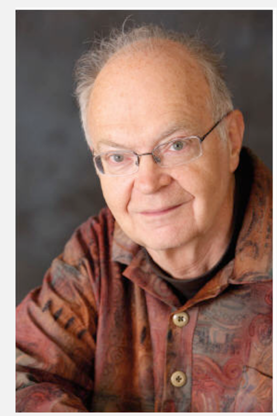
Researchers have cracked many hard problems since 1 January 1900, but we are passing some even harder ones on to the next century. In spite of a lot of good work, the question of how to extract information from extremely large masses of data is still almost untouched. There are still very big challenges coming from more "traditional" tasks, too. For example, we need efficient methods to tell when the result of a large floating-point calculation is likely to be correct. Think of the way that check sums function. The added computational cost is very small, but the added confidence in the answer is large. Is there an analog for things such as huge, multidisciplinary optimizations? At an even deeper level is the issue of reasonable methods for solving specific cases of "impossible" problems. Instances of NP-complete problems crop up in attempting to answer many practical questions. Are there efficient ways to attack them?

I suspect that in the 21st century, things will be ripe for another revolution in our understanding of the foundations of computational theory. Questions already arising from quantum computing and problems associated with the generation of random numbers seem to require that we somehow tie together theories of computing, logic, and the nature of the physical world.

The new century is not going to be very restful for us, but it is not going to be dull either!

2 COMPUTING IN SCIENCE & ENGINEERING

> " An algorithm must be seen to be believed. " — Donald Knuth



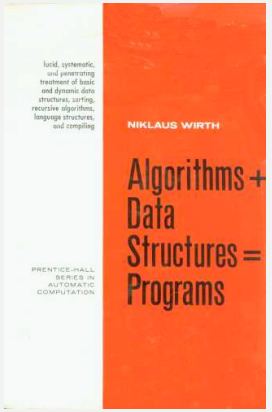
Why study algorithms?

To become a proficient programmer.

" I will, in fact, claim that the difference between a bad programmer and a good one is whether he considers his code or his data structures more important. Bad programmers worry about the code. Good programmers worry about data structures and their relationships. "

— Linus Torvalds (creator of Linux)

" Algorithms + Data Structures = Programs. " *— Niklaus Wirth*



Why study algorithms?

They may unlock the secrets of life and of the universe.

Computational models are replacing mathematical models in scientific inquiry.

$$E = mc^2$$

$$F = ma \qquad F = \frac{Gm_1m_2}{r^2}$$

$$\left[-\frac{\hbar^2}{2m}\nabla^2 + V(r)\right]\Psi(r) = E\,\Psi(r)$$

20th century science
(formula based)

```
for (double t = 0.0; true; t = t + dt)
   for (int i = 0; i < N; i++)
   {
      bodies[i].resetForce();
      for (int j = 0; j < N; j++)
         if (i != j)
            bodies[i].addForce(bodies[j]);
   }
```

21st century science
(algorithm based)

" Algorithms: a common language for nature, human, and computer. " — *Avi Wigderson*

Why study algorithms?

For fun and profit.

Why study algorithms?

- Their impact is broad and far-reaching.
- Old roots, new opportunities.
- To solve problems that could not otherwise be addressed.
- For intellectual stimulation.
- To become a proficient programmer.
- They may unlock the secrets of life and of the universe.
- For fun and profit.

The usual suspects

Lectures. Introduce new material.

Precepts. Discussion, problem-solving, background for programming assignment.

What	When	Where	Who
L01	MW 11–12:20	Robertson 100	Kevin Wayne
P01	Th 12:30–1:20	Friend 112	Diego Botero
P01A	Th 12:30–1:20	Sherrerd 101	Dave Shue
P01B	Th 12:30–1:20	Friend 008	Joey Dodds
P02	Th 1:30–2:20	Sherrerd 101	Josh Hug †
P03	Th 3:30–4:20	Friend 108	Josh Hug †
P04	F 11–11:50	Friend 112	Joey Dodds
P04A	F 11–11:50	CS 102	Jacopo Cesareo

† lead preceptor

Where to get help?

Piazza. Online discussion forum.

- Low latency, low bandwidth.
- Mark solution-revealing questions as private.

http://www.piazza.com/class#spring2012/cos226

Office hours.

- High bandwidth, high latency.
- See web for schedule.

http://www.princeton.edu/~cos226

Computing laboratory.

