# Enrichment Reading

In this page we give some suggestions for further reading for each lecture. These readings are entirely optional and some of them may be very challenging.

- Lecture 1:
  - For more about the algorithm "PIERCE" see this paper. Vlado Keselj has the best bound known.
  - Jon Bentley's two books, *Programming Pearls* and *More Programming Pearls* are essential reading for any programmer.
- Lecture 2:
  - Many people have studied the subrange sum problem, creating efficient algorithms and generalizations. For example, look at Brodal and Jorgensen, A linear time algorithm for the k maximal sums problem, MFCS 2007, pp. 442-453 and Fan, Lee, et al., An optimal algorithm for maximum-sum segment and its application in bioinformatics, CIAA 2003, pp. 252-257.
- Lecture 3:
  - There are classic papers on the RAM model, where it is proved that the log-cost model is polynomially-equivalent to a Turing machine. See, for example, Stephen A. Cook and Robert A. Reckhow, Time-bounded random access machines, *Journal of Computer Systems Science* **7** (1973), 354-375.
- Lecture 4:
  - For more about the discrete logarithm, this survey by Andrew Odlyzko, although a little out of date, is useful.
  - For more about the convex hull problem, see this paper, which discusses an algorithm by Waterloo's Timothy Chan.
- Lecture 5:
  - Probably the best reference on algorithms to enumerate combinatorial objects is still the old book of Nijenhuis and Wilf, *Combinatorial Algorithms*.
  - Frank Ruskey has a page here to generate combinatorial objects of many types.
- Lecture 6:
  - It's actually possible to multiply two n-bit numbers in O(n log n log log n) time, but the algorithm is not simple. The book of Crandall and Pomerance, *Prime Numbers - A Computational Perspective*, is probably the best place to start.
  - In 2007, Martin Fürer found an even faster way to multiply numbers here. But again, the algorithm is not simple.
  - Recently, Umans and Cohn found a new approach to matrix multiplication that may eventually lead to much faster algorithms.
- Lecture 8:
  - For more about the number of possible ways to parenthesize things, see this paper by Guy and Selfridge with the amusing title, "The Nesting and Roosting Habits of The Laddered Parenthesis".
  - Our dynamic programming algorithm for finding the optimal multiplication order uses $\Theta(n^3)$ time if there are n matrices. But this can be improved to O(n log n). See this paper by Hu and Shing in *SIAM J. Comput.* **13** (1984), 228-251.
- Lecture 10:
  - In 1997, Cummings and Smyth found a better algorithm for abelian squares than the one we presented here, in that it does not depend on the alphabet size, but unfortunately it is still $\Omega(n^2)$. The lower bound "proof" in the paper is apparently incorrect and we don't know how to repair it.

- Writing x = 200a + 100b + 25c + 10d + 5e + f where a, b, c, d, e, f are integers ≥ 0, and such that a+b+c+d+e+f is minimized, is an example of an *integer program*. Unfortunately, integer programming in general is NP-hard, so there is no general fast way known to solve systems like this.
- This paper on change-making might amuse you.

- Lecture 11:
  - One of the best sources for papers about Egyptian fractions is the book of Richard Guy, *Unsolved Problems in Number Theory*. See Section D11.
  - For more about Egyptian fractions, see David Eppstein's page.

- Lecture 12:
  - For the story of Finck and his little-known analysis of the Euclidean algorithm, see here. (J. Shallit, Origins of the analysis of the Euclidean algorithm, *Historia Mathematica* **21** (1994), 401-419.)

- Lecture 12:
  - The Mersenne twister algorithm is a good way to generate pseudorandom numbers.

- Lectures 13-14:
  - For a different perspective on some of our lower bound arguments, see Eric Bach, Energy arguments in the theory of algorithms, **104** (1997), 831-837. Perhaps you can read it here.

- Lectures 15-16:
  - Prim's algorithm dates from 1957. In 1986, Gabow, Galil, and Spencer found a faster algorithm for minimum spanning tree: it runs in O(|E| log* (|E|/|V|)) time, where log* is the slowly-growing function that gives the number of logs needed to reduce the argument to < 1. See here. In 1990, Fredman and Willard found a minimum-spanning tree algorithm that runs in linear time, but it depends on operations on the binary representation of the edge weights. See here. In 1996, Karger, Klein, and Tarjan found a randomized algorithm that runs in O(|E|) expected time and uses only comparisons between the edge weights. See here.

- Lecture 17:
  - Shortest-path algorithms continue to attract attention. At the FOCS 2010 conference, Peres et al. presented a paper showing you can find shortest paths in n-vertex graphs where the edge weights are chosen uniformly and independently at random from [0, 1] in about $O(n^2)$ expected time. See here for a video of the talk.

- Lectures 19-23:
  - The lower bound on extended regular expression equivalence is due to Meyer and Stockmeyer, and can be found here.
  - This page maintained by Gerhard Woeginger is a sad list of people who have claimed to resolve P versus NP. Needless to say, none of their proofs have been accepted by the general computer science community.
  - The standard reference for NP-completeness is Garey and Johnson, *Computers and Intractibility*. Johnson wrote a column for many years in the *Journal of Algorithms*, and they can be found here.

- Lectures 23-24:
  - For more about Hilbert's 10th problem, see *Hilbert's Tenth Problem* by Yuri Matijasevich, Davis Centre library, QA242.M4213 1993.
  - For more about Gödel's proof, see *Gödel's Proof* by Nagel and Newman, Davis Centre library, QA9.65.N34 2001.
  - For a biography of Gödel, see Rebecca Goldstein, *Incompleteness*.
  - For more about the life of Alan Turing, see *Alan Turing: The Enigma* by Hodges, Davis Centre library, QA29.T87H64x 1983 or this website maintained by Hodges.

- For more about FRACTRAN, and the prime-producing program, see Guy's article, "Conway's Prime Producing Machine" *Math. Mag.* **56** (1983), 26-33. You may be able to access it here.