Something for almost nothing: algorithms for big data

Ronitt Rubinfeld

MIT CSAIL

Algorithms for REALLY big data

No time

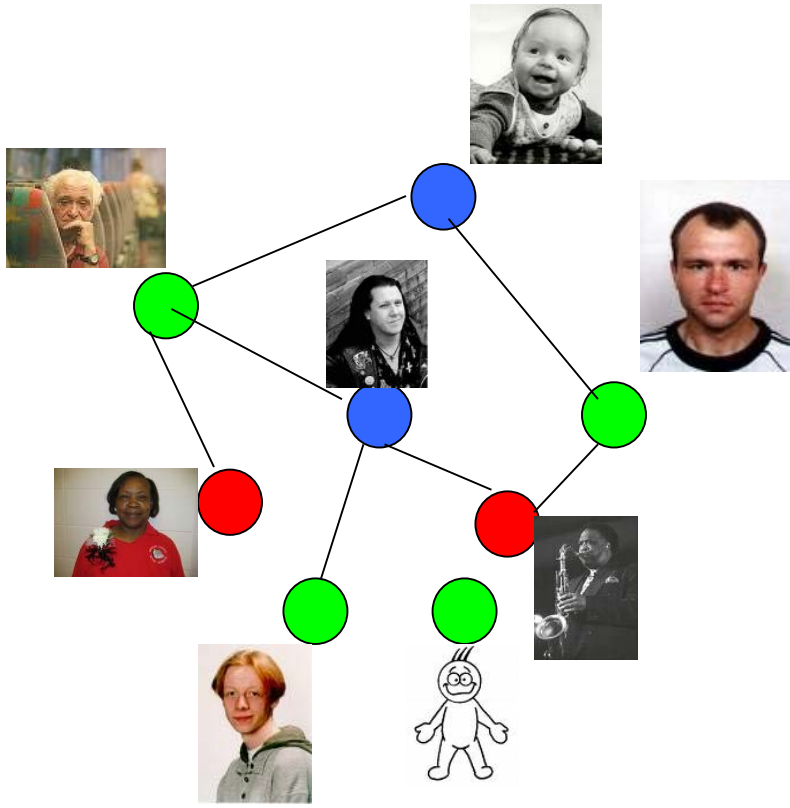
What can we hope to do without viewing most of the data?

Vast data

- Impossible to access all of it
- Accessible data is too enormous to be viewed by a single individual
- Once accessed, data can change

Small world phenomenon

- The social network is a graph:
  - "node" is a person
  - "edge" between people that know each other
- "6 degrees of separation"
  - Are all pairs of people connected by path of distance at most 6?



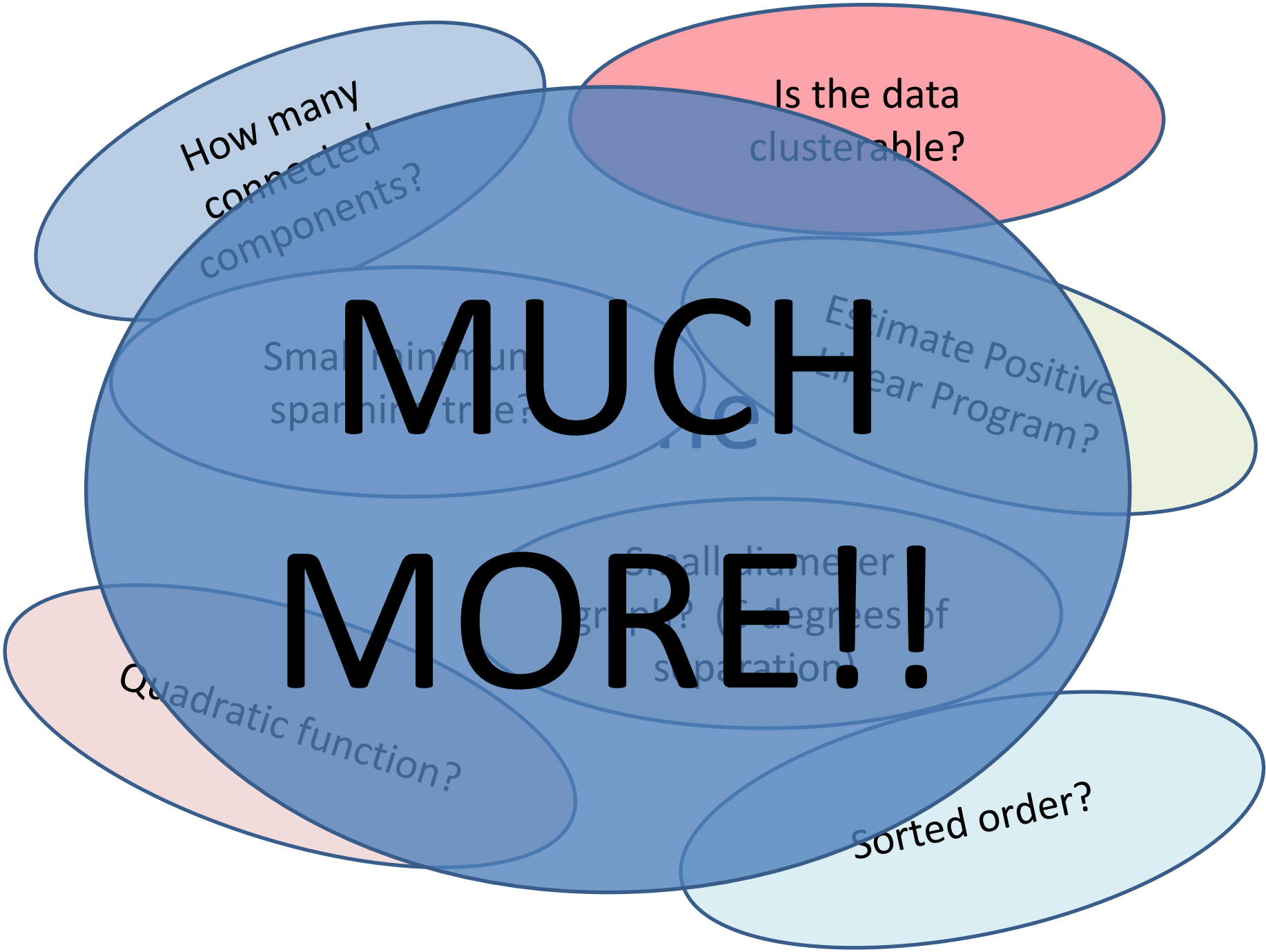
What can we hope to do without viewing most of the data?

