Lecture 38

CSE 331

Dec 5, 2018

Quiz 2 is graded

note

0 views

Actions

Quiz 2 grade and stats

Quiz 2 has been graded and scores have been released on Autolab.

Here are the **stats:**

Problem	Mean	Median	StdDev	Max	Min
Q1(a)	1.3	2.0	1.0	2.0	0.0
Q1(b)	1.5	2.0	0.8	2.0	0.0
Q2- part 1	0.9	1.0	0.9	2.0	0.0
Q2- part 2	1.2	1.0	1.1	4.0	0.0
Total	4.9	5.0	2.2	10.0	0.0

Peer eval grades assigned by SATURDAY

Q&A session: Friday lecture

Extra OH on Friday

note

stop following 107 views

Actions

Extra OH on Friday, Dec 7

In prep for the final exam (and in particular, to give y'all an opportunity to pickup HW solutions before the exam), the following TAs will hold the following extra OH (all in Salvador Lounge):

- Iman, 11am-1pm
- Angus, 1:30-3pm
- Charles, 3-5pm
- Steven, 5-6pm

#pin

office_hours final

edit good note 0

Updated 4 days ago by Atri Rudra

Bring UB card to final exam

note

0 views

Assigned seating for final exam

Your seating for the final in Norton 112 will be assigned (and won't be sit where you find a spot as it was for the mid-term).

I will release more details by Saturday. In the meantime, two important things to remember:

- **You will HAVE to have your UB card on you during the exam**
    - A TA will come and verify that you are seated in the correct row
- To facilitate the TAs checking your UB IDs, **please keep your bag in the front of the room** (i.e. not with you).

final

edit good note 0

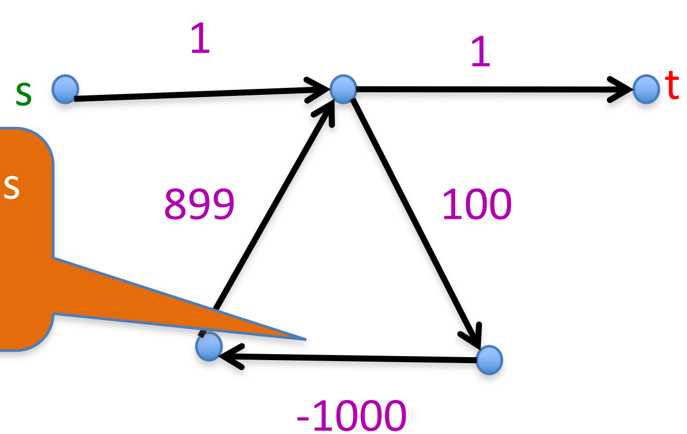
Updated Just now by Atri Rudra

Shortest Path Problem

Input: (Directed) Graph $G=(V,E)$ and for every edge e has a cost c_e (can be <0)

t in V

Output: Shortest path from every s to t



Shortest path has cost negative infinity

Assume that G has no negative cycle

Bellman-Ford Algorithm

Runs in $O(n(m+n))$ time

Only needs $O(n)$ additional space

Reading Assignment

Sec 6.8 of [KT]

Longest path problem

Given G, does there exist a simple path of length n-1 ?

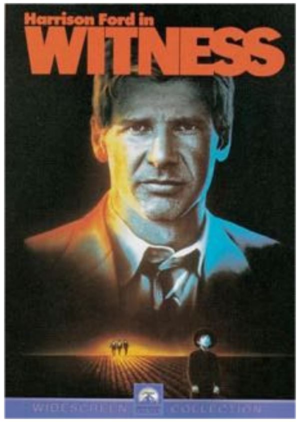
Longest vs Shortest Paths

Two sides of the “same” coin

Shortest Path problem

Can be solved by a polynomial time algorithm

Is there a longest path of length n-1?



Given a path can verify in polynomial time if the answer is yes

Poly time algo for longest path?

P vs NP question

P: problems that can be solved by poly time algorithms

Is P=NP?

NP: problems that have polynomial time verifiable witness to optimal solution

Alternate NP definition: Guess witness and verify!

