Lecture 37

CSE 331

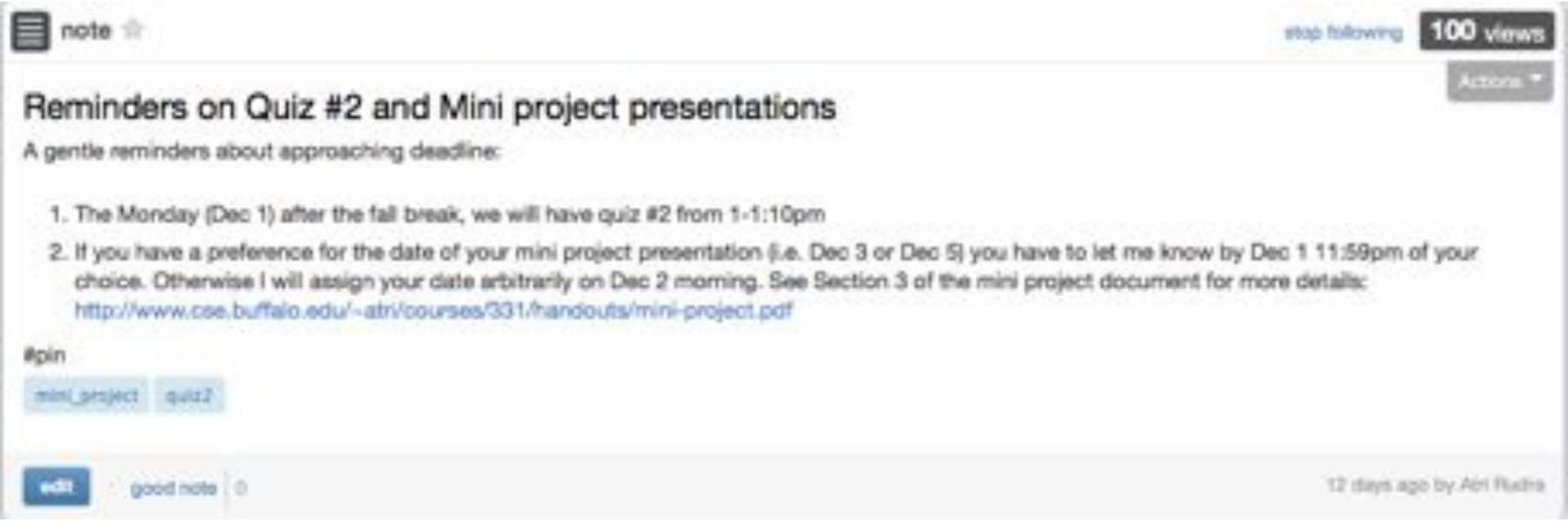
Dec 1, 2014

Quiz starts at 1pm and ends at 1:10pm

Time left

Lecture starts at 1:15pm

Deadline for presentation slots



Final exam post

note

stop following 40 views

Actions

Final exam post

I'll start off with some generic comments:

- The final exam will be based on all the material we have seen in class till the lecture on Monday, Nov 24 (i.e. up to the P vs NP stuff).
- The lecture on Monday, Dec 1 (i.e. Monday after the fall break and right after Quiz 2) will partly be a Q & A session (where you can ask any 331 related questions).
- Exam will be from **noon to 2:30** on Friday, **Dec 12** in class (**NSC 215**). Note that the exam will be for 2.5 hours and not 3 hours as it says on HUB.

Next are comments related to **preparing for the finals**:

1. Take a look at the sample final (@506) and spend some quality time solving it. Unlike the homeworks, it might be better to try to do this on your own. Unlike the sample mid-term, this one is an actual 331 final exam so in addition to the format, you can also gauge how hard the final exam is going to be (your final exam will be the same ballpark). However as with the sample mid-term, you make deduction about the coverage of topics at your own peril (but see points below). Once you have spent time, on it on your own, take a look at the sample final solutions (@524).
2. Go to Atri/Frank/Zulkar's extra office hours (@507).
3. Attend the Q&A session (Monday, Dec 1) in class.
4. The actual final will have the same format as the sample final: The first question will be T/F, 2nd will be T/F with justification, the rest of the three will be longer questions and will ask you to design algorithms (parts of them might be just analyzing an algorithm.)
5. For the T/F questions (i.e. the first two questions), anything that was covered in class is fair game. If you want to refresh your memory on what was covered, take a look at the schedule page. If you want quick summaries of (almost all) the lectures, review the lecture notes or slides.
6. To get more practice for the T/F questions, prepare and take Quiz 2 (@512) on Monday, Dec 1.

Now relax…

Randomized algorithms

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

What is different?

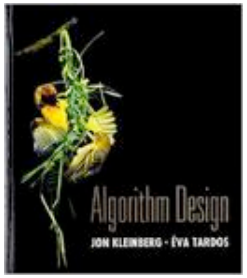
Algorithms can toss coins and make decisions

A Representative Problem

Hashing

Further Reading

Chapter 13 of the textbook

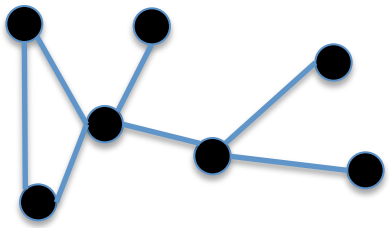


Approximation algorithms

What is different?

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

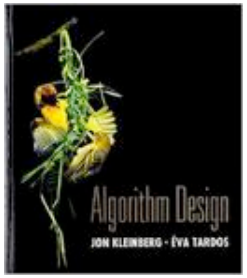
A Representative Problem



Vertex Cover

Further Reading

Chapter 12 of the textbook



Online algorithms

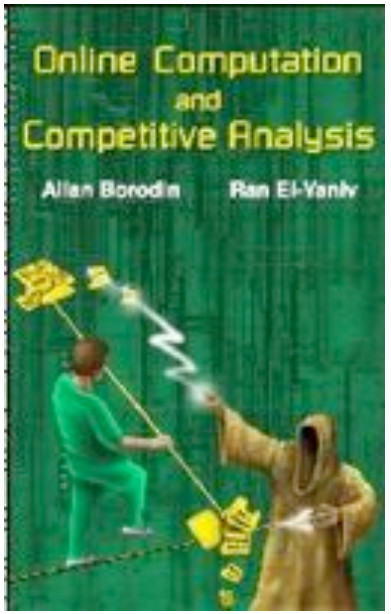
What is different?

