19

Algorithmic Resources

This chapter briefly describes resources that the practical algorithm designer should be familiar with. Although some of this information has appeared elsewhere in the catalog, the most important pointers are collected here for general reference.

19.1 Software Systems

In this section, we describe several particularly comprehensive implementations of combinatorial algorithms, all of which are downloadable over the Internet. Although these codes are mentioned in the relevant sections of the catalog, they are substantial enough to warrant further attention.

A good algorithm designer does not reinvent the wheel, and a good programmer does not rewrite code that other people have written. Picasso put it best: "Good artists borrow. Great artists steal."

However, here is a word of caution about stealing. Many of the codes described in this book have been made available for research or educational use only. Commercial use may require a licensing arrangement with the author. I urge you to respect this. Licensing terms from academic institutions are usually quite modest. The recognition that industry is using a particular code is important to the authors, often more important than the money involved. This can lead to enhanced support or future releases of the software. Do the right thing and get a license. Information about terms or whom to contact is usually available embedded within the documentation, or available at the source's website.

Although many of the systems we describe here may be available by accessing our algorithm repository, *http://www.cs.sunysb.edu/∼algorith* we encourage you to get them from the original sites instead of Stony Brook. There are three reasons. First, the version on the original site is *much* more likely to be up-to-date. Second,

S.S. Skiena, *The Algorithm Design Manual*, 2nd ed., DOI: 10.1007/978-1-84800-070-4_19,

there are often supporting files and documentation that we did not download that may be of interest to you. Finally, many authors monitor the downloads of their codes, and so you deny them a well-earned thrill if you don't take them from the homepage.

19.1.1 LEDA

LEDA, for *Library of Efficient Data types and Algorithms*, is perhaps the best single resource available to support combinatorial computing. *LEDA* was originally developed by a group at Max-Planck-Instutut in Saarbrücken, Germany, including Kurt Mehlhorn, Stefan Näher, Stefan Schirra, Christian Uhrig, and Christoph Burnikel. *LEDA* is unique because of (1) the algorithmic sophistication of its developers, and (2) the level of continuity and resources invested in the project.

What *LEDA* offers is a complete collection of well-implemented C++ data structures and types. Particularly useful is the graph type, which supports all the basic operations one needs in an intelligent way, although this generality comes at some cost in size and speed over handcrafted implementations. A useful library of graph algorithms is included, which illustrates how cleanly and concisely these algorithms can be implemented using the LEDA data types. Good implementations of the most important data structures supporting such common data types as dictionaries and priority queues are provided. There are also algorithms and data structures for computational geometry, including support for visualization. For more, see the book [MN99].

Since 2001, *LEDA* has been exclusively available from Algorithmic Solutions Software GmbH (*http://www.algorithmic-solutions.com/*). This ensures professional-quality support, and new releases appear often. The great news is that a free edition containing all the basic data structures (including dictionaries, priority queues, graphs, and numerical types) was released February 2008. No source code or advanced algorithms are provided with the free edition. However, the licencing fees for the full library are not outlandish and free trial downloads are available.

19.1.2 CGAL

The *Computational Geometry Algorithms Library* or *CGAL* provides efficient and reliable geometric algorithms in C++. It is extremely comprehensive, offering a rich variety of triangulations, Voronoi diagrams, operations on polygons and polyhedra, line/curve arrangements, alpha-shapes, convex-hull algorithms, geometric search structures, and more. Many work in three dimensions and some beyond.

CGAL (*www.cgal.org*) should be the first place to go for serious geometric computing, although you should expect to spend some start-up time getting oriented to the *CGAL* way of thinking. *CGAL* is distributed under a dual-license scheme. It can be used together with open source software free of charge but using *CGAL* in other contexts requires obtaining a commercial license.

19.1.3 Boost Graph Library

Boost (*www.boost.org*) provides a well-regarded collection of free peer-reviewed portable C++ source libraries. The Boost license encourages both commercial and noncommercial use.

The Boost Graph Library [SLL02] (*http://www.boost.org/libs/graph/doc*) is perhaps most relevant for the readers of this book. Implementations of adjacency lists, matrices, and edge lists are included, along with a reasonable library of basic graph algorithms. Its interface and components are generic in the same sense as the C++ Standard Template Library (STL). Other Boost libraries of interest include those for string/text processing and math/numeric computation.

19.1.4 GOBLIN

The *Graph Object Library for Network Programming Problems* (GOBLIN) is a C++ class library that broadly focuses on graph optimization problems. These include several varieties of shortest path, minimum spanning tree, and connected component algorithms, plus particularly strong coverage of network flows and matching. Finally, a generic branch-and-bound module is used to solve such hard problems as independent set and vertex coloring.

GOBLIN was written and is maintained by Christian Fremuth-Paeger at the University of Augsberg. It is available under GNU lesser public licence from *http://www.math.uni-augsburg.de/~fremuth/goblin.html*. A spiffy Tcl/Tk interface is provided. GOBLIN is presumably not as robust as *Boost* or *LEDA*, but contains several algorithms not present in either of them.