Proving P ≠ NP

Pick any one problem in NP and show it cannot be solved in poly time

Pretty much all known proof techniques *provably* will not work

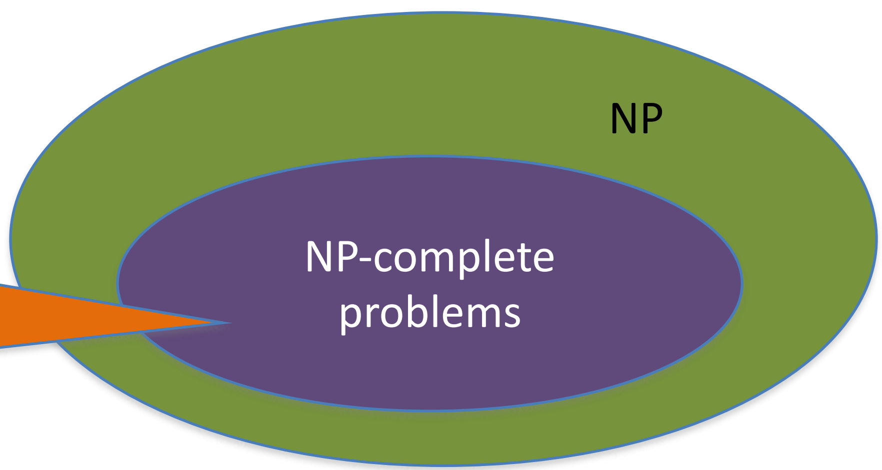
Proving P = NP

Will make cryptography collapse

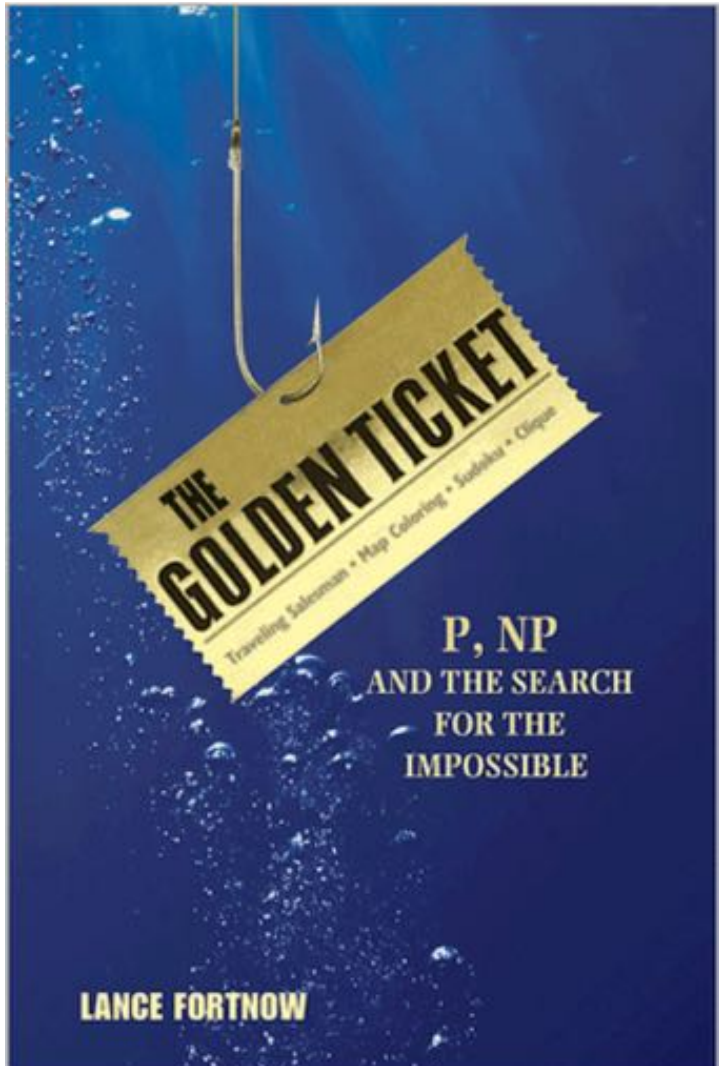
Compute the encryption key!

Prove that all problems in NP can be solved by polynomial time algorithms

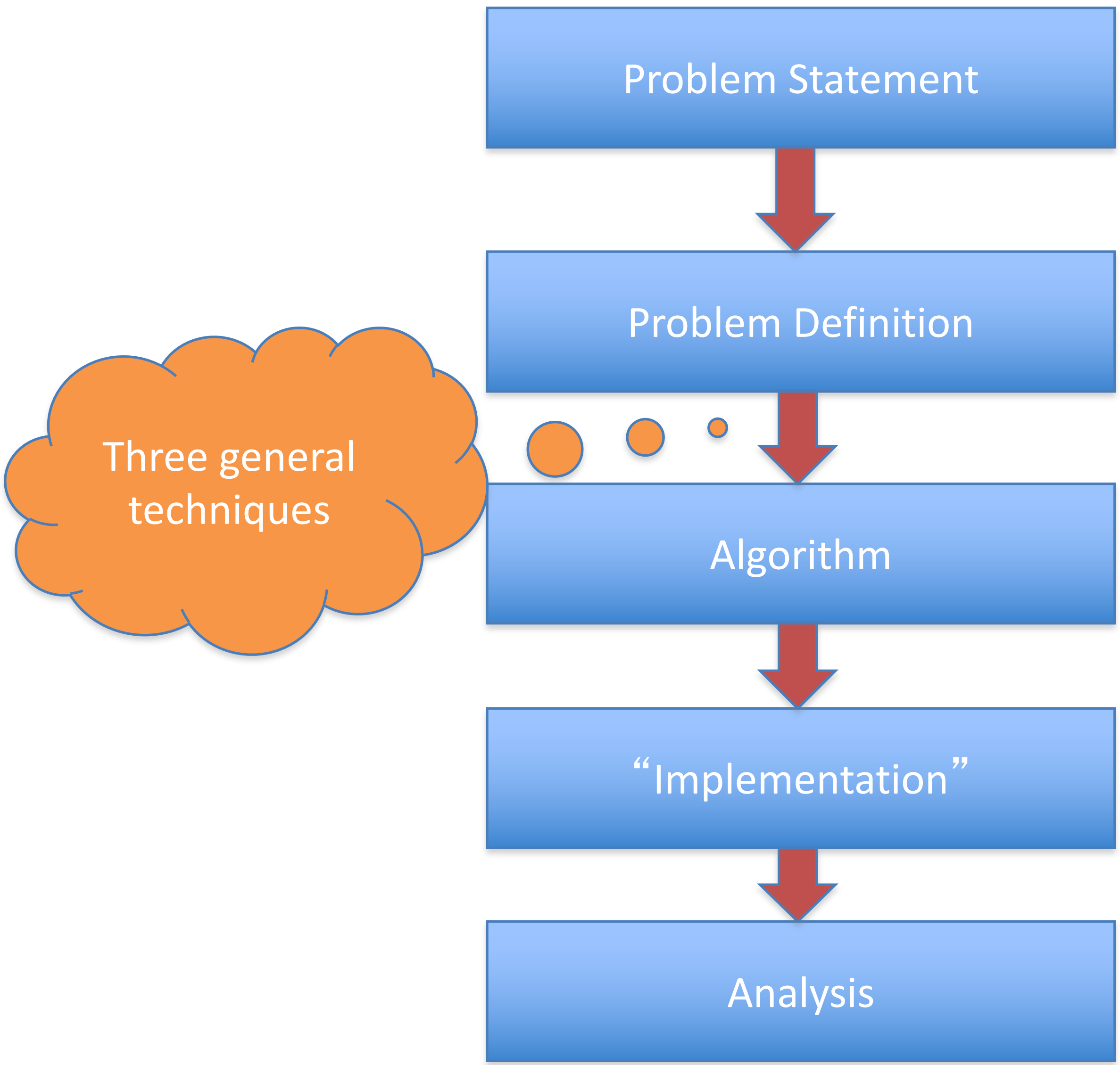
Solving any ONE problem in here in poly time will prove P=NP!



A book on P vs. NP



High level view of CSE 331



Data Structures

Correctness+Runtime Analysis

If you are curious for more

CSE 429 or 431: Algorithms

CSE 396: Theory of Computation

Now relax…

Randomized algorithms

http://calculator.mathcaptain.com/coin-toss-probability-calculator.html

What is different?

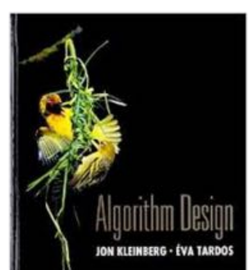
Algorithms can toss coins and make decisions

A Representative Problem

Hashing

Further Reading

Chapter 13 of the textbook



CSE 430/432 in Spring 19!

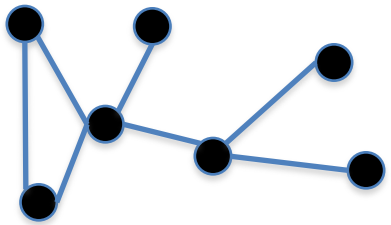
Approximation algorithms

What is different?

