The Mathematics of Computer Science

Sam Roweis
Department of Computer Science
University of Toronto

Computer Science

"Never trust a scientific discipline that feels the need to put the word Science in its name."

Computer Science & Mathematics

"Never trust a scientific discipline that feels the need to put the word Science in its name."

- Since computers are so common, the mystery of how to use and program computers is gone.
- Q: What then, if anything, does the field of Computer Science have to offer the world?
- A: The ability to apply a wide variety of mathematical and statistical results to designing better computer algorithms.

Questions Math/Stats can answer

- Some abstract mathematical problems:
  - What is the k^{th} Fibonnacci number?
  - What is the solution to the differential equation:

$$p_l(t+1) = [p_l(t) + \gamma(x^l(t) - c_l)]^+$$

  - What is the expected gap between two random points in the unit interval?
- Who cares about their answers?
  Computer Scientists do!
  I'll show you why.

Google

- Q: How does Google work?
  A: Almost nobody knows.

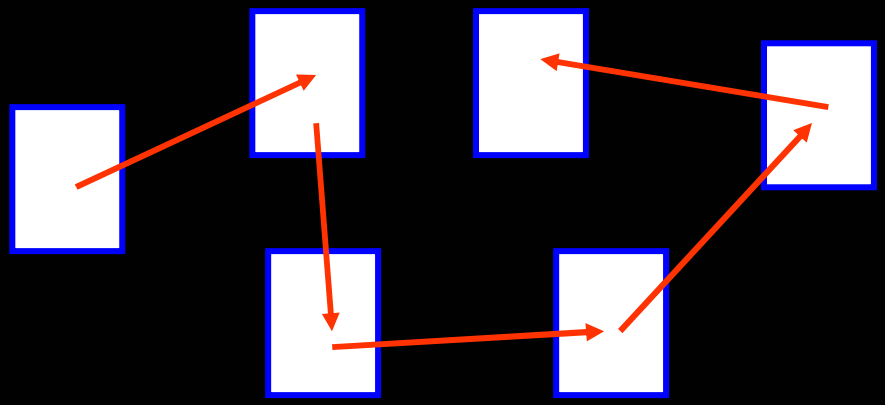
Google

- Q: How does Google work?
  A: Almost nobody knows.

- Q: What was the basic idea of how Google used to work?
- A: Filter queries by text matching in crawled pages, and rank results using the “random surfer” model. (PageRank)

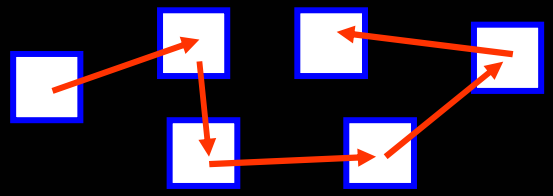
Google's Random Surfer

- Imagine repeating this procedure:
  - Whatever web page you are on, with probability (1-c) follow a random outgoing link.
    With probability c, teleport to a completely random page, uniformly over the whole web.



Google's Random Surfer

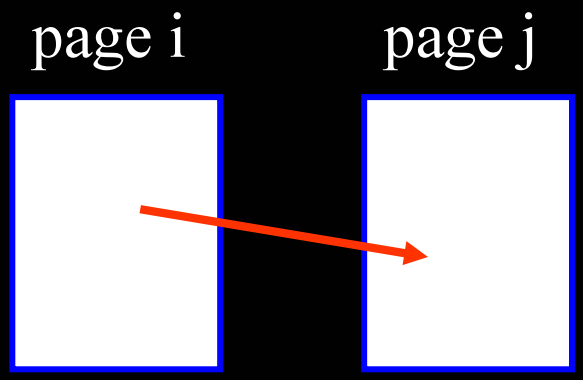
- Whatever web page you are on, with probability (1-c) follow a random outgoing link. With probability c, teleport to a completely random page, uniformly over the whole web.

