Why Algorithms? And What Should You Expect From This Course?

Andrew B. Kahng, abk@ucsd.edu  http://vlsicad.ucsd.edu/~abk

Algorithms are the “heart of computer science”. They connect real-world needs and applications to architectures, interconnection networks, programming languages, data structures and other aspects of CS and CE. Computation is ubiquitous in improving the quality of human life – whether through computational science, information processing, security, graphics, bioinformatics, machine learning, networking, or any number of other domains. And it is algorithms that achieve the enabling efficiencies (with respect to time, storage, or energy) when performing sorting, FFT, map-reduce, packet routing, rendering, genome sequencing, search, etc. within such computations.

I believe that studying algorithms ultimately boils down to studying the art and science of problem-solving – that is, problem-solving using computers.

Three questions are always present in the study and development of algorithms.

First: What is the problem for which we need an efficient computational solution?

You’ll practice abstracting and formally stating the problem (including any constraints on its solution).

Second: How do we solve the problem?

You’ll practice applying problem-solving paradigms (develop notation, study small examples, recall similar problems, look at limiting cases, …).

You’ll also practice applying basic algorithmic paradigms at an introductory level of sophistication. This “basic algorithm toolkit” includes greed, divide-and-conquer, dynamic programming, mathematical programming, network flow, and branch-and-bound enumeration.

Third: How do we know that our solution is correct, and good?

You’ll practice the analysis of algorithms in multiple senses: proving correctness, and establishing tight upper and lower bounds on (worst-case, average-case) (runtime, space) complexity with respect to natural parameters of problem instances.

About “Problem-Solving”. When you get a job in the real world, “cookbook” goes out the window: your added value will come from doing what others cannot do, from creating what others cannot create. (In some sense, “outsourcing” always starts with whatever is “cookbook” or “commoditized”.) So, my goal in this course is to push you toward the problem-solving side of algorithms – the abstraction, formulation, and solution of meaningful problems. Many students are uncomfortable in contexts where “you either see the solution or you don’t”. Nevertheless, a goal of this course is to impart mental strategies along with toolkits – i.e., both of (1) problem-solving paradigms and (2) algorithmic paradigms – to strengthen your chances of success in such contexts. See page 14 of the Paul Zeitz book, “The Art and Craft of Problem Solving”, linked from the course webpage. See also the “Research and Graduate School Advice” slides at http://vlsicad.ucsd.edu/ (a version of this talk will be presented during 10th week).
What should you achieve by the end of the course?

You should understand the basic algorithm techniques ("toolkit"): greed, divide-and-conquer, dynamic programming, mathematical programming, network flow, and branch-and-bound enumeration.

You should be able to “map” new problems to previously-seen problems and to appropriate algorithm techniques.

You should be able to design efficient algorithms, and you should be able to prove efficiency through analysis of time and space complexity.

You should be able to clearly communicate both your algorithm designs and your algorithm analyses.

Feedback?

Feedback and questions regarding the course and its goals are always welcome. Please don’t hesitate to contact me with these at any time.