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/

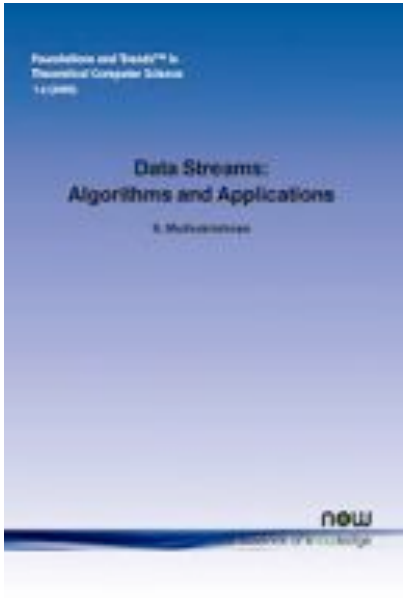
What is different?

One pass on the input with severely limited memory

A Representative Problem

Compute the top-10 source IP addresses

Further Reading



Distributed algorithms

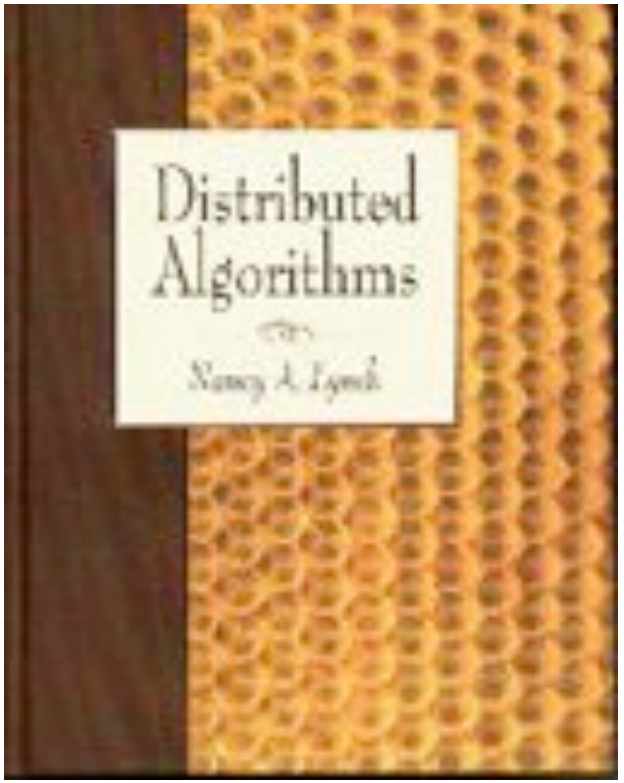
What is different?

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

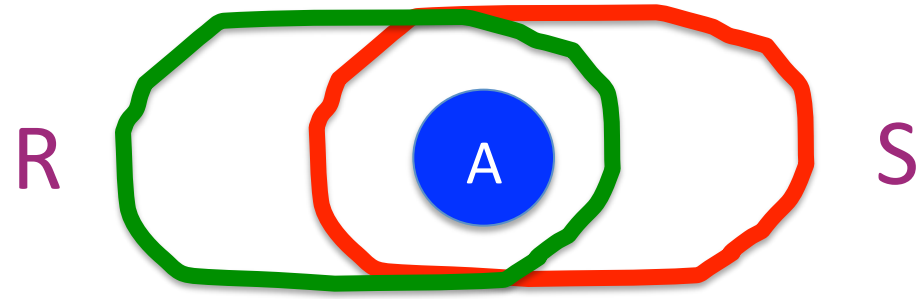
Further Reading

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

Q & A Session

The simplest non-trivial join query

Intersection of R and S

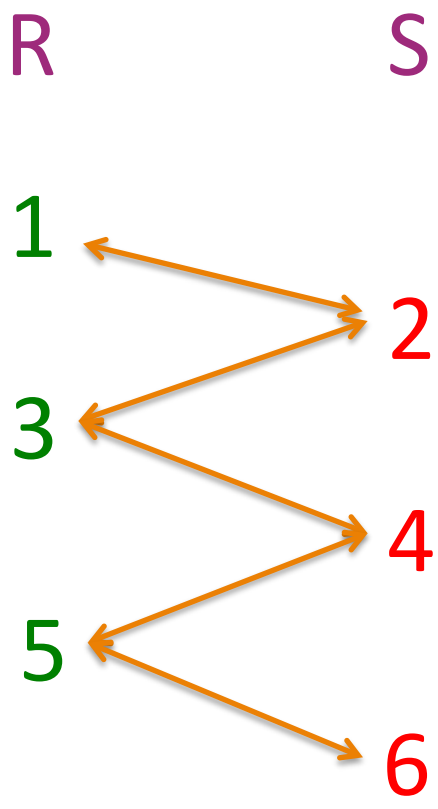


Assume R and S are sorted

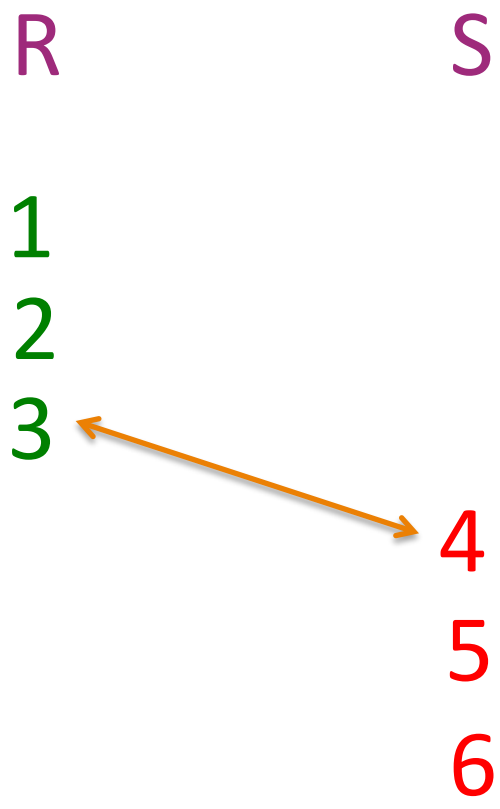
Let us concentrate on comparison based algorithms