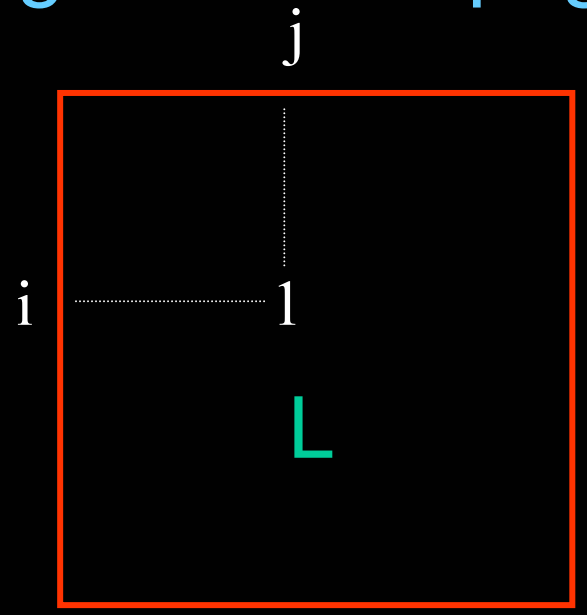


- Now ask: what fraction of the time do I spend on each web page? That's the rank.
- Intuition: you'll spend more time on important pages. "All roads lead to Rome."

Formalizing the Random Surfer

- Imagine a square binary matrix L, with as many rows as web pages.
  The entry L(i,j) is 1 if page i links to page j and 0 otherwise.





Formalizing the Random Surfer

- Now normalize the rows of L, by dividing each row by its sum.
  Multiply this normalized matrix L*
  by (1-c) and add c to every entry.
- The resulting matrix W has entries W(i,j) which give the probability of going to page j if you are now on page i.

$$W=(1-c)L^{*} + c$$

Computing Occupation Times

- How can we calculate the fraction of time spent on each web page?
- First observation: if we surf for long enough, it doesn't matter where we start.
- Imagine maintaining a vector x, whose entry x(i) is the fraction of the time we expect to spend on page i.
- One iteration of surfing: $x \leftarrow Wx$

Finding the Fixed Point

- We want the fixed point of the iterated map x := Wx. Certainly it will satisfy Wx=x, but how can we find it? Is it unique?
- It turns out that this is a standard problem in linear algebra. The answer is to find the eigenvector of W with the largest eigenvalue.
- The components of this vector tell us what fraction of time is spent on each page.

Google is Linear Algebra

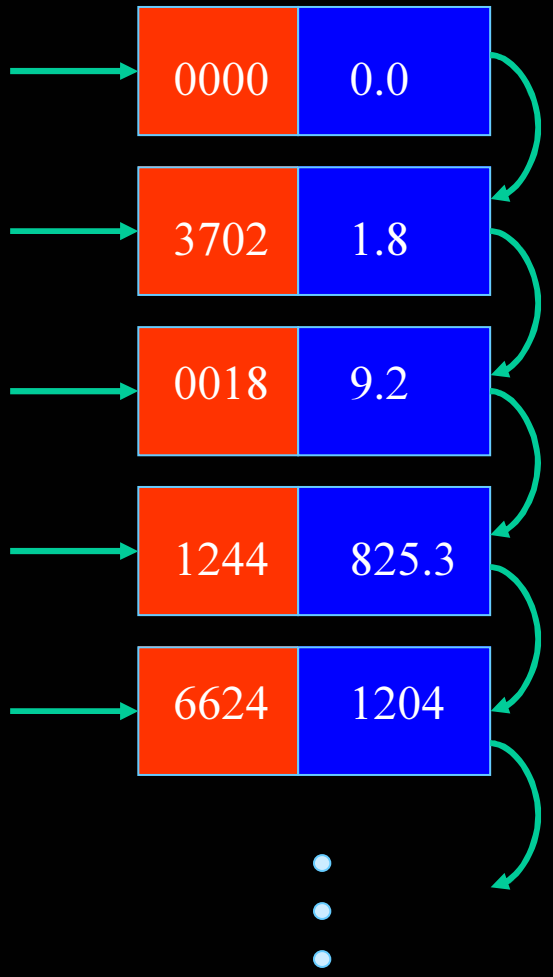
- We've transformed what seemed to be a computer science question:
  How do we compute the rank of a page in Google?
  Into a purely mathematical answer:
  Find the eigenvector of the matrix $(1-c)L^{*} + c$ having the largest eigenvalue.
- Google is just applied Linear Algebra!

A Database Search Problem

- Assume you have a very large list of records, each of which has a unique ID (arbitrarily & randomly assigned) as well as a positive key value (which is what we really care about).