- Can't answer "for all" or "exactly" type statements:
  - exactly how many individuals on earth are left-handed?
  - are all individuals connected by at most 6 degrees of separation?

- Compromise?
  - approximately how many individuals on earth are left-handed?
  - is there a very large group of individuals connected by at most 6 degrees of separation?

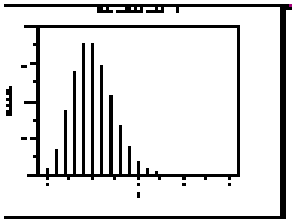
Sublinear time algorithms

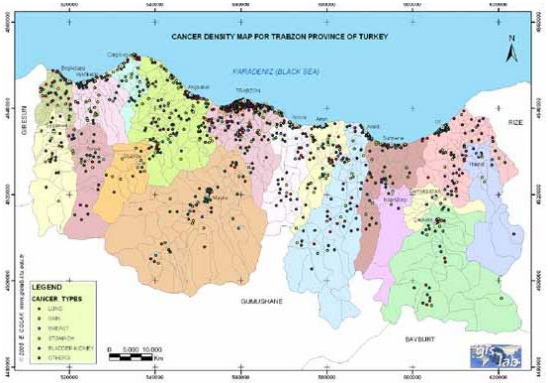
- Estimate value of *optimization problem* after viewing only small portion of data
  - Classically studied problems: Minimum spanning tree, vertex cover, max cut, positive linear program, edit distance, …
    - some problems are “hard” some are “easy”…

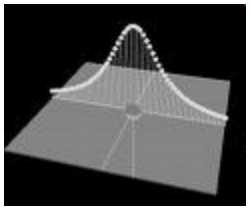

How many connected components?
Is the data clusterable?
Small minimum spanning tree?
Estimate Positive Linear Program?
Small diameter graph? (6 degrees of separation)
Quadratic function?
Sorted order?
MUCH MORE!!

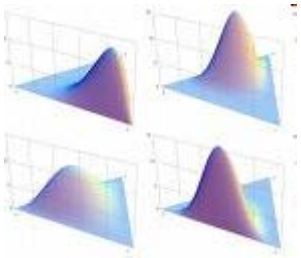
No samples

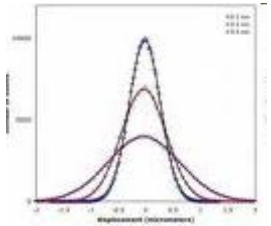
What if data only accessible via random samples?