Assume |R| = |S| = N

Not all inputs are created equal



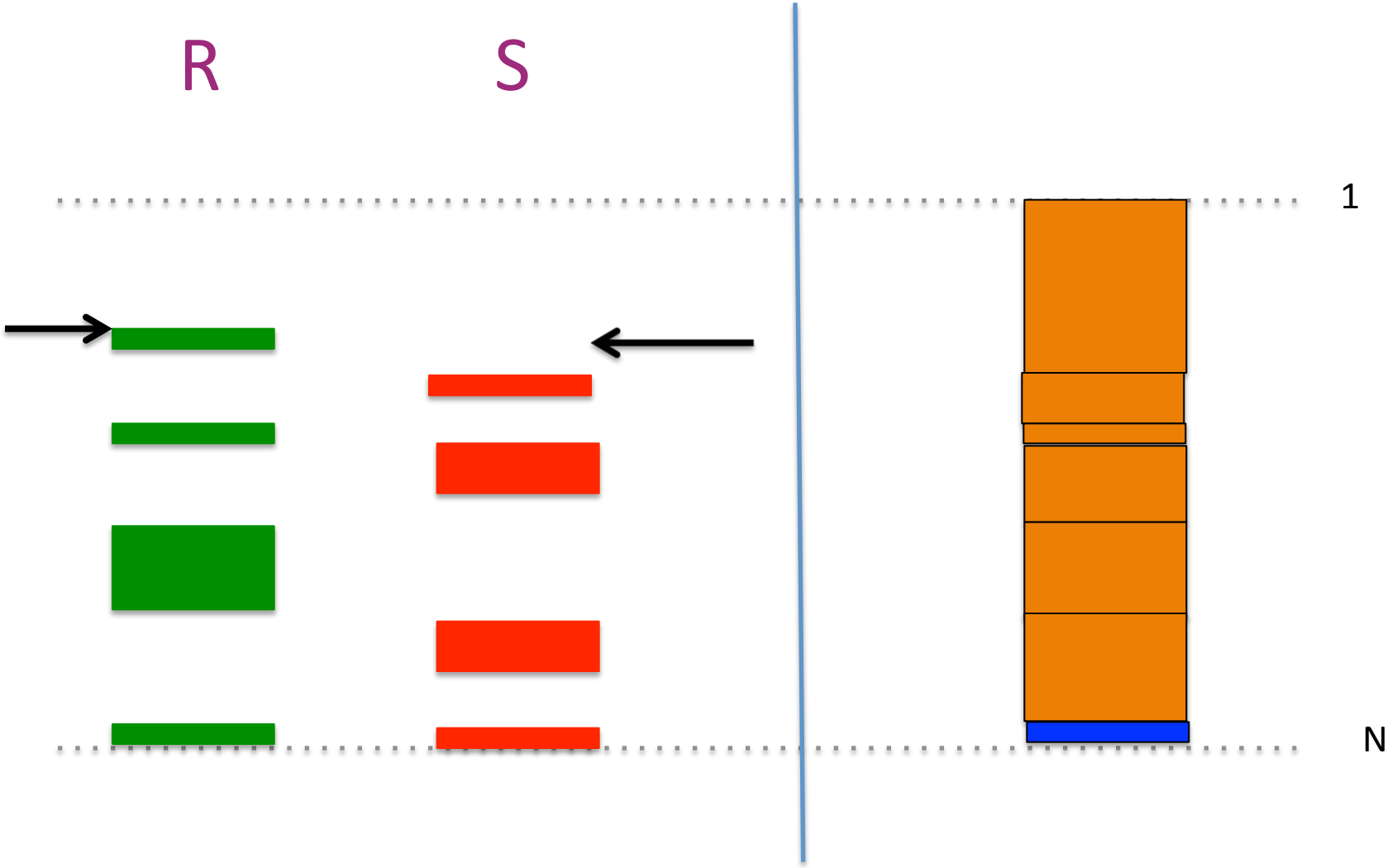
Ω(N) comparisons



1 comparison!

