Programming Contests, Algorithms, and the Real World

Steven Skiena

Department of Computer Science
State University of New York
Stony Brook, NY 11794–4400

http://www.cs.sunysb.edu/∼skiena

What Does it Mean to Be a TopCoder Champ?

Making the finals of an international programming contest such as TopCoder is a tremendous achievement.

Congratulations!

I have long been interested in programming contests, and how they relate to the real world of computing.

What important skills do they accurately measure?

What important skills don't they measure?

What does this say about your future?

You have Fast Heads

You are amazingly quick at solving algorithm problems using standard techniques.

The average CS student would *never* solve some of the problems you do instantly.

You are very quick at reading and understanding problems, even if English is not your native language.

You have a very intuitive grasp of whether an algorithm will be difficult to program, and can make instant judgments of "how fast is fast enough".

You see through boundary conditions and traps that would foil most of your peers.

You have Fast Hands

You are completely fluent in your favorite programming language(s), including libraries and I/O.

You have completely mastered your favorite programming environment(s): editor, personal code libraries, compiler, profiler, and debugger.

You are all fast typists – but watch your wrists!

You program defensively but efficiently – few comments but not terrible style.

It is said (and I believe) that certain programmers are 100 times more productive than others.

I was never anywhere near as fast as any of you.

You have Balls

There is a competitive, obsessive, pressure-driven, go-for-the-jugular mentality required for top-level programming competitions.

Women can be tremendous algorists – the best student I ever taught algorithms to is female.

Women can be excellent programmers – in my current research group, the two graduate students whose programs I trust most are the two women.

But top-level programming competitions remain a largely male domain.

But So What?

I recognize only two people from ACM ICPC championship teams for a non-programming contest accomplishments: Amit Sahai (Princeton) and Craig Silverstein (Google).

From several hundred IOI medalists since 1989, I recognize only the names of two professors.

There is often surprisingly little correlation between achievement in one domain and success in another.

Studies find that people from the middle of the class are "more successful" than those from the top quarter of the class.

My experience is that many *top* students fail to realize their true potential...

Cautionary Tale: “M”

- Finished third in the Putnam mathematics competition as a college freshman. . .
- Went to work writing Cobol programs for IBM. . .
- Came back and got a PhD in Computer Science in only two years. . .
- Went back to work writing programs for IBM. . .

Cautionary Tale: “L”

- Top-50 Univ. de Valladolid and Topcoder contestant...
- Does research with me on random walks models for finance...
- Missed deadlines to apply to top graduate schools...
- Burned out on school, leaves to travel the world...
- Now working as a programmer for a small financial company.

Cautionary Tale: “H”

- Member of world champion ACM programming team. . .
- Comes to Stony Brook for Computer Science PhD program, easily passes quals. . .
- But has surprising difficulty with fuzzily-specified research problems. . .
- Takes summer job at Microsoft and never comes back.

What Can be Bad about Programming Contests?

- Skills become over-optimized for particular aspects of software design.
- Participants work on *other people's* well-defined problems.
- Participants basically work individually – interpersonal skills are not a factor.
- The time horizon of hours is very different than a time horizon of months.

My Advice for Your Future

You are all awesome and will be successful – but follow my advice to maximize your success:

- Look out for yourself and your career, since no one else is going to do it for you . . .
- Look for where the big challenges are happening, and then focus on setting the agenda . . .
- Avoid self-destructive behavior/career-limiting moves. . .
- Learn to communicate. . .
- *Always* keep learning and growing. . .

Algorithms and the Real World

It is sad but true that interesting algorithmics plays a relatively small role in most software development.

Efficiency seems to matter less as machines get faster.

Sluggish application performance is more often due to system and storage issues than CPU time.

Interesting algorithmics are (by volume) a small part of most large applications.

Improved solution quality for fuzzily-defined problems is becoming more important.

Research Issues in Algorithms

An important aspect of algorithmic research is that of *formulating* problems.

What is the right abstraction of a real-life application?

What algorithmic problems are difficult enough to be interesting (i.e. requires new techniques) yet realistically might be solvable?

Algorithms Today: Approximation

Most interesting optimization problems are NP-complete, including traveling salesman, set cover, and clique.

This means that no efficient algorithm for these optimization problems (presumably) exists.