Algorithms can output a solution that is say 50% as good as the optimal

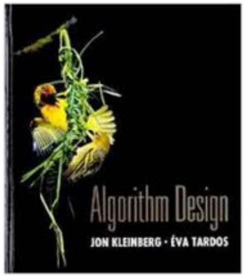
A Representative Problem

Vertex Cover



Further Reading

Chapter 12 of the textbook



Online algorithms

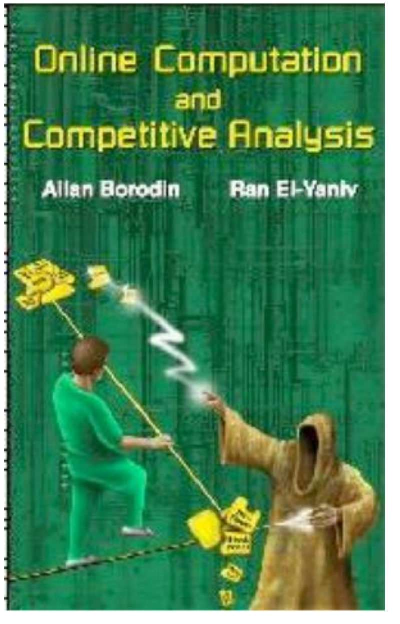
What is different?

Algorithms have to make decisions before they see all the input

A Representative Problem

Secretary Problem

Further Reading



Data streaming algorithms

https://www.flickr.com/photos/midom/2134991985/

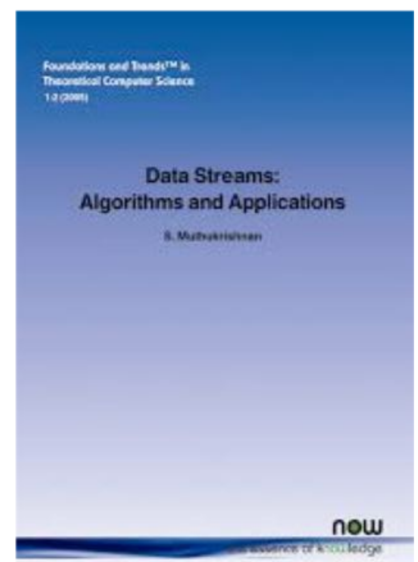
What is different?

One pass on the input with severely limited memory

A Representative Problem

Compute the top-10 source IP addresses

Further Reading



Distributed algorithms

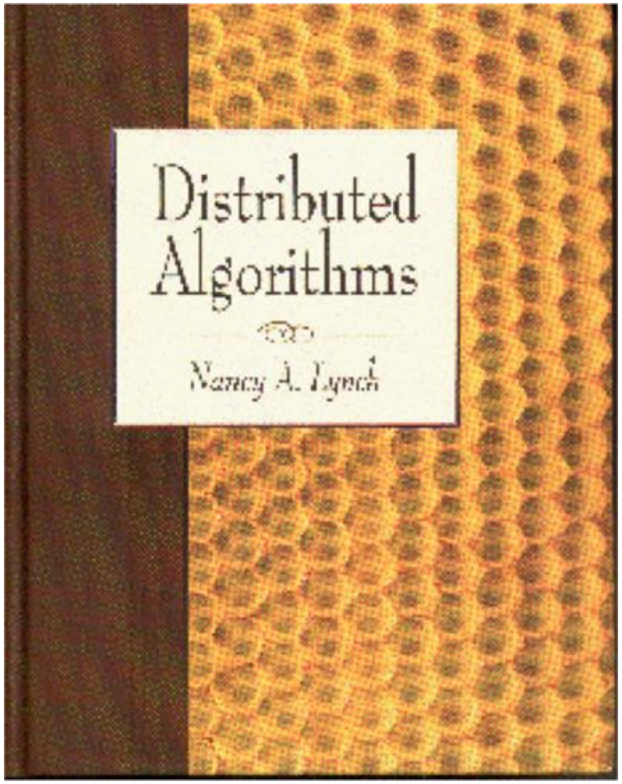
What is different?

Input is distributed over a network

A Representative Problem

Consensus

Further Reading



Beyond-worst case analysis

What is different?

Analyze algorithms in a more instance specific way

A Representative Problem

Intersect two sorted sets

Further Reading

http://theory.stanford.edu/~tim/f14/f14.html

Algorithms for Data Science

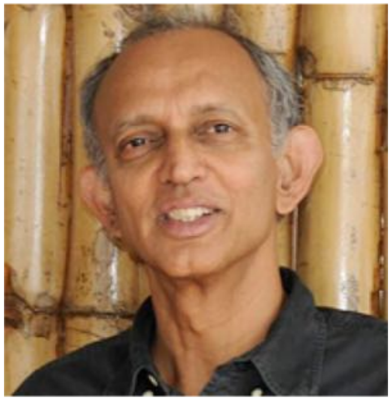
What is different?

Algorithms for non-discrete inputs

A Representative Problem

Compute Eigenvalues

Further Reading

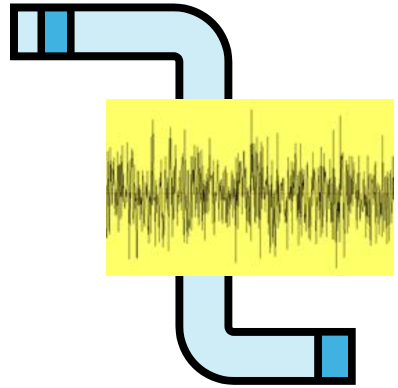


http://research.microsoft.com/en-US/people/kannan/book-no-solutions-aug-21-2014.pdf

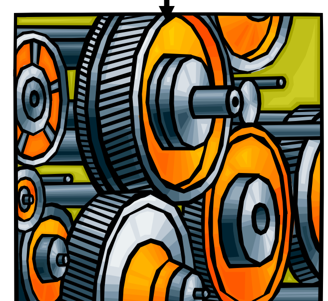
Communicating with my 3 year old

x

C(x)



y = C(x)+error



x

Give up

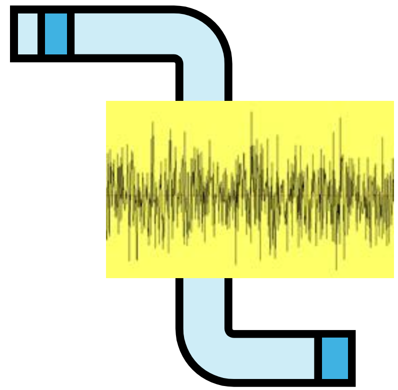
"Code" **C**

- "Kiran English"
- C(x) is a "codeword"

