Preface

This book evolved over the past ten years from a set of lecture notes developed while teaching the undergraduate Algorithms course at Berkeley and U.C. San Diego. Our way of teaching this course evolved tremendously over these years in a number of directions, partly to address our students' background (undeveloped formal skills outside of programming), and partly to reflect the maturing of the field in general, as we have come to see it. The notes increasingly crystallized into a narrative, and we progressively structured the course to emphasize the "story line" implicit in the progression of the material. As a result, the topics were carefully selected and clustered. No attempt was made to be encyclopedic, and this freed us to include topics traditionally de-emphasized or omitted from most Algorithms books.

Playing on the strengths of our students (shared by most of today's undergraduates in Computer Science), instead of dwelling on formal proofs we distilled in each case the crisp mathematical idea that makes the algorithm work. In other words, we emphasized rigor over formalism. We found that our students were much more receptive to mathematical rigor of this form. It is this progression of crisp ideas that helps weave the story.

Once you think about Algorithms in this way, it makes sense to start at the historical beginning of it all, where, in addition, the characters are familiar and the contrasts dramatic: numbers, primality, and factoring. This is the subject of Part I of the book, which also includes the RSA cryptosystem, and divide-and-conquer algorithms for integer multiplication, sorting and median finding, as well as the fast Fourier transform. There are three other parts: Part II, the most traditional section of the book, concentrates on data structures and graphs; the contrast here is between the intricate structure of the underlying problems and the short and crisp pieces of pseudocode that solve them. Instructors wishing to teach a more traditional course can simply start with Part II, which is self-contained (following the prologue), and then cover Part I as required. In Parts I and II we introduced certain techniques (such as greedy and divide-and-conquer) which work for special kinds of problems; Part III deals with the "sledgehammers" of the trade, techniques that are powerful and general: dynamic programming (a novel approach helps clarify this traditional stumbling block for students) and linear programming (a clean and intuitive treatment of the simplex algorithm, duality, and reductions to the basic problem). The final Part IV is about ways of dealing with hard problems: NP-completeness, various heuristics, as well as quantum algorithms, perhaps the most advanced and modern topic. As it happens, we end the story exactly where we started it, with Shor's quantum algorithm for factoring.

The book includes three additional undercurrents, in the form of three series of separate