But how close can we come to the optimal answer in polynomial time?

The *approximation ratio guarantee* varies dramatically according to the problem: TSP ($O(1)$), set cover ($O(\lg n)$), and clique ($\Theta(n^{1/3})$)

Quality of the solution is usually more interesting than running time.

Algorithms Today: Randomization

How do you find the mode (most frequent element) of a data set?

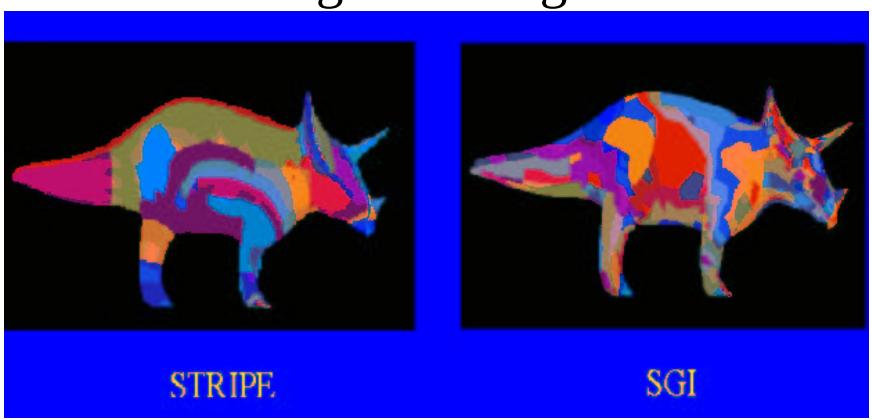
Now how do you do it when your data stream is so large you cannot store it all?

Can you guarantee finding the mode (or an element near the mode) with high probability?

Randomized strategies such as random sampling are necessary to avoid bad worst-case behavior.

Research Project: Rendering Triangle Strips

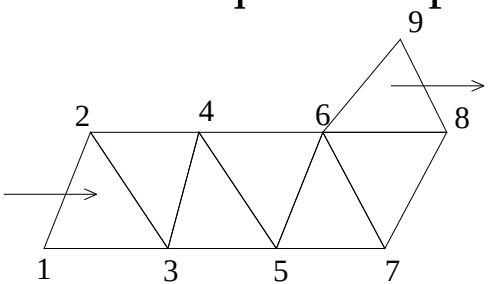
In rendering large polyhedral models, the performance bottleneck is in describing the triangulation to the hardware.



By partitioning triangles into strips, we can describe the *next* triangle using only one additional vertex.

What's the Optimization Criteria?

We seek to cover the vertices of the dual graph of the triangulation using as few 'sequential' paths as possible.



Many heuristics suggest themselves, including greedy longest path, low-degree walks, and DFS partitioning.

Since many models are not fully triangulated, we are free to triangulate polygonal faces so as to optimize strip lengths.

Results

It is NP-complete to minimize the sequential strip decomposition for general models.

We developed a good heuristic, based on finding large 'patches' in the mesh and appropriate greedy walks.

Rendering times for arbitrary images are typically improved by about 10%, which is clearly visible in helicopter races.

Our resulting code, Stripe, has been widely used and is available from http://www.cs.sunysb.edu/∼stripe

Modern graphics hardware yields somewhat different trade-offs for optimal rendering.

Research Project: Text Analysis

The increasing volume of online information coupled with decreasing costs of communications and computation creates exciting new opportunities in text mining.

We are interested in identifying all people, places and things in text and how they relate to each other.

Our Lydia system can analyze all of the 1000+ on-line daily English-language newspapers on a single commodity computer

Our ultimate goal is to build a relational encyclopedia of much of the world's knowledge through text analysis.

System Architecture

Spidering – text is retrieved from a given site on a daily basis using semi-custom spidering agents.

Normalization – clean text is extracted with semi-custom parser and formatted for our pipeline.

Text Markup – annotates parts of the source text (POS tagging, entity recognition) for storage and analysis.

