Updated 1/23/2021

Principles of Scalable Data Management: theory, algorithms and database systems

Be prepared to very succinctly state:

- What do you hope to get out of this course ☺
- What is your biggest fear for this course ☹
- Something interesting/ surprising about you

Wolfgang Gatterbauer

CS7240 Principles of scalable data management (sp21)

https://northeastern-datalab.github.io/cs7240/sp21/

1/19/2021

"Principles of Scalable Data Management"

Relational databases (and related technologies) are the core technology used for managing data at scale

Our intention it to build solid foundations and look at the algorithmic principles

"Principles of Scalable Data Management"

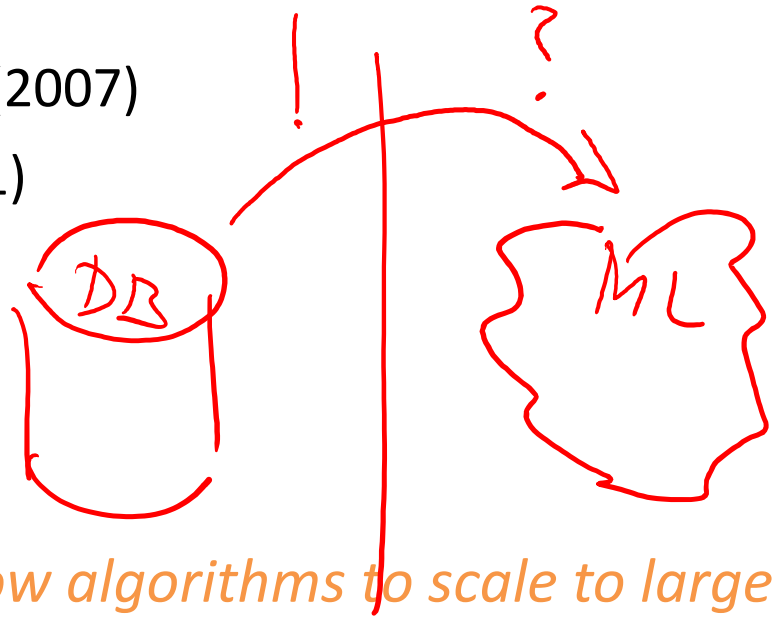
Relational databases (and related technologies) are the core technology used for managing data at scale

Our intention it to build solid foundations and look at the algorithmic principles

Background of instructors: Wolfgang Gatterbauer

Background

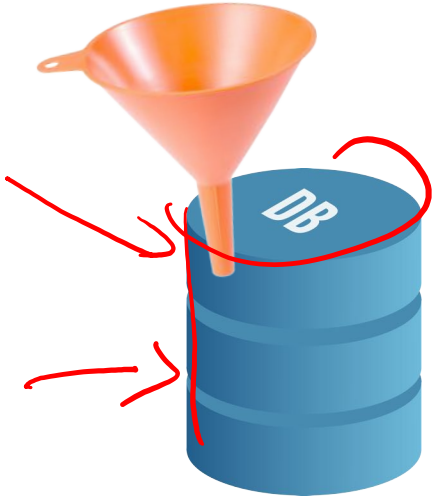
- PhD, Computer Science, Vienna University of Technology (2007)
- PostDoc, Database Group, University of Washington (2011)
- Assistant Professor, Tepper School of Business @ CMU
- At Khoury since 2017



Combining theory with database systems

What are the fundamental algebraic properties that allow algorithms to scale to large amounts of data? How to apply these principled to new data management problems?

- Inconsistencies & Trust
- Provenance & Explanations
- Uncertainty ("Probabilistic data")
- Graphs & Linear Algebra



Let's take turns

We call your name, please succinctly state:

1. What do you hope to get out of this course ☺
2. What is your biggest fear for this course ☹
3. Something interesting/ surprising about you

This helps us get to know each other better / helps me understand your goals and expectations for the course

Foundations of relational databases

Some "birth-years". When was SQL born?

- 2004: Facebook

- 1998: Google
- 1995: Java, Ruby
- 1993: World Wide Web
- 1991: Python

- 1985: Windows

Some "birth-years". When was SQL born?

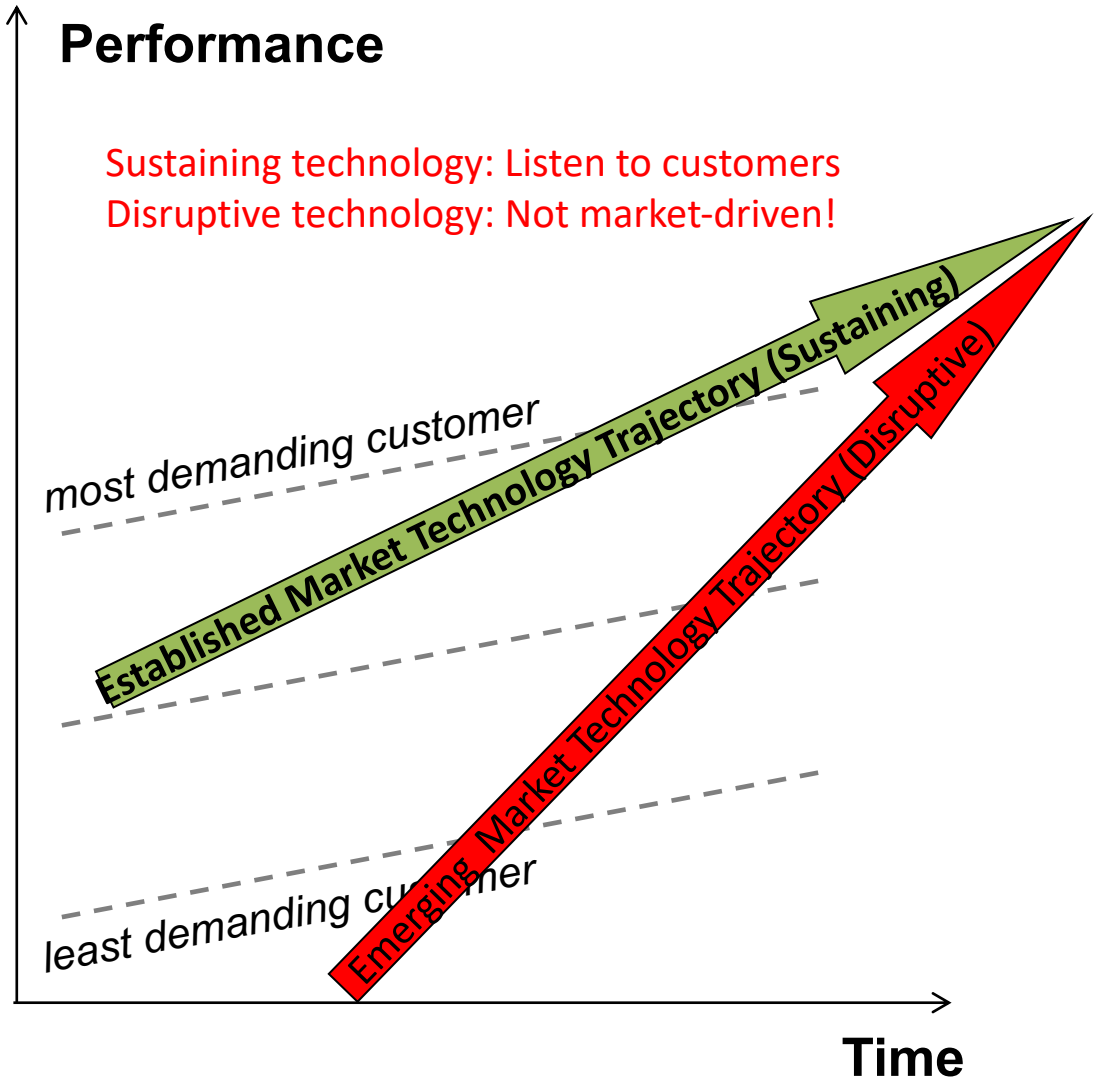
- 2004: Facebook

- 1998: Google
- 1995: Java, Ruby
- 1993: World Wide Web
- 1991: Python

- 1985: Windows

- 1974: SQL

Disruptive Innovation



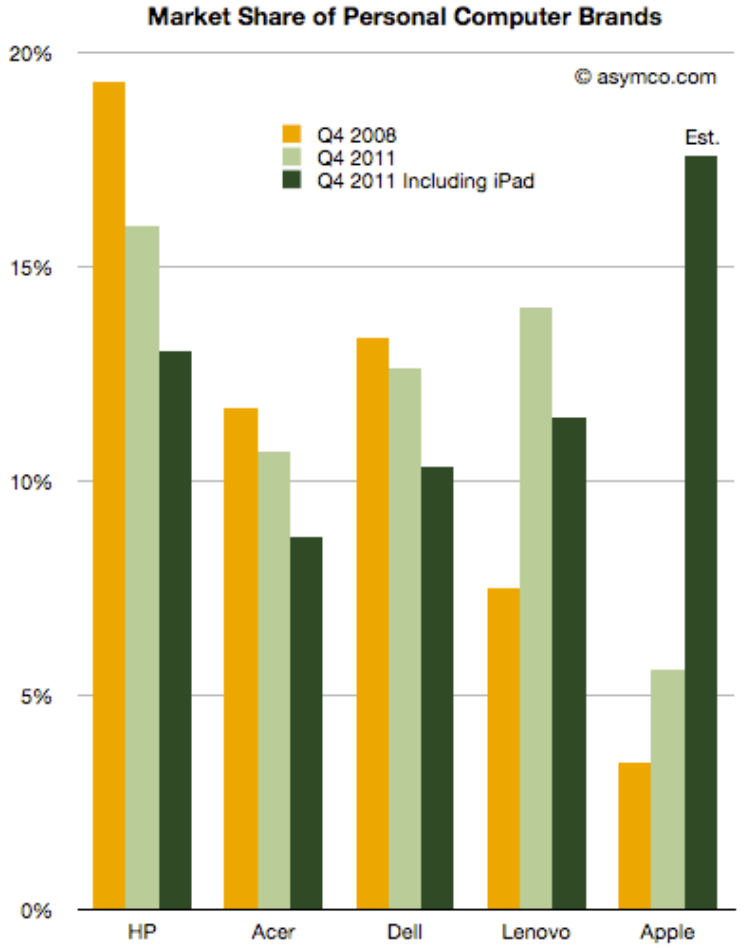
- Disruptive innovations are generally not acceptable for the mass market when they are introduced. Only the fringes of the market pick up the innovation in the first iteration
- It performs worse in one or more areas, but is typically simpler, more reliable, or more convenient than existing technologies.
- It is less profitable than existing technologies. Leading firms' most profitable customers generally can't use it and don't want it.
- As the innovator continues to refine their product the utility value to the market increases
- Its performance trajectory is steeper than that of existing technologies.
- Large organizations are fundamentally incapable of successfully bringing it to market.

Source: Compiled from “The Innovator’s Dilemma” by Clayton Christensen, 1997.

iPhone: Disruptive Innovation or not?