The setup

x

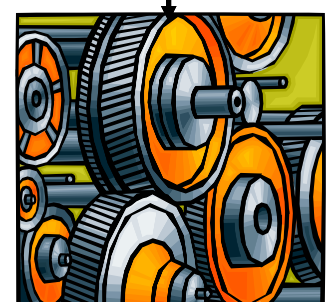
C(x)



y = C(x)+error

Mapping **C**

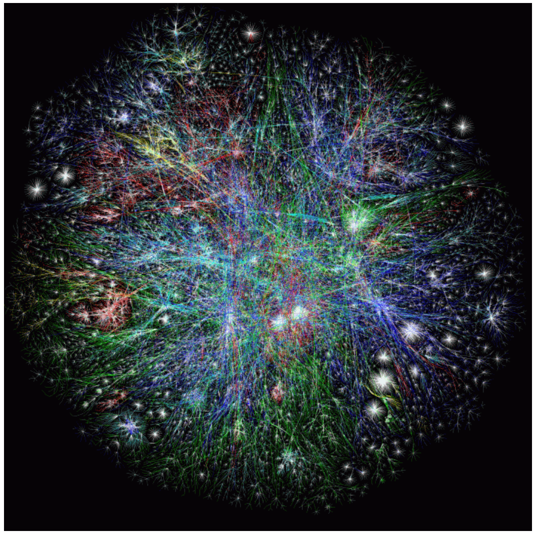
- Error-correcting code or just code
- Encoding: x → C(x)
- Decoding: y → x
- C(x) is a codeword



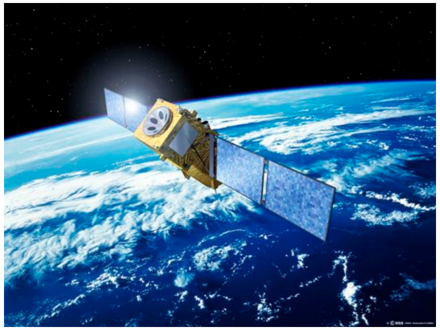
x

Give up

Different Channels and Codes

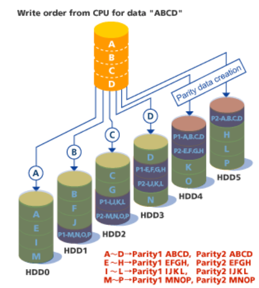


- Internet
  - Checksum used in mult layers of TCP/IP stack
- Cell phones
- Satellite broadcast
  - TV
- Deep space telecommunications
  - Mars Rover

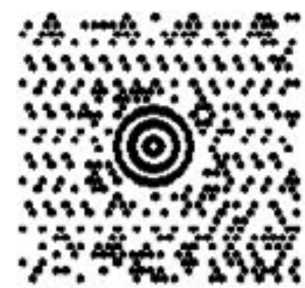


“Unusual” Channels

- Data Storage
  - CDs and DVDs
  - RAID
  - ECC memory



- Paper bar codes
  - UPS (MaxiCode)



Codes are all around us

Redundancy vs. Error-correction

- Repetition code: Repeat every bit say 100 times
  - Good error correcting properties
  - Too much redundancy
- Parity code: Add a parity bit
  - Minimum amount of redundancy
  - Bad error correcting properties
    - Two errors go completely undetected
- Neither of these codes are satisfactory

1 1 1 0 0	1

1 0 0 0 0	1

Two main challenges in coding theory

- Problem with parity example
  - Messages mapped to codewords which do not differ in many places
- Need to pick a lot of codewords that differ a lot from each other

- Efficient decoding
  - Naive algorithm: check received word with all codewords

The fundamental tradeoff

- Correct as many errors as possible with as little redundancy as possible

Can one achieve the "optimal" tradeoff with *efficient* encoding and decoding ?

Interested in more?

CSE 445/545, Spring 2019

Superfast Matrix Vector Multiplication (and some deep learning mumbo-jumbo)

Stanford
University

$Ax = y$

$$\begin{bmatrix} a_{0,0} & a_{0,2} & \cdots & a_{0,N-1} \\ \vdots & \vdots & \cdots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,N-1} \end{bmatrix} \times \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_{N-1} \end{bmatrix}$$

$$\begin{array}{ccc} A & x & y \end{array}$$

$\Theta(N^2)$ time in worst-case

In practice A has structure

$$\begin{bmatrix} a_{0,0} & a_{0,2} & \cdots & a_{0,N-1} \\ \vdots & \vdots & \cdots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,N-1} \end{bmatrix} \times \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ \vdots \\ y_{N-1} \end{bmatrix}$$

$$\underset{A}{} \qquad \underset{x}{} \qquad \underset{y}{}$$

Can we exploit the structure for faster algorithms?

Discrete Fourier Transform

$$
\begin{bmatrix}
a_{0,0} & a_{0,2} & \cdots & a_{0,N-1} \\
\vdots & \vdots & \cdots & \vdots \\
a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,N-1}
\end{bmatrix}
\times
\begin{bmatrix}
b_0 \\
\vdots \\
b_{N-1}
\end{bmatrix}
$$

A b

$$\boxed{a_{x,y} = \exp(2\pi i\, x \bullet y/N)}$$

Cooley

Tukey

FFT (1965)

Can compute DFT in O(N log N) time

Cauchy Matrix

$$\underset{A}{\begin{bmatrix} a_{0,0} & a_{0,2} & \cdots & a_{0,N-1} \\ \vdots & \vdots & \cdots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,N-1} \end{bmatrix}} \times \underset{b}{\begin{bmatrix} b_0 \\ \vdots \\ b_{N-1} \end{bmatrix}}$$

$$a_{x,y} = \frac{1}{r_x - s_y}$$

Can be computed in $O(N \log^2 N)$ time

Superfast = N poly-log(N)

The main Question

What is the largest class of matrices A for which we can have superfast algo to compute Ax?