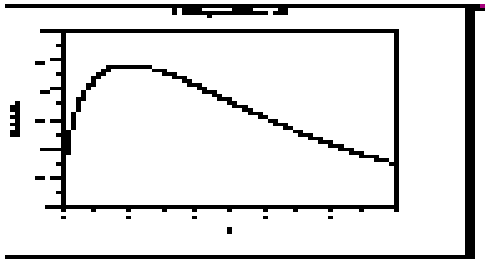


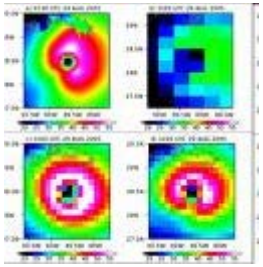


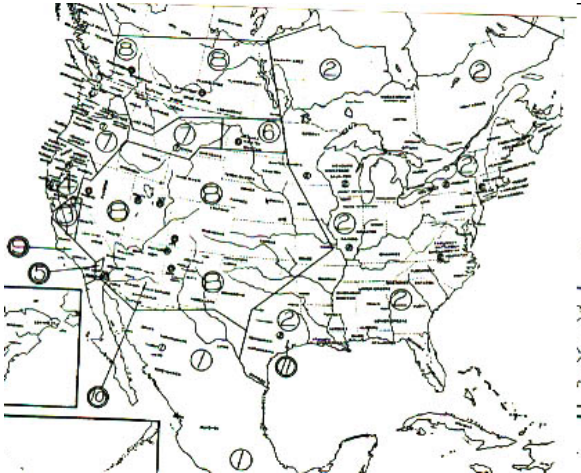




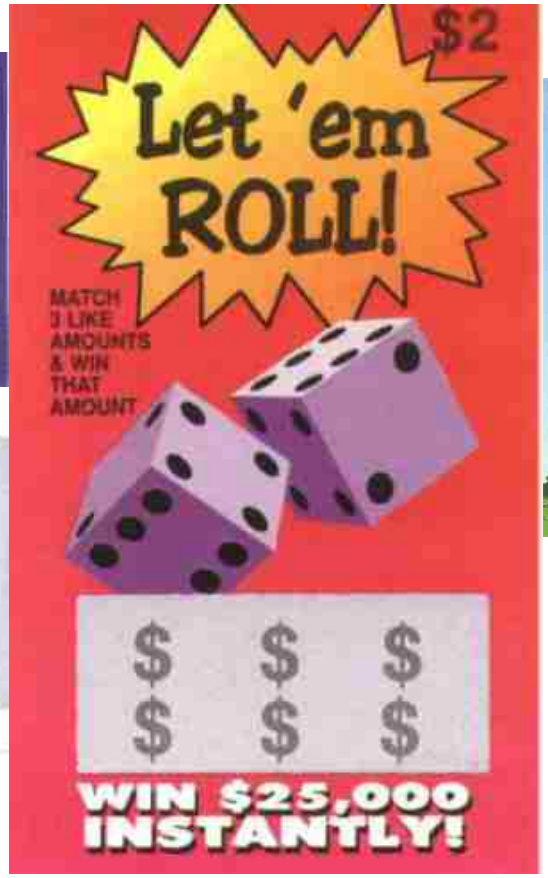
Distributions







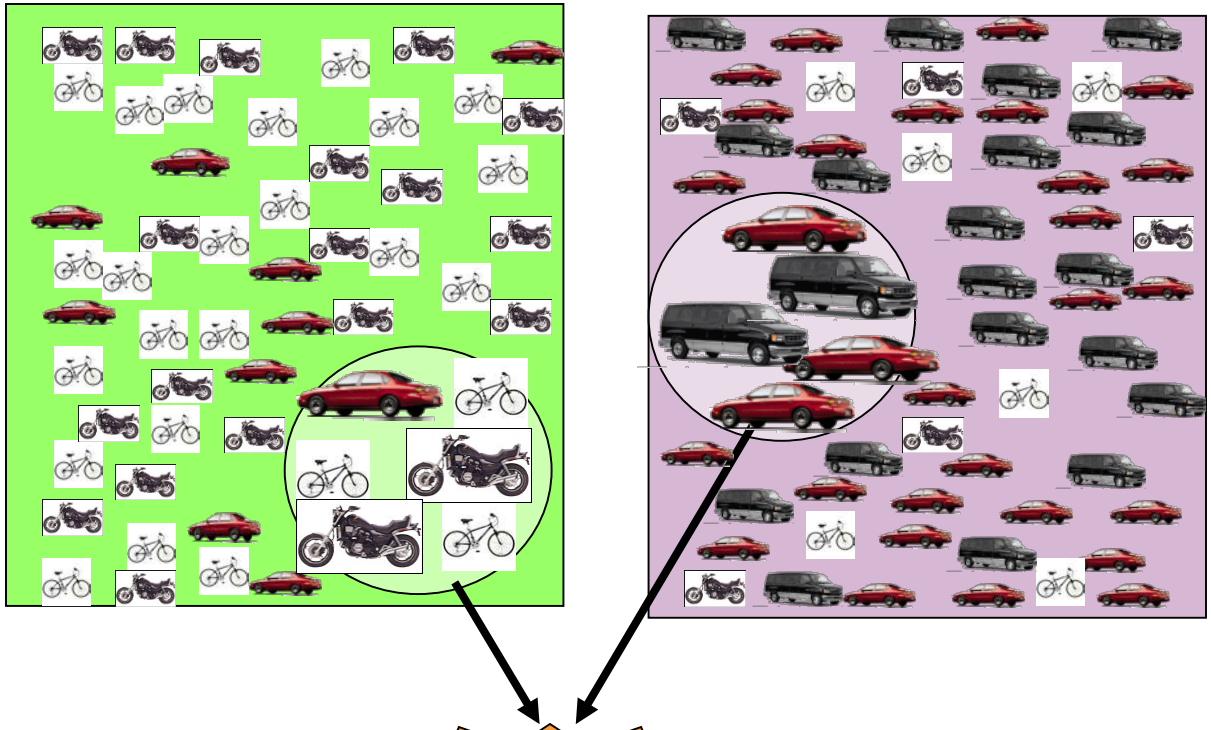
Play the lottery?