1: "Business Phones"
Microsoft in 2007

2: Laptops



Source: http://www.youtube.com/watch?v=eywi0h_Y5_U , http://www.asymco.com/2012/01/16/apple-is-the-top-personal-computer-vendor/

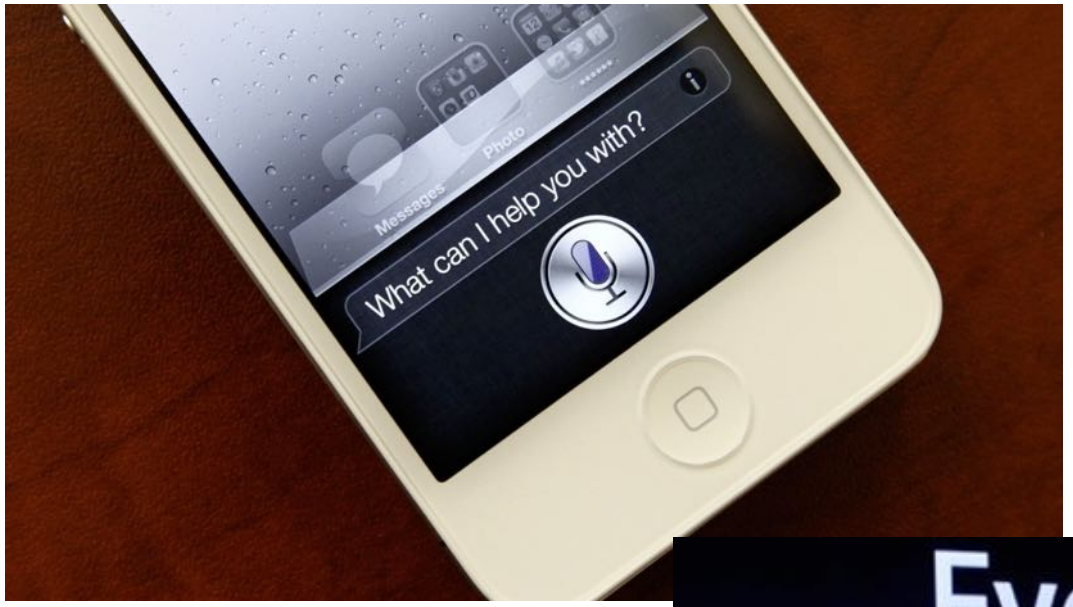
What keyboards without keys can do...

In Feb 2016, SwiftKey was purchased by Microsoft, for 250 M$

Sources: https://en.wikipedia.org/wiki/Swype, https://en.wikipedia.org/wiki/SwiftKey

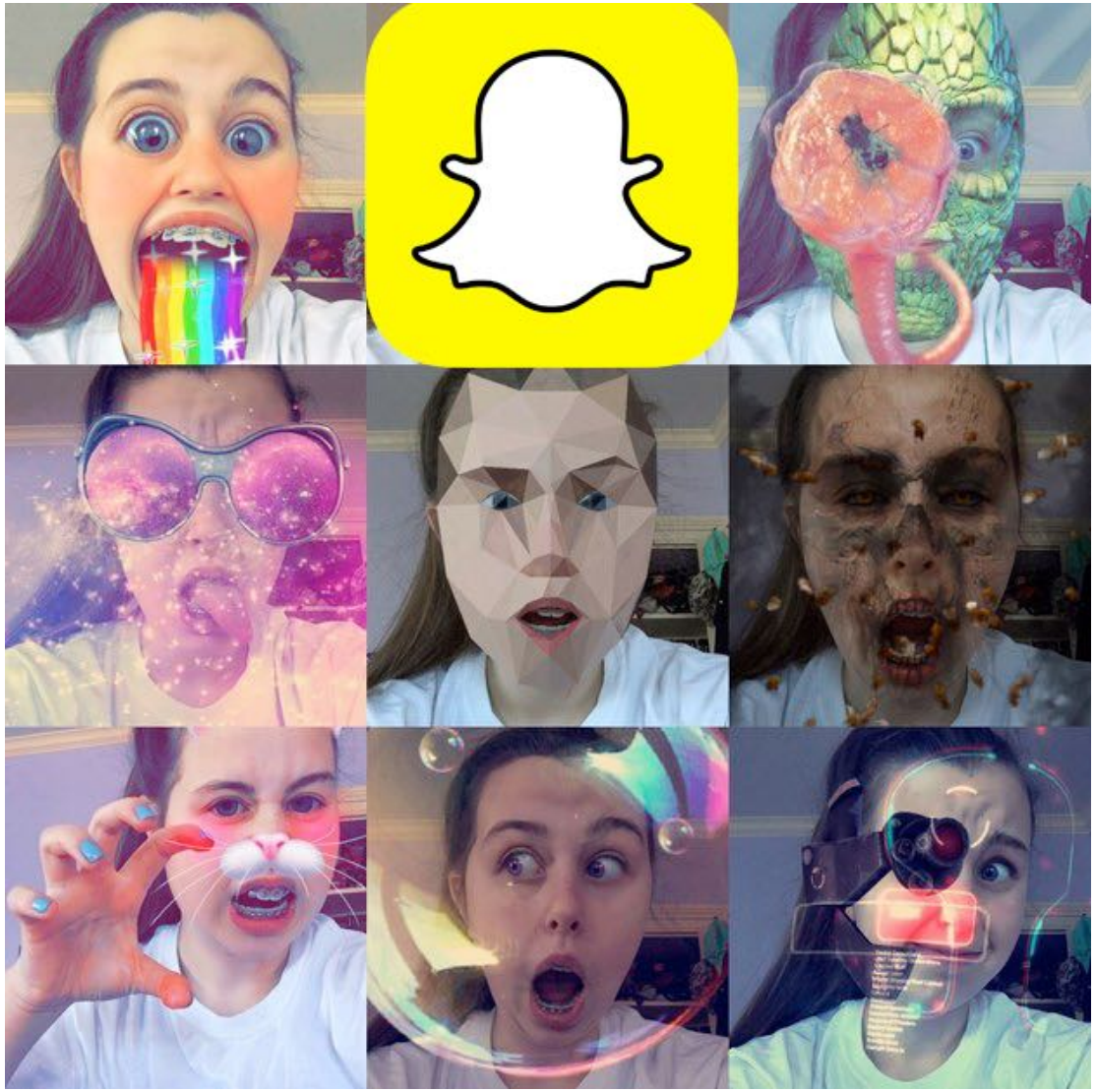
The keyboard of the future?

Sources: http://www.wired.com/2014/06/siri_ai/


Messages
Photo
What can I help you with?


Eyes Free
MODE
MODE
FM/AM
engadget

Keyboards? Do we need text to communicate?



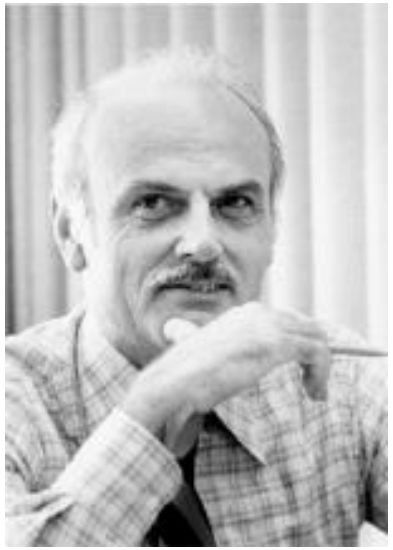
Sources: http://www.buzzfeed.com/benrosen/how-to-snapchat-like-the-teens (2/2016)

What is this? (1975)

Source: http://pluggedin.kodak.com/pluggedin/post/?id=687843

SQL: some history

- Dr. Edgar Codd (IBM)
  - CACM June 1970: "A Relational Model of Data for Large Shared Data Banks" http://seas.upenn.edu/~zives/03f/cis550/codd.pdf
- Standardized
  - 1986 by ANSI: SQL1
  - 1992: Revised: SQL2
    - Approx 580 page document describing syntax and semantics
  - Revised: 1999, 2003, 2008, ...
- Players
  - Oracle (Relational Software), Microsoft, IBM, ….
- Every vendor has a slightly different version of SQL
- But the main commands are standardized



Codd's (disruptive ?) innovation

A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on n-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. Introduction

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of *data independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of *data inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

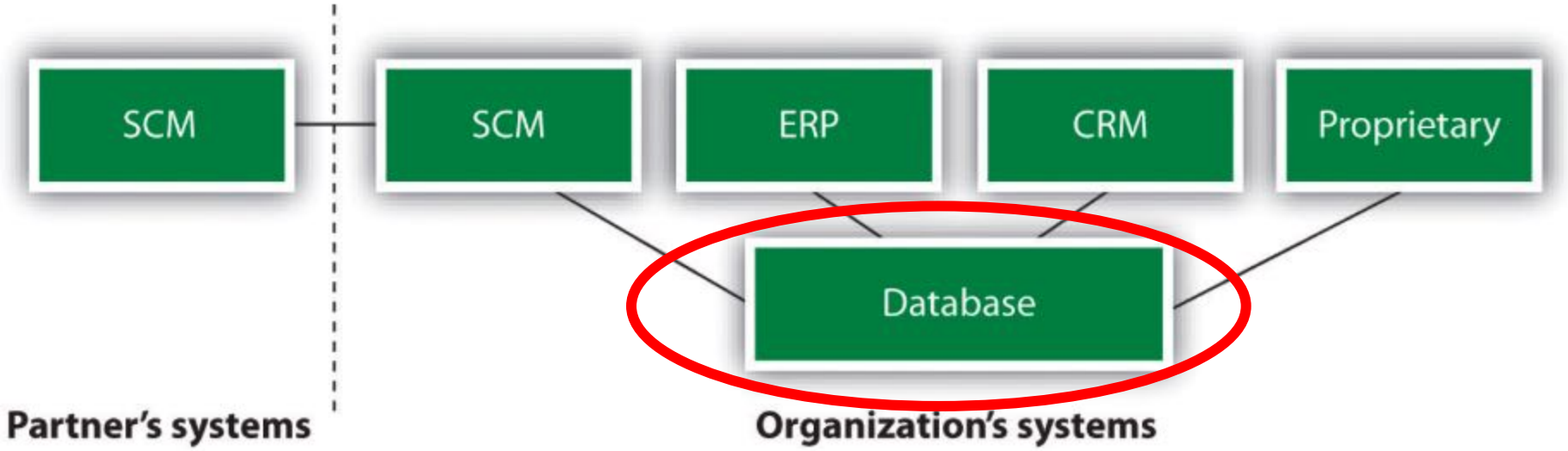
1.2. Data Dependencies in Present Systems

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

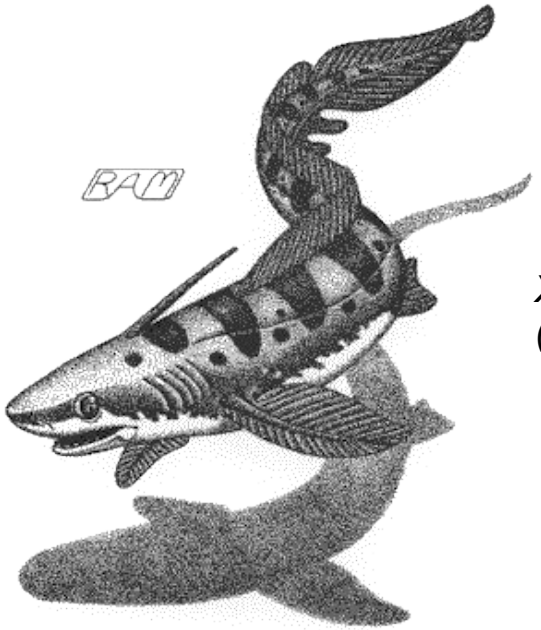
Source: EF Codd, "A Relational Model of Data for Large Shared Data Banks"
https://scholar.google.com/scholar?cluster=1624408330930846885

SQL and the relational model as standard

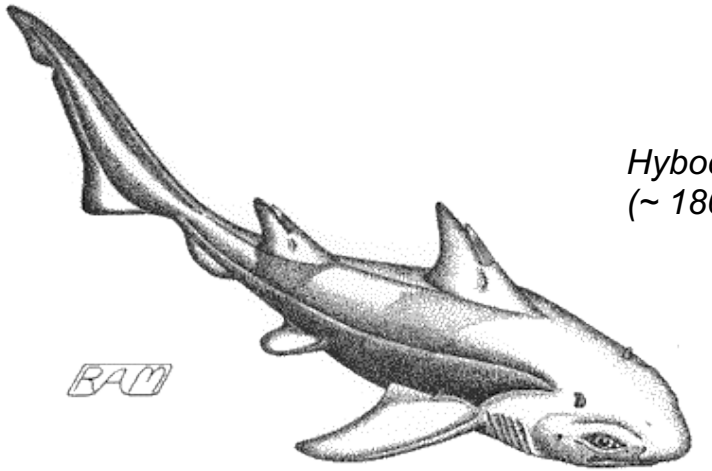


Source: "Information Systems: A Manager's Guide to Harnessing Technology (book v1.4)," p.185, Gallaugher, 2012.