- Undergrad lab TAs in Friend 017.
- For help with debugging.
- See web for schedule.

http://www.princeton.edu/~cos226

Coursework and grading

Programming assignments. 45%

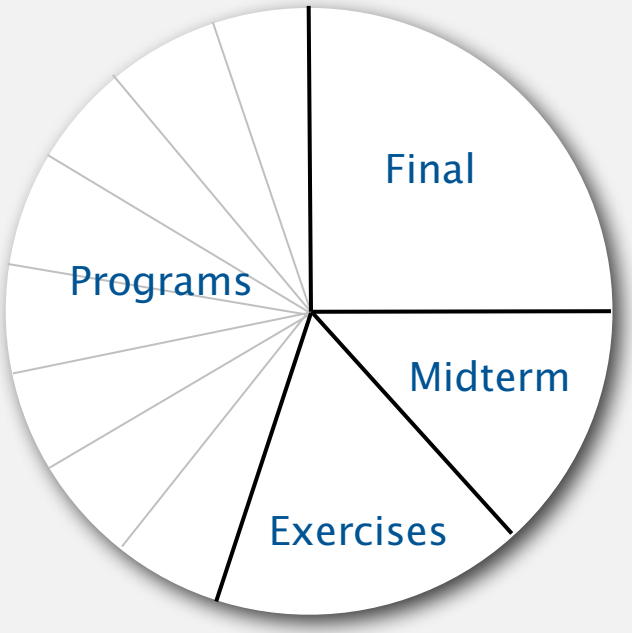
- Due on Tuesdays at 11pm via electronic submission.
- Collaboration/lateness policies: see web.

Written exercises. 15%

- Due on Mondays at 11am in lecture.
- Collaboration/lateness policies: see web.

Exams. 15% + 25%

- Midterm (in class on Monday, March 12).
- Final (to be scheduled by Registrar).

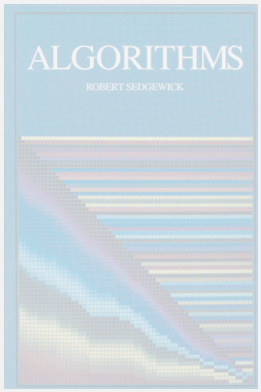


Staff discretion. To adjust borderline cases.

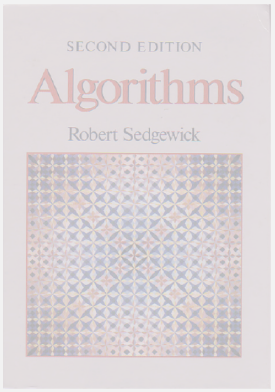
- Report errata.
- Contribute to Piazza discussions.
- Attend and participate in precept/lecture.

Resources (textbook)

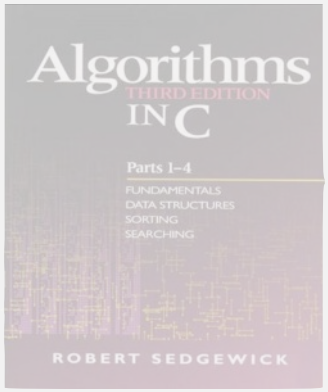
Required reading. Algorithms 4th edition by R. Sedgewick and K. Wayne, Addison-Wesley Professional, 2011, ISBN 0-321-57351-X.



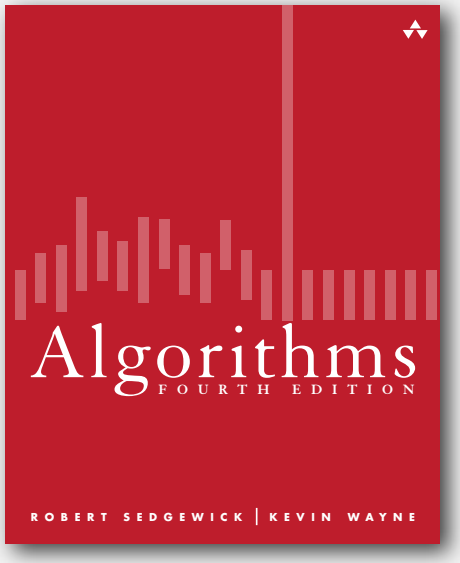
1st edition (1982)



2nd edition (1988)



3rd edition (1997)



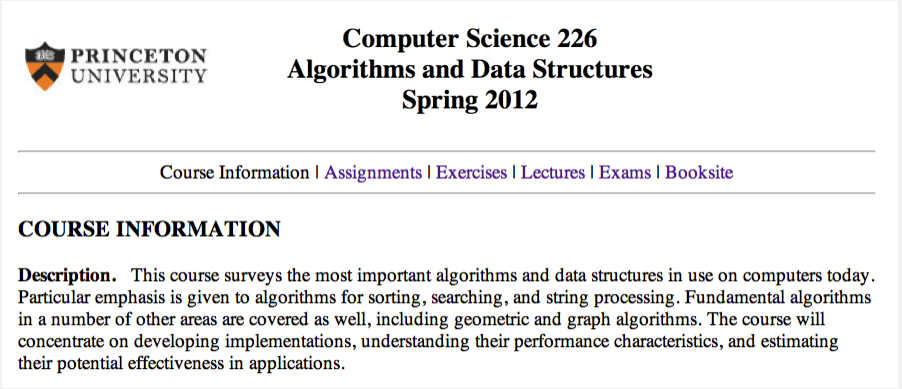
Available in hardcover and Kindle.