Testing closeness of two distributions:

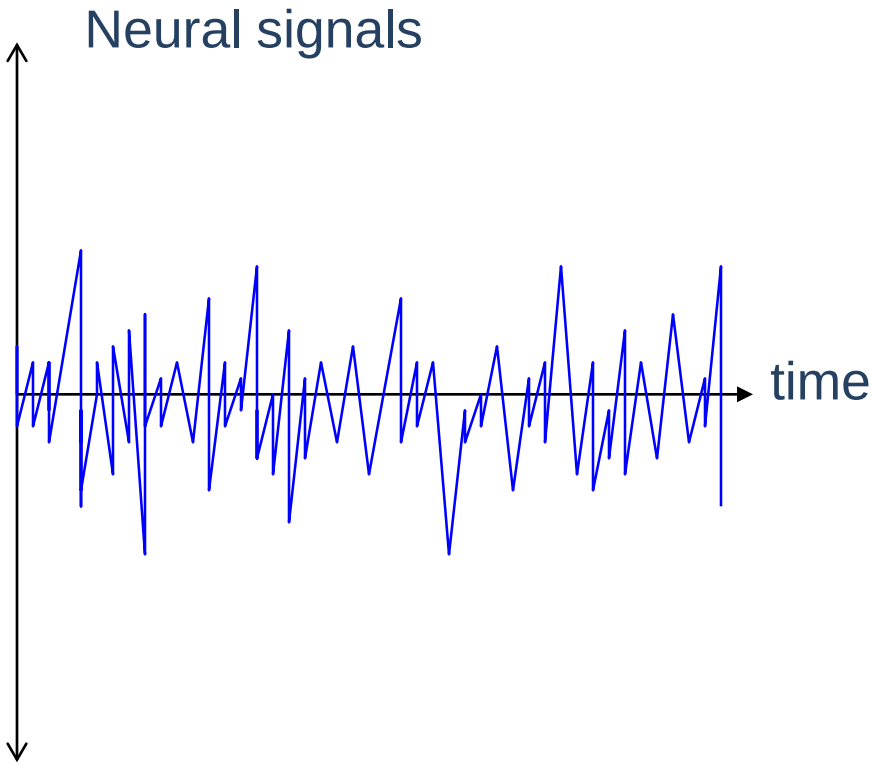
Transactions of 20-30 yr olds

Transactions of 30-40 yr olds



Information in neural spike trails

[Strong, Koberle, de Ruyter van Steveninck, Bialek '98]



Each application of stimuli gives sample of signal (spike trail)

Entropy of (discretized) signal indicates which neurons respond to stimuli

Working with samples of distributions on BIG domains

- Given samples of a distribution, need to know, e.g.,
  - entropy
  - number of distinct elements
  - "shape" (monotone, bimodal,…)
  - Independence/correlations
  - closeness to uniform, Gaussian, Zipfian…
- Questions considered in statistics, information theory/EE, machine learning, databases, physics, biology,…
- Key question:
  - Without assumptions on the distribution, how many samples do you need in terms of domain size?
    - Do you need to estimate the probabilities of each domain item?

Distributions on BIG domains

- Previous work (including in other communities):
  - E.g., Can you beat *n log n* samples for estimating entropy?
    - *O(n)* samples is a big success!

- Great progress!
  - Some optimal bounds:
    - Additive estimates of entropy, support size, closeness of two distributions: $n/\log n$ [Valiant Valiant 2011]
    - 2-multiplicative estimate of entropy: $n^{1/4}$ [Batu Dasgupta Kumar Rubinfeld 2005] [Raskhonikova Ron Shpilka Smith 2007] [Valiant 2008]
    - Two distributions - the same or far (in L1 distance)? $n^{2/3}$ [Goldreich Ron][Batu Fortnow Rubinfeld Smith White 2000] [Valiant 2008]
  - Can do better for data with special properties

In conclusion

- Lots of important algorithmic questions
- Lots of algorithmic ideas
- Lots left to think about!