Evolution of Sharks



Xenacanthus
(~ 280 million years ago)

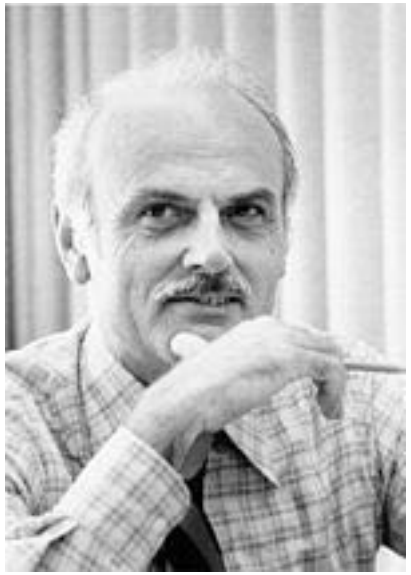


Hybodus sp.
(~ 180 million years ago)

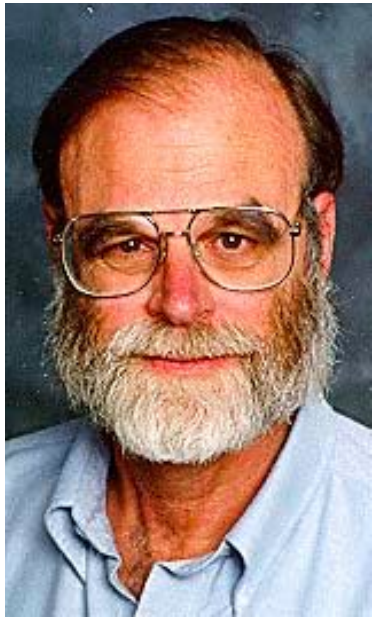
Source for picture on the right: http://themetapicture.com/evolution-it-makes-you-lose-friends/

Four Turing Award Winners

Charles Bachmann 1973



Edgar Codd 1981



Jim Gray 2004

Michael Stonebraker 2014

Seminal contributions made in Industry

Some example highlights

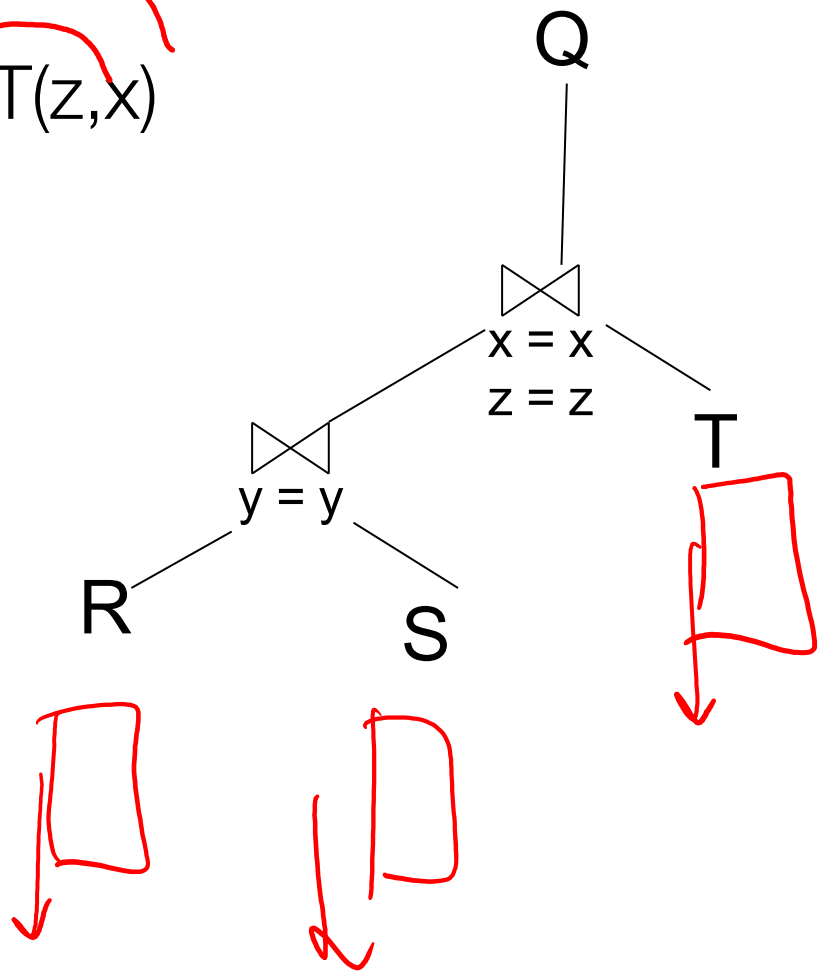
Joins in databases

Efficient multi-way join processing

$$Q(x,y,z) = R(x,y), S(y,z), T(z,x)$$

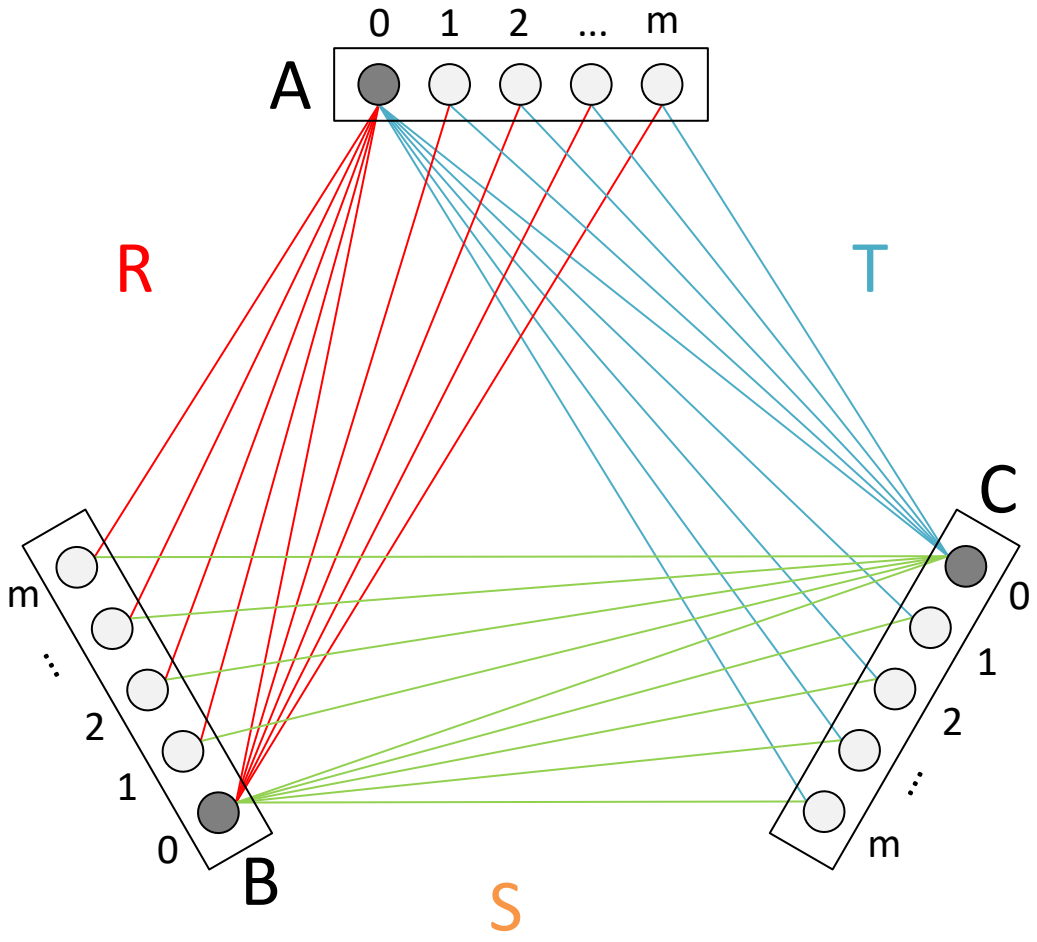
Three plans

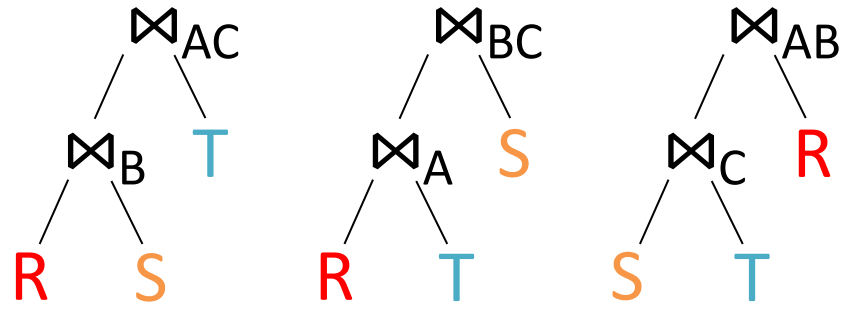
- $(R \bowtie S) \bowtie T$
- $(S \bowtie T) \bowtie R$
- $(T \bowtie R) \bowtie S$