We need a faster/adaptive algorithm

The MERGE algorithm works



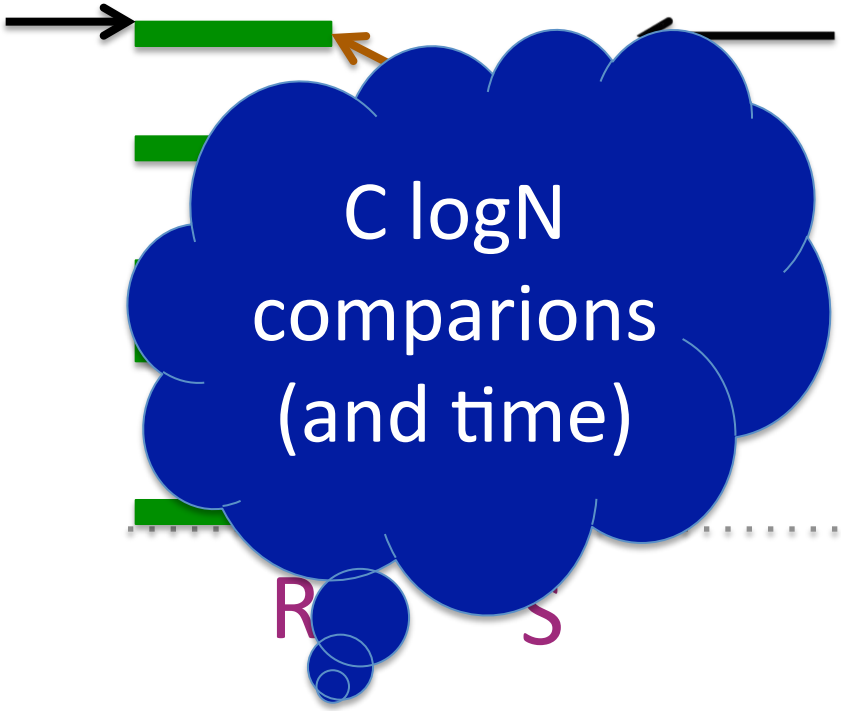
An assumption

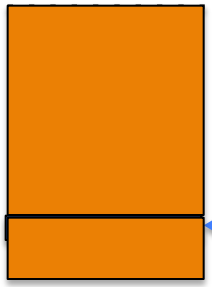
Output of the join is empty

MERGE is (near) instance optimal

Benchmark: Minimum number of comparisons (C) to "certify" output

Demaine Lopez-Oritz Munro





1

Value not ruled out yet

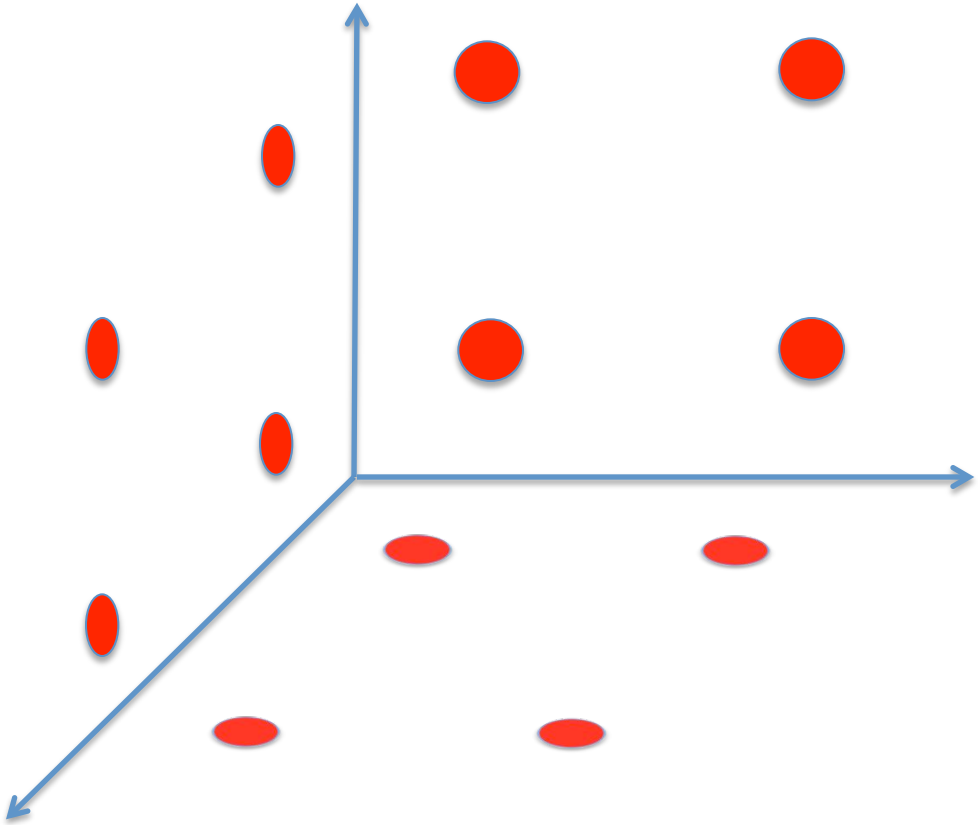
Need a comparison to rule the value out

Each value involved with ≤2 comparisons

Once the pointer moves the value is never seen again

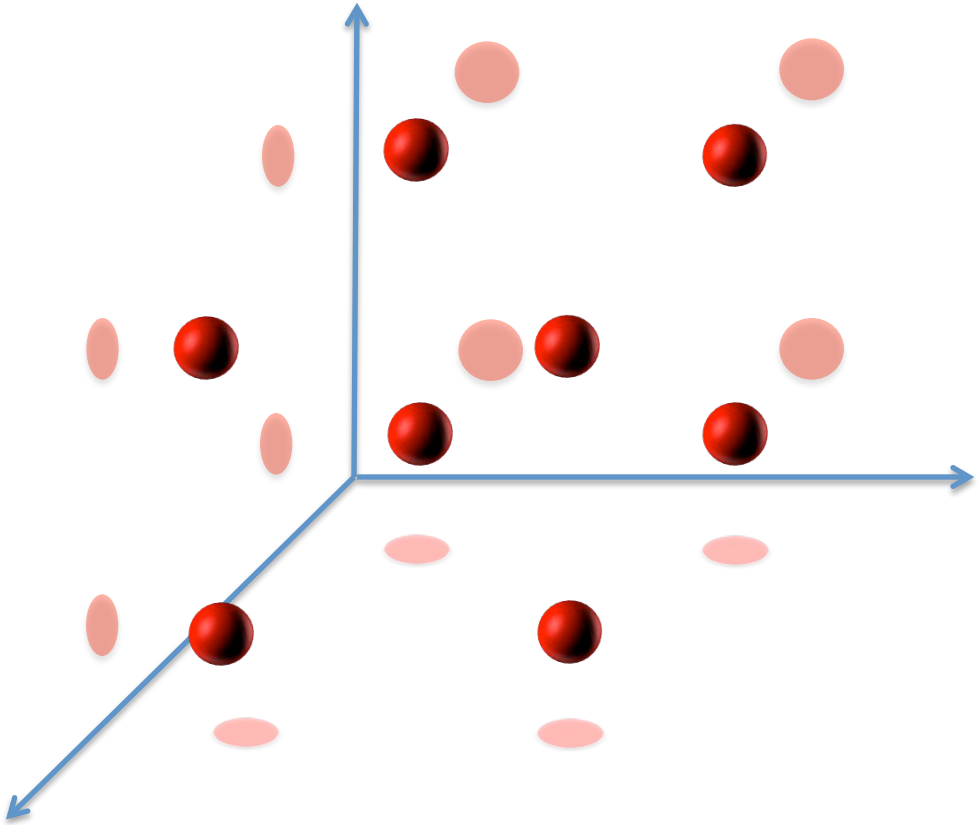
Each move takes log N comparisons

A toy problem



Given the three projections, what is the largest size of the original set of points?