Overview

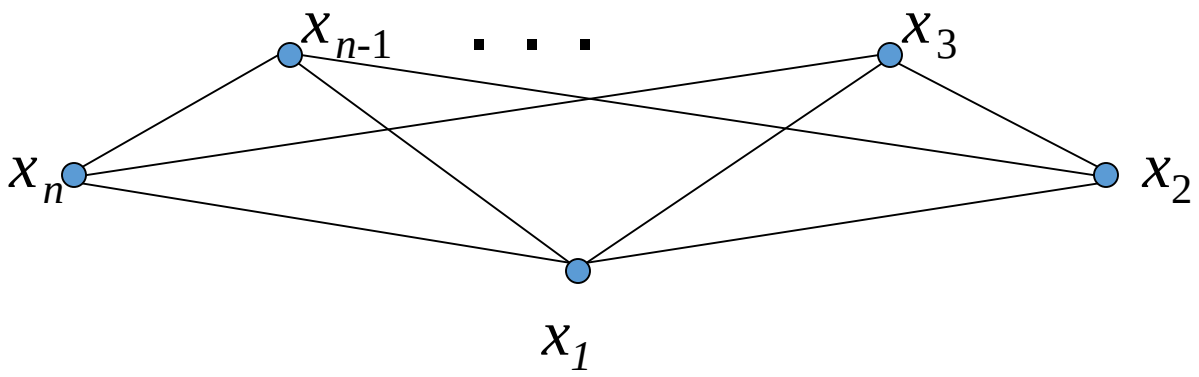
- Intuitive ideas of secrecy
  - Encrypted data are safe; but they appear to be unusable
- If people have secrets, and want to do a joint computation
  - They can use a trusted third party
  - Dating, Millionaire's, Sorority pledges, Battle-unit coordination, Satellites
- Modern cryptography
  - You may not have to decrypt
  - There may not need a trusted third party

- Have your cake and eat it too
- Secret inputs; joint computation of desired output
- Yao, Nebuchadnezzar/Daniel
  - Trusted Third Party: God
- SMPC or Secure Function Evaluation (SFE)

Secure Function Evaluation Contd.

- SMPC or Secure Function Evaluation (SFE)
- State of the art
  - 2P-SFE: Currently fast enough for many applications
  - S**M**PC: Fast enough in some cases
  - Fully Homomorphic Encryption: Slow, but improving
- We'll be looking at 2P-SFE using *garbled circuits*

Secure Function Evaluation



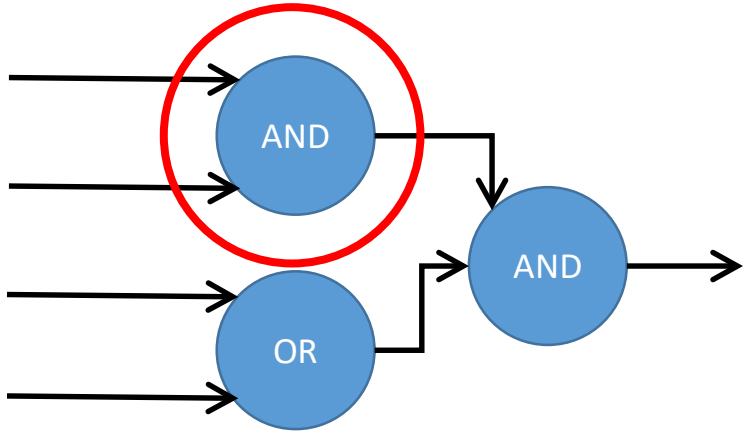
$y = F(x_1, \ldots, x_n)$

- Each *i* learns *y*
- No *i* can learn anything about x_j
  (except what he can infer from x_i and *y*)
- Very general positive results (*e.g.*, GMW87, BGW88)
- Not very widely used in practice … YET!

Garbled Circuits

- Function represented as a Boolean circuit
- Generator encrypts/garbles the circuit
- Evaluator evaluates the circuit