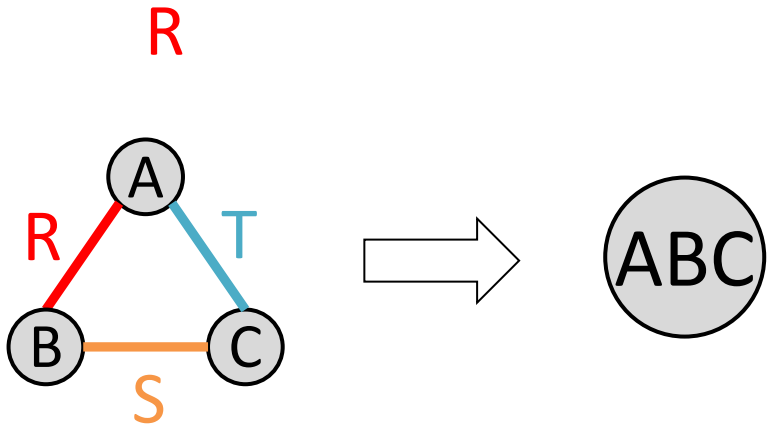
Can we do better? ☺

$Q_\Delta = R(A,B) \bowtie S(B,C) \bowtie T(A,C)$





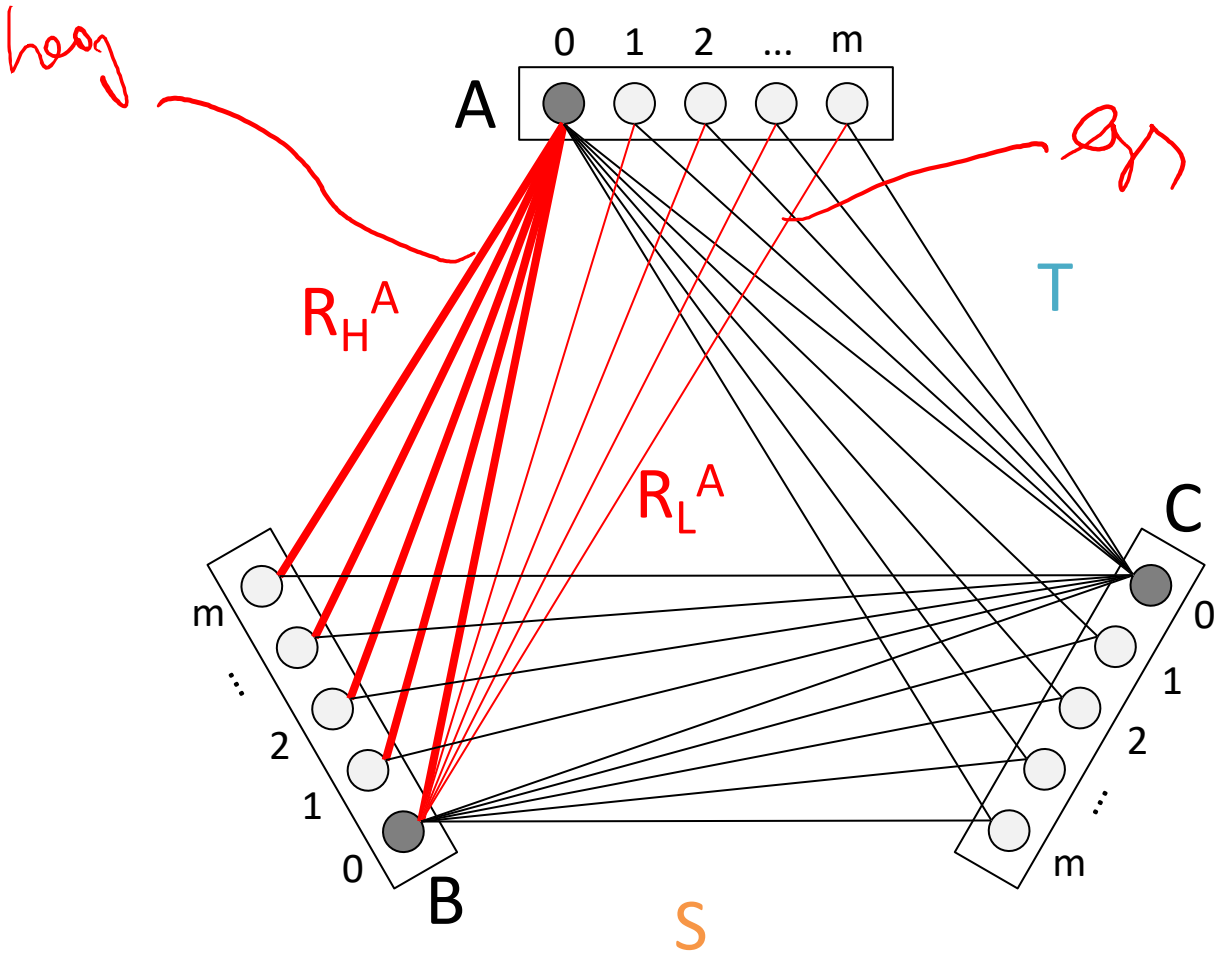
Query plans (correspond to variable elimination orders)

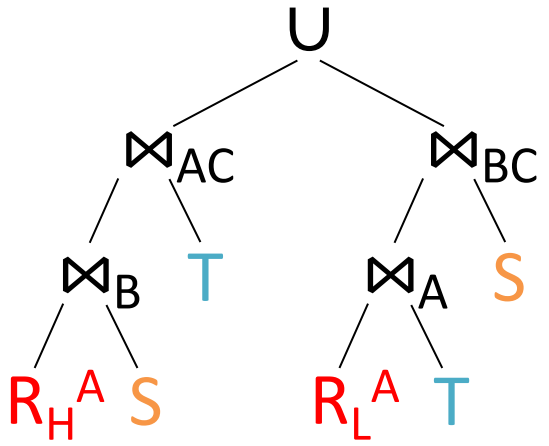


Query (hyper)graph Tree decomposition

$Q_\Delta = R(A,B) \bowtie S(B,C) \bowtie T(A,C)$

$R = R_H^A \cup R_L^A$





Using multiple query plans

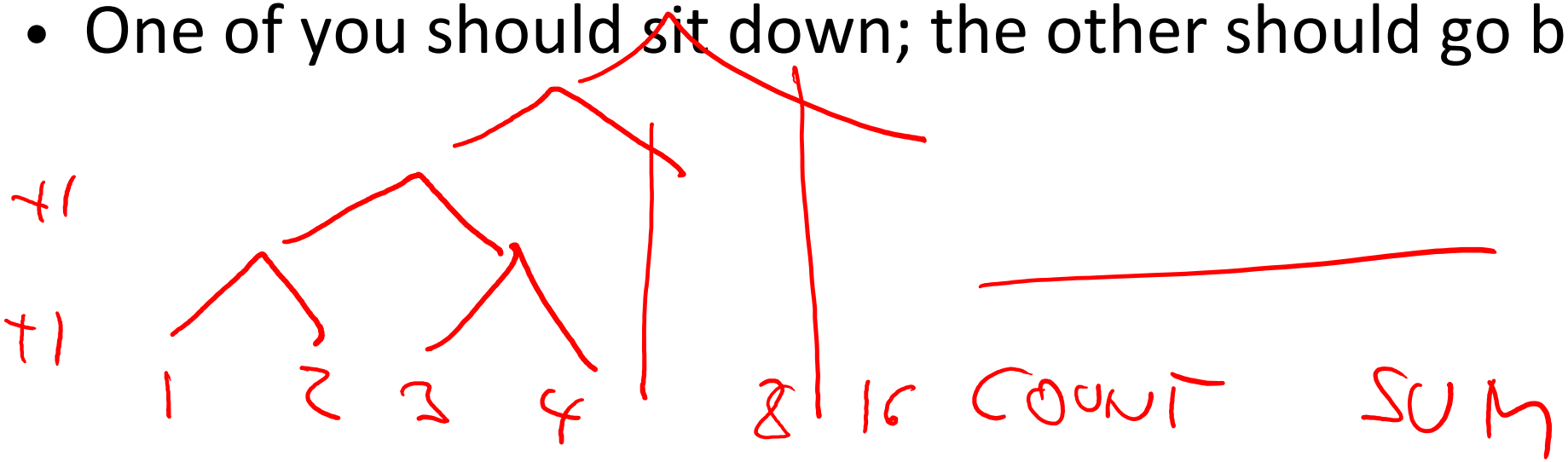
$O(n^{3/2})$

Generalizing Dynamic Programming

- DP: solves optimization problems that show the "optimal subproblem property":
  - an optimal solution for the problem needs to include optimal solutions to any subproblem
  - Forward pointer: algebraic properties that allow such "factorization"
- Assume you don't just want to find the optimal solution = top-1
  - But you want to *enumerate* all solution. First best (top-1), then 2nd best, then 3rd, etc.
- Can you do this in an "optimal time"? How do define "optimal"?

Parallel query processing by example: An Algorithm

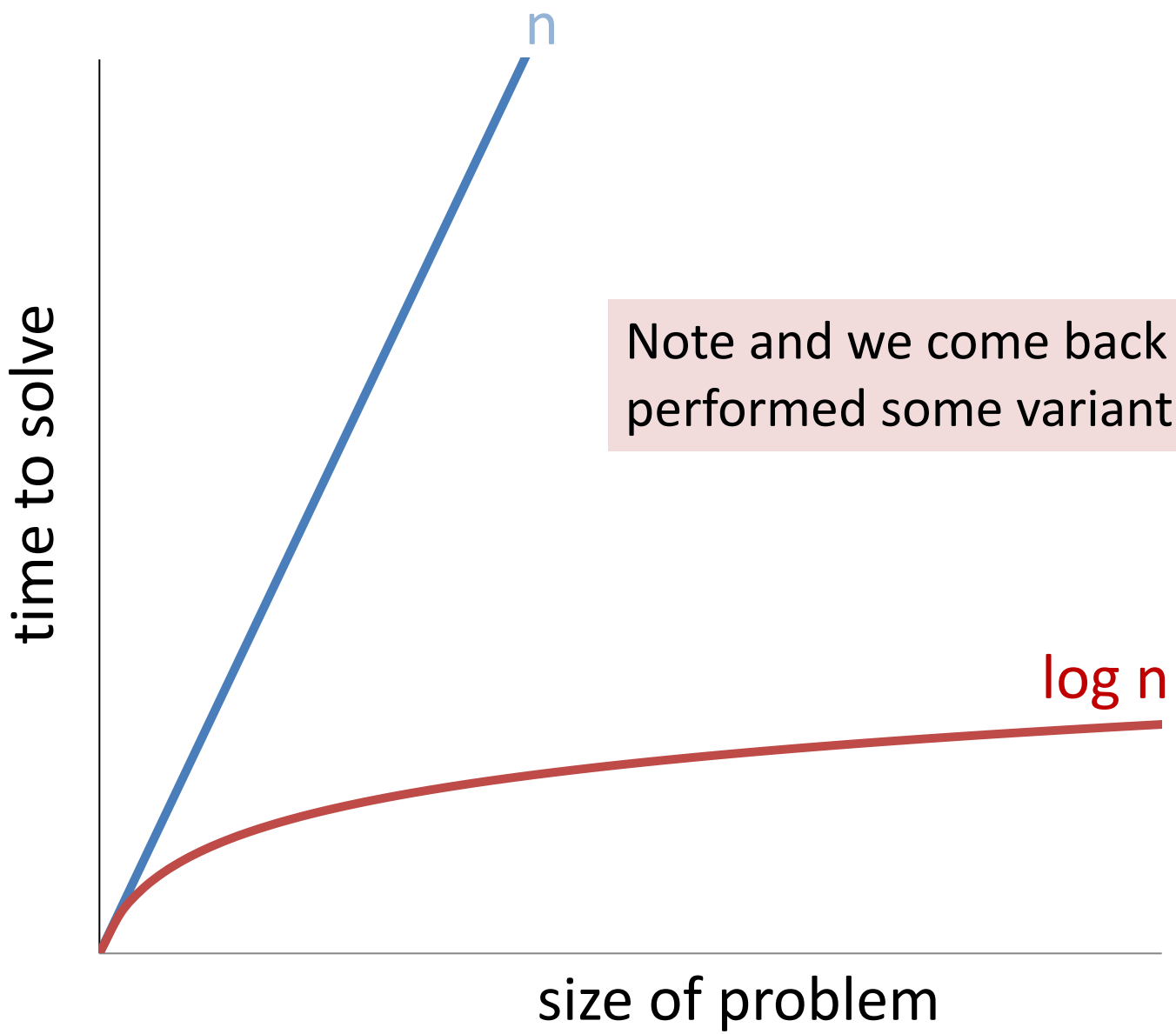
- Stand up and think of the number 1
- Pair off with someone standing, add your numbers together, and take the sum as your new number
- One of you should sit down; the other should go back to step 2



COUNT SUM

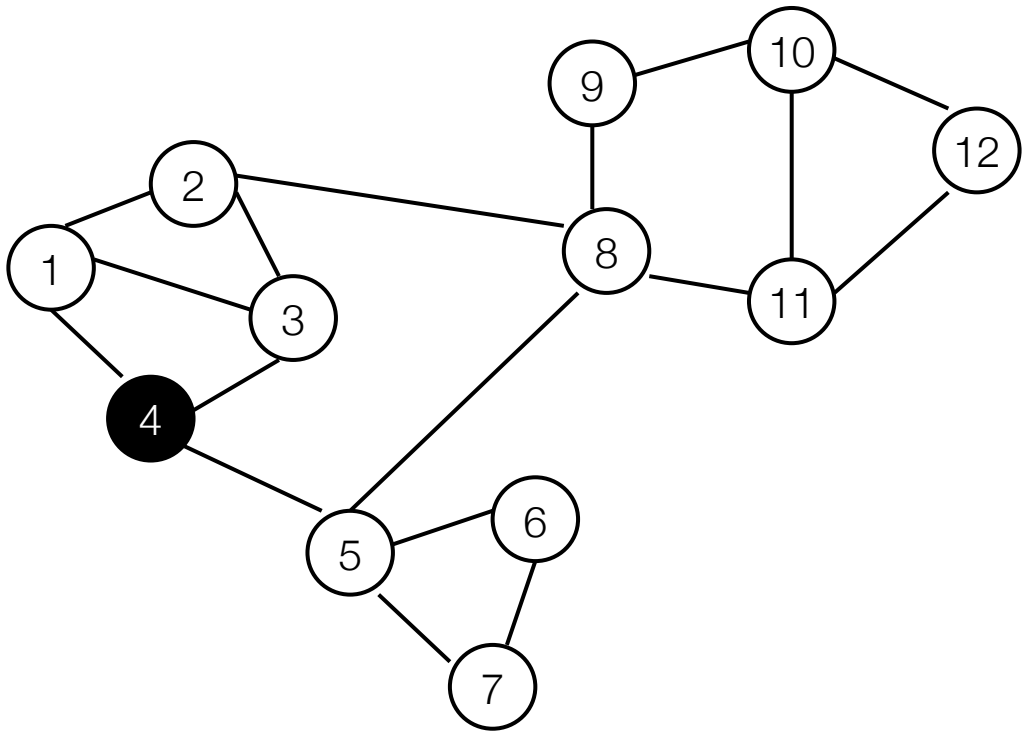
AVG = SUM / COUNT

Parallel query processing: Scalability



Note and we come back to this later: you performed some variant of message passing!