Back Office Operations – we aggregate entity frequency and relational data for statistical / algorithmic analysis

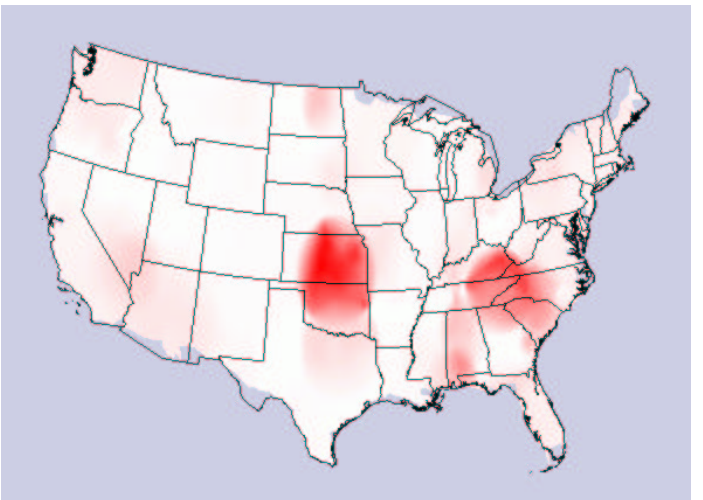
Back Office Operations

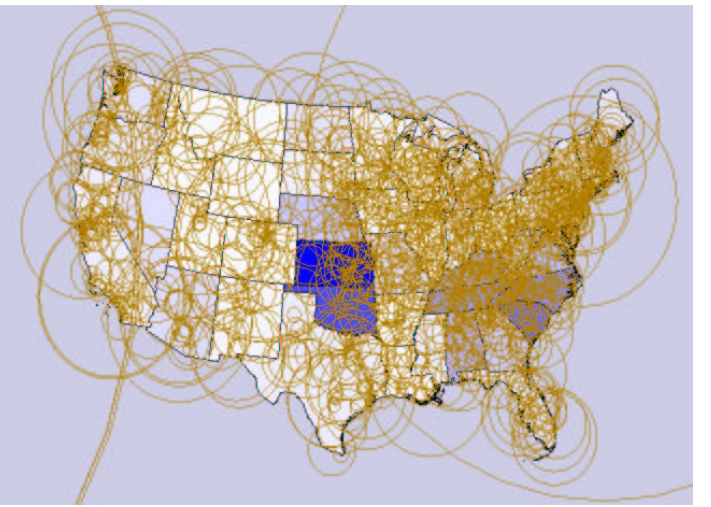
The most interesting analysis occurs after markup, using our MySQL database of occurrences of interesting entities.

Each day's worth of analysis yields about 10 million occurrences of about 1 million different entities, so efficiency matters.

Linkage of each occurrence to source and time facilitates a variety of interesting analysis. . .

Heatmaps: Who cares about NASCAR?





The two NASCAR races run during this period (October 2004) were the Banquet 400 at the Kansas Motor Speedway and the UAW-GM Quality 500 in Concord, NC.

Juxtapositions

Statistical analysis of neighboring entities in the text enables us to identify important associations between them.

From an analysis of CNN since 1999:

Yahoo: AOL, Web, Lycos, Google, Internet

Bill Gates: Windows, Paul Allen, Steve Ballmer, Microsoft, David Boies

George Bush: Al Gore, Ronald Reagan, Republican, Texas, John Kerry

Synonym Sets

JFK, John Kennedy, John F. Kennedy, and John Fitzgerald Kennedy all refer to the same person.

We need a mechanism to link multiple entities that have slightly different names but refer to the same thing.

We say two entities belong in the same synonym set if: (1) their names are morphologically compatible and (2) if the entities they are related to are similar.

Here edit distance / hashing help measure morphological similarity, where statistical analysis measures contextual similarity.

Research Project: Computational Biology

Biology is the science where the most exciting things are happening. . .

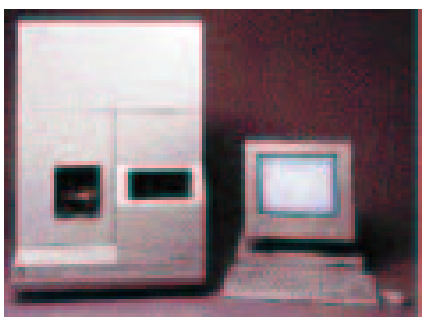
Computer Science is playing an enormous role in modern biology because DNA sequences are both fundamental to life and can be usefully modeled as a strings on $\{ACGT\}$.

