11

A Catalog of Algorithmic Problems

This is a catalog of algorithmic problems that arise commonly in practice. It describes what is known about them and gives suggestions about how best to proceed if the problem arises in your application.

What is the best way to use this catalog? First, think about your problem. If you recall the name, look up the catalog entry in the index or table of contents and start reading. Read through the entire entry, since it contains pointers to other relevant problems. Leaf through the catalog, looking at the pictures and problem names to see if something strikes a chord. Don't be afraid to use the index, for every problem in the book is listed there under several possible keywords and applications. If you *still* don't find something relevant, your problem is either not suitable for attack by combinatorial algorithms or else you don't fully understand it. In either case, go back to step one.

The catalog entries contain a variety of different types of information that have never been collected in one place before. Different fields in each entry present information of practical and historical interest.

To make this catalog more easily accessible, we introduce each problem with a pair of graphics representing the problem instance or input on the left and the result of solving the problem in this instance on the right. We have invested considerable thought in creating stylized examples that illustrate desired behaviors more than just definitions. For example, the minimum spanning tree example illustrates how points can be clustered using minimum spanning trees. We hope that people will be able to flip through the pictures and identify which problems might be relevant to them. We augment these pictures with more formal written input and problem descriptions to eliminate the ambiguity inherent in a purely pictorial representation.

S.S. Skiena, *The Algorithm Design Manual*, 2nd ed., DOI: 10.1007/978-1-84800-070-4_11,

Once you have identified your problem, the discussion section tells you what you should do about it. We describe applications where the problem is likely to arise and the special issues associated with data from them. We discuss the kind of results you can hope for or expect and, most importantly, what you should do to get them. For each problem, we outline a quick-and-dirty algorithm and provide pointers to more powerful algorithms to try next if the first attempt is not sufficient.

For each problem, we identify available software implementations that are discussed in the implementation field of each entry. Many of these routines are quite good, and they can perhaps be plugged directly into your application. Others maybe inadequate for production use, but they hopefully can provide a good model for your own implementation. In general, the implementations are listed in order of descending usefulness, but we will explicitly recommend the best one available for each problem if a clear winner exists. More detailed information for many of these implementations appears in Chapter 19. Essentially all of the implementations are available via the WWW site associated with this book—reachable at *http://www.cs.sunysb.edu/~algorith*.

Finally, in deliberately smaller print, we discuss the history of each problem and present results of primarily theoretical interest. We have attempted to report the best results known for each problem and point out empirical comparisons of algorithms or survey articles if they exist. This should be of interest to students and researchers, and also to practitioners for whom our recommended solutions prove inadequate and need to know if anything better is possible.

Caveats

This is a catalog of algorithmic problems. It is not a cookbook. It cannot be because there are too many recipes and too many possible variations on what people want to eat. My goal is to point you in the right direction so that you can solve your own problems. I try to identify the issues you will encounter along the way—problems that you will have to work out for yourself. In particular:

- For each problem, I suggest algorithms and directions to attack it. These recommendations are based on my experiences, and are aimed toward what I see as typical applications. I felt it was more important to make concrete recommendations for the masses rather than to try to cover all possible situations. If you don't agree with my advice, don't follow it, but before you ignore me, try to understand the reasoning behind my recommendations and articulate a reason why your application violates my assumptions.

- The implementations I recommend are not necessarily complete solutions to your problem. I point to an implementation whenever I feel it might be more useful to someone than just a textbook description of the algorithm. Some programs are useful only as models for you to write your own codes. Others are embedded in large systems and so might be too painful to extract and run

on their own. Assume that all of them contain bugs. Many are quite serious, so beware.

- Please respect the licensing conditions for any implementations you use commercially. Many of these codes are not open source and have licence restrictions. See Section 19.1 for a further discussion of this issue.
- I would be interested in hearing about your experiences with my recommendations, both positive and negative. I would be especially interested in learning about any other implementations that you know about. Feel free to drop me a line at feedback@algorist.com.