Graphs (Matrix Algebra)



$\mathbf{W}^{\text{col}}$

column-normalized
adjacency matrix

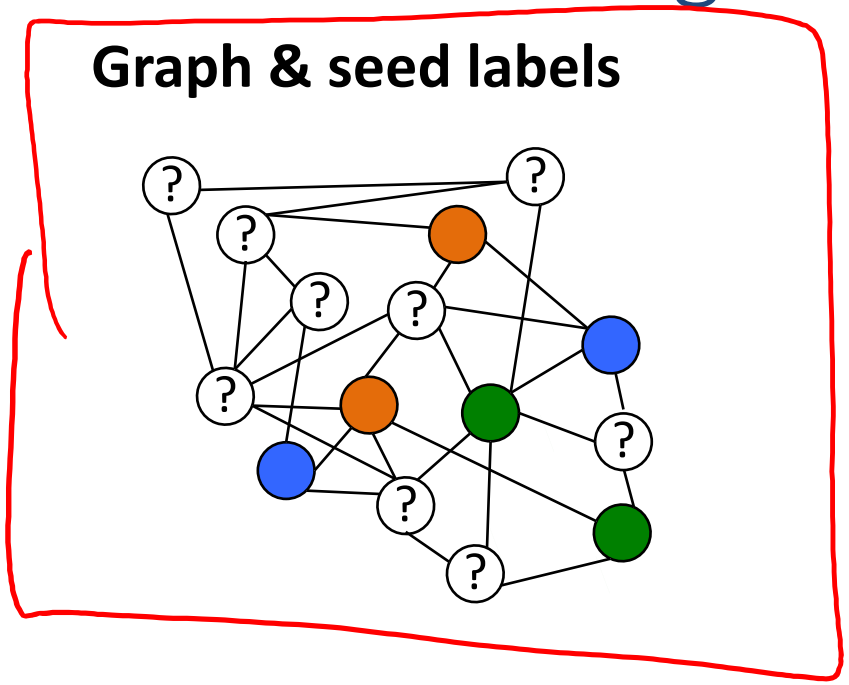
$$\begin{bmatrix}
0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 & 0 \\
1/3 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1/3 & 0 & 1/3 & 0 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1/3 & 0 & 1/2 & 1/2 & 1/4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/4 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/4 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1/3 & 0 & 0 & 1/4 & 0 & 0 & 0 & 1/2 & 0 & 1/3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 1/3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 1/3 & 1/2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 1/3 & 0 & 1/2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 0
\end{bmatrix}$$

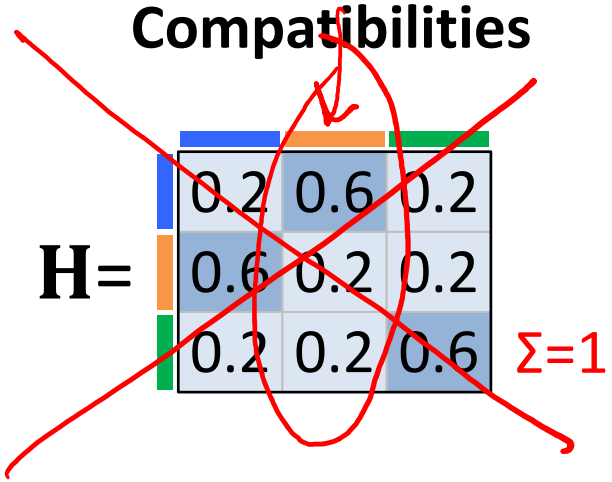
"How close" is each node to node 4?

-> "Personalized PageRank"

Linear algebra ☺

A problem where algebra give surprising speed-ups





Prob 1: Propagating arbitrary compatibilities

Given:
- undirected graph **W**
- seed labels
- compatibilities **H**

Find:
- remaining labels

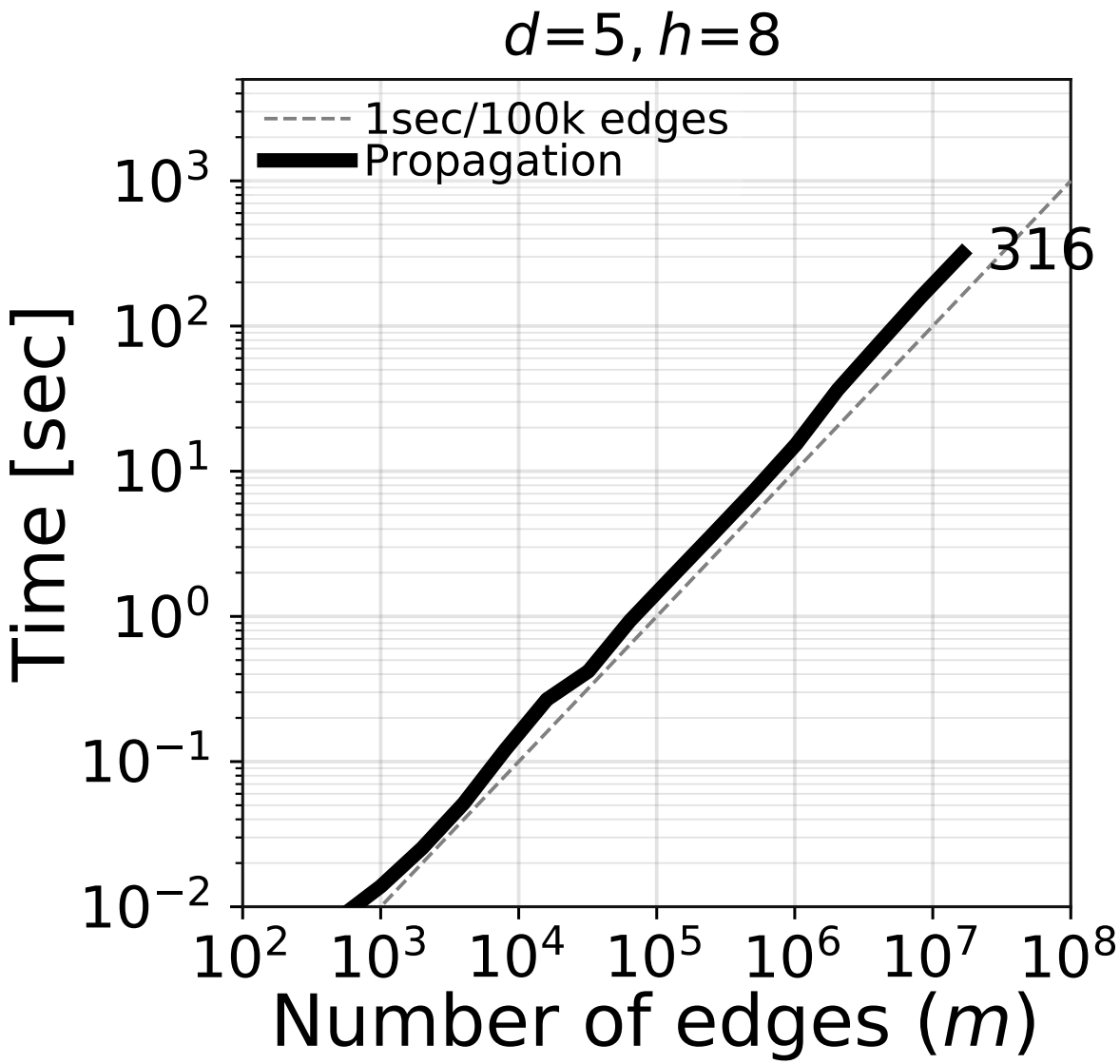
Prob 2: Learning & prop. compatibilities

Given

Find

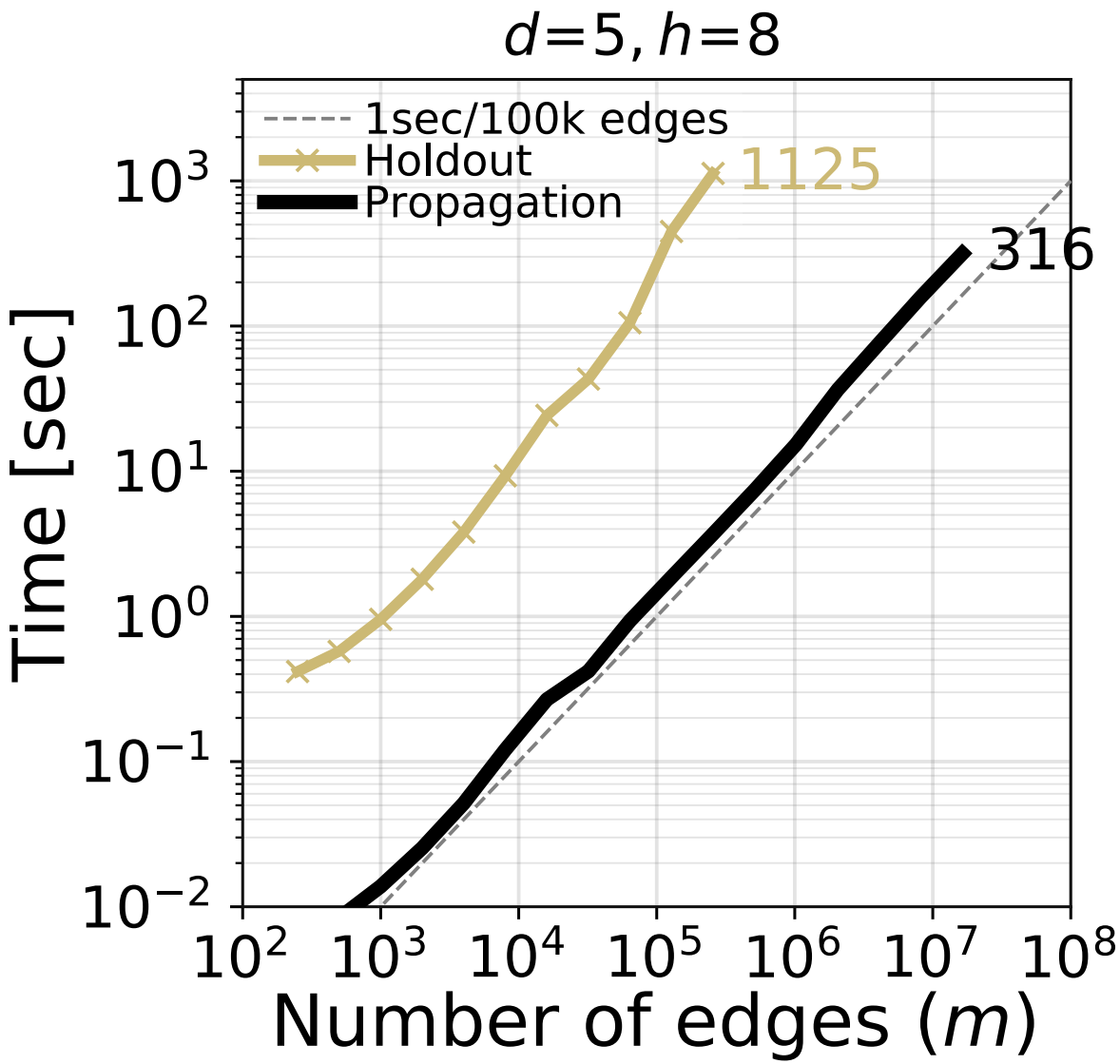
No more heuristics!

What is the overhead of compatibility estimation?