The key technical problem

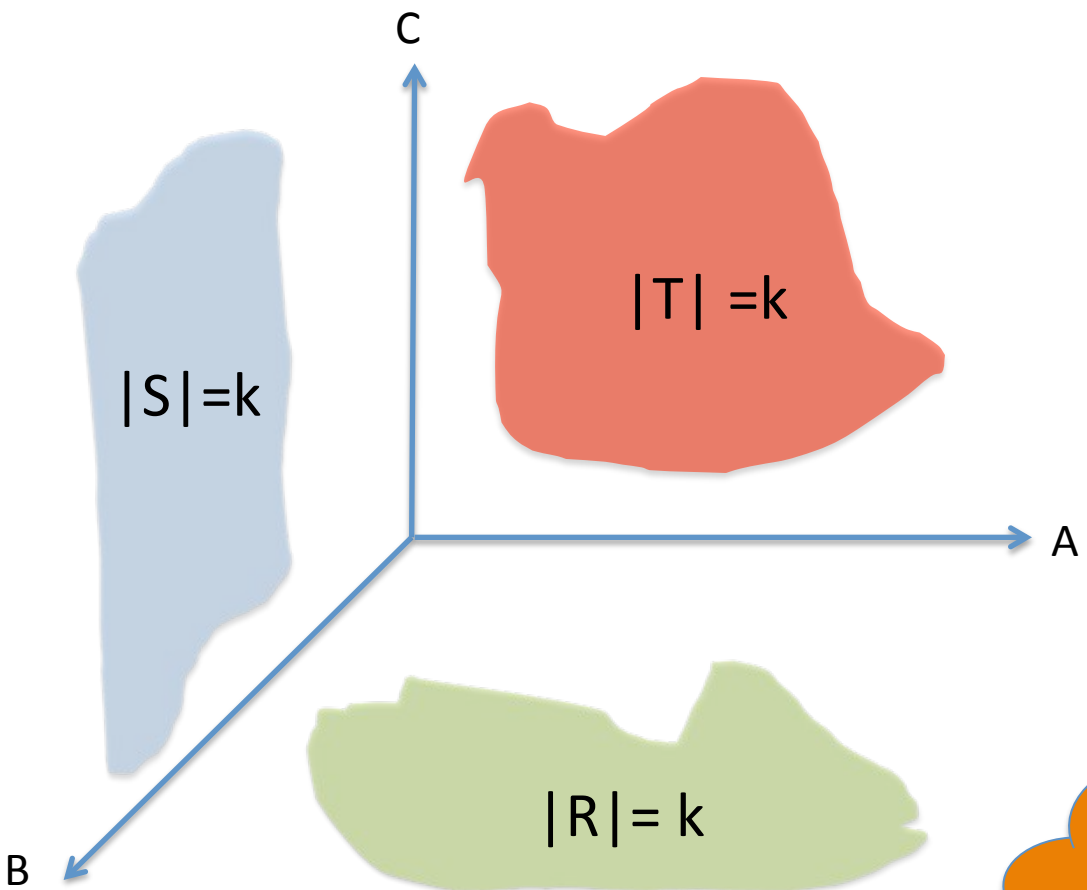


Highly trivial: $4^3 = 64$

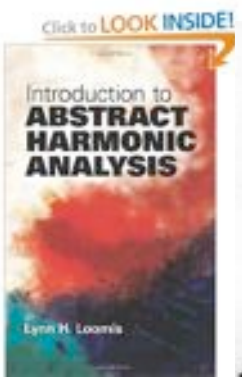
Still trivial: $4^2 = 16$

Correct answer: $4^{1.5} = 8$

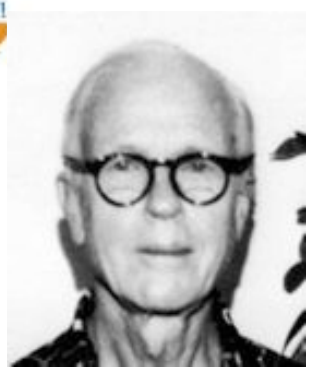
The key technical problem



$k^{3/2}$



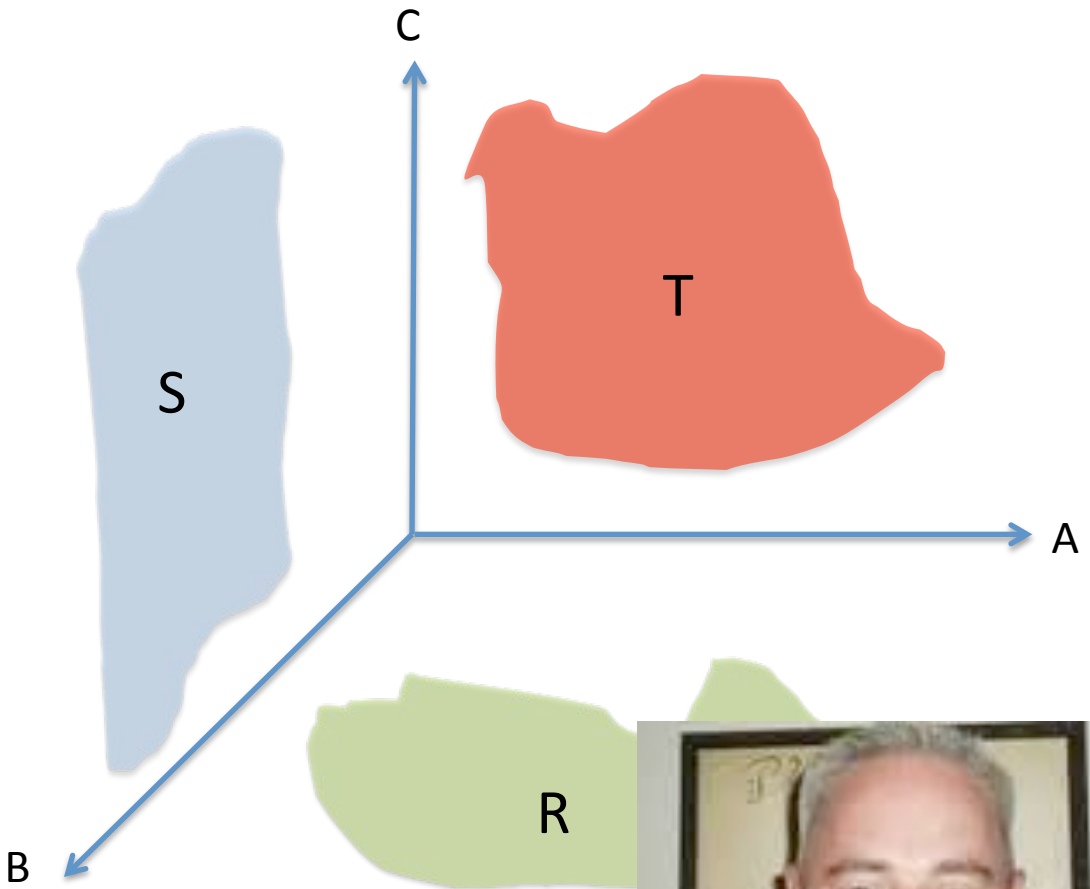
Loomis

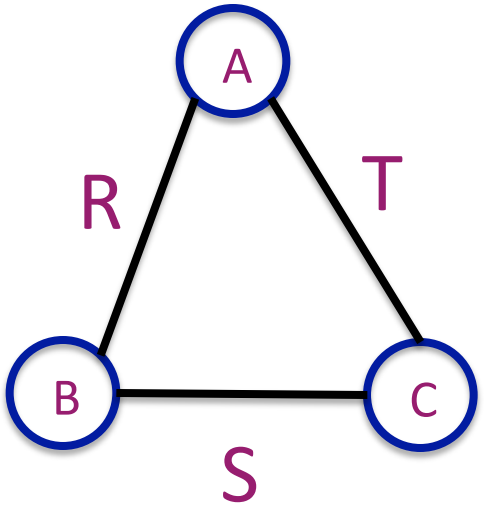


Whitney

Algorithmic Loomis-Whitney?

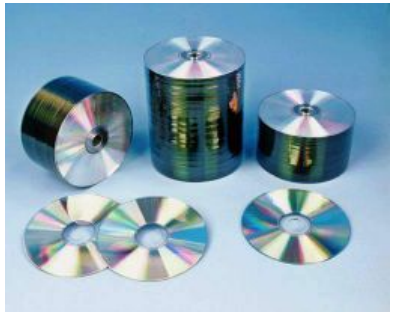
An equivalent view

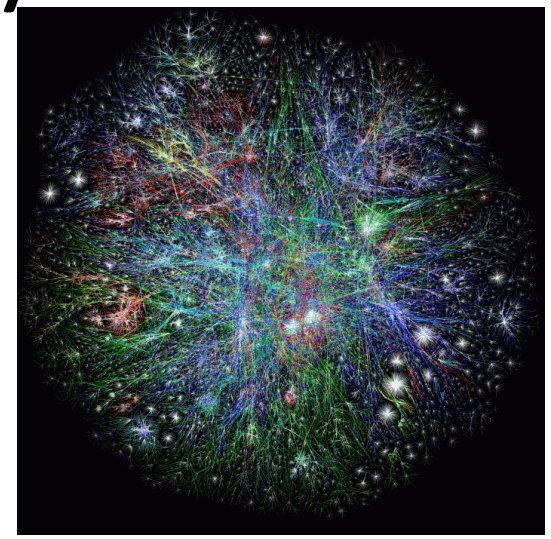




Output all (a,b,c) s.t. (a,b) in R, (b,c) in S and (c,a) in T

Coding Theory

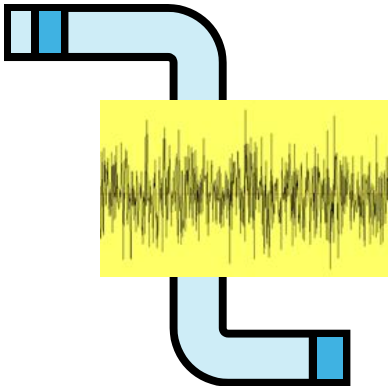