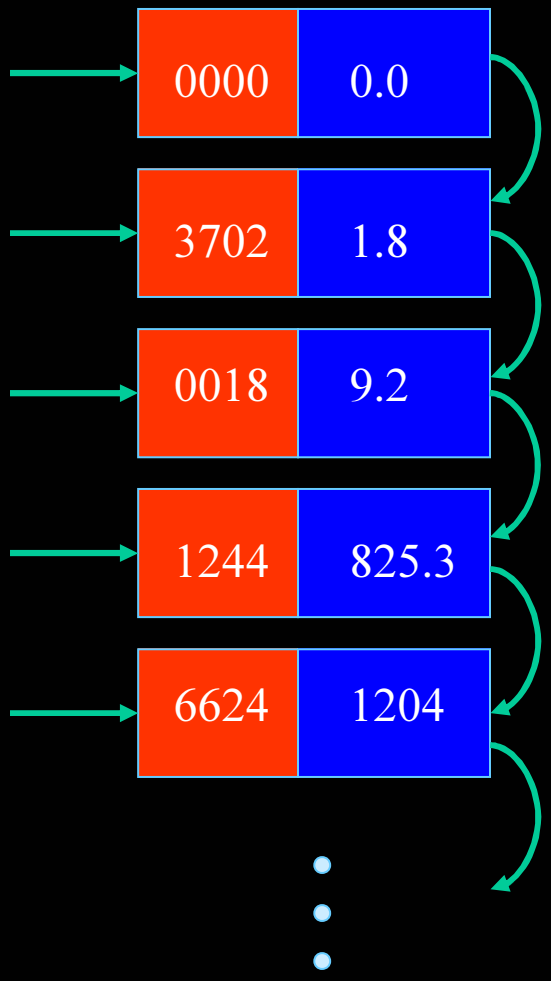
Accessing Records by ID/Key

- You can look up any record quickly by ID.
- But when searching by keys, we can only find the record with the next highest key.



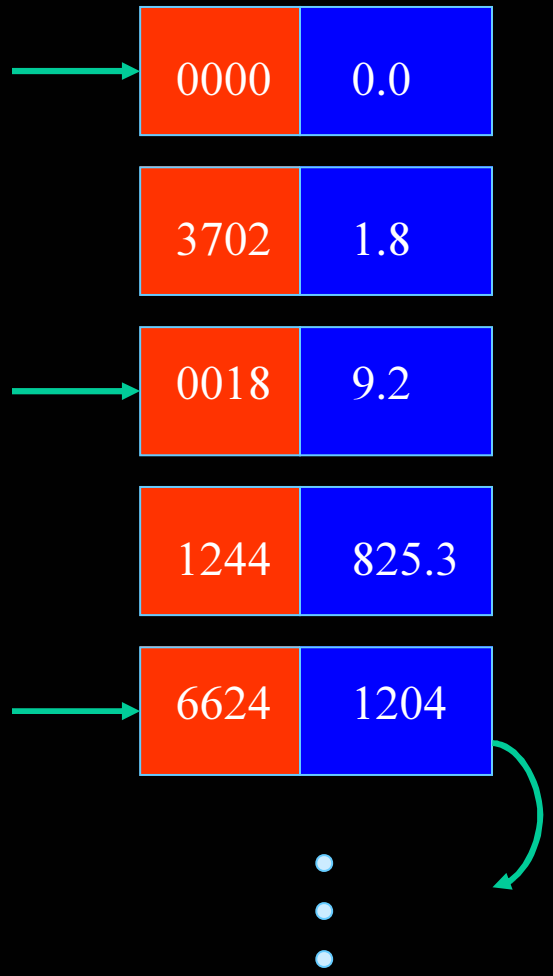
How can we find the largest key?

- Two obvious ways to find the ID with largest key.
  - Scan all IDs sequentially, and remember the biggest one.
  - Start at ID zero and get the next highest key repeatedly until you come to the biggest.
- Both strategies access all records in the database.
- Other ideas?