19.1.5 Netlib

Netlib (*www.netlib.org*) is an online repository of mathematical software that contains a large number of interesting codes, tables, and papers. Netlib is a compilation of resources from a variety of places, with fairly detailed indices and search mechanisms. Netlib is important because of its breadth and ease of access. Whenever you need a specialized piece of mathematical software, you should look here first.

The Guide to Available Mathematical Software (GAMS), is an indexing service for Netlib and other related software repositories that can help you find what you want. Check it out at *http://gams.nist.gov*. GAMS is a service of the National Institute of Standards and Technology (NIST).

19.1.6 Collected Algorithms of the ACM

An early mechanism for the distribution of useful algorithm implementations was the *Collected Algorithms of the ACM* (*CALGO*). It first appeared in *Communications of the ACM* in 1960, covering such famous algorithms as Floyd's linear-time build heap algorithm. More recently, it has been the province of the *ACM Transactions on Mathematical Software*. Each algorithm/implementation is described

in a brief journal article, with the implementation validated and collected. These implementations are maintained at *http://www.acm.org/calgo/* and at Netlib.

Over 850 algorithms have appeared to date. Most of the codes are in Fortran and are relevant to numerical computing, although several interesting combinatorial algorithms have slithered their way into CALGO. Since the implementations have been refereed, they are presumably more reliable than most comparable software.

19.1.7 SourceForge and CPAN

SourceForge (*http://sourceforge.net/*) is the largest open source software development website, with over 160,000 registered projects. Most are of highly limited interest, but there is a lot of good stuff to be found. These include graph libraries such as *JUNG* and *JGraphT*, optimization engines such as *lpsolve* and *JGAP*, and much more.

CPAN (*http://www.cpan.org/*) is the Comprehensive Perl Archive Network. This enormous collection of Perl modules and scripts is where you should look before trying to implement anything in Perl.

19.1.8 The Stanford GraphBase

The Stanford GraphBase is an interesting program for several reasons. First, it was composed as a "literate program," meaning that it was written to be read. If anybody's programs deserve to be read, it is Knuth's, and [Knu94] contains the full source code of the system. The programming language/environment is CWEB, which permits the mixing of text and code in particularly expressive ways.

The GraphBase contains implementations of several important combinatorial algorithms, including matching, minimum spanning trees, and Voronoi diagrams, as well as specialized topics like constructing expander graphs and generating combinatorial objects. Finally, it contains programs for several recreational problems, including constructing word ladders (flour-floor-flood-blood-brood-broad-bread) and establishing dominance relations among football teams. Check it out at *http://www-cs-faculty.stanford.edu/~knuth/sgb.html*.

Although the GraphBase is fun to play with, it is not really suited for building general applications on top of. The GraphBase is perhaps most useful as an instance generator for constructing a wide variety of graphs to serve as test data. It incorporates graphs derived from interactions of characters in famous novels, Roget's thesaurus, the Mona Lisa, and the economy of the United States. Furthermore, because of its machine-independent random number generators, the GraphBase provides a way to construct random graphs that can be reconstructed elsewhere, making them perfect for experimental comparisons of algorithms.

19.1.9 Combinatorica

Combinatorica [PS03] is a collection of over 450 algorithms for combinatorics and graph theory written in Mathematica. These routines have been designed to work together, enabling one to readily experiment with discrete structures. *Combinatorica* has been widely used for both research and education.

Although (in my totally unbiased opinion) *Combinatorica* is more comprehensive and better integrated than other libraries of combinatorial algorithms, it is also the slowest such system available. Credit for all of these properties is largely due to Mathematica, which provides a very high-level, functional, interpreted, and thus inefficient programming language. *Combinatorica* is best for finding quick solutions to small problems, and (if you can read Mathematica code) as a terse exposition of algorithms for translation into other languages.

Check out *http://www.combinatorica.com* for the latest release and associated resources. It is also included with the standard Mathematica distribution in the directory *Packages/DiscreteMath/Combinatorica.m*.

19.1.10 Programs from Books

Several books on algorithms include working implementations of the algorithms in a real programming language. Although these implementations are intended primarily for exposition, they can also be useful for computation. Since they are typically small and clean, they can prove the right foundation for simple applications.

The most useful codes of this genre are described below. Most are available from the algorithm repository, *http://www.cs.sunysb.edu/∼algorith*.

Programming Challenges

If you like the C code that appeared in the first half of the text, you should check out the programs I wrote for my book *Programming Challenges* [SR03]. Perhaps most useful are additional examples of dynamic programming, computational geometry routines like convex hull, and a bignum integer arithmetic package. This algorithm library is available at *http://www.cs.sunysb.edu/∼skiena/392/programs/* or at *http://www.programming-challenges.com*.