Generating a Garbled Circuit



A	B	C
0	0	c_0
0	1	c_1
1	0	c_2
1	1	c_3

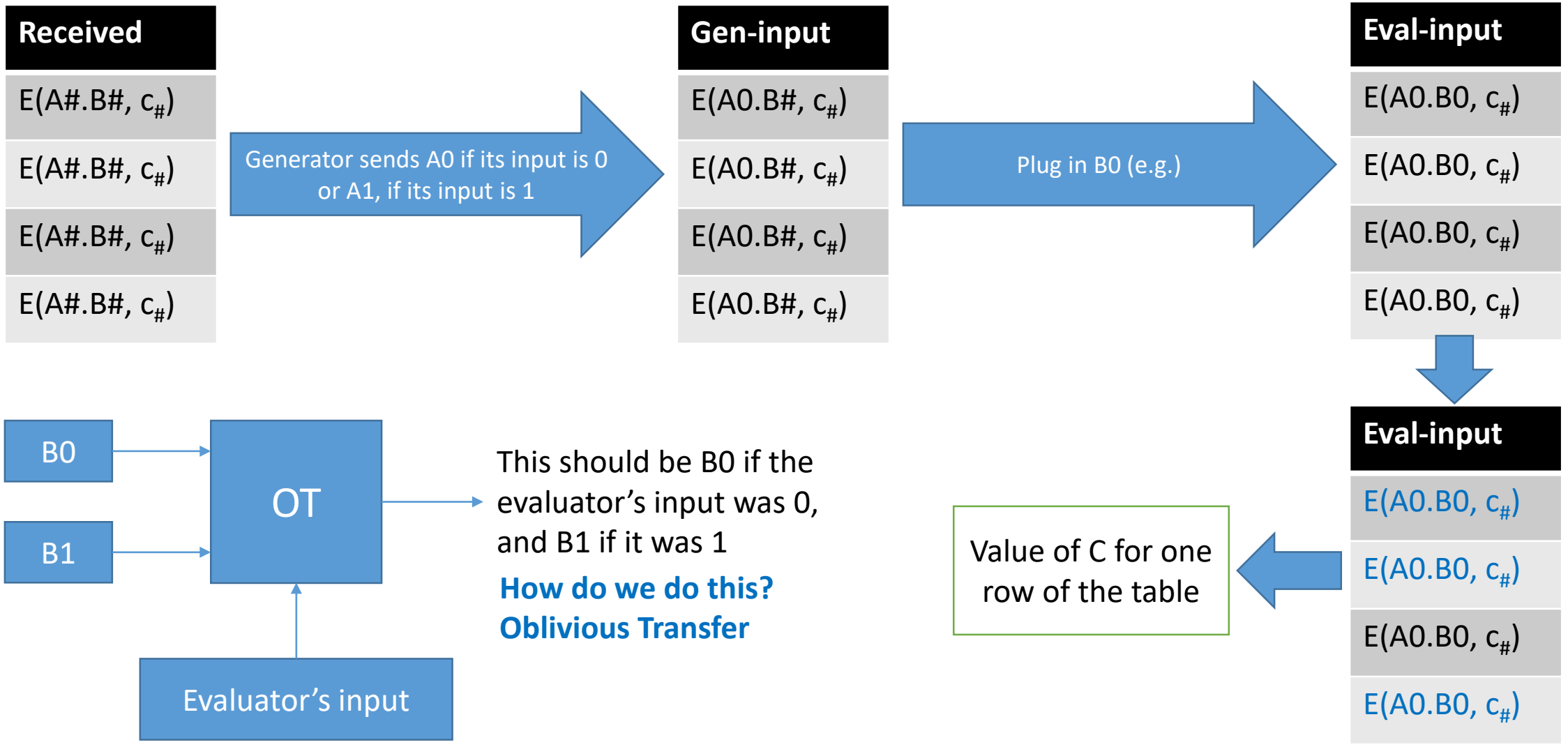
Generator creates 4 keys, one each for A=0, A=1, B=0, and B=1

A	B	C
0	0	E(A0.B0, c_0)
0	1	E(A0.B1, c_1)
1	0	E(A1.B0, c_2)
1	1	E(A1.B1, c_3)

Permute
E(A1.B1, c_3)
E(A1.B0, c_2)
E(A0.B0, c_0)
E(A0.B1, c_1)

This is sent to the evaluator

Evaluating a Garbled Circuit



The End

- Questions?

6.006	Lecture 24	Dec. 13, 2016

Erik's main research areas:

http://erikdemaine.org/

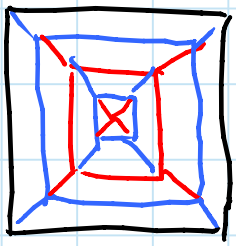
- computational geometry [6.850]
  - geometric folding algorithms [6.849]
  - self-assembly
- data structures [6.851]
- graph algorithms [6.889]
- recreational algorithms/hardness [6.890 & SP.268]
- algorithmic sculpture

Geometric folding algorithms:

NEXT SEMESTER! SPRING'17 [6.849, videos online]

- design: algorithms to fold any polyhedral surface from a square of paper
  [Demaine, Demaine, Mitchell 2000; Demaine & Tachi 2016]
  - bicolor paper ⇒ can 2-color faces
  - OPEN: how to best optimize "scale factor"
  - e.g. best $n \times n$ checkerboard folding DEMO
    recently improved from $\sim n/2 \to \sim n/4$
- foldability: given a crease pattern, can you fold it flat?
  - NP-complete in general [Bern & Hayes 1996]
    [Akitaya, Cheung, Demaine, Horiyama, Hull, Ku, Tachi, Uehara 2015]
  - OPEN: $m \times n$ map with creases specified as mountain/valley [Edmonds 1997]
  - recently solved: $2 \times n$ [Demaine, Liu, Morgan 2012]