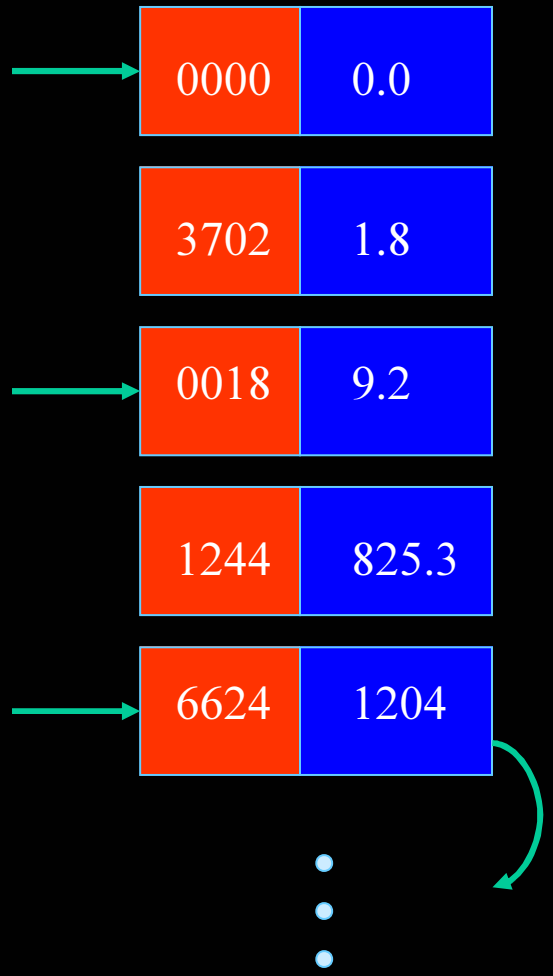
Can we do better?

- Say there are N records.
- Now pick R random IDs. Take the one with the biggest key, and start with it, getting the next highest until you reach the largest.
- What's the expected number of records accessed?



Throwing Darts

- Pick R random IDs, walk from largest to the end.
- Q: How many records are accessed, on average?
- Same Q: What's the average size of the final interval if we throw R darts into the real line segment [0,1]?



Performance of Dart Throwing

- Average size of the final interval if we throw R darts into [0,1] is 1/R.
- Therefore, on average (R+N/R) records are accessed by our method.
- We've transformed the CS question:
  What's the expected cost of our new database search algorithm?
  into a math answer:
  $E[\min_r(1-u_r)]$, $1<=r<=R$, $u_r \sim U(0,1)$

Optimal Database Access

- To minimize expected accesses, what's the best value of R?

  $A=(R+N/R)$; $dA/dR = 1-N/R^2$

  $R^* = \sqrt{N}$; $A^* = 2\sqrt{N}$!

- We've transformed the question:
  What's the optimal DB search parameter?
  into a simple calculus problem!

  (In fact, a lot of database and distributed systems research uses calculus & probability.)

Internet Network Flow Control

- You are given a network of unidirectional links (transmission lines) with fixed capacities.
- The network is shared by a set of customers, each of whom is willing to pay a certain amount to get a certain bandwidth usage.
- Goal: find the best routing of customers along the links and the optimal transmission rates so as to maximize the revenue.
- This is exactly the flow control problem faced on the internet, except with utility instead of money.

Formalizing Flow Control

- Unidirectional links k with capacities c(k).
- Sources s, each characterized by a transmission rate x(s) and a monotonic utility function U[x(s)].
- Routing matrix R(s,k)=1 if source s uses link k.
- Mathematical problem:

$$\max \sum_s U_s \qquad \text{subject to}$$

$$\sum_{s:R(s,k)=1} x(s) <= c(k) \qquad \text{for all } k$$

(maximize utility respecting capacity)

Except…nobody's driving

- There is one more important constraint: the optimization has to be performed using only local computations at the links and sources!
- Basic algorithm:
  Coupled differential equations which implement stochastic gradient ascent in the utility:

$$p_l(t+1) = [p_l(t) + \gamma(x^l(t) - c_l)]^+$$

$$x_s(t+1) = U_s'^{-1}(p^s(t))$$

Existing Protocols

- Reno

$$U_s^{reno}(x_s) = \frac{\sqrt{2}}{D_s} \tan^{-1}\left(\frac{x_s D_s}{2}\right)$$

- Reno/RED

$$U_s^{reno/red}(x_s) = \begin{cases} b_1 x_s + \rho_1 \frac{\sqrt{2}}{D_s} \tan^{-1}\left(\frac{x_s D_s}{2}\right), & x_s \text{ large} \\ b_2 x_s + \rho_2 \frac{\sqrt{2}}{D_s} \tan^{-1}\left(\frac{x_s D_s}{2}\right), & x_s \text{ small} \end{cases}$$