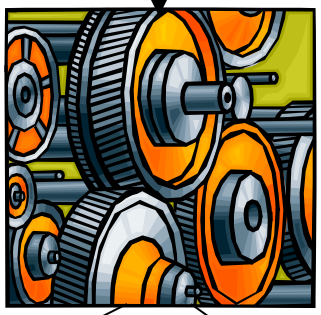
Communicating with my 5 year old

x

C(x)



y = C(x)+error



x

Give up

"Code" **C**

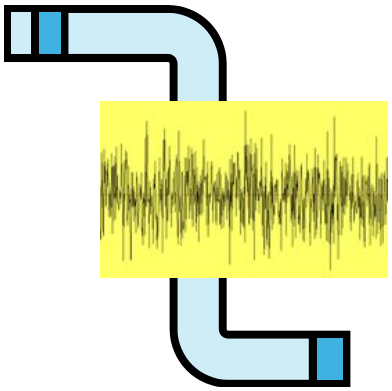
"Akash English"

C(x) is a "codeword"

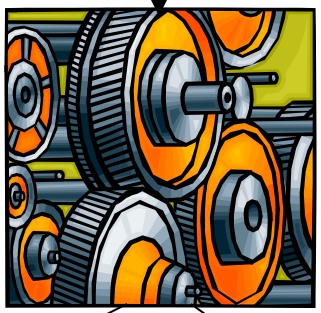
The setup

x

C(x)



y = C(x)+error



x

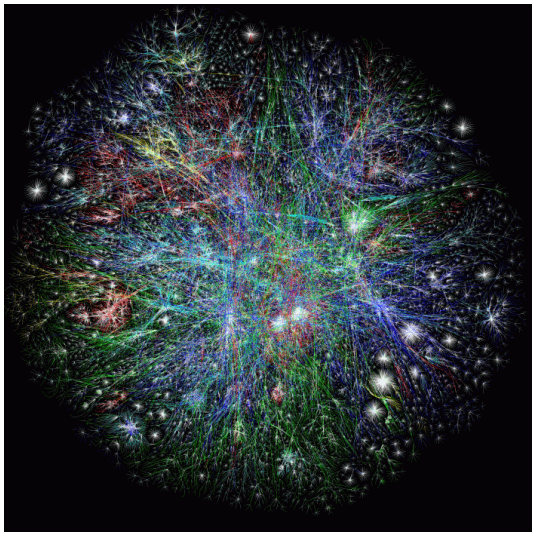
Give up

Mapping **C**

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

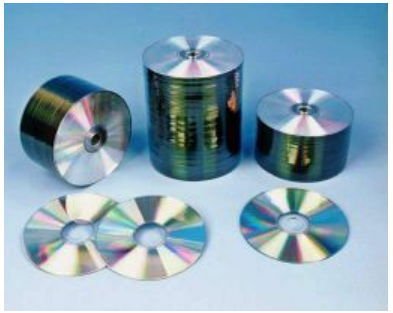
Different Channels and Codes

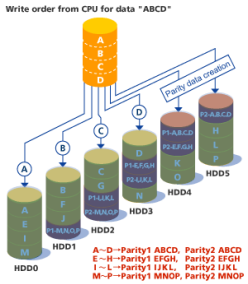
- Internet
  - Checksum used in mult layers of TCP/IP stack
- Cell phones
- Satellite broadcast
  - TV
- Deep space telecommunications
  - Mars Rover