- Online: Amazon ($60 to buy), Chegg ($40 to rent), ...
- Brick-and-mortar: Labyrinth Books (122 Nassau St). ← 30% discount with PU student ID
- On reserve: Engineering library.

Resources (web)

Course content.

- Course info.
- Programming assignments.
- Exercises.
- Lecture slides.
- Exam archive.
- Submit assignments.



PRINCETON UNIVERSITY

Computer Science 226
Algorithms and Data Structures
Spring 2012

Course Information | Assignments | Exercises | Lectures | Exams | Booksite

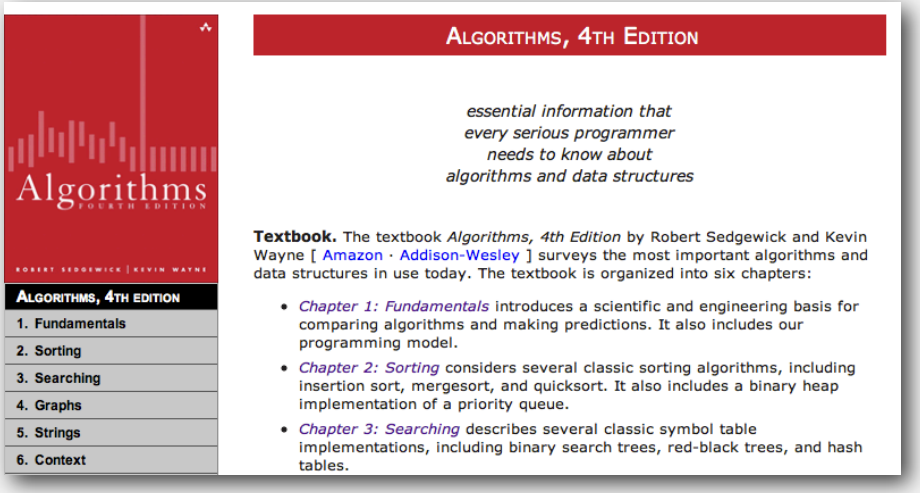
COURSE INFORMATION

Description. This course surveys the most important algorithms and data structures in use on computers today. Particular emphasis is given to algorithms for sorting, searching, and string processing. Fundamental algorithms in a number of other areas are covered as well, including geometric and graph algorithms. The course will concentrate on developing implementations, understanding their performance characteristics, and estimating their potential effectiveness in applications.

http://www.princeton.edu/~cos226

Booksites.

- Brief summary of content.
- Download code from book.



Algorithms
FOURTH EDITION
ROBERT SEDGEWICK | KEVIN WAYNE

ALGORITHMS, 4TH EDITION

1. Fundamentals
2. Sorting
3. Searching
4. Graphs
5. Strings
6. Context

ALGORITHMS, 4TH EDITION

essential information that every serious programmer needs to know about algorithms and data structures

Textbook. The textbook *Algorithms, 4th Edition* by Robert Sedgewick and Kevin Wayne [Amazon · Addison-Wesley] surveys the most important algorithms and data structures in use today. The textbook is organized into six chapters:

- *Chapter 1: Fundamentals* introduces a scientific and engineering basis for comparing algorithms and making predictions. It also includes our programming model.
- *Chapter 2: Sorting* considers several classic sorting algorithms, including insertion sort, mergesort, and quicksort. It also includes a binary heap implementation of a priority queue.
- *Chapter 3: Searching* describes several classic symbol table implementations, including binary search trees, red-black trees, and hash tables.

http://www.algs4.princeton.edu

What's ahead?

Lecture 1. Union find. ← today

Lecture 2. Analysis of algorithms. ← Wednesday

Precept 1. Meets this week. ← Thursday or Friday

Exercises 1 + 2. Due via hardcopy in lecture at 11am on Monday.

Assignment 1. Due via electronic submission at 11pm on Tuesday.

Right course? See me.

Placed out of COS 126? Review Sections 1.1–1.2 of *Algorithms*, 4th edition (includes command-line interface and our I/O libraries).

Not registered? Go to any precept this week.

Change precept? Use SCORE. ← see Colleen Kenny-McGinley in CS 210 if the only precept you can attend is closed