- Reno/REM

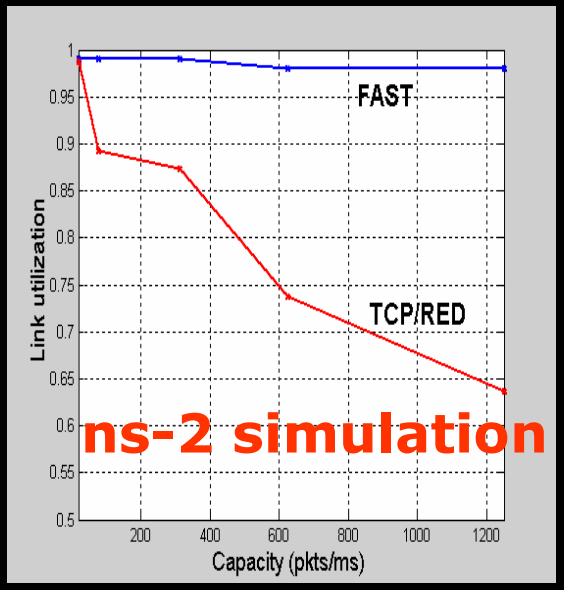
$$U_s^{reno/rem}(x_s) = (\log \phi)^{-1}\left(x \log\left(1 + \frac{2}{x_s^2 D_s^2}\right) + \frac{2\sqrt{2}}{D_s} \tan^{-1}\left(\frac{x_s D_s}{2}\right)\right)$$

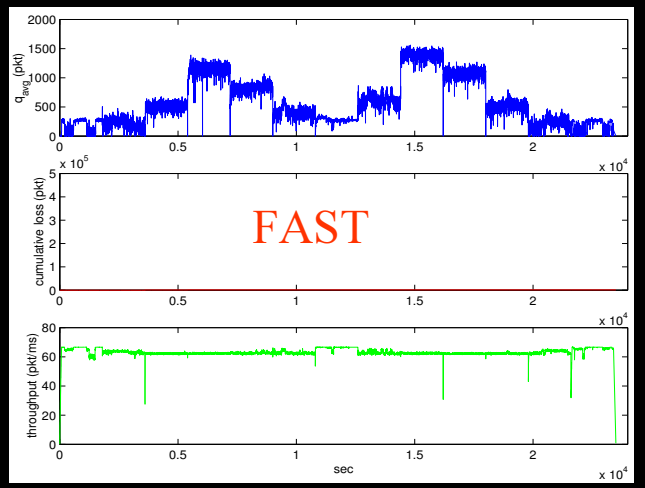
- Vegas, Vegas/REM

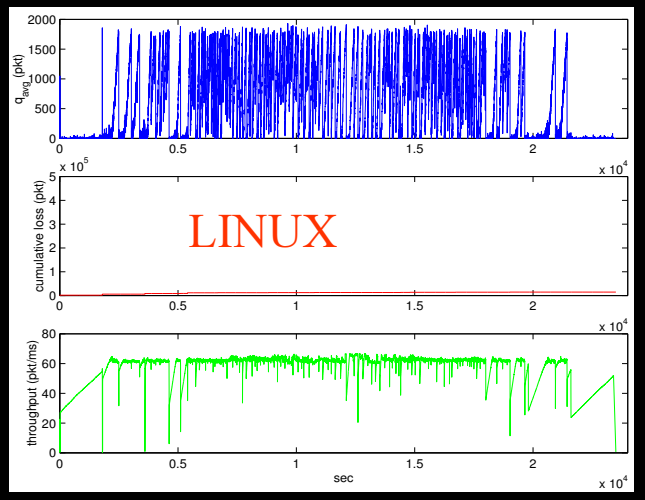
$$U_s^{vegas}(x_s) = \alpha_s d_s \log x_s$$

Caltech: FAST TCP protocol

- By analyzing these equations and considering the effects of different utility functions, we can derive better flow control algorithms directly from mathematics!







The Power of Mathematics

- The development of FAST represents the realization of every mathematician's dream:

  To come up with an equation, whose solution will change the world, making everybody's life better.

- And in this case, the dream was achieved by research done in Computer Science.
  Hah!

A Very Hard CS Problem

- What's going on in this video?

- How could we ever write a program to do that?!

Machines that Learn

- Instead of programming the solution to hard problems by hand, we can show the computer some examples of how to do what we want, and let it figure out the details for itself.
- For example, show it examples of pictures that contain faces and examples that do not:

Faces:

Not face:

You try it: Grus

Grus

Not Grus

Gru?

Learning from Examples

- What we write is actually a computer program that has lots of tuneable numbers inside it.
- Another program looks at the example data and modifies the numbers inside the first program to make it work better. How?

- Using statistics and optimization theory!

Computers that See

- Using this statistical learning approach, we can teach a computer to track moving objects in videos.

Computers Vision

- We can also learn to separate several moving objects from one another.

