Lecture 38

CSE 331

Dec 5, 2016

Quiz 2

1:00-1:10pm

Lecture starts at 1:15pm

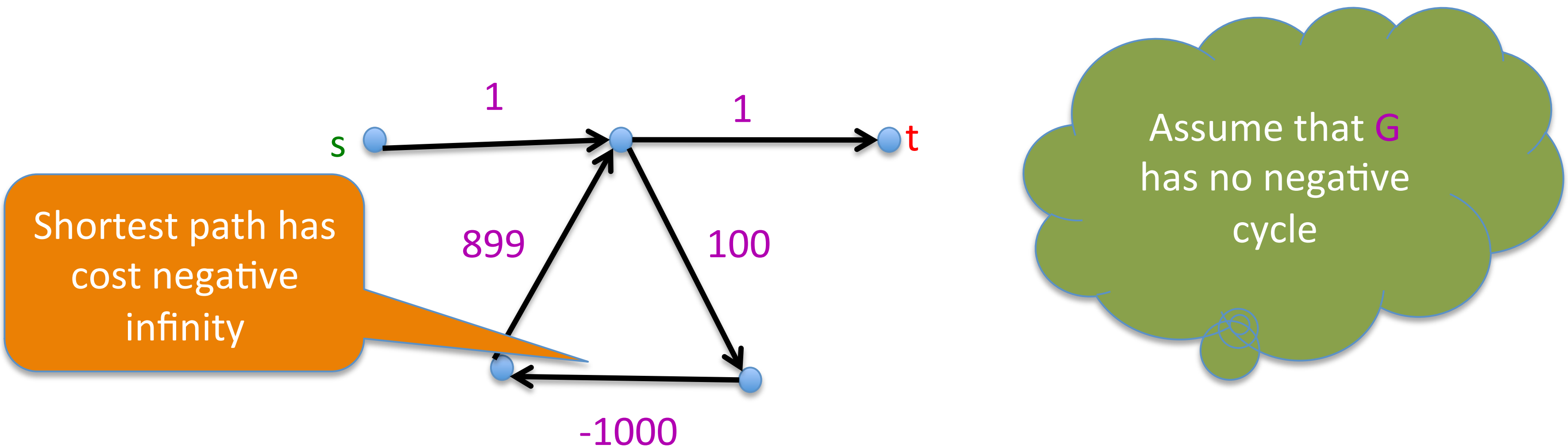
Please write your UBIT name

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



Longest path problem

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

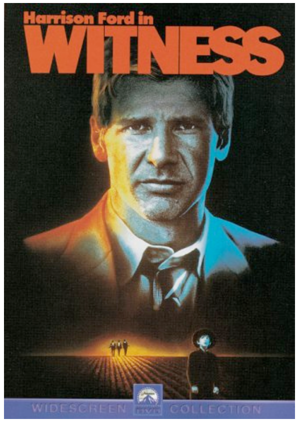
Longest vs Shortest Paths

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

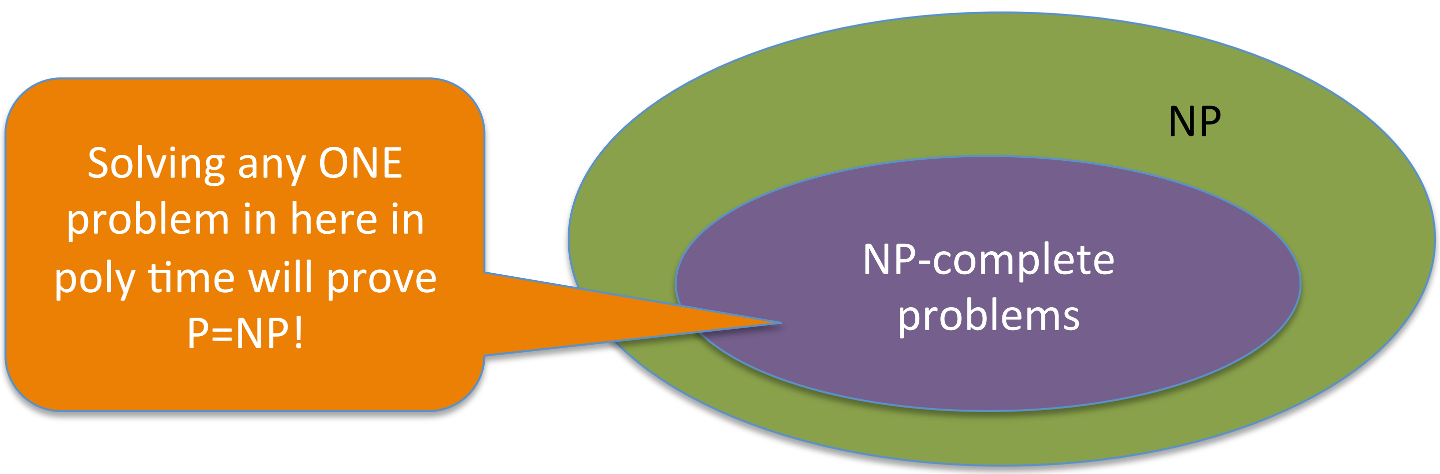
Pretty much all known proof techniques *provably* will not work