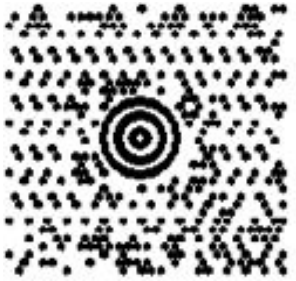
“Unusual” Channels

- 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 545, Spring 201?

Passwords

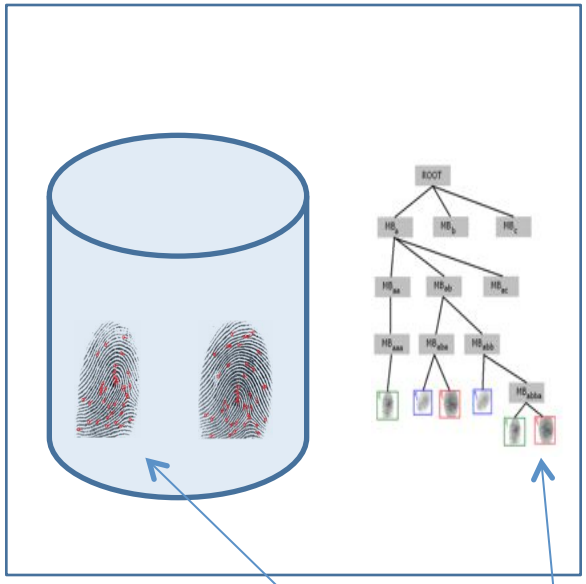
- Fast matching
- Secure
- "Cancelable"
- One can forget them!

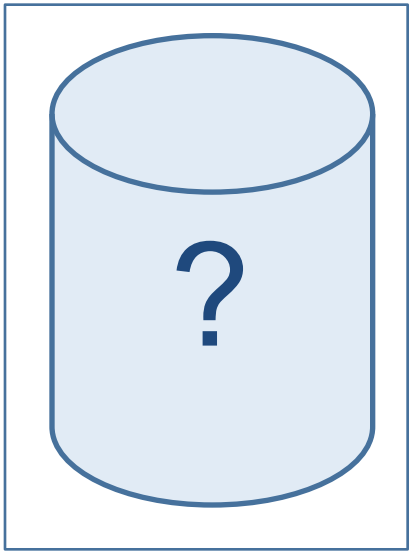
Fingerprints

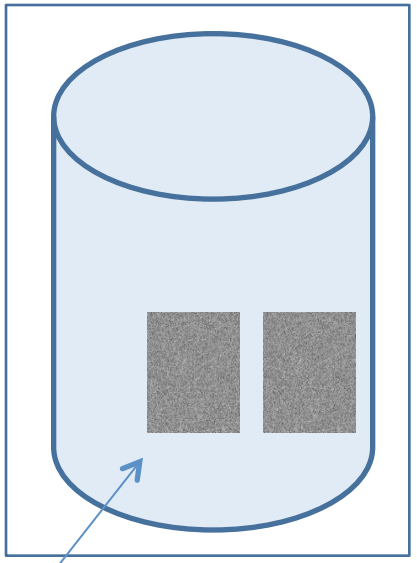
- \+ Fast matching
- ? Secure
- ? “Cancelable”
- \+ Cannot forget them!