Structure 1: Recurrence

$$\boxed{a_{x,y} = \exp(2\pi i\, x\bullet y/N)}$$

$$a_{x,y+1} = a_{x,y}\bullet \boxed{\exp(2\pi i\, x/N)}$$

Multiplier only depends on x

$$\begin{bmatrix} a_{0,0} & a_{0,2} & \cdots & a_{0,N-1} \\ \vdots & \vdots & \cdots & \vdots \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,N-1} \end{bmatrix} \times \begin{bmatrix} b_0 \\ \vdots \\ b_{N-1} \end{bmatrix}$$

A b

“Multiplier matrix” only depends on x

$$\begin{bmatrix} a_{0,0} & a_{0,2} & \cdots & a_{0,N-1} \\ \vdots & \vdots \quad a_{x,y} & & \vdots \\ a_{N-1,0} & a_{N-1,1} & \cdots & a_{N-1,N-1} \end{bmatrix}$$

Structure 2: Low Displacement Rank

$$\boxed{a_{x,y} = \frac{1}{r_x - s_y}}$$

$$r_x \bullet a_{x,y} - a_{x,y} \bullet s_y = 1$$

LA – AR has low rank

$$\begin{bmatrix} r_0 & & 0 \\ & \ddots & \\ 0 & & r_{N-1} \end{bmatrix} \times A - A \times \begin{bmatrix} s_0 & & 0 \\ & \ddots & \\ 0 & & s_{N-1} \end{bmatrix} = 1$$

Known Results

Recurrences

Low Displacement Rank

$O(N \log^2 N)$

Driscoll Healy Rockmore

FFT

Olshevsky Shokrollahi

$O(N \log^2 N)$

Our Main Result*

One Result that recovers all existing results*

Recurrences

Low Displacement Rank

O(N $\log^2$ N)

Driscoll Healy Rockmore

FFT

Olshevsky Shokrollahi

O(N $\log^2$ N)

For more….

arXiv.org > cs > arXiv:1611.01569

Search or Artic

Help | Advanced sea

Computer Science > Data Structures and Algorithms

A Two Pronged Progress in Structured Dense Matrix Multiplication

Christopher De Sa, Albert Gu, Rohan Puttagunta, Christopher Ré, Atri Rudra

(Submitted on 4 Nov 2016 (v1), last revised 18 Nov 2017 (this version, v3))

Matrix-vector multiplication is one of the most fundamental computing primitives. Given a matrix $A \in \mathbb{F}^{N\times N}$ and a vector b, it is known that in the worst case $\Theta(N^2)$ operations over $\mathbb{F}$ are needed to compute Ab. A broad question is to identify classes of structured dense matrices that can be represented with $O(N)$ parameters, and for which matrix-vector multiplication can be performed sub-quadratically. One such class of structured matrices is the orthogonal polynomial transforms, whose rows correspond to a family of orthogonal polynomials. Other well known classes include the Toeplitz, Hankel, Vandermonde, Cauchy matrices and their extensions that are all special cases of a ldisplacement rank property. In this paper, we make progress on two fronts:

1. We introduce the notion of recurrence width of matrices. For matrices with constant recurrence width, we design algorithms to compute Ab and $A^T b$ with a near-linear number of operations. This notion of width is finer than all the above classes of structured matrices and thus we can compute multiplication for all of them using the same core algorithm.

2. We additionally adapt this algorithm to an algorithm for a much more general class of matrices with displacement structure: those with low displacement rank with respect to quasiseparable matrices. This class includes Toeplitz-plus-Hankel-like matrices, Discrete Cosine/Sine Transforms, and more, and captures all previously known matrices with displacement structure that we are aware of under a unified parametrization and algorithm.

Our work unifies, generalizes, and simplifies existing state-of-the-art results in structured matrix-vector multiplication. Finally, we show how applications in areas such as multipoint evaluations of multivariate polynomials can be reduced to problems involving low recurrence width matrices.

Where is the deep learning stuff?

1 Layer : $y = g(Ax)$

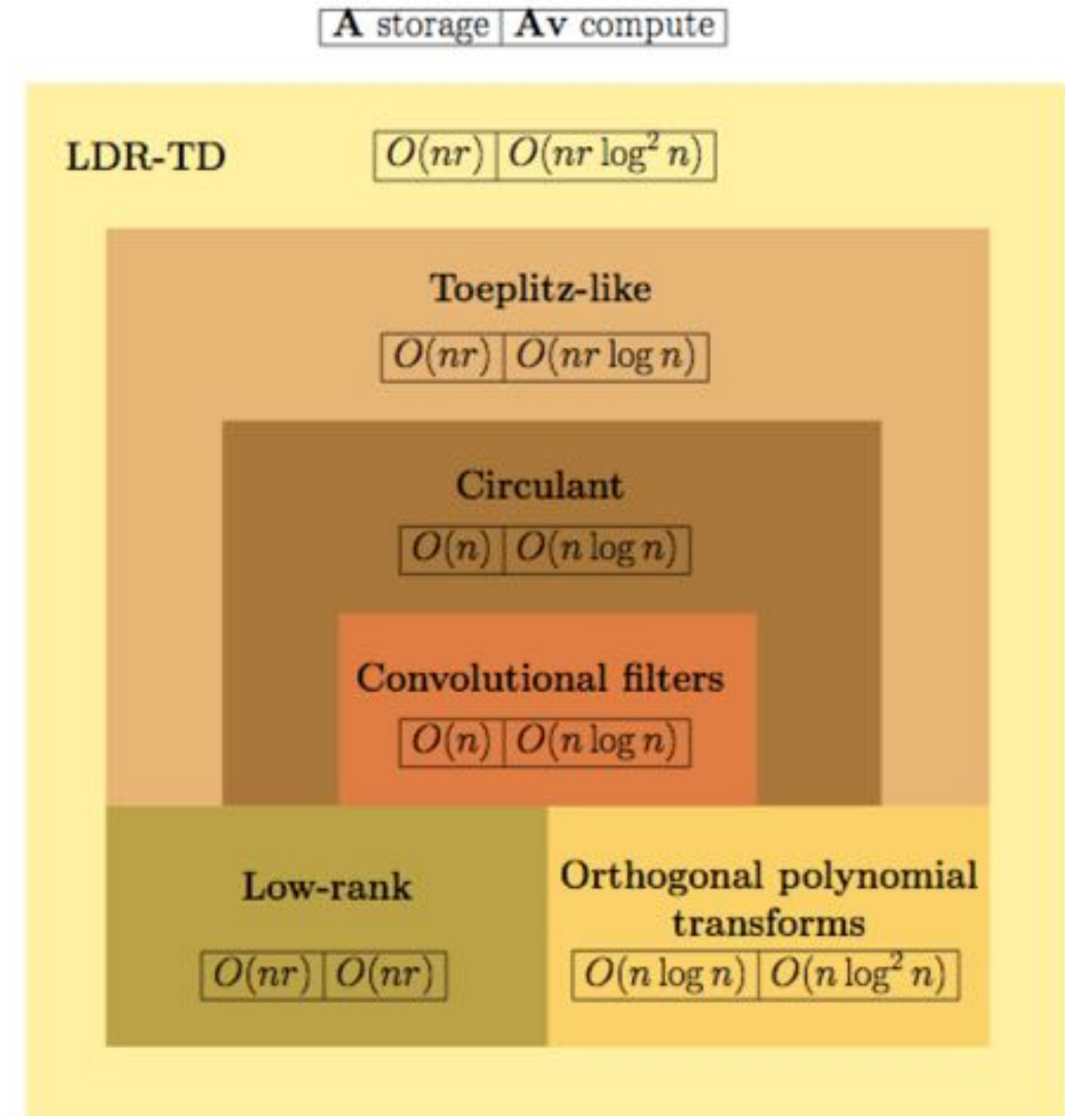
Non-linear function

Better accuracy than unconstrained A and with less parameters in some image classification tasks