Proving P = NP

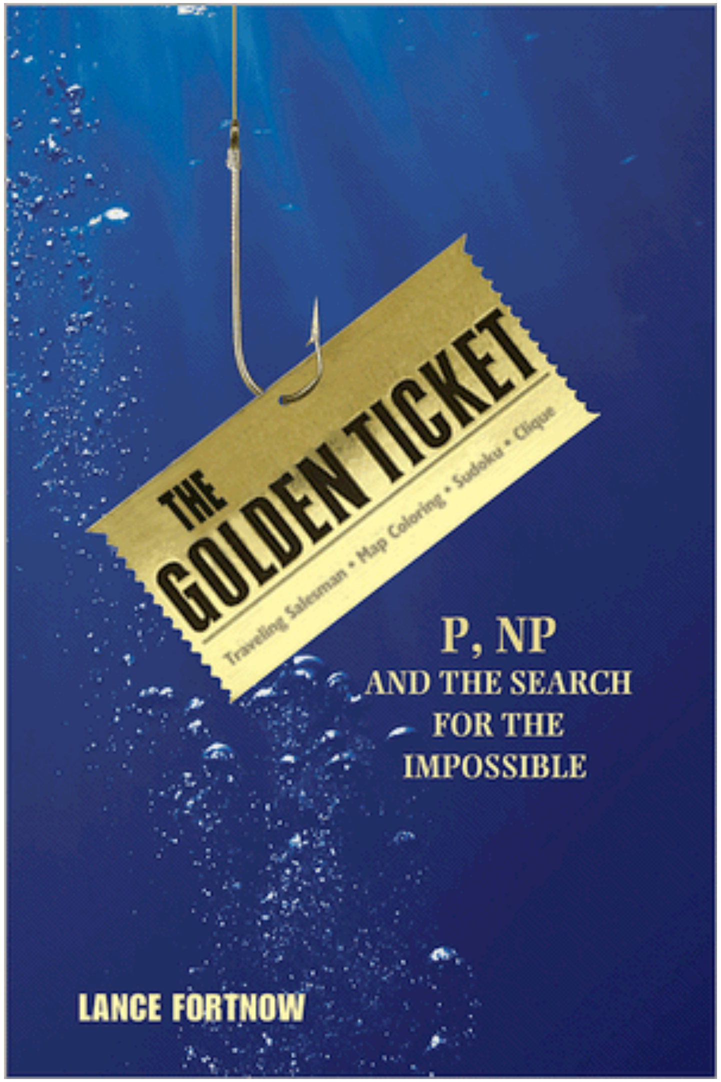
Will make cryptography collapse

Compute the encryption key!

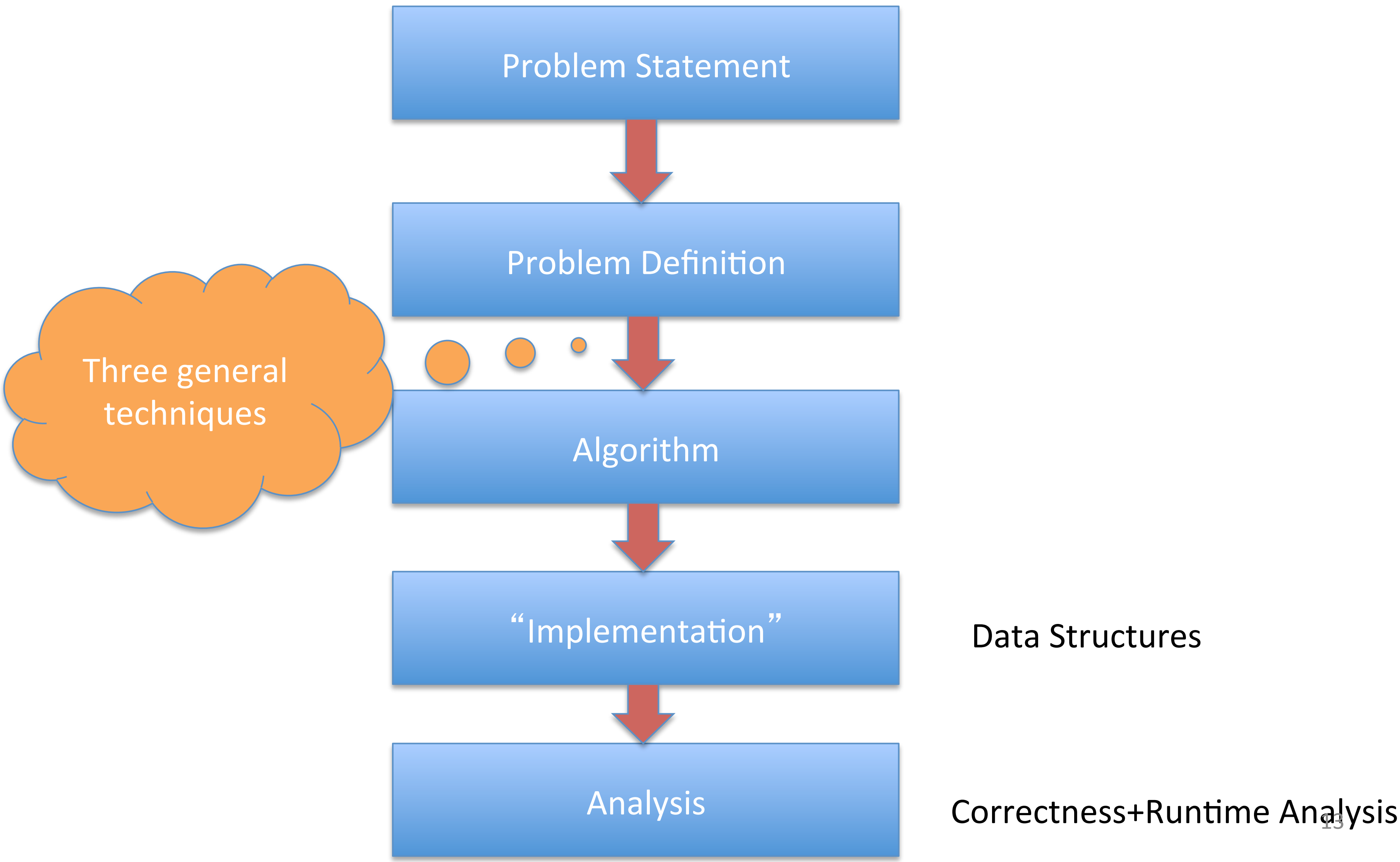
Prove that all problems in NP can be solved by polynomial time algorithms



A book on P vs. NP



High level view of CSE 331



If you are curious for more

CSE 429 or 431: Algorithms

CSE 396: Theory of Computation

Now relax…

Randomized algorithms

What is different?

Algorithms can toss coins and make decisions

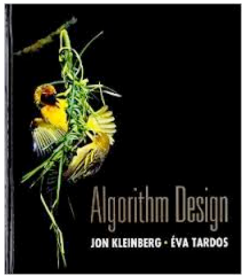
http://calculator.mathcaptain.com/coin-toss-probability-calculator.html

A Representative Problem

Hashing

Further Reading

Chapter 13 of the textbook



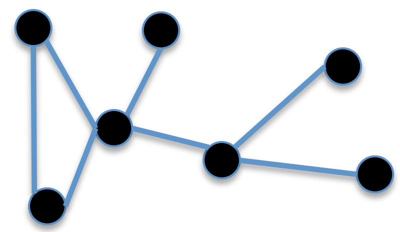
Approximation algorithms

What is different?