Propagation scales linearly with graph size (number of edges)

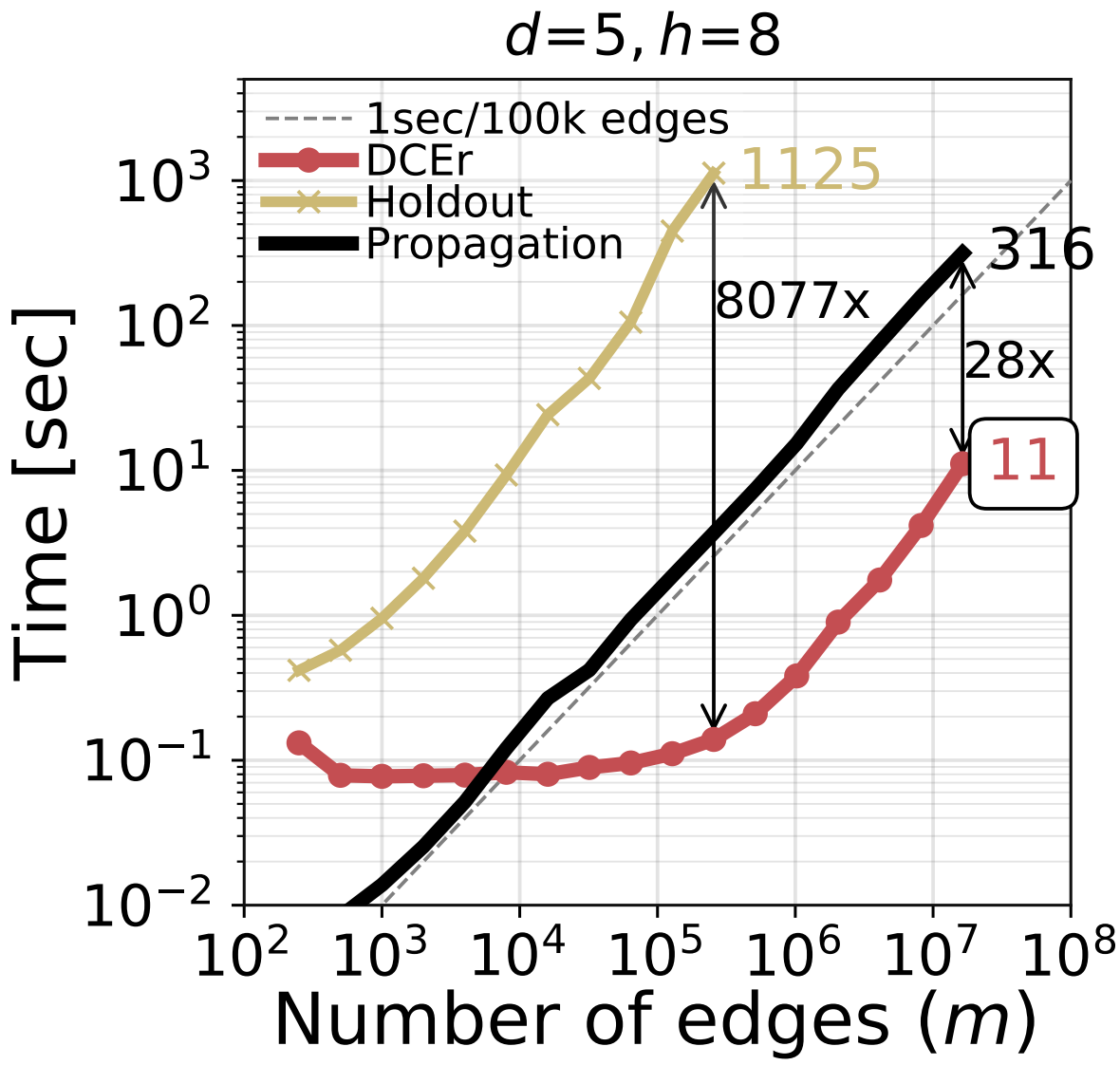
What is the overhead of compatibility estimation?



Propagation scales linearly with graph size (number of edges)

Learning commonly uses inference as subroutine (> 10^2 times slower)

What is the overhead of compatibility estimation?

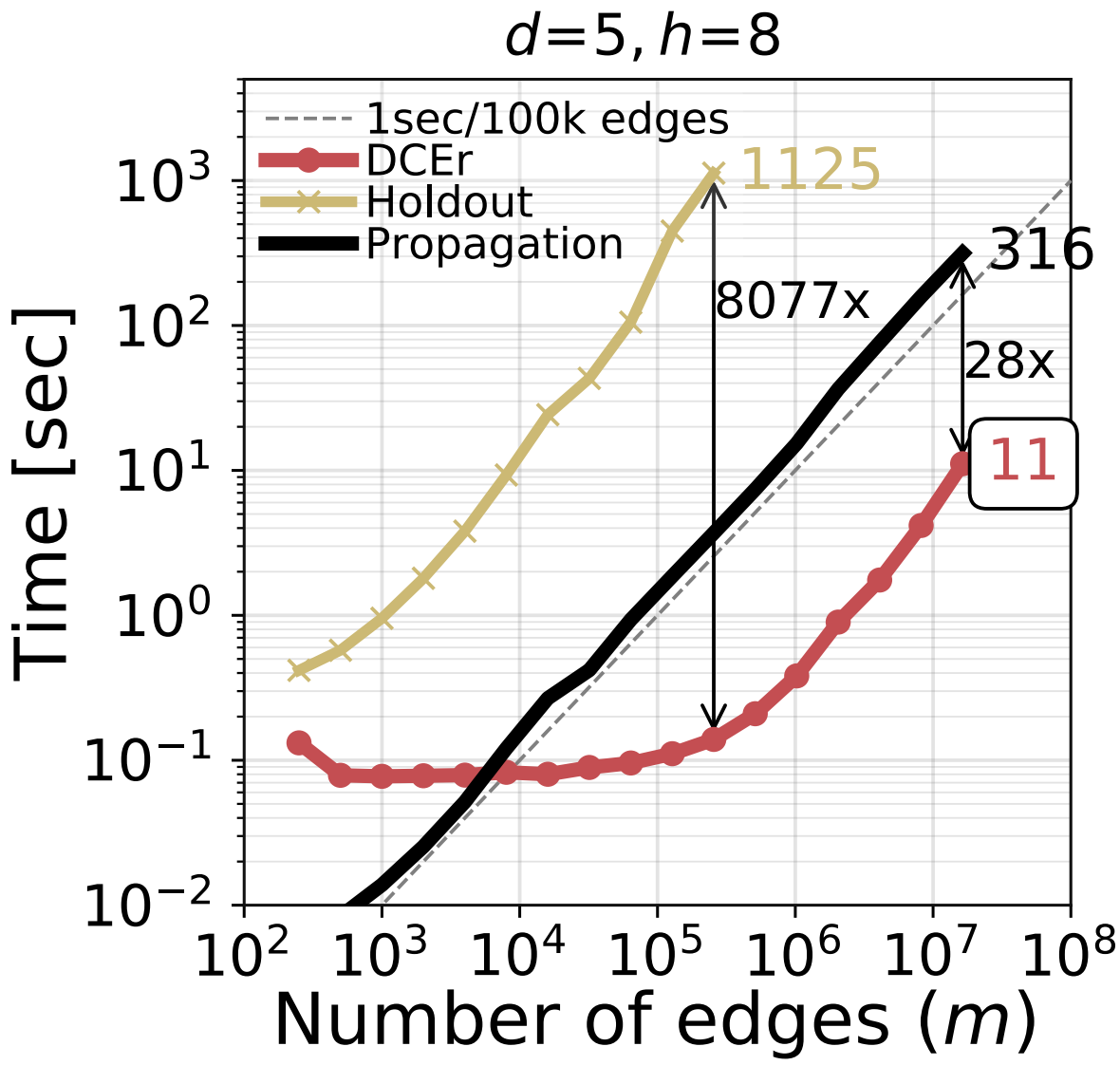


Propagation scales linearly with graph size (number of edges)

Learning commonly uses inference as subroutine (> 10^2 times slower)

Our estimation is *faster* then inference (> 10 times faster)

What is the overhead of compatibility estimation? Basically for free!



Propagation scales linearly with graph size (number of edges)

Learning commonly uses inference as subroutine (> 10^2 times slower)

Our estimation is *faster* then inference (> 10 times faster)

It is basically for free. No more need for heuristics or domain experts!

"Factors"

Assume for each course, we can independently choose a lecturer and a book. Can we represent this table more compactly?

Classes

Course	Lecturer	Book
cs3200	Renee	Complete book
cs3200	Wolfgang	Complete book
cs3200	Renee	Cow book
cs3200	Wolfgang	Cow book