Class of displacement rank ops

$$\begin{bmatrix} 0 & & \cdots & 0 & f \\ 1 & 0 & & \ddots & 0 \\ \vdots & 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & & \cdots & 0 & x_0 \\ x_1 & 0 & & \ddots & 0 \\ \vdots & x_2 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \\ 0 & 0 & \dots & x_{n-1} & 0 \end{bmatrix} \quad \begin{bmatrix} b_0 & a_0 & \cdots & 0 & s \\ c_0 & b_1 & a_1 & & 0 \\ \vdots & c_1 & \ddots & \ddots & \vdots \\ 0 & & \ddots & b_{n-1} & a_{n-2} \\ t & 0 & \dots & c_{n-2} & b_{n-1} \end{bmatrix}$$

Another view



Some copy and paste from paper

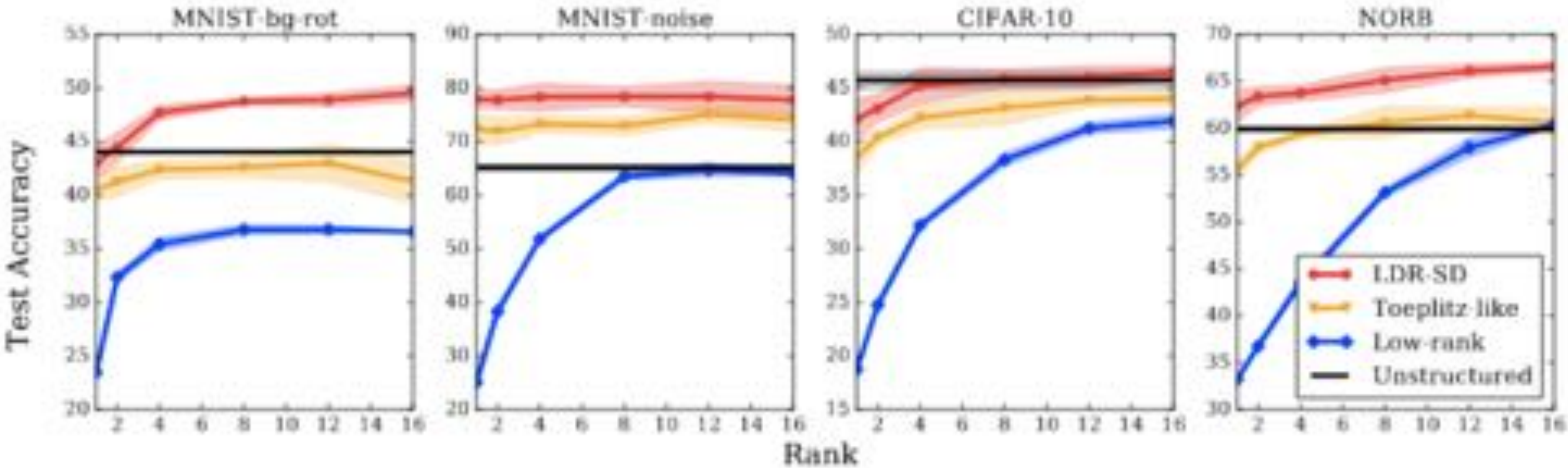
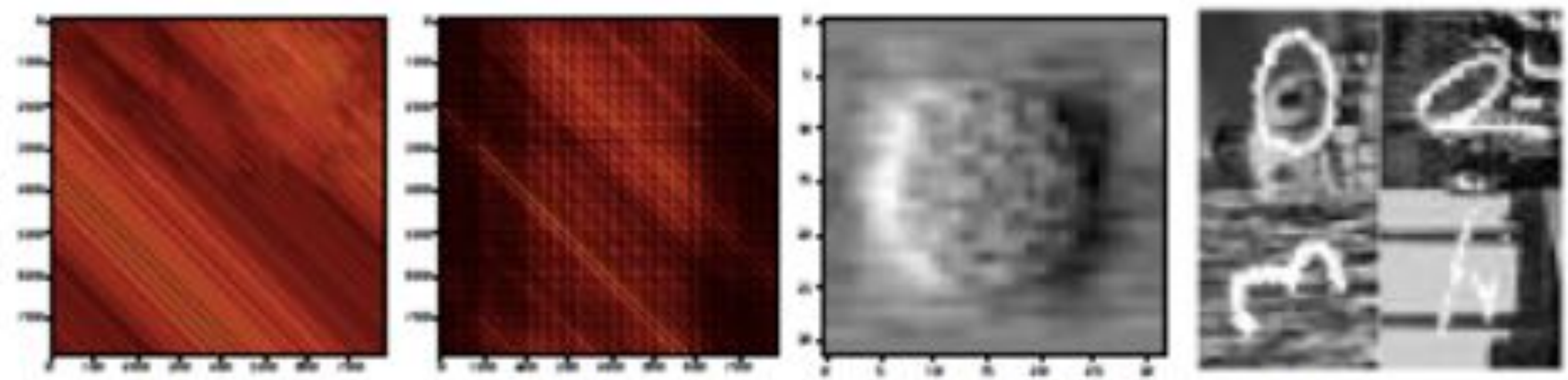


Figure 3: Test accuracy vs. rank for unstructured, LDR-SD, Toeplitz-like, low-rank classes. On each dataset, LDR-SD meets or exceeds the accuracy of the unstructured fully-connected baseline at higher ranks. At rank 16, the compression ratio of an LDR-SD layer compared to the unstructured layer ranges from 23 to 30. Shaded regions represent two standard deviations from the mean, computed over five trials with randomly initialized weights.