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

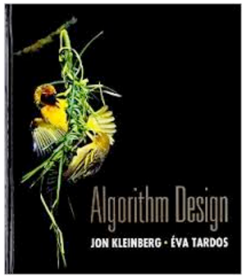
A Representative Problem

Vertex Cover



Further Reading

Chapter 12 of the textbook



Online algorithms

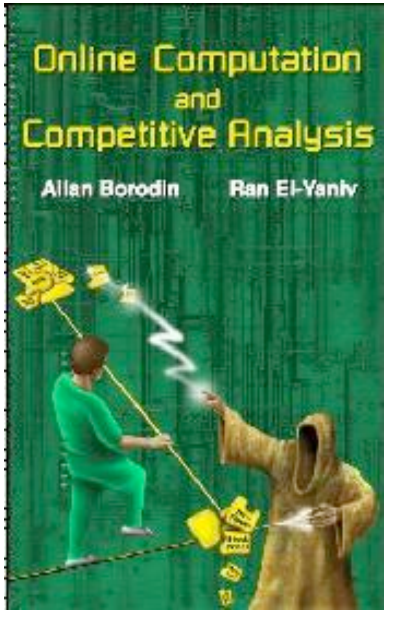
What is different?

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/

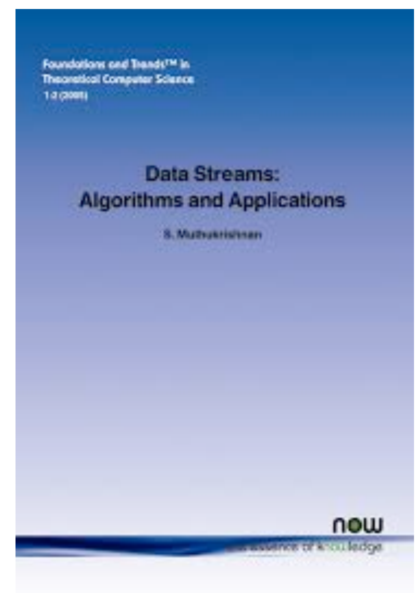
What is different?

One pass on the input with severely limited memory

A Representative Problem

Compute the top-10 source IP addresses

Further Reading



Distributed algorithms

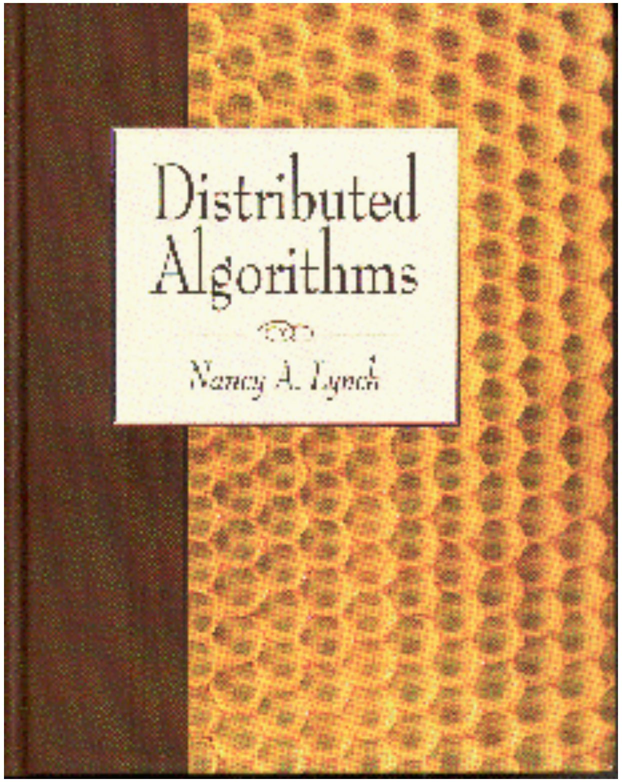
What is different?

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

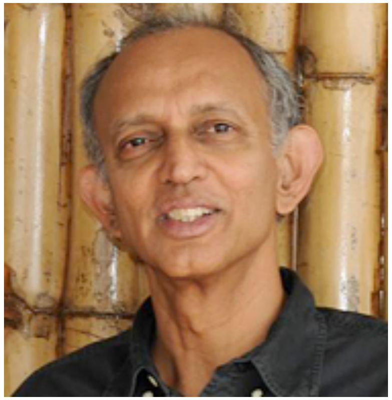
What is different?