=

Lecturer

Course	Lecturer
cs3200	Renee
cs3200	Wolfgang

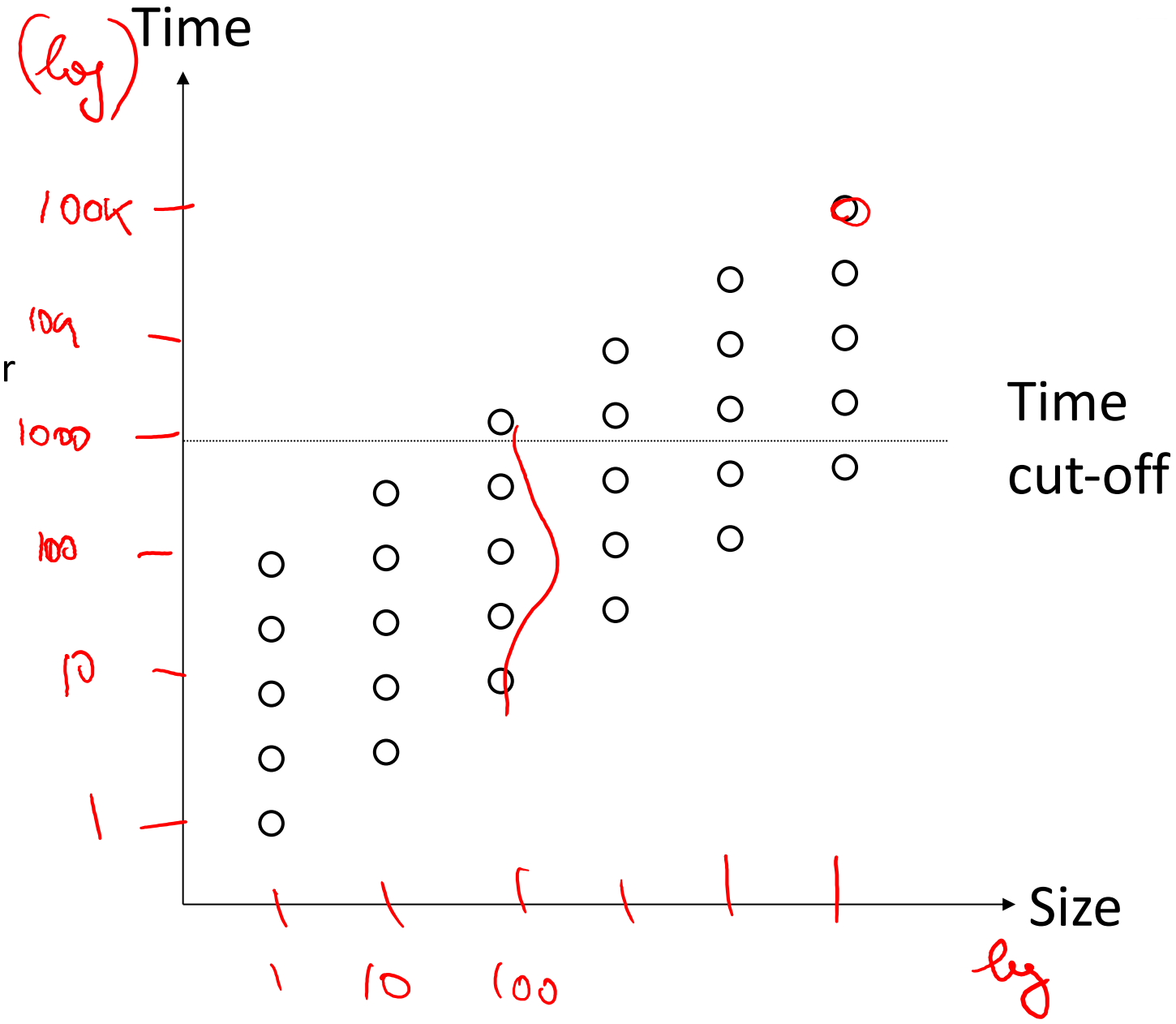
⋈

Book

Course	Book
cs3200	Complete book
cs3200	Cow book

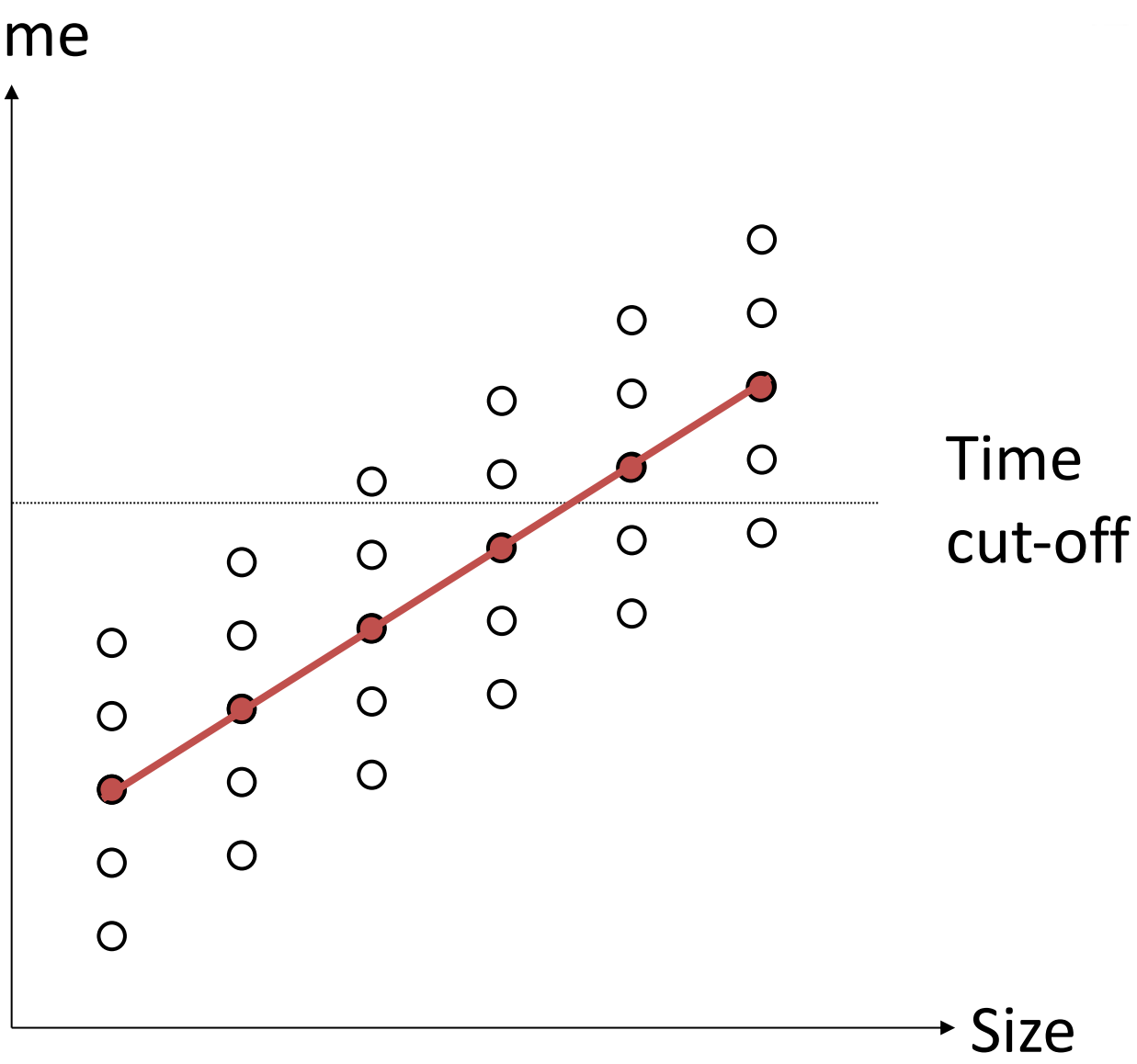
How to deal with cut-offs when binning

- These are the true points that you would get if you could run the experiments long enough.
  - Assume loglog scale
- However, we can't and thus in practice cut-off the experiments after some time.
- There is an overall trend, yet some variation for each experiment. We would still like to capture the trend with some smart aggregations



How to deal with cut-offs when binning

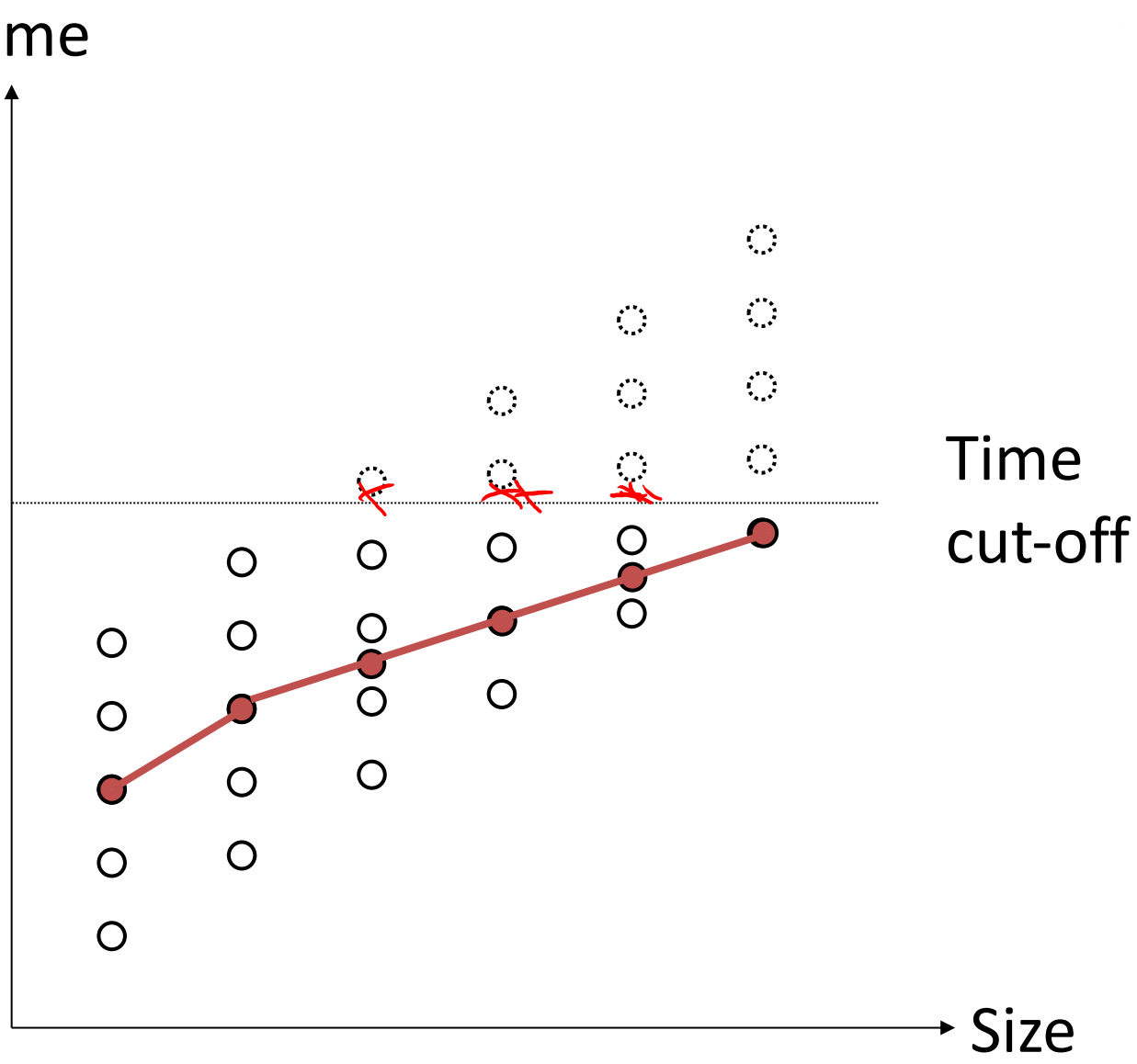
- Here is what the aggregate would look like like if we could get all points and then aggregated for each size



How to deal with cut-offs when binning

- Here is what happens if we throw away all those points that take longer than the cut-off, and only average over the "seen points"

- What would you do?



Logistics

Coursework/Evaluation

50%: Course project: The main component of this course will be a research project in the latter part of this class. This project can be a new application of one of the techniques presented or theoretically-oriented. The topic will be flexible, allowing students to explore scalable data management and analysis aspects related to their PhD research. This will involve an initial project proposal, an intermediate report, a project presentation and a final report. The final report should resemble a conference paper, and will be evaluated on the basis of soundness, significance, novelty, and clarity. Deliverables and dates are posted on the project page.

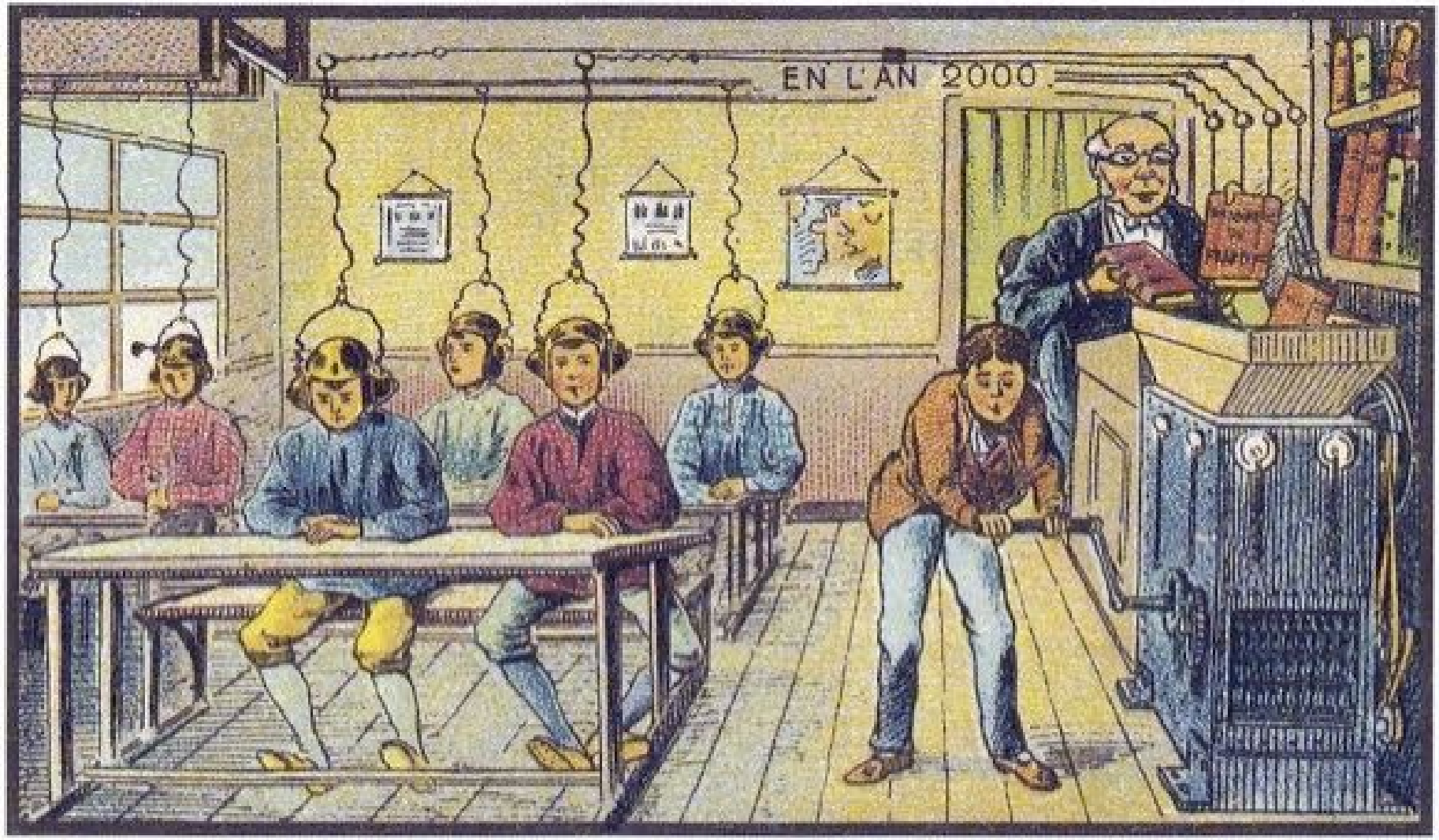
20%: Class scribes: Students take turns in "illustrating" (to be consistent with standard notation we refer to it as "scribing") 3-5 lectures. Graduate theory classes often ask students to scribe the lecture content. Yet since this course provides well-curated lectures slides, we change the rule of the game in this graduate course. Rather than scribing (repeating and summarizing) the content of the class, the students are asked to "illustrate" the covered topics with imaginative and ideally tricky illustrating examples. Those in turn can help other students practice and solidify their understanding of the class material. Scribes are due 1 week after class at midnight (Tue for Tue classes, Fri for Fri classes) and can be done individually or in teams of two. If you work in teams of two, you are expected to illustrate classes more often. We have a signup sheet in Piazza. You can either use PowerPoint or Latex. For PPTX, please use our PPTX template and share your file with me via the Share option in PowerPoint after you are logged in with your Northeastern email. For Latex, please use our Overleaf latex template and share your write-up with me on Overleaf.