Combinatorial Algorithms for Computers and Calculators

Nijenhuis and Wilf [NW78] specializes in algorithms for constructing basic combinatorial objects such as permutations, subsets, and partitions. Such algorithms are often very short, but they are hard to locate and usually surprisingly subtle. Fortran routines for all of the algorithms are provided, as well as a discussion of the theory behind each of them. The programs are usually short enough that it is reasonable to translate them directly into a more modern programming language, as I did in writing *Combinatorica* (see Section 19.1.9). Both random and sequential

generation algorithms are provided. Descriptions of more recent algorithms for several problems, without code, are provided in [Wil89].

These programs are now available from our algorithm repository website. We tracked them down from Neil Sloane, who had them on a magnetic tape, while the original authors did not! In [NW78], Nijenhuis and Wilf set the proper standard of statistically testing the output distribution of each of the random generators to establish that they really appear uniform. We encourage you to do the same before using these programs to verify that nothing has been lost in transit.

Computational Geometry in C

O'Rourke [O'R01] is perhaps the best practical introduction to computational geometry, because of its careful and correct C language implementations of all the main algorithms of computational geometry. Fundamental geometric primitives, convex hulls, triangulations, Voronoi diagrams, and motion planning are all included. Although they were implemented primarily for exposition rather than production use, they should be quite reliable. The codes are available from *http://maven.smith.edu/∼orourke/code.html*.

Algorithms in C++

Sedgewick's popular algorithms text [Sed98, SS02] comes in several different language editions, including C, C++, and Java. This book distinguishes itself through its use of algorithm animation and in its broad topic coverage, including numerical, string, and geometric algorithms.

The language-specific parts of the text consist of many small code fragments, instead of full programs or subroutines. They are best thought of as models, instead of working implementations. Still, the program fragments from the C++ edition have been made available from *http://www.cs.princeton.edu/∼rs/*.

Discrete Optimization Algorithms in Pascal

This is a collection of 28 programs for solving discrete optimization problems, appearing in the book by Syslo, Deo, and Kowalik [SDK83]. The package includes programs for integer and linear programming, the knapsack and set cover problems, traveling salesman, vertex coloring, and scheduling, as well as standard network optimization problems. They have been made available from the algorithm repository at *http://www.cs.sunysb.edu/∼algorith*.

This package is noteworthy for the operations-research flavor of the problems and algorithms selected. The algorithms have been selected to solve problems, rather than purely expository purposes.

19.2 Data Sources

It is often important to have interesting data to feed your algorithms, to serve as test data to ensure correctness or to compare different algorithms for raw speed. Finding good test data can be surprisingly difficult. Here are some pointers:

- *TSPLIB* – This well-respected library of test instances for the traveling salesman problem [Rei91] provides the standard collection of hard instances of TSPs. TSPLIB instances are large, real-world graphs, derived from applications such as circuit boards and networks. TSPLIB is available from *http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/* Somewhat older instances are also available from Netlib.
- *Stanford GraphBase* – Discussed in Section 19.1.8, this suite of programs by Knuth provides portable generators for a wide variety of graphs. These include graphs arising from distance matrices, arts, and literature, as well as graphs of more theoretical interest.
- *DIMACS Challenge data* – A series of DIMACS Challenge workshops have focused on evaluating algorithm implementations of graph, logic, and data structure problems. Instance generators for each problem have been developed, with the focus on constructing difficult or representative test data. The products of the DIMACS Challenges are all available from *http://dimacs.rutgers.edu/Challenges*.

19.3 Online Bibliographic Resources

The Internet has proven to be a fantastic resource for people interested in algorithms, as it has for many other subjects. What follows is a highly selective list of the resources that I use most often. All should be in the tool chest of every algorist.

- *ACM Digital Library* – This collection of bibliographic references provides links to essentially every technical paper ever published in computer science. Check out what is available at *http://portal.acm.org/*.
- *Google Scholar* – This free resource (*http://scholar.google.com/*) restricts Web searches to things that look like academic papers, often making it a sounder search for serious information than a general Web search. Particularly useful is the ability to see which papers cite a given paper. This lets you update an old reference to see what has happened since publication, and helps to judge the significance of a particular article.
- *Amazon.com* – This comprehensive catalog of books (*www.amazon.com*) is surprisingly useful for finding relevant literature for algorithmic problems, particularly since many recent books have been digitized and placed in its index.

19.4 Professional Consulting Services

Algorist Technologies (*http://www.algorist.com*) is a consulting firm that provides its clients with short-term, expert help in algorithm design and implementation. Typically, an Algorist consultant is called in for one to three days worth of intensive, onsite discussion and analysis with the client's own development staff. Algorist has built an impressive record of performance improvements with several companies and applications. We provide longer-term consulting and contracting services as well.

Call 212-222-9891 or email info@algorist.com for more information on services provided by Algorist Technologies.

Algorist Technologies
215 West 92nd St. Suite 1F
New York, NY 10025
http://www.algorist.com