Algorithms for non-discrete inputs

A Representative Problem

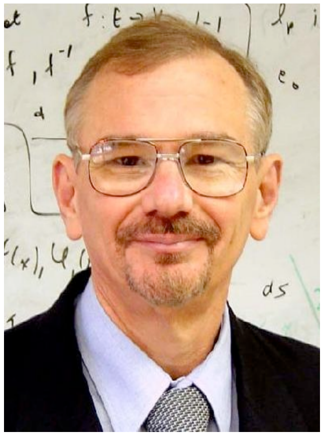
Compute Eigenvalues

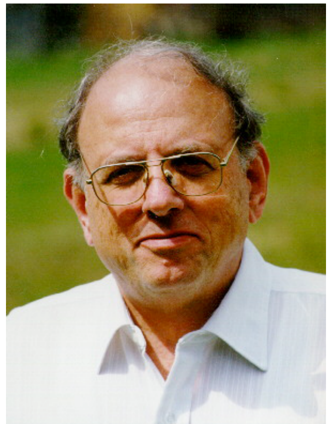
Further Reading

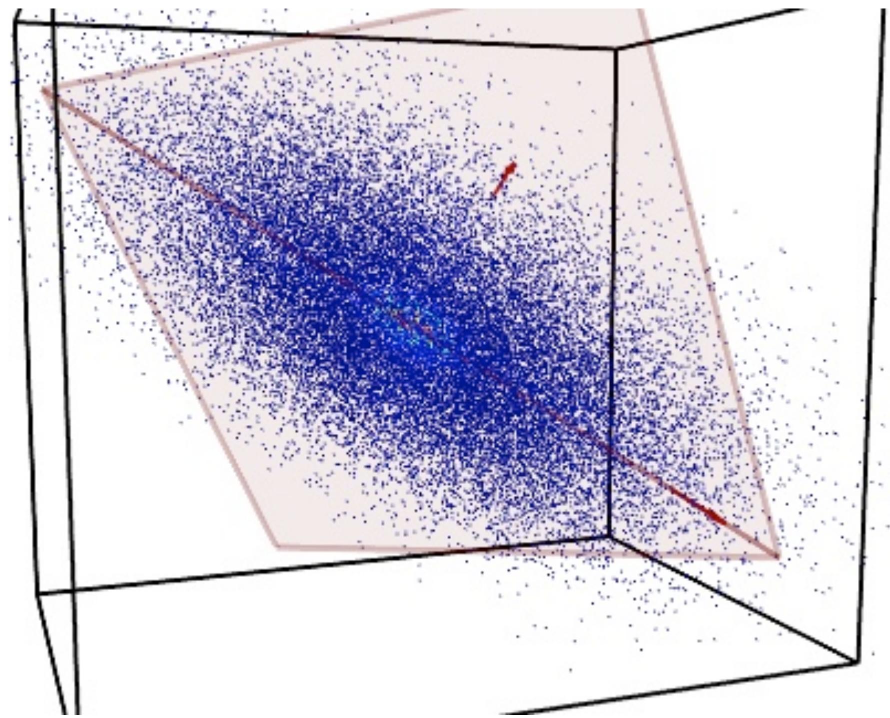


http://research.microsoft.com/en-US/people/kannan/book-no-solutions-aug-21-2014.pdf

