

I have just started to read "The Algorithm Design Manual" by Steven Skiena, a computer science professor at SUNY, Stony Brook. I found the book to be very useful and well written. Another good book by Skiena is "Calculated Bets", in which he describes his attempts to bet on the outcome of Jai Alai matches using mathematical models and computer simulations.

The Algorithm Design Manual is divided into two parts, titled Techniques and Resources respectively. The first part, Techniques, introduces Data Structures and fundamental algorithms. After discussing the basic data structures like binary search trees and priority queues, the author moves on to specialized data structures for graphs, sets, strings and geometric structures (to represent polygons) and spatial data structures like kd-trees.

Most of the important ideas in algorithms like Divide and Conquer, Dynamic Programming, Graph Traversals (breadth first and depth -first search), the use of Minimum Spanning Trees, Shortest Path problems and Dijkstra's algorithm, heuristics for Combinatorial problems (e.g., backtracking) and newer techniques like simulated annealing and genetic algorithms are quite well covered.

The second part is a resources part. It has one very long chapter, (chapter 8) titled " A Catalog of Algorithmic Problems". This chapter describes a lot of common algorithms and tells the reader what is known about the problem, list of references for further reading and references to commercial/free implementations of the algorithm. The algorithmic problems are broken up into numerical problems, combinatory problems, two chapters on polynomial-time and hard Graph problems, Computational Geometry and Set and String problems. The last chapter (chapter 9) is titled Algorithmic Resources and describes some well known academic algorithm repositories - LEDA (library of Efficient Data Types and Algorithms), Netlib, Collected Algorithms of the ACM, the Stanford Graphbase, Combinatorica, and Xtango. A very good list of references to internet resources is also given. The book comes with a CD ROM which has the entire book with links and cross references, thirty hours of the author's audio lectures and some video lectures as well.

The biggest appeal of the book is the way the first part and the second part mesh together. I will pick an example that might be of particular interest, in order to illustrate the point. The data structure kd-tree is mentioned in the first part and described in detail in section 8.1, and then its applications/implementations are described in the section on Nearest Neighbour Search. Nearest Neighbour search is particularly important in classification and local regression problems. (Dr Brett had mentioned the book Expert Trading Systems: Modeling Financial Markets with Kernel Regression by John Wolberg a while back which discusses the use of kd-trees in nearest neighbour searches). The author discusses the various aspects of the problem, and how one of several different approaches might best be utilized depending upon the dimension of the data, size of the data set, whether the data set is static or dynamic; then he describes, Ranger, a tool for visualizing and experimenting with nearest neighbours in high dimensions and supports different types of kd-trees. He also describes Voronoi diagrams and provides references to its implementations; e.g., LEDA, which has a C++ implementation. (A Voronoi diagram is

a partitioning of space around each of N given points such that all points in the region around the i -th point are closer to this i -th point than they are to any of the other $(N-1)$ given points. Applications include nearest neighbour searches; facility location (e.g., locate the site for a new McDonald's restaurant such that it is as far as possible from the nearest existing McDonald's); locating largest empty contiguous sites for location of a factory; path planning, where we choose a route that maximizes the distance from a given set of obstacles or hazards; Quality triangulations like the Delauney triangulation which maximizes the minimum angle over all possible triangulations

I particularly enjoyed the wonderful treatment of Dynamic Programming with several examples - for instance, approximate string matching. (E.g., how can we search for the string closest to a given pattern in order to account for spelling errors?) This subject is very useful in text classification as well as in homology or similarity search in DNA sequences.