“Automatically” learning invariance



(a) Toeplitz-like (b) LDR-SD (c) Subdiagonal of B (d) Input examples

For more…

arXiv.org > cs > arXiv:1810.02309

Search or Art

(Help | Advanced

Computer Science > Machine Learning

Learning Compressed Transforms with Low Displacement Rank

Anna T. Thomas, Albert Gu, Tri Dao, Atri Rudra, Christopher Ré

(Submitted on 4 Oct 2018)

The low displacement rank (LDR) framework for structured matrices represents a matrix through two displacement operators and a low-rank residual. Existing use of LDR matrices in deep learning has applied fixed displacement operators encoding forms of shift invariance akin to convolutions. We introduce a rich class of LDR matrices with more general displacement operators, and explicitly learn over both the operators and the low-rank component. This class generalizes several previous constructions while preserving compression and efficient computation. We prove bounds on the VC dimension of multi-layer neural networks with structured weight matrices and show empirically that our compact parameterization can reduce the sample complexity of learning. When replacing weight layers in fully-connected, convolutional, and recurrent neural networks for image classification and language modeling tasks, our new classes exceed the accuracy of existing compression approaches, and on some tasks even outperform general unstructured layers while using more than 20X fewer parameters.

Subjects: **Machine Learning (cs.LG)**; Machine Learning (stat.ML)

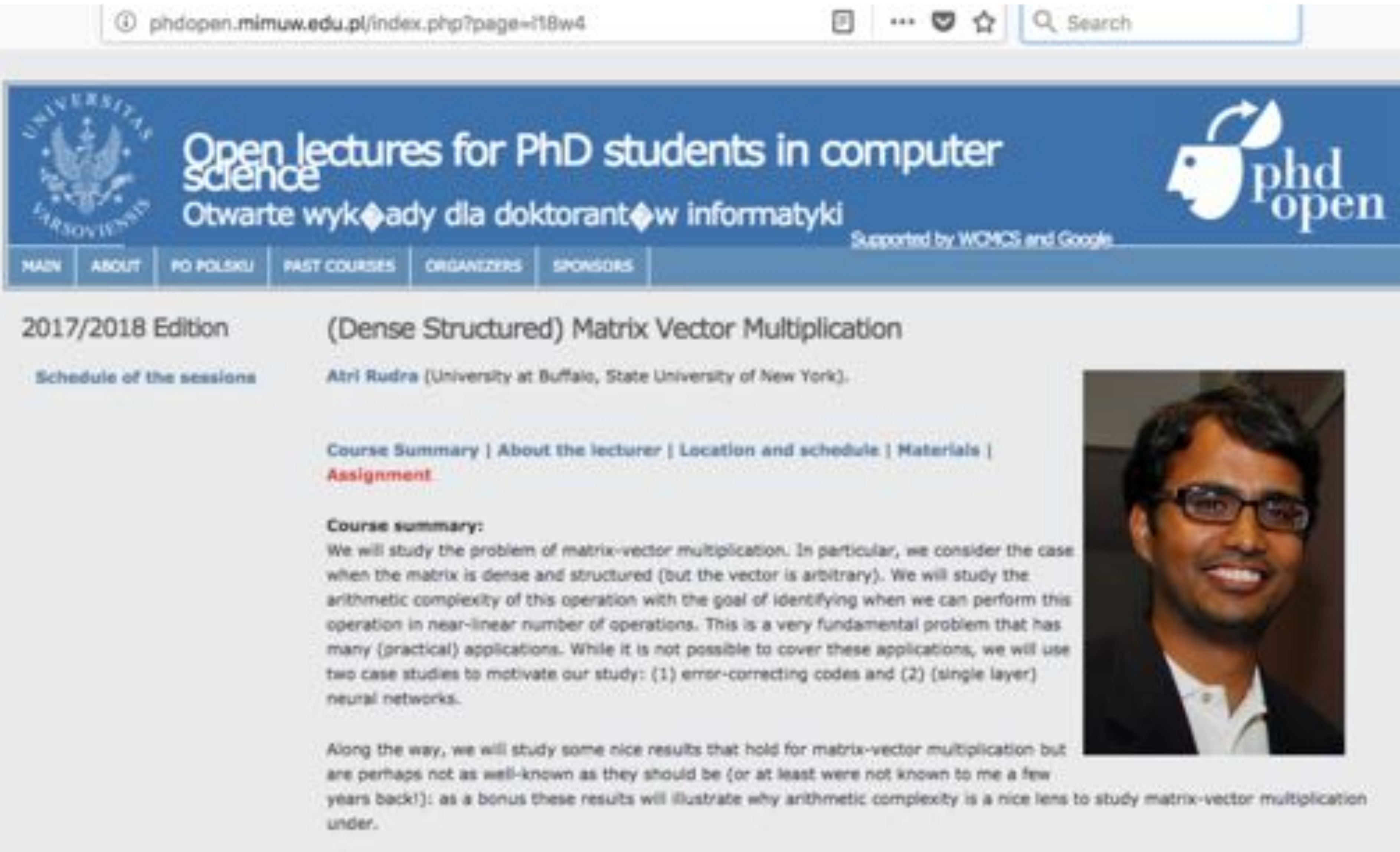
Cite as: arXiv:1810.02309 [cs.LG]

(or arXiv:1810.02309v1 [cs.LG] for this version)

Bibliographic data

[Enable Bibex (What is Bibex?)]

A better source



phdopen.mimuw.edu.pl/index.php?page=l18w4

Open lectures for PhD students in computer science

Otwarte wyk◆ady dla doktorant◆w informatyki

Supported by WCMCS and Google

MAIN | ABOUT | PO POLSKU | PAST COURSES | ORGANIZERS | SPONSORS

2017/2018 Edition

Schedule of the sessions

(Dense Structured) Matrix Vector Multiplication

Atri Rudra (University at Buffalo, State University of New York).