Johnson Lindenstrauss Lemma



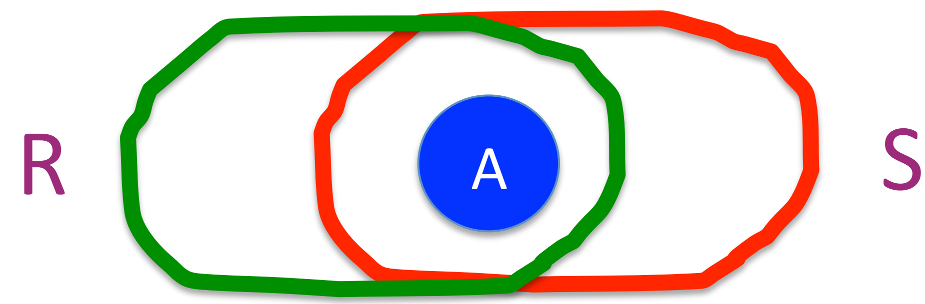




http://www.scipy-lectures.org/_images/pca_3d_axis.jpg

The simplest non-trivial join query

Intersection of R and S

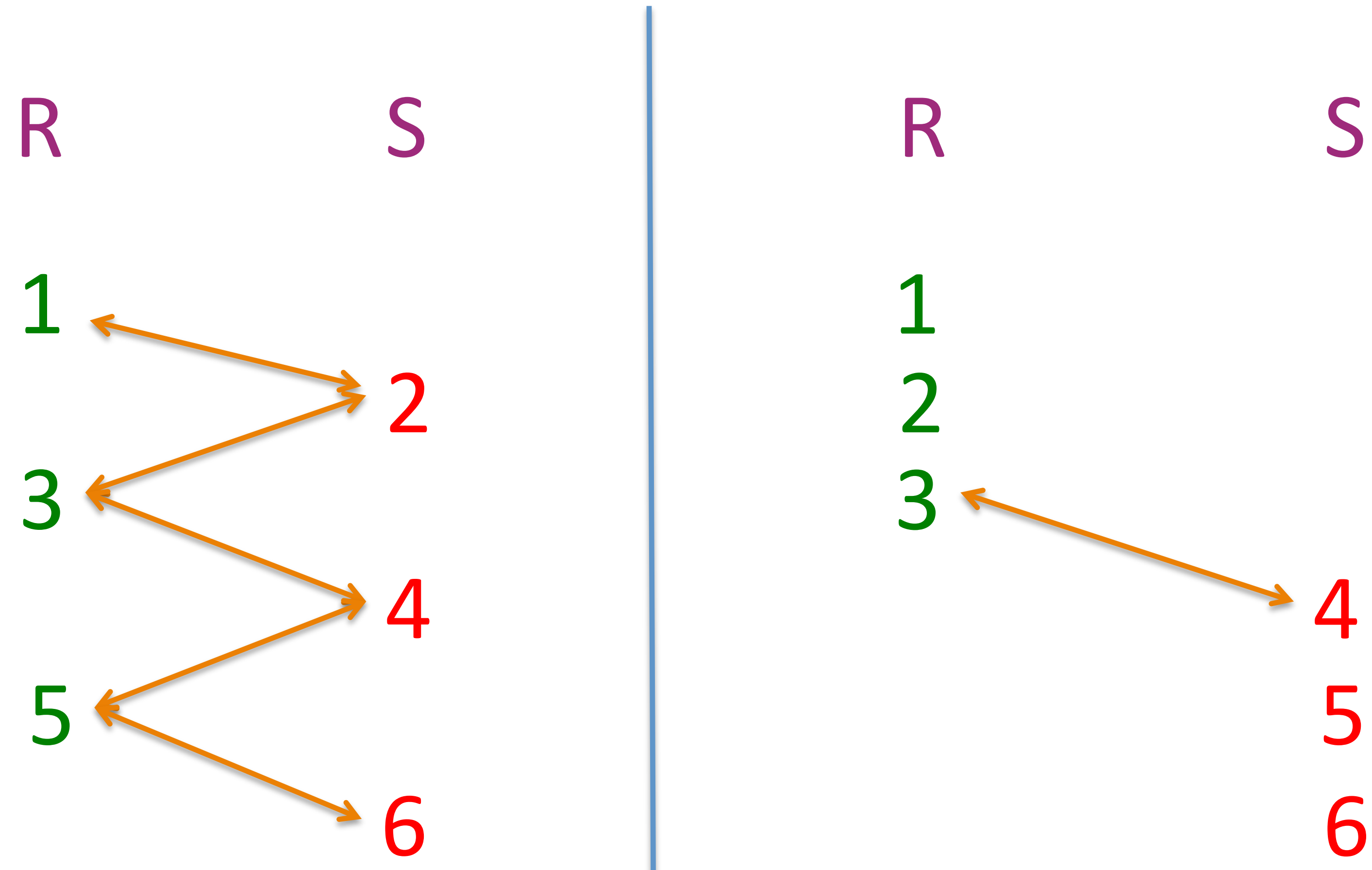


Assume R and S are sorted

Let us concentrate on comparison based algorithms

Assume $|R| = |S| = N$

Not all inputs are created equal

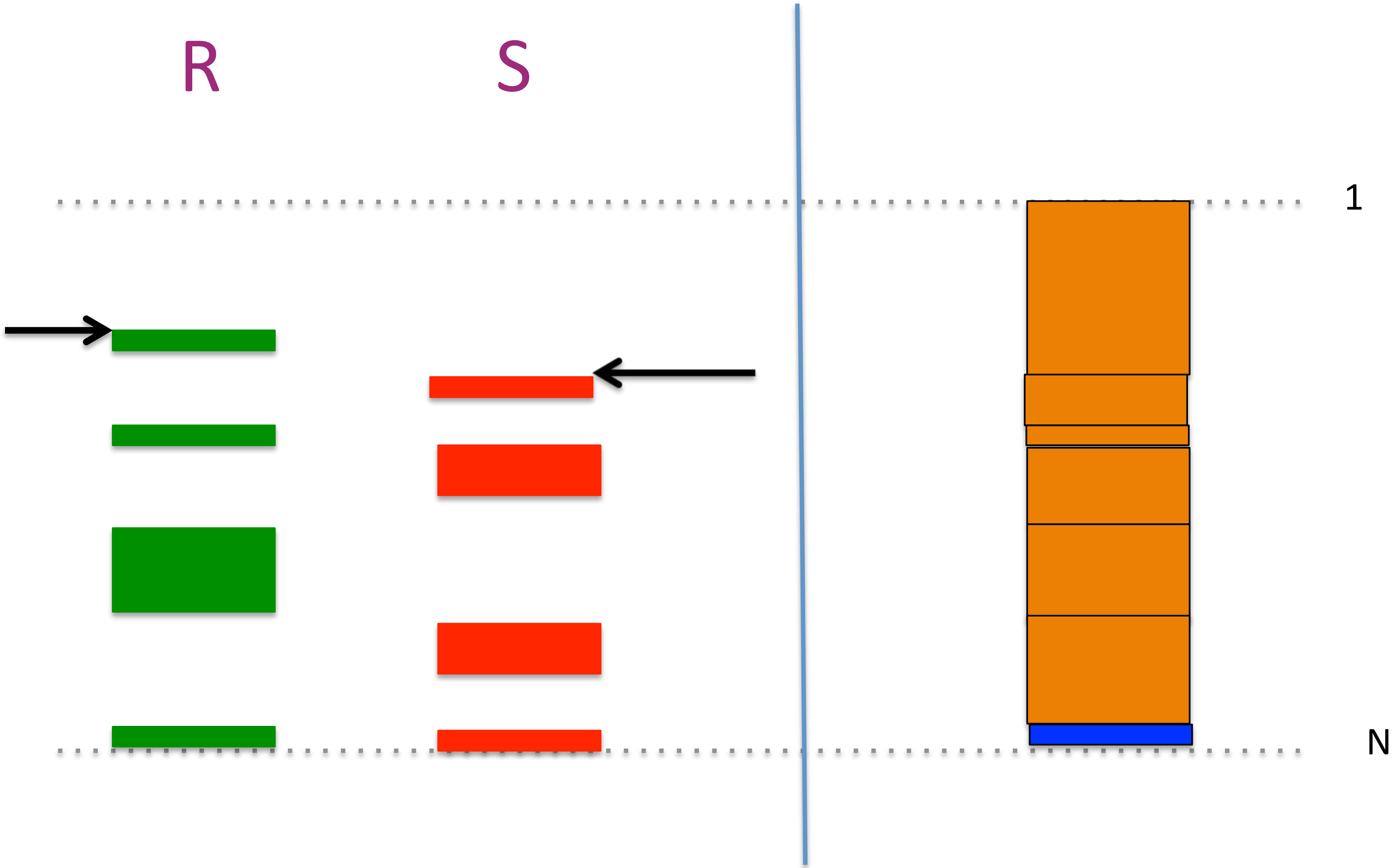


Ω(N) comparisons

1 comparison!