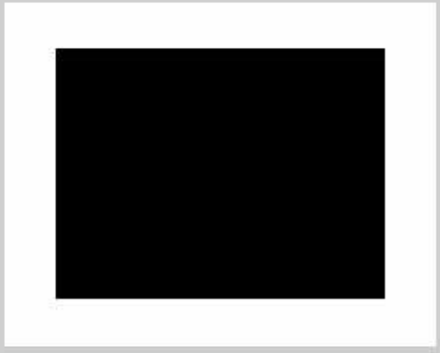
Flexible panoramic sprites, Jojic & Frey 2001

Original video

Learned sprites

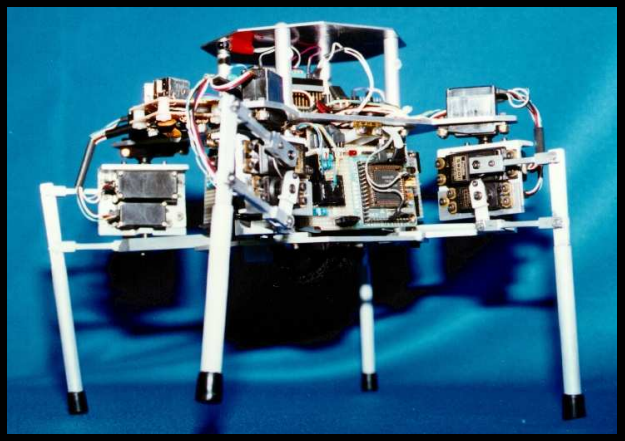
Learned mask means

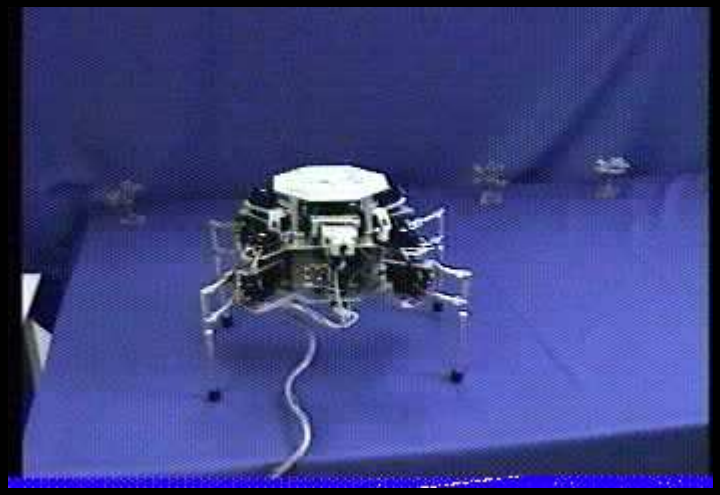
Sprite stabilization



Intelligent Robots

- We can even take a robot that we built, and automatically generate a program that makes it walk, by just telling it that we want it to move forward or stand up.
- This has also be done with submarines and helicopters!





Autonomous Helicopter Control

Robot Soccer

- Computer programs that run robot soccer players use machine learning also.

Machine Learning

- Learning algorithms automatically write programs to solve all kinds of very hard problems just by examing examples of the behaviours we want.
- The key is statistical modeling of the examples. The better our statistical assumptions, the more carefully we fit the models and the more data we have, the better everything works.
- This very cool area of computer science turns out to be mostly applied statistics and optimization!

Many more examples

- A CS question:
  Does the blue key in my web browser really keep my credit card number safe?
- This is the same question as:
  Is there any polynomial time algorithm to factor a large prime number?

Many more examples

- Scheduling of sports tournaments, factory jobs, or computer processes can often be reduced to an optimization problem of the form: maximize a convex function F(x) subject to convex constraints on x.
- Algorithms like linear programming, dynamic programming, semidefinite programming can solve these efficiently.

Conclusions

- Mathematics, Statistics and Computer Science are the perfect mix for doing fun research that actually impacts people.
- CS benefits from math and stats as the core engines used to attack, analyze and solve problems once they have been formalized.
- Math and stats benefit from CS as a venue in which their abstract power can be brought to bear on real world problem.

Research Groups in our Department

- We have professors working in all of the areas I have touched on, and more:
  - Artificial Intelligence, HCI & Graphics
  - Theory, Applied Discrete Math, Cryptography
  - Databases, Systems, Networks
  - Programming Languages, Software Engineering, Numerical Methods
- Come visit our table.
  Better yet, come to grad school here!