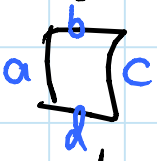
- hyperbolic paraboloid [Bauhaus 1929] doesn't exist! DEMO
  [Demaine, Demaine, Hart, Price, Tachi 2009]



- understanding circular creases DEMO
  [Demaine, Demaine, Huffman, Koshitz, Tachi 2014]
- any straight-line graph can be made by folding flat & one straight cut DEMO
  [Demaine, Demaine, Lubiw 1998; Bern, Demaine, Eppstein, Hayes 1999]

Self-assembly: geometric model of computation

- glue: e.g. DNA strands, each pair has strength
- square tiles with glue on each side

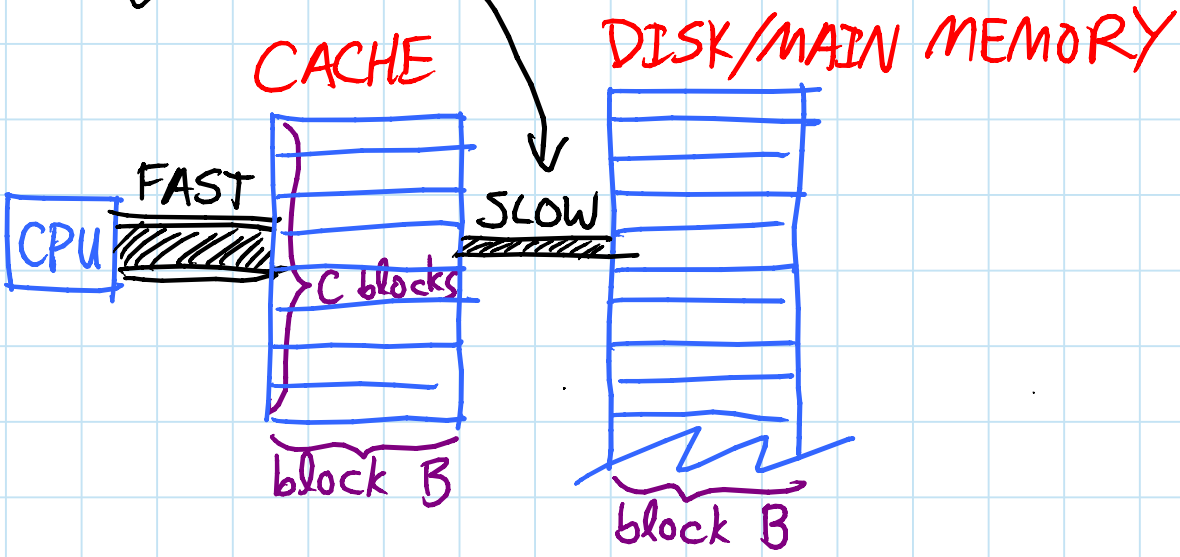


- Brownian motion: tiles/constructions stick together if $\sum$ glue strengths $\geq$ temperature
- can build $n \times n$ square using $O\left(\frac{\lg n}{\lg \lg n}\right)$ tiles
  [Rothemund & Winfree 2000]
  or using $O(1)$ tiles & $O(\lg n)$ "stages"
  algorithmic steps by the bioengineer
  [Demaine, Demaine, Fekete, Ishaque, Rafalin, Schweller, Souvaine 2007]
- can replicate ∞ copies of given unknown shape using $O(1)$ tiles & $O(1)$ stages
  [Abel, Benbernou, Damian, Demaine, Demaine, Flatland, Kominers, Schweller 2010] [Chalk, Demaine, Demaine, Martinez, Schweller, Vega, Wylie – SODA 2017]

Data structures: [6.851, videos online]

- integer data structures: store n integers in $\{0, 1, \ldots, u-1\}$ subject to insert, delete, predecessor, successor (on word RAM)
  - hashing does exact search in $O(1)$
  - AVL trees do all in $O(\lg n)$
  - $O(\lg \lg u)$/op. [van Emde Boas]
  - $O(\lg n / \lg \lg u)$/op. [fusion trees: Fredman & Willard]
  - $O(\sqrt{\lg n / \lg \lg n})$/op. [min of above]

