Syllabus for Introduction to Algorithms
Summer 2019

Adam Frank
Bard Prison Initiative

June 3, 2019

1 The Goal of this Course

The goal of this course is to make you self-sufficient at understanding, analyzing, using, and even designing your own algorithms. No one course can get a student all the way there, Algorithms is too large and difficult a subject, and of course a field of active research. Nobody can be said to fully understand Algorithms, since the subject is beyond the power of any one person.

I hope that this course will equip you with

- the tools necessary to carry on your studies as far as you need them for your own personal goals.
- the capacity to understand other courses which require basic knowledge of Algorithms as a prerequisite.

2 Prerequisites

This course requires you to already have a *very* strong foundation in two topics: Programming in Java and Discrete Mathematics. This course will spend little to no time explaining Java programming, and you will be expected to complete Java code that builds complex projects.

This course will also depend on substantial fluency with mathematics. The most important mathematics we will handle includes Advanced Algebra, Combinatorics, Proofs, Graph Theory, Probability, and modular arithmetic. A course in Discrete Mathematics should prepare a student adequately in each of these, perhaps also supplemented by a course in Probability. It is helpful although not necessary for the student to understand modular arithmetic from a Group- and Field-Theoretic perspective, such as one would encounter in a course on Abstract Algebra. You will want to understand divisibility, the GCD, and Bezout's identity at least a little bit before those topics are introduced in the notes and in the textbook.

3 What We'll Study

3.1 Algorithm Analysis, Searching and Sorting

Algorithm analysis (run-time, asymptotic analysis, proof of correctness), LinearSearch, BinarySearch, SelectionSort, InsertionSort, MergeSort, QuickSort, Data structures, invariants, LinkedList, ArrayList, The Master Theorem.

3.2 Algorithm Strategies on Lists

Dynamic programming, greedy algorithms, amortized analysis, the selection problem.

3.3 Graph Algorithms

DFS, BFS, Trees, Stacks, Queues, Records, Prim's, Kruskal's, Dijkstra's, Bellman-Ford's, Topological sort, Kahn's, SCCs, Kosaraju's.

3.4 Graph Invariants

BSTs, Minheaps, 2-3 Trees

3.5 Hashing and Compression Algorithms

Huffman coding, Hash functions, Probability, Modular Arithmetic, RSS, Rabin-Karp

3.6 Radix Representation, Fast Multiplication

Radix sort, Karatsuba's, FFT, Complex Numbers

3.7 NP-Completeness

P, NP, NP-Complete, NP-Hard, 2-SAT, 3-SAT, SAT, CIRCUIT-SAT, Hamiltonian Paths, TSP.

4 Textbook

Cormen, Thomas, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms. 3rd ed.* MIT Press, 2009. ISBN: 9780262033848.

I will work from my own lecture notes, but these follow the course textbook, the notorious CLRS. Hopefully between the clear but technical presentation in the textbook, and the typo-riddled but more conversational lecture notes, and the hurried class lectures, the material will become clear.

5 Schedule

If we are extremely ambitious (even ambitious beyond all reasonable expectations) we will spend one week on each topic mentioned in the "What We'll Study" section above. With eight weeks of classroom meetings, that leaves one week for slack. Much more likely is that we won't get to everything. Topics I'll most quickly choose to drop are, in the order in which I'll first decide to drop them: FFT and complex numbers, the selection problem, 2-3 trees, Radix sort, Karatsuba's. Everything else, we just have to get to somehow.

Classes meet during the time PM slot, 1 to 3:30, on SOME Fridays. In particular, the days we meet are June 14th, 21st, 28th, and July 5th. NOT July 12th, but then July 19th, 26th, August 2nd, and August 9th.

6 Grading

The lecture notes contain the assignments (sections of the notes labelled "Exercise" and colored green). These are intended to be completed and submitted for grading on a weekly basis. There is no formal due date other than submission on or before the last day of class. However, it behooves you to finish these early in order to receive feedback. Not only that, but you have an unlimited number of available re-submissions to improve your grade without penalty.

Each question is worth 10 points. You earn 5 points if you demonstrate a significant amount of understanding. You earn an additional 3 points if, in addition, your answer is correct. You earn an additional 2 points if, in addition, your answer is well-written. Demonstrating understanding is a squishy idea but I will do my best to apply it consistently. It generally means that even if your code won't compile or you may have missed some steps in a proof, I can see that you have a mostly complete mental picture of the concepts being exercised. "Correct" is of course a firmer idea: Either your code compiles with the right inputs and corresponding outputs, or it doesn't. Either your proof states true sentences with valid inferences, or it doesn't. "Written well" is again squishy but it amounts to having clean hand-writing, well-commented code, explained in short but precise prose.

Programming exercises in Java will be distributed with code stubs. You are to complete the code stubs, print your code, and hand it in. You should include the main class with print lines demonstrating some tests that you performed to ensure the correctness of your implementations. You will not be graded for the actual run-time of your algorithms, as I have not provided code stubs that will accomplish the theoretically minimal run-time on these data structures. I've instead opted for simpler and easier to read code stubs that in practice run slower than they theoretically could. (In actual practice you wouldn't typically implement these data structures yourself anyway. You'd typically import a library and make use of it as a user and not as a designer. So I don't believe much is lost in this trade-off, relative to what is gained.)

I'll distribute a mid-term on or before July 5th and a final exam on or before August 9th. The same rules that apply for weekly written and programming assignments also apply to exams, except that each question is worth 200 points. The relative distribution of points for understanding, correctness, and writing style are the same. Hence for a single exam question, merely demonstrating understanding earns half, or 100 points. Correctness earns another three-tenths or 60 points. Writing style earns 40 points.

There is an option for a final project if you choose to do so. It is worth a bonus 200 points. To earn these points or some portion of them, select a topic we will not cover in class, but which is relevant to Algorithms and Data Structures. Implement the concept in a Java program. You can select any project of your choosing as long as you get approval from the instructor (aka *moi*). Some ideas include:

- Implementing pointers and reference in a language that doesn't support them (in Java we will just pretend that Java doesn't have them and build them from scratch).

- Implement an approximation of the TSP.
- Build a webpage class to store in webpages as nodes of a graph, and compute a graph algorithm based on links between pages.
- Build a network simulation in Java and implement some basic network algorithms.
- Implement a red-black tree.
- Implement a disjoint set.
- Implement fast primality testing.

A score greater than 90% of the possible points (excluding the bonus from a project) is an A. A score between 80% and 90% is a B. Between 70% and 80% a C. Between 60% and 70% a D. Anything else is an F.