Fingerprints as Passwords

Or making “Forgot password” links obsolete



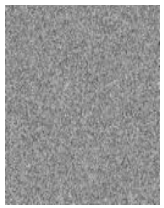




Only Fast

Fast & Secure

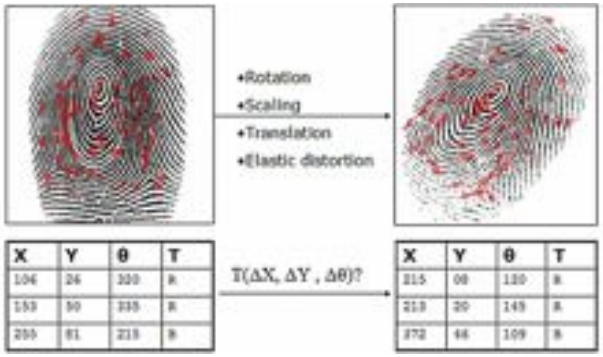
Only Secure



Challenges in Fingerprint Matching

Fingerprint readings are inconsistent

Matching Algorithms exist

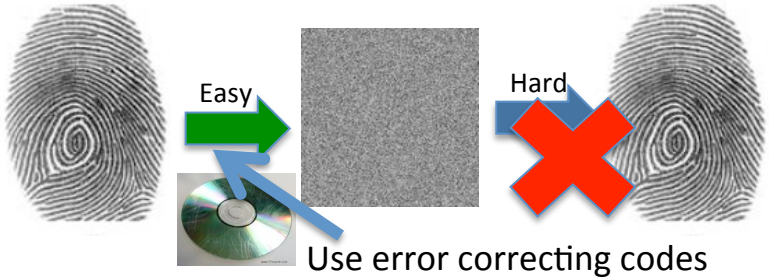


even in practice…

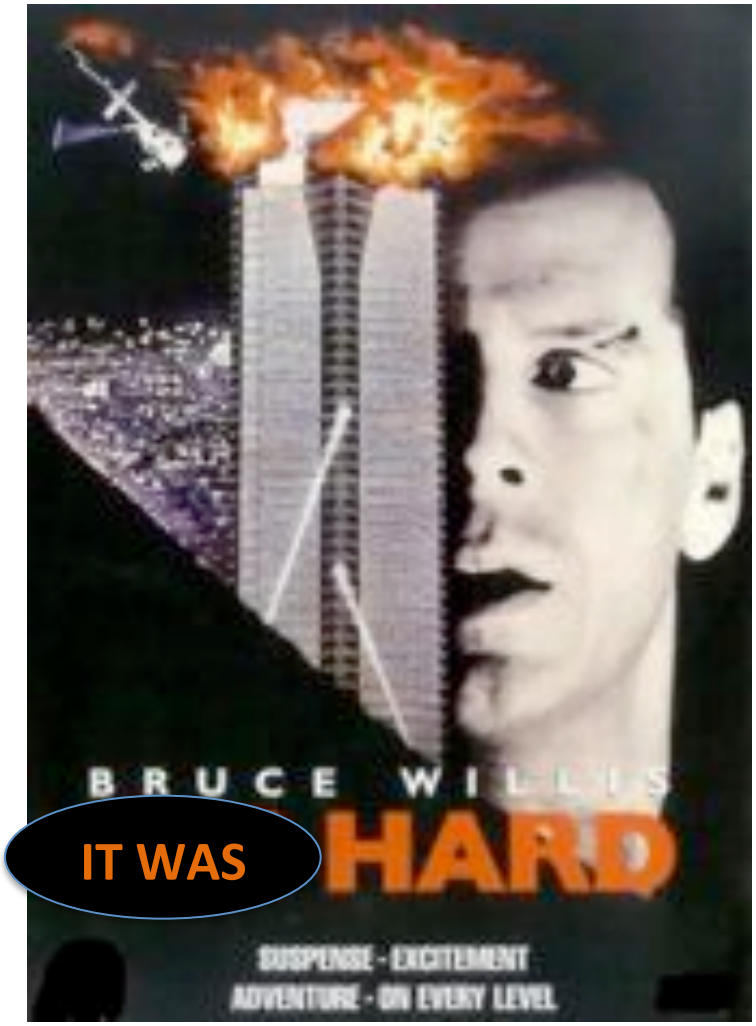
Security for fingerprints?

Stored fingerprints can be stolen

Main idea: obfuscate the fingerprint!

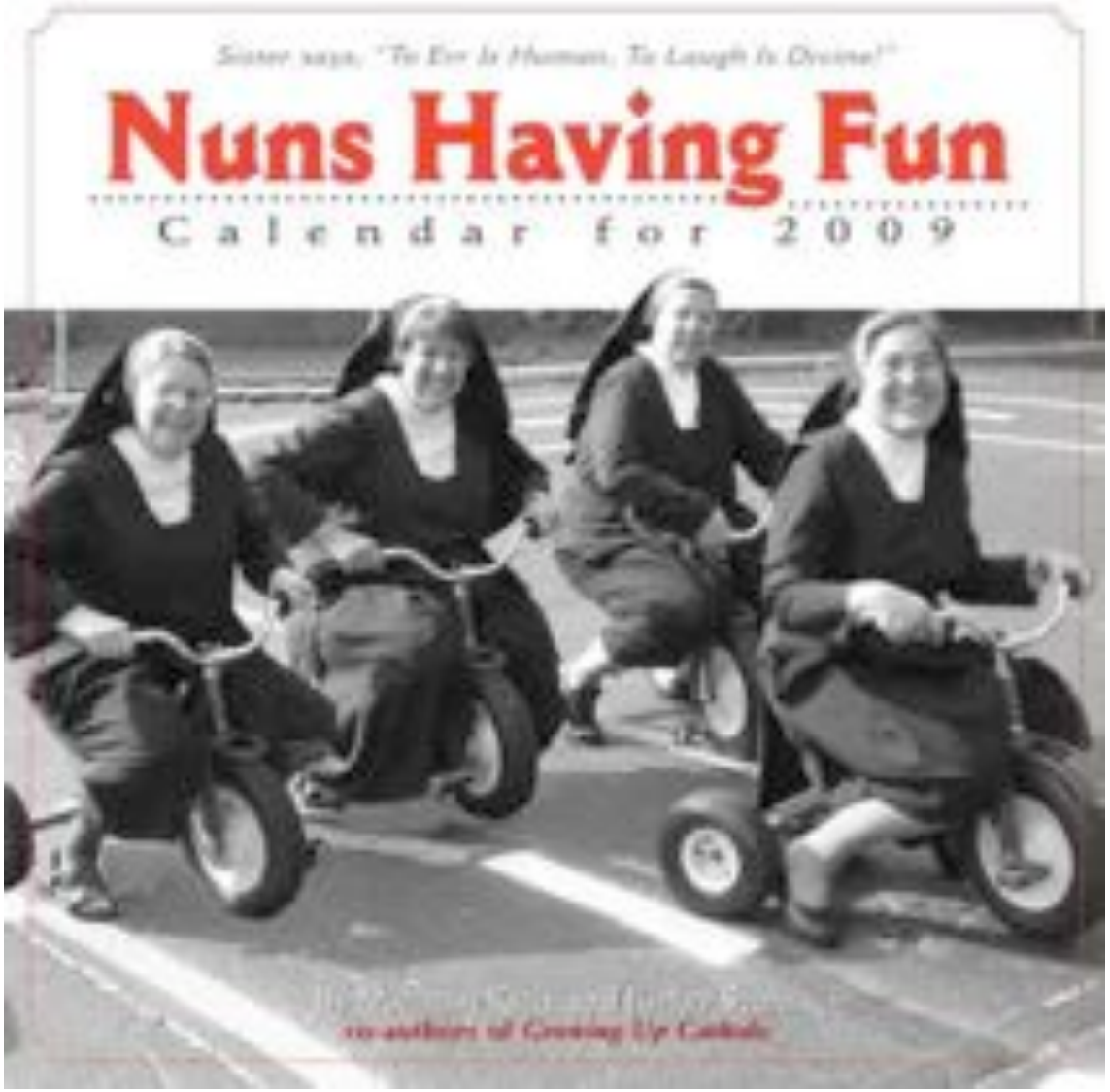


Whatever your impression of the 331



IT WAS

Hopefully it was fun!



Thanks!

Except of course, HW 10, presentations and the final exam