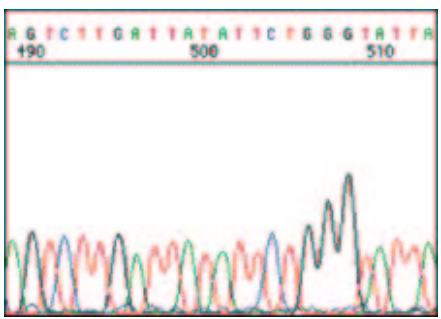
Sequence similarity programs like BLAST use fast algorithms/heuristics for approximate string matching.

Other important algorithmic problems include data clustering, evolutionary tree reconstruction, and sequence assembly

Contemporary Sequencing Machines

Sequencing machines today use the same basic principles as the original Gilbert-Sanger method.

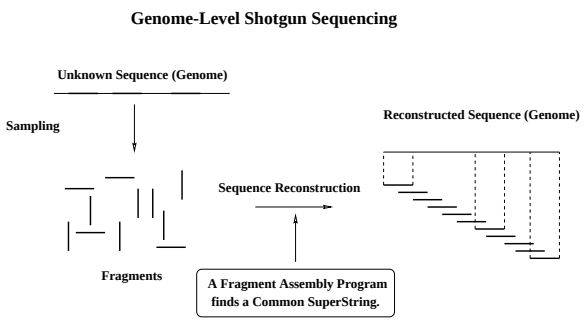




Read lengths have gotten slightly longer with time, perhaps from 500 bp to 700 bp, with a base error rate of about 2%, at a cost of about $1-$2 per read.

Fragment Assembly

In *shotgun sequencing*, whole genomes are sequenced by making clones, breaking them into small pieces, and trying to put the pieces together again based on overlaps.

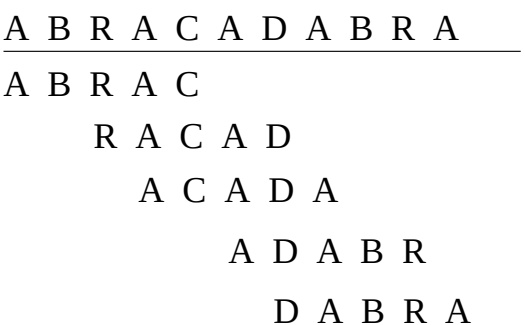


Note that the fragments are *randomly* sampled, and thus no positional information is available.

Why is Assembly Difficult?

The most natural notion of assembly is to order the fragments so as to form the shortest string containing all of them.

A B R A C
A C A D A
A D A B R
D A B R A
R A C A D



However, the problem of finding the shortest common superstring of a set of strings is NP-complete.

Even Worse…

- We must deal with significant errors in the sequence fragments.
- Genomes have many repeats (approximate copies of the same sequence), which are very hard to identify and reconstruct.
- *Paired read* sequencing strategies are necessary to help deal with repeats.
- The size of the problem is very large. Celera's Human Genome sequencing project contained roughly 26.4 million fragments, each about 550 bases long.

But Difficult Does Not Mean Impossible

Genome assembly projects have increased from tens of thousands of bases to billions of bases over 10 years or so.

In 1996, our *Stroll* shotgun sequence assembler (Chen and Skiena) was used by Brookhaven National Laboratory to sequence the bacteria *Borrelia burgdorferi*.

Today, assembler design for bacterial projects (and to a lesser extent mammalian projects) is largely a solved problem.

However, the algorithms in today's assemblers rely heavily on today's assumptions of read length, error rate, and coverage.

Trends in Sequencing Technology

Very short reads (20 bases on each end) are soon to become amazingly cheap. But can we use them for assembly in the face of repeats?

We are currently developing algorithms for assembly from massive numbers of short reads.

Ideas include Eulerian paths in de Bruijn graphs, combinatorial / statistical analysis of errors, and hashing for speed/error correction.

Consider Graduate School!

If you are interested in algorithms and advanced computer science, you should consider graduate school.

Advanced course work is an important part of the program, but the most important part of a Ph.D is research.

Any decent Ph.D. program will provide enough financial support for you to live comfortably although not lavishly.

I would love to have every one of you at Stony Brook!

Congratulations!

By no means be discouraged by anything I said.

Every one of you is awesome. . .

By keeping a big picture vision of your career you will all accomplish amazing things.

Now on to the finals – may the best man win!

For Further Reading