Schedule

Overview

Each Monday at 12:01am EDT, we will release the course materials for the week: two lectures, two sets of exercises, a programming assignment (in weeks 1, 2, 3, and 5), and two sets of job interview questions.

- Exercises: due 14 days after they are released.
- Programming assignments: due 14 days after they are released.
- Job interview questions: ungraded and for your own enrichment.

In the final week, there is a final exam.

Week 1

We begin our exploration of advanced algorithms by looking at data structures for graphs, which are essential models for a broad variety of real-world problems. A number of classic algorithms are easy to implement, but worthy of careful study both because convincing oneself that they operate as intended can be a challenge and because their performance is strongly dependent on proper use of the basic data structures that we covered in Algorithms Part I.

Lecture: Undirected Graphs. We define an undirected graph API and consider the adjacency-matrix and adjacency-lists representations. We introduce two classic algorithms for searching a graph—depth-first search and breadth-first search. We also consider the problem of computing connected components and conclude with related problems and applications.

Lecture: Directed Graphs. In this lecture we study directed graphs. We begin with depth-first search and breadth-first search in digraphs and describe applications ranging from garbage collection to web crawling. Next, we introduce a depth-first search based algorithm for computing the topological order of an acyclic digraph. Finally, we implement the Kosaraju-Sharir algorithm for computing the strong components of a digraph.

Exercises. Drill exercises on the lecture material.

Programming Assignment: WordNet. Determine the semantic relatedness of two nouns using the WordNet lexicon.

Job Interview Questions. Algorithmic interview questions based on the lecture material.

Suggested readings. Section 4.1 and 4.2 in *Algorithms, 4th edition.*

Week 2

This week is devoted to two classic graph problems that have found extensive applications for decades. While the basic algorithms are easy to understand, efficient implementations in practical situations typically require deployment of data structures such as the priority queue and union-find structures that we considered in Part I. We also consider the fact that simple variations can lead to deep and difficult computational problems.

Lecture: Minimum Spanning Trees. In this lecture we study the minimum spanning tree problem. We begin by considering a generic greedy algorithm for the problem. Next, we consider and implement two classic algorithm for the problem—Kruskal's algorithm and Prim's algorithm. We conclude with some applications and open problems.

Lecture: Shortest Paths. In this lecture we study shortest-paths problems. We begin by analyzing some basic properties of shortest paths and a generic algorithm for the problem. We introduce and analyze Dijkstra's algorithm for shortest-paths problems with nonnegative weights. Next, we consider an even faster algorithm for DAGs, which works even if the weights are negative. We conclude with the Bellman-Ford-Moore algorithm for edge-weighted digraphs with no negative cycles. We also consider applications ranging from content-aware fill to arbitrage.

Exercises. Drill exercises on the lecture material.

Programming Assignment: Content-Aware Resizing. Implement a seam-carving algorithm for content-aware resizing.

Job Interview Questions. Algorithmic interview questions based on the lecture material.

Suggested readings. Section 4.3 and 4.4 in *Algorithms, 4th edition.*

Week 3

This week we finish our study of graph algorithms by considering the classic maxflow and mincut problems, two problem-solving models that are useful in a broad variety of applications. Then we pivot to begin our study of string-processing algorithms, starting with substantially improved algorithms for sorting in the case when keys are strings.

Lecture: Maximum Flow and Minimum Cut. In this lecture we introduce the maximum flow and minimum cut problems. We begin with the Ford-Fulkerson algorithm. To analyze its correctness, we establish the maxflow-mincut theorem. Next, we consider an efficient implementation of the Ford-Fulkerson algorithm, using the shortest augmenting path rule. Finally, we consider applications, including bipartite matching and baseball elimination.

Lecture: Radix Sorts. In this lecture we consider specialized sorting algorithms for strings and related objects. We begin with a subroutine to sort integers in a small range. We then consider two classic radix sorting algorithms—LSD and MSD radix sorts. Next, we consider an especially efficient variant, which is a hybrid of MSD radix sort and quicksort known as 3-way radix quicksort. We conclude with suffix sorting and related applications.

Exercises. Drill exercises on the lecture material.

Programming Assignment: Baseball Elimination. Given the standings in a sports division at some point during the season, determine which teams have been mathematically eliminated from winning their division.

Job Interview Questions. Algorithmic interview questions based on the lecture material.

Suggested readings. Section 6.4 (pp. 886-902) and 5.1 in *Algorithms, 4th edition*.