15%: Three assignments: We have three assignments in the first part of this class. Like research projects, assignment solutions must be typeset in LaTeX.

15%: Class participation: Classes will be interactive and require participation of the students in class (discussing contributions or shortcomings of algorithms covered, small group break-out sessions with exercises). The class provides extensive readings for those interested, yet those are optional unless otherwise stated in class.

Source: https://northeastern-datalab.github.io/cs7240/sp21/

The year 2000 imagined in 1900



At School

Source: http://5.mshcdn.com/wp-content/gallery/the-year-2000-as-imagined-in-1900/future.jpg

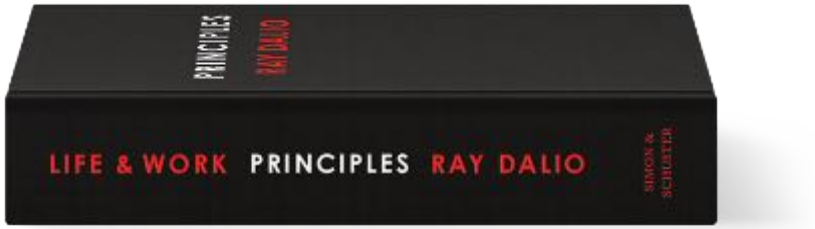
Lectures are not recorded

If gaps in knowledge are the seeds of curiosity, exploration is the sunlight. Hundreds of studies with thousands of students have shown that when science, technology and math courses include active learning, students are less likely to fail and more likely to excel. A key feature of active learning is interaction. But too many online classes have students listening to one-way monologues instead of having two-way dialogues. Too many students are sitting in front of a screen when they could be exploring out in the world.

Source: https://www.nytimes.com/2020/09/07/opinion/remote-school.html

A suggestion on how to best use class time!

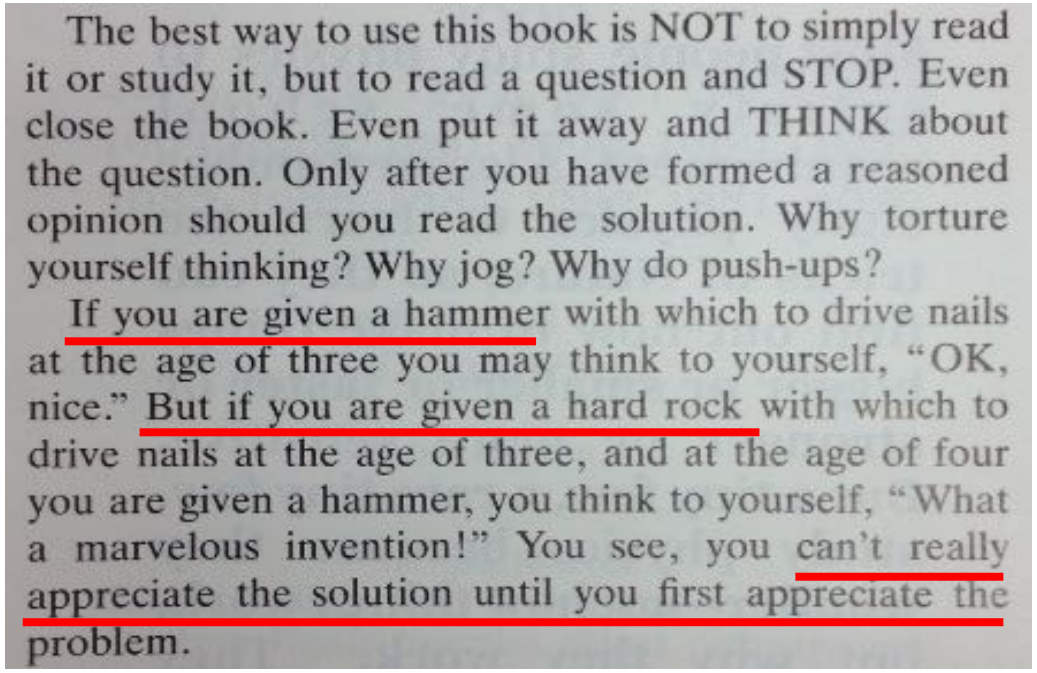
- It is ok to make mistakes in class. Making mistakes in class is the best thing that can happen to you. You learn and will never make it again.
  - "Create a Culture in Which It Is Okay to Make Mistakes and Unacceptable Not to Learn from Them" ... "Recognize that mistakes are a natural part of the evolutionary process." ... "Don't feel bad about your mistakes or those of others. Love them!"
    Ray Dalio. Principles. 2017



Source: https://www.principles.com/

Why I don't post slides *before* lecture

From the Preamble of one of the best physics books: „How to read this book“

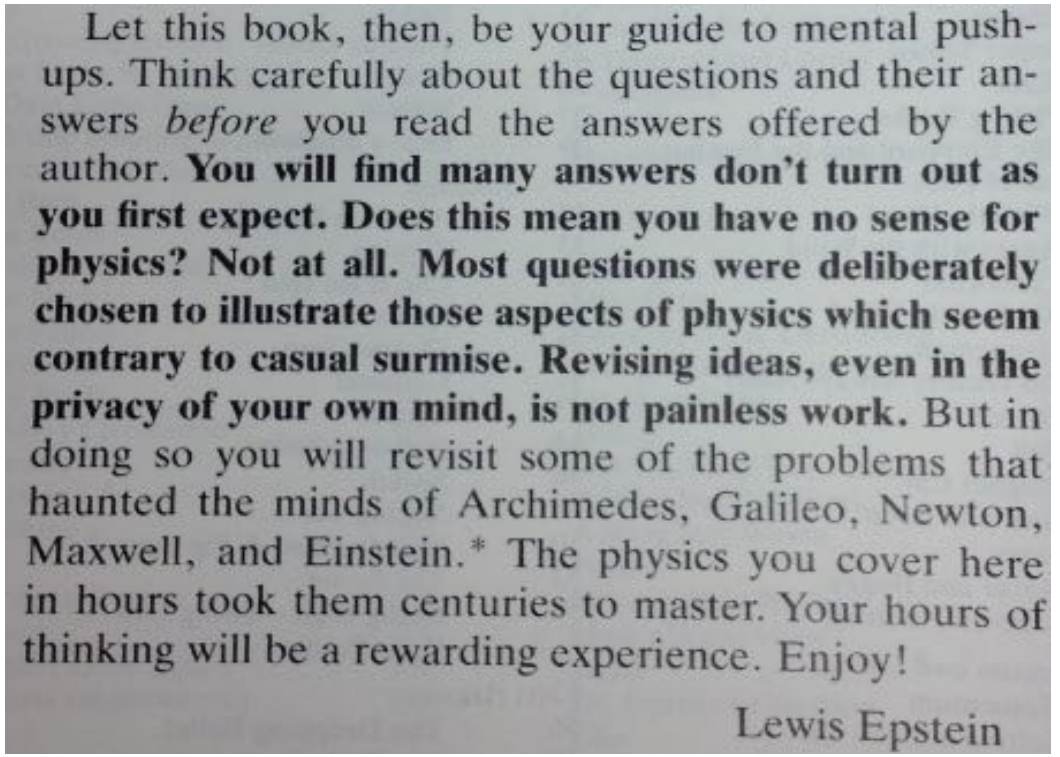


The best way to use this book is NOT to simply read it or study it, but to read a question and STOP. Even close the book. Even put it away and THINK about the question. Only after you have formed a reasoned opinion should you read the solution. Why torture yourself thinking? Why jog? Why do push-ups?

If you are given a hammer with which to drive nails at the age of three you may think to yourself, "OK, nice." But if you are given a hard rock with which to drive nails at the age of three, and at the age of four you are given a hammer, you think to yourself, "What a marvelous invention!" You see, you can't really appreciate the solution until you first appreciate the problem.

...

...



Let this book, then, be your guide to mental push-ups. Think carefully about the questions and their answers *before* you read the answers offered by the author. **You will find many answers don't turn out as you first expect. Does this mean you have no sense for physics? Not at all. Most questions were deliberately chosen to illustrate those aspects of physics which seem contrary to casual surmise. Revising ideas, even in the privacy of your own mind, is not painless work.** But in doing so you will revisit some of the problems that haunted the minds of Archimedes, Galileo, Newton, Maxwell, and Einstein.* The physics you cover here in hours took them centuries to master. Your hours of thinking will be a rewarding experience. Enjoy!

Lewis Epstein

Source: "Thinking Physics: Understanding Practical Reality", Lewis Carroll Epstein, 1979-2009. http://www.amazon.com/Thinking-Physics-Understandable-Practical-Reality/dp/0935218084

The "Surfer Analogy" for time management

Source: http://stwww.surfermag.com/files/2013/10/Yak_Charlie-970x646.jpg

Other Thoughts

- Active participation is important in this class. If you spot any errors or inconsistencies across slides/web page, typos (even if minor) do let me know! I appreciate
- Please keep your webcams on
- You can check last year's slides (topics 1-2 are online), but they will be updated this year
- Project topics: do look also through the topics later in class
  - Individual project (except in rare circumstances)
  - But you should work together on everything else in the class!

Study groups are great for learning material!

- "... The groups of students who were doing best spontaneously formed study groups...
- Students who were not doing as well tended to do as the instructor suggested-study two hours out of class for every hour in class-but did it by themselves with little social support...
- ... even well-prepared students (high math SATs) are often disadvantaged by high school experiences that lead them to work alone."

Source: Craig Nelson, Dysfunctional Illusions of Rigor: Part 1 - Basic Illusions, http://cgi.stanford.edu/~dept-ctl/tomprof/posting.php?ID=1058

Tools

- Website: calendar, links to optional readings
- Piazza: discussions, follow-up instructions beyond web page
- Overleaf & Latex: used for project deliverables (optional scribes)
- PowerPoint Office 365: recommended for scribes ("illustrations")
- Canvas: copies of some papers, due dates calendar, links to other tools (Zoom, Piazza, Gradescope)
- Gradescope: submissions of homeworks

Let's look through the website (project topics)