Course: Algorithms in the Real World (15-853)

Instructors: Guy Blelloch and Bruce Maggs

URL:
www.cs.cmu.edu/~guyb/realworld/indexF03.html

Expected Background

Asymptotic analysis

Graph Terminology and Algorithms
- e.g. what is max flow

NP-complete problems and reductions

Matrix algebra
- e.g. what is the null-space of a matrix

Probability

Course Requirements

Readings
- Chapters, papers, course-notes, web documents
- There is NO course textbook

Assignments
- about 6 assignments (at least one will involve programming)

Midterm and Final
- takehome

Why care about algorithms?

Just about every field uses algorithms
- Engineering, Physical sciences, Social sciences
- Business, Medicine
- Probably used more widely than Calculus

The level of sophistication has increased significantly in recent years

Asymptotics becoming more important
- Linear programming (interior-point methods)
- Factoring (number-field sieve)
- …

An Exercise

What are the 10 most useful applications of algorithms in the "Real World"

Work in groups of 2 or 3.

You have 15 minutes.

Some Examples

Do cities use interesting algorithms for trash collection?

GeoRoute

Does your mailman use sophisticated algorithms to pick his routes?

RouteSmart

Does your janitor select the order of rooms using sophisticated algorithms?

Probably not

Topics

Cryptography (5)
- ssh, RSA, Rijdael, kerberos, …

Error Correcting Codes (3)
- Reed-Solomon, tornado-codes, CDs

Graph Separators (4)
- Genome sequencing, BLAST, evolutionary trees, …

Linear and Integer Programming (4)
- Interior point methods, airline crew scheduling, …

Web Algorithms (5)
- Searching, Google pagerank, duplicate removal, …

(n) = number of lectures (approximate)

6 Standard Misconceptions about the Theory of Algorithms

Or "why theory is only of limited use in the real world"

Note: all the following are partially true, so to protect the innocent (and guilty) I will refrain from using names, sometimes.

6: Theory is about worst-case analysis

or, yes there is some average-case analysis, but it is irrelevant also. I care about my problem instances.

Many solutions are of this form, but

1. understanding worst case helps understand what the bad examples are
2. there has been significant work on considering bounds given certain restrictions on data distributions
3. there has been significant work on studying what fraction of cases are hard.

Side note: a hot topic

Characterizing input classes or properties and analyzing algorithms for those inputs

Any ideas for graphs?

Any ideas for sorting?

Any ideas for point distributions?

5: Machine model used to analyze algorirhtms is broken

Yes the RAM is not a perfect model, but

1. there are many results using various enhancements to the model (IO, cache, streaming). If you care about more accurate costs, consider results for these models.
2. most results do not change qualitatively across models. P vs. NP translates across most models
3. many great ideas come from simplified models that are later translated to more sophisticated models

4: Theory is about Runtime Analysis

No, theory is about formalizing models. Consider

1. cryptography
   - what does it mean to be hard to break?
   - what is a public-key cryptosystem?
   - what is a zero-knowledge proof?
2. learning theory
3. information theory
4. graph theory

3: Most problems studied in theory are irrelevant

Today's irrelevant problem might be tomorrow's multibillion-dollar industry.

I mean, who cares about factoring numbers, or about Galois Fields, or about expander graphs?

2: Problems that are not irrelevant are oversimplified

Sometimes, but

1. solving simplified version is often a step on the way to solve the full version.
2. solving a simplified version can help understand why the full version is hard

1: Theory is all about big-O and ignores constant factors

Many important results do use big-o analysis, but

1. There are tons of results that use exact analysis, sometimes just in highest-order term, e.g. $7n + O(\log n)$
2. Often algorithms with large constant factors are later improved, sometimes with trivial modifications
3. Sometimes the large constant factor is just in the proof (perhaps to simplify it), and not in the real algorithm.

On Strassen's Matrix Multiply

From Numerical Recipes ([redacted] 1986)

> "We close the chapter with a little entertainment, a bit of algorithmic pretidigitation

and later in the same section

> "This is fun, but let's look at practicabilities: If you estimate how large N has to be before the difference between exponent 3 and exponent 2.807 is substatial enough to outweigh the bookkeeping overhead, arising from eht compicated nature of the recursive Strassen algorithm, you will find that LU decomposition is in no immediate danger of coming obsolete."

From an Experimental Paper

From an experimental paper (Bailey, Lee and Simon):

"For many years it was thought that [the level at which Strassen's algorithm is more efficient] was well over 1000x1000. In fact, for some new workstations, such as the Sun-4 and SGI IRIS 4D, Strassen is faster for matrices as small as 16x16.

Most modern matrix libraries now supply Strassen as a choice

- IBM ESSL library
- Cray Libraries
- LAPACK library

What each topic will cover

Key problems in the area

- Formal definitions
- How they relate to practice

Many algorithms

- Theory
- Practice

Key applications along with case studies

For Next Monday

Read the readings on Cryptography:
http://www.cs.cmu.edu/~guyb/realworld/indexF03.html

Some General Themes

Measures of "performance" beyond "Time"

- Quality of results
- Generality
- Simplicity

Transition from theory to practice

- Often an iterative process

Characterizing input

- Input data is rarely "worst case" or even "expected case" when the expectation is over all possible inputs.