- cache-efficient data structures:
  - memory transfers happen in blocks



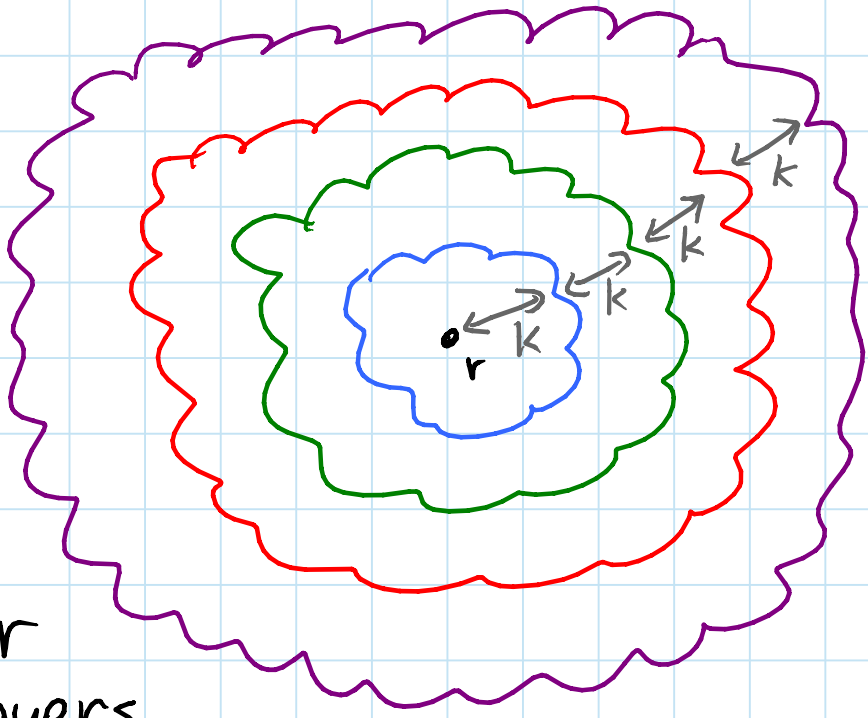
  - searching takes $\Theta(\log_B N)$ transfers (vs. $\lg n$)
  - sorting takes $\Theta(\frac{N}{B} \log_C \frac{N}{B})$ transfers

  - possible even if you don't know B & C!

(Almost) planar graphs: [6.889, videos online]

- Dijkstra in $O(n)$ time
  [Henzinger, Klein, Rao, Subramanian 1997]
- Bellman-Ford in $O(n \lg^2 n / \lg\lg n)$ time
  [Mozes & Wolff-Nilson 2010]

- many problems NP-hard, even on planar graphs
- but can find a solution within $1+\varepsilon$ factor of optimal, for any ε
- run BFS from any root vertex r
- delete every k layers (guess initial offset)
- for many problems, solution messed up by only $1 + 1/k$ factor ($\Rightarrow k = 1/\varepsilon$)
- connected components of remaining graph have $<k$ layers $\sim$ can solve via DP typically in $\sim 2^k \cdot n$ time



[Baker 1994 & others]

Recreational algorithms:

- many algorithms & complexities of games
  [6.890, SP.268, & our book *Games, Puzzles, & Computation* (2009)] DEMO
- $n \times n \times n$ Rubik's Cube diameter is $\Theta(n^2/\lg n)$
  [Demaine, Demaine, Eisenstat, Lubiw, Winslow 2011] DEMO
- Tetris is NP-complete
  [Breukelaar, Demaine, Hohenberger, Hoogeboom, Kosters, Liben-Nowell 2004]
- Super Mario Bros., Legend of Zelda, Donkey Kong Country, Pokémon NP-complete
  [Aloupis, Demaine, Guo 2012]
- Super Mario Bros. is PSPACE-complete!
  [Demaine, Viglietta, Williams 2016]
- Portal is PSPACE-complete
  [Demaine, Lockhart, Lynch 2016]
- balloon twisting any polyhedron DEMO
  [Demaine, Demaine, Hart 2008]
- algorithmic magic tricks
  - coin flipping
    [Benbernou, Demaine, Demaine, Rossman 2008]
  - picture hanging DEMO
    [Demaine, Demaine, Minsky, Mitchell, Rivest, Pătraşcu 2012]

Algorithms classes at MIT: (post-6.006)

- #1: 6.046: Intermediate Algorithms
  (more adv. algorithms & analysis, less coding)
- 6.047: Computational Biology
  (genomes, phylogeny, etc.)
- 6.854: Advanced Algorithms
  (intense survey of whole field)
- 6.850: Geometric Computing
  (working with points, lines, polygons, meshes, ...)
- 6.849: Geometric Folding Algorithms
  (origami, robot arms, protein folding, ...)
- 6.851: Advanced Data Structures
  (sublogarithmic performance)
- 6.852: Distributed Algorithms
  (reaching consensus in a network with faults)
- 6.853: Algorithmic Game Theory
  (Nash equilibria, auction mechanism design, ...)
- 6.855: Network Optimization
  (optimization in graph: beyond shortest paths)
- 6.856: Randomized Algorithms
  (how randomness makes algs. simpler & faster)
- 6.857: Network and Computer Security
  (cryptography)
- 6.816: Multicore Programming

Other theory classes:

- 6.045: Automata, Computability, & Complexity
- 6.840: Theory of Computing
- 6.841: Advanced Complexity Theory
- 6.842: Randomness & Computation
- 6.845: Quantum Complexity Theory
- 6.440: Essential Coding Theory
- 6.441: Information Theory