We need a faster/adaptive algorithm

The MERGE algorithm works

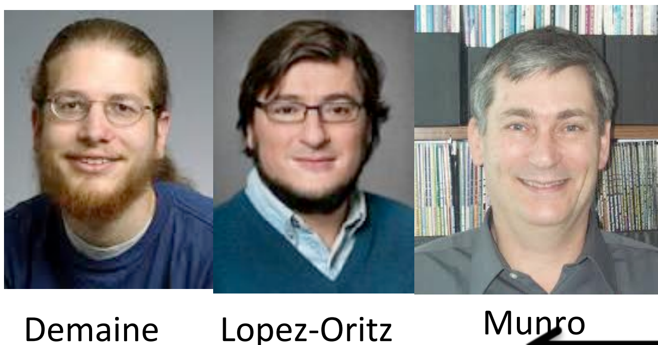


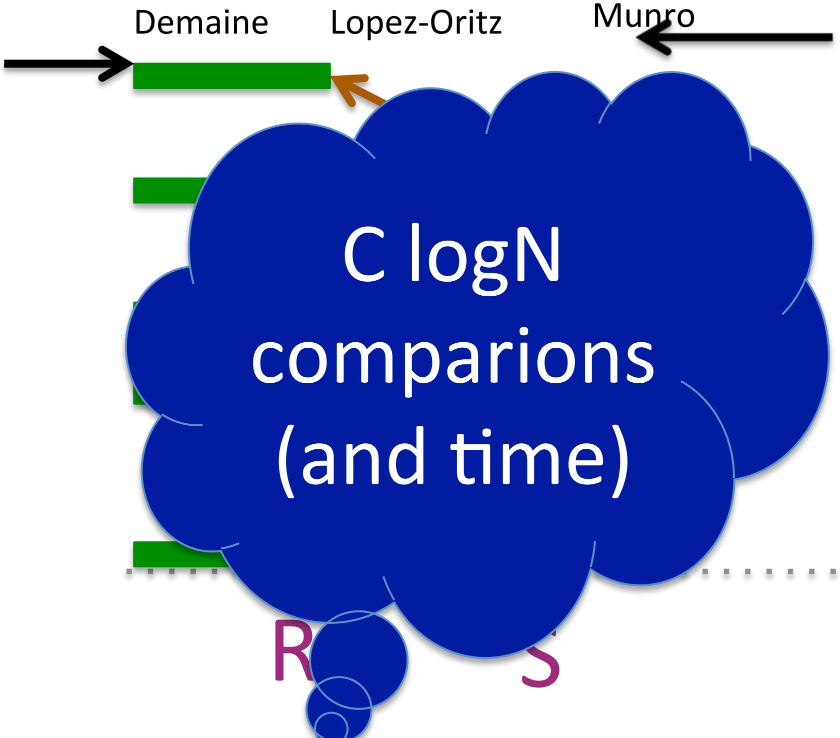
An assumption

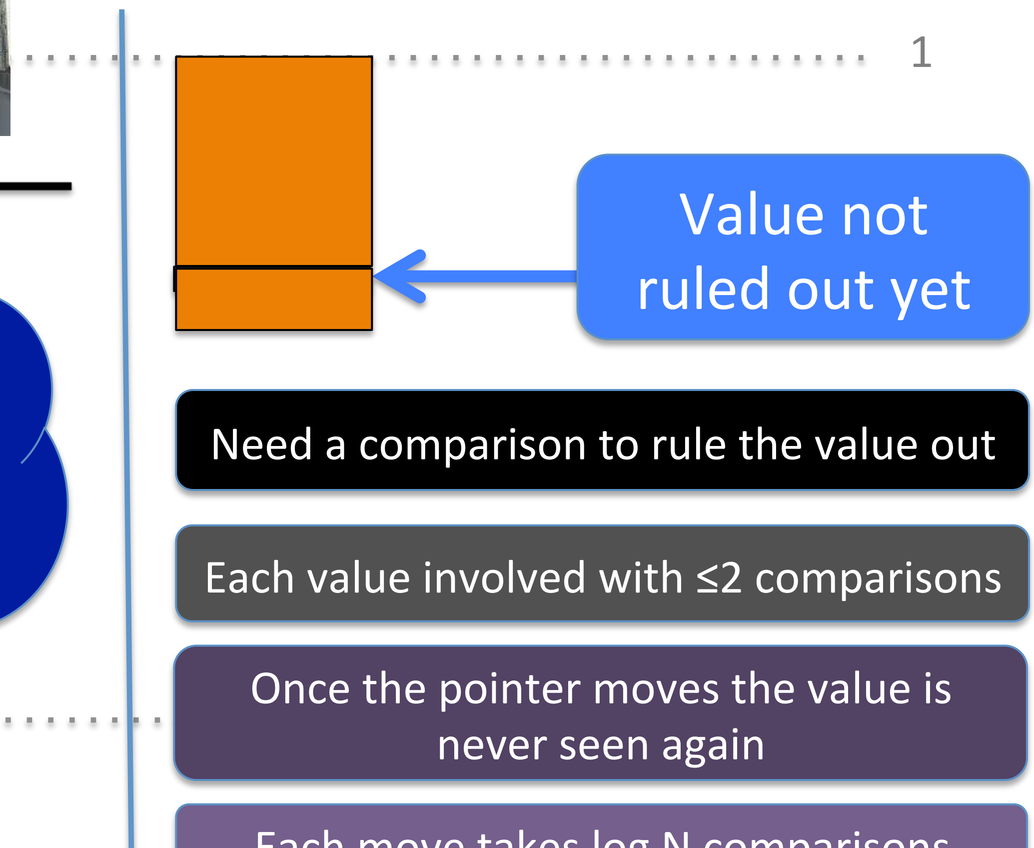
Output of the join is empty

MERGE is (near) instance optimal

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







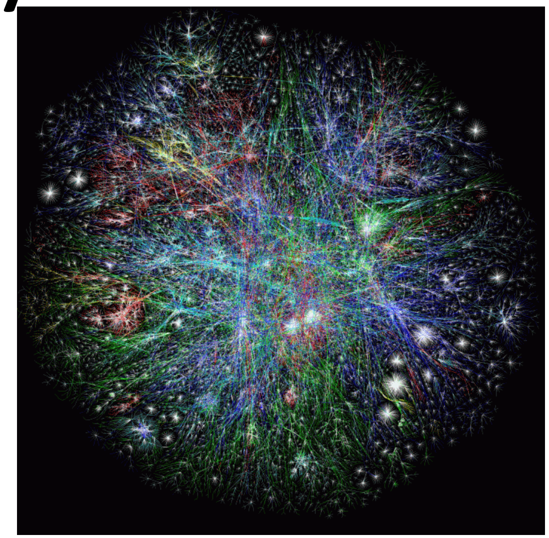
Need a comparison to rule the value out

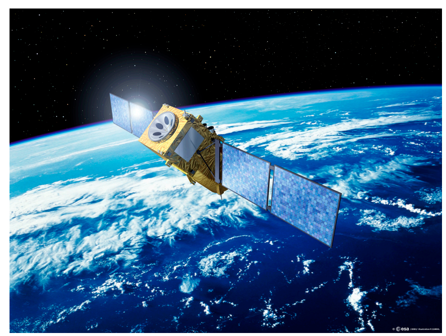
Each value involved with ≤2 comparisons

Once the pointer moves the value is never seen again

Each move takes log N comparisons

Coding Theory

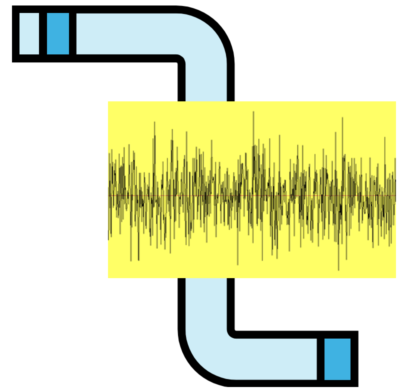




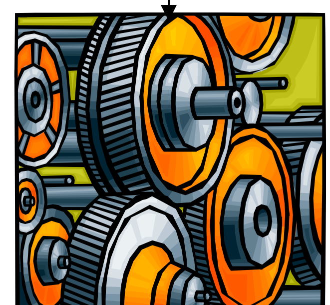
Communicating with my 5 year old

x

C(x)