Course Summary | About the lecturer | Location and schedule | Materials | Assignment

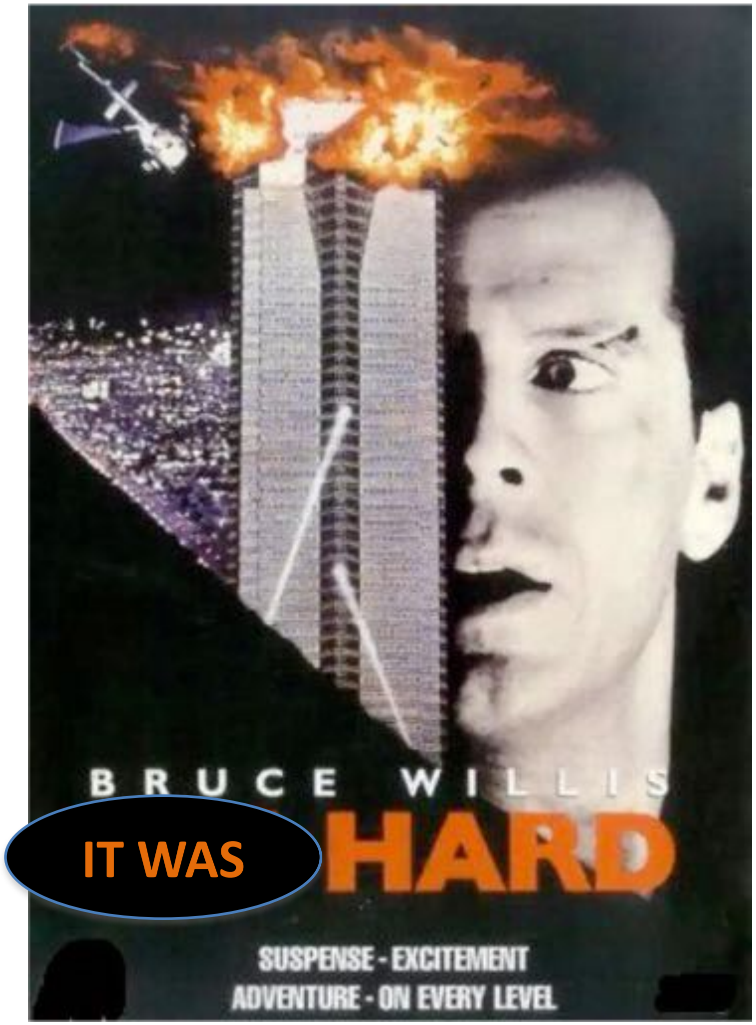
Course summary:

We will study the problem of matrix-vector multiplication. In particular, we consider the case when the matrix is dense and structured (but the vector is arbitrary). We will study the arithmetic complexity of this operation with the goal of identifying when we can perform this operation in near-linear number of operations. This is a very fundamental problem that has many (practical) applications. While it is not possible to cover these applications, we will use two case studies to motivate our study: (1) error-correcting codes and (2) (single layer) neural networks.

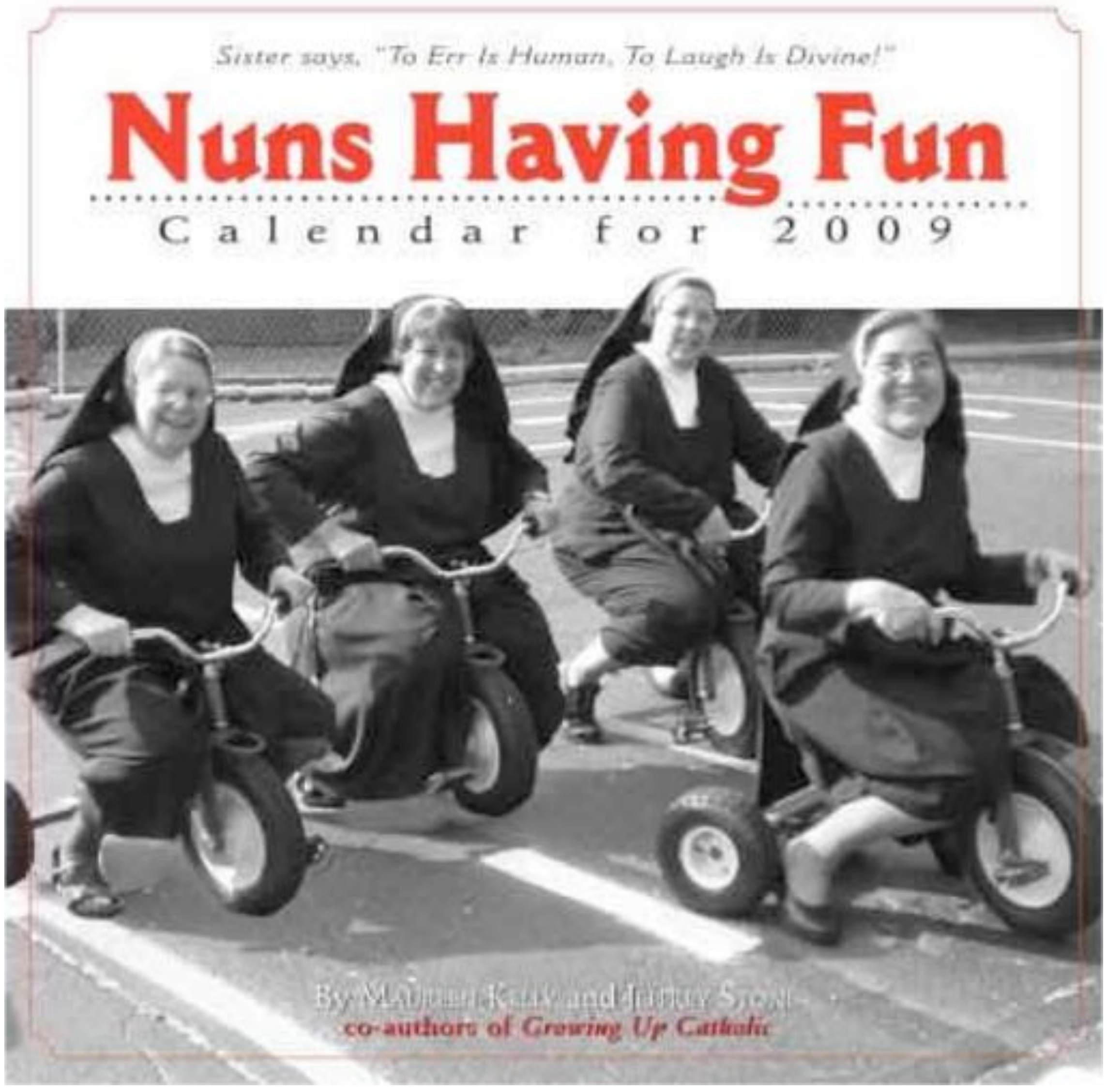
Along the way, we will study some nice results that hold for matrix-vector multiplication but are perhaps not as well-known as they should be (or at least were not known to me a few years back!): as a bonus these results will illustrate why arithmetic complexity is a nice lens to study matrix-vector multiplication under.

Questions?

Whatever your impression of the 331



Hopefully it was fun!



Thanks!

Except of course, HW 10 and the final exam