Week 4

This week, we consider two different search problems involving strings. In the first lecture, we build upon the ideas we considered for string sorts to develop string search methods that are even faster than hashing and even more flexible than binary search trees. In the second lecture we consider classic algorithms for the substring search problem, where the goal is to find a given substring in a large text.

Lecture: Tries. In this lecture we consider specialized algorithms for symbol tables with string keys. Our goal is a data structure that is as fast as hashing and even more flexible than binary search trees. We begin with multiway tries; next we consider ternary search tries. Finally, we consider character-based operations, including prefix match and longest prefix, and related applications.

Lecture: Substring Search. In this lecture we consider algorithms for searching for a substring in a piece of text. We begin with a brute-force algorithm, whose running time is quadratic in the worst case. Next, we consider the ingenious Knuth-Morris-Pratt algorithm whose running time is guaranteed to be linear in the worst case. Then, we introduce the Boyer-Moore algorithm, whose running time is sublinear on typical inputs. Finally, we consider the Rabin-Karp fingerprint algorithm, which uses hashing in a clever way to solve the substring search and related problems.

Exercises. Drill exercises on the lecture material.

Programming Assignment: none.

Job Interview Questions. Algorithmic interview questions based on the lecture material.

Suggested readings. Section 5.2 and 5.3 in *Algorithms, 4th edition.*

Week 5

We complete our study of string processing by considering a number of ingenious algorithms that take full advantage of several of the fundamental methods covered earlier in the course. In the first lecture, we consider regular expression pattern matching, where the goal is to find strings from a specified set in a given text. In the second lecture, we consider classic methods for data compression.

Lecture: Regular Expressions.A regular expression is a method for specifying a set of strings. Our topic for this lecture is the famous grep algorithm that determines whether a given text contains any substring from the set. We examine an efficient implementation that makes use of our digraph reachability implementation from Week 1.

Lecture: Data Compression. We study and implement several classic data compression schemes, including run-length coding, Huffman compression, and LZW compression. We develop efficient implementations from first principles using a Java library for manipulating binary data that we developed for this purpose, based on priority queue and symbol table implementations from earlier lectures.

Exercises. Drill exercises on the lecture material.

Programming Assignment: Burrows-Wheeler algorithm. Implement the Burrows-Wheeler data compression algorithm.

Job Interview Questions. Algorithmic interview questions based on the lecture material.

Suggested readings. Section 5.4 and 5.5 in *Algorithms, 4th edition.*

Week 6

Our lectures this week are centered on the idea of problem-solving models like maxflow and shortest path, where a new problem can be formulated as an instance of one of those problems, and then solved with a classic and efficient algorithm.

Lecture: Reductions. In this lecture our goal is to develop ways to classify problems according to their computational requirements. We introduce the concept of

reduction as a technique for studying the relationship among problems. People use reductions to design algorithms, establish lower bounds, and classify problems in terms of their computational requirements.

Lecture: Linear Programming. The quintessential problem-solving model is known as linear programming, and the simplex method for solving it is one of the most widely used algorithms. In this lecture, we given an overview of this central topic in operations research and describe its relationship to algorithms that we have considered.

Exercises. Drill exercises on the lecture material.

Programming Assignment: none.

Job Interview Questions. Algorithmic interview questions based on the lecture material.

Suggested readings. Section 6.5 (pp. 903-910) in *Algorithms, 4th edition*.

Week 7

To complete the course, we describe the classic unsolved problem from theoretical computer science that is centered on the concept of algorithm efficiency and guides us in the search for efficient solutions to difficult problems.

Lecture: Intractability. Is there a universal problem-solving model to which all problems that we would like to solve reduce and for which we know an efficient algorithm? You may be surprised to learn that we do no know the answer to this question. In this lecture we introduce the complexity classes $\mathcal{P}$, $\mathcal{NP}$, and $\mathcal{NP}$-complete, pose the famous $\mathcal{P} = \mathcal{NP}$ question, and consider implications in the context of algorithms that we have treated in this course.

Final Exam.

Job Interview Questions. Algorithmic interview questions based on the lecture material.

Suggested readings. Section 6.6 (pp. 910-921) in *Algorithms, 4th edition*.

Created Wed 11 Jul 2012 3:48 AM PDT (UTC -0700)
Last Modified Sun 24 Mar 2013 10:39 PM PDT (UTC -0700)