y = C(x)+error



x

Give up

“Code” **C**

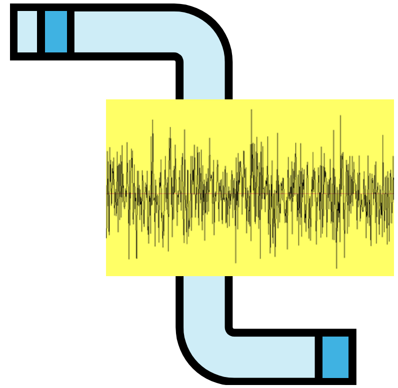
“Akash English”

C(x) is a “codeword”

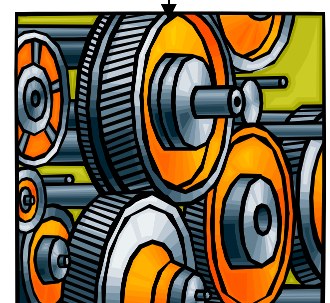
The setup

x

C(x)



y = C(x)+error



x

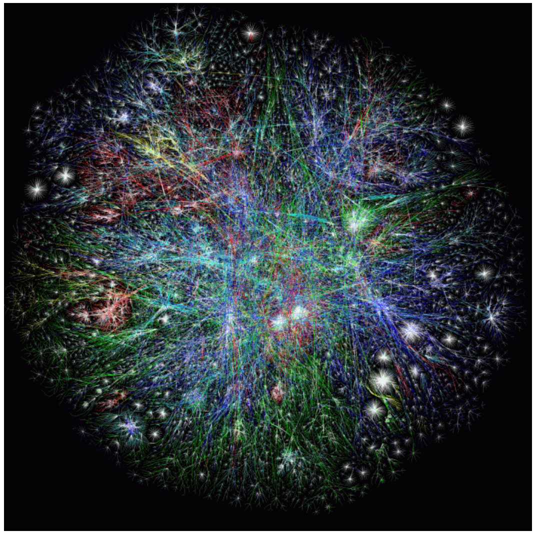
Give up

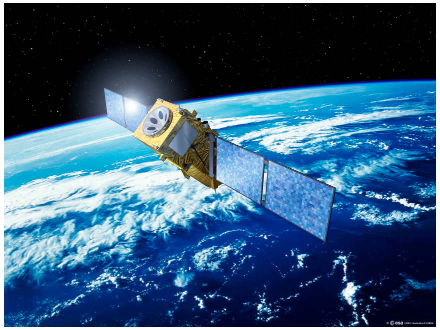
Mapping **C**

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

Different Channels and Codes

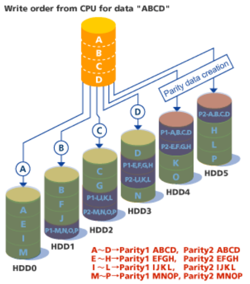
- Internet
  - Checksum used in mult
    layers of TCP/IP stack
- Cell phones
- Satellite broadcast
  - TV
- Deep space telecommunications
  - Mars Rover





“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

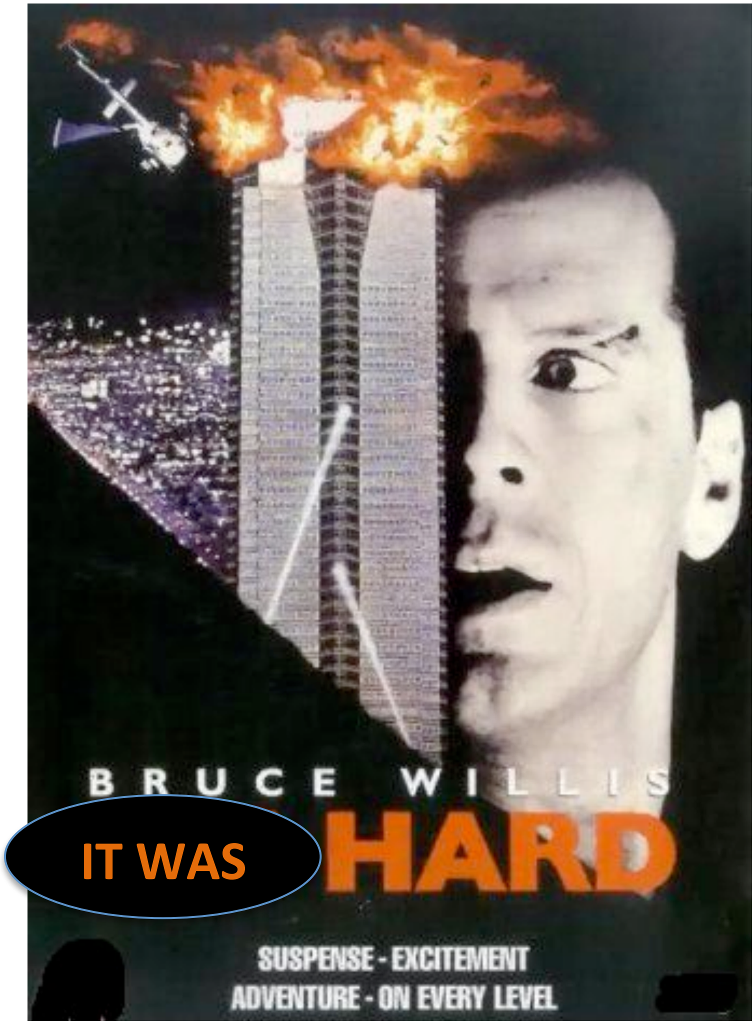
- Correct as many errors as possible with as little redundancy as possible

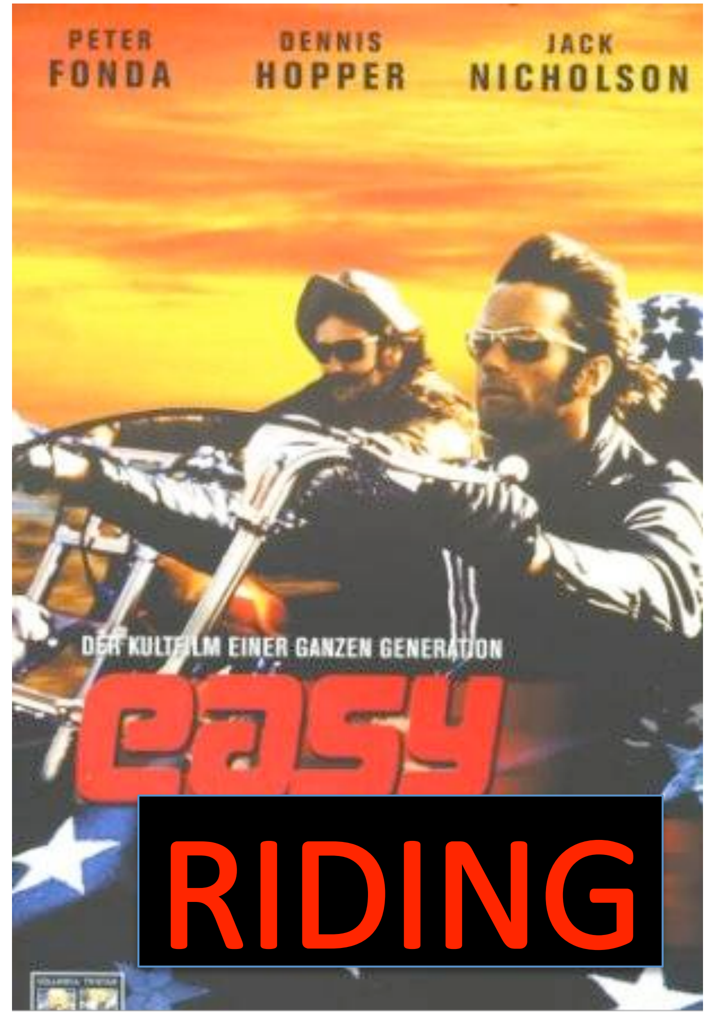
Can one achieve the "optimal" tradeoff with *efficient* encoding and decoding ?

Interested in more?

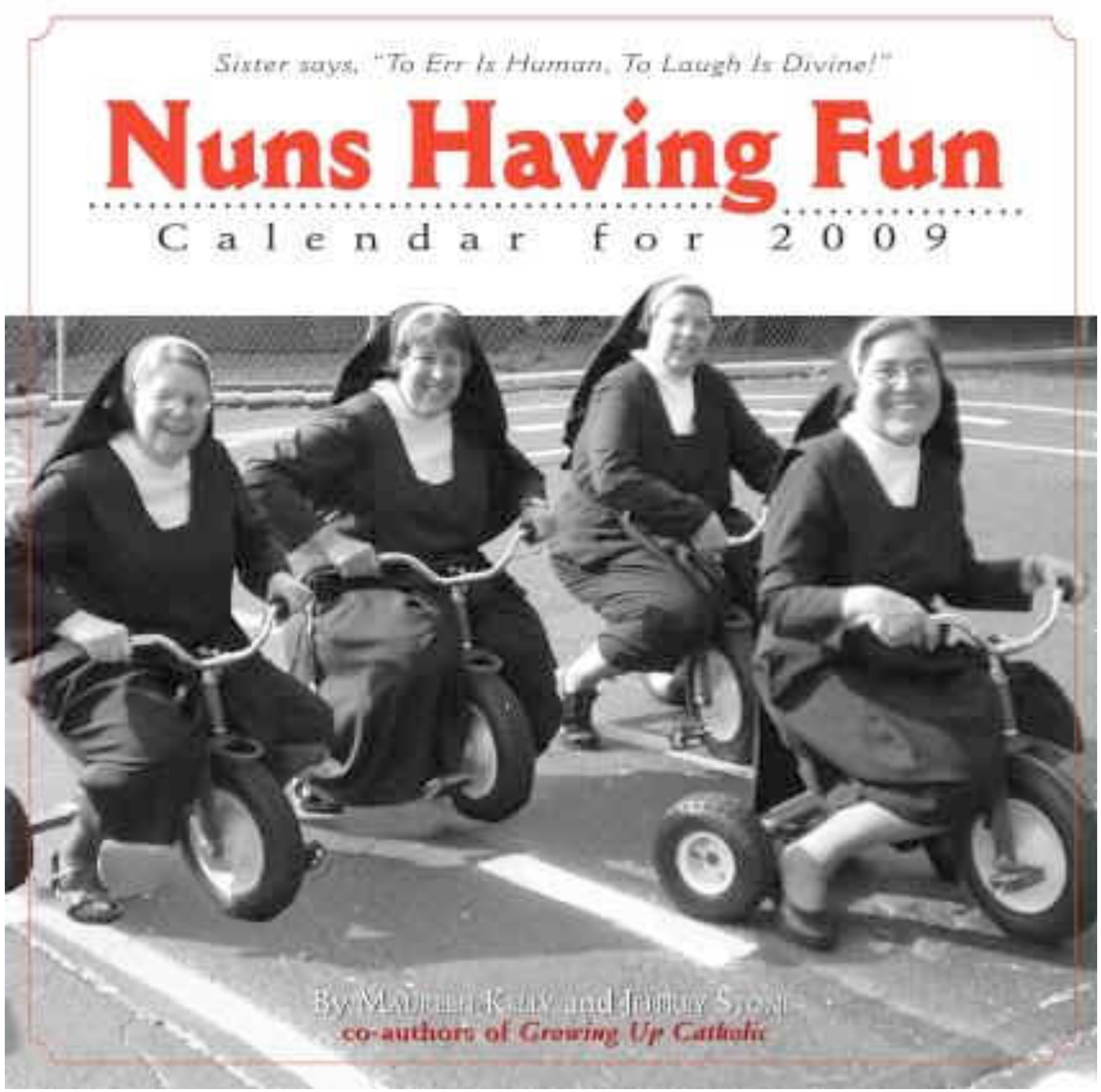
CSE 545, Spring 201?

Whatever your impression of the 331





Hopefully it was fun!



Thanks!

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