CS 7240 Spring 2020

# 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

Office: 450 WVH, Office hours: Tue 2pm

Course web page: https://northeastern-datalab.github.io/cs7240/sp20

# "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

How to extend the relational data model to new forms of data?

- 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

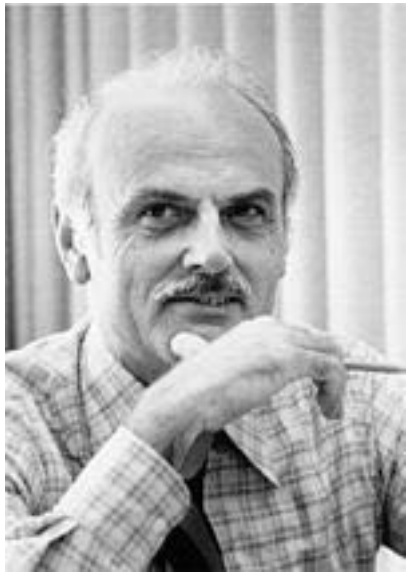
- 1998: Google
- 1995: Java, Ruby
- 1993: World Wide Web
- 1991: Python

- 1985: Windows

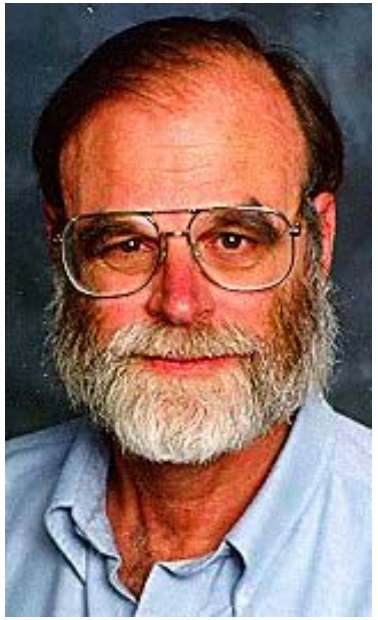
- 1974: SQL

# Four Turing Award Winners

Charles Bachmann 1973



Edgar Codd 1981

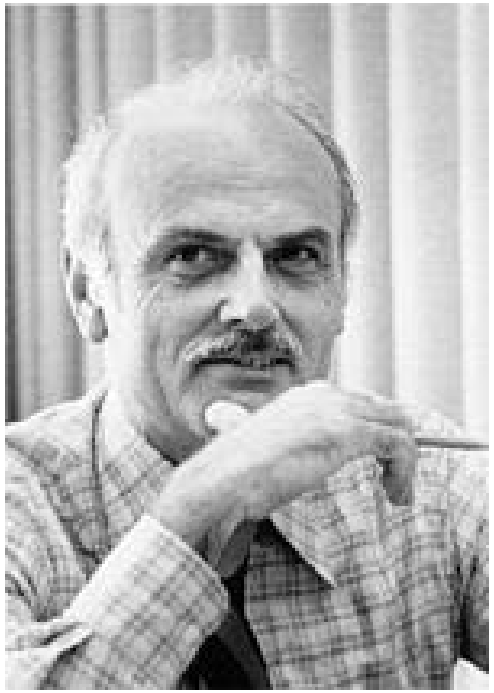


Jim Gray 2004

Michael Stonebraker 2014

Seminal contributions made in Industry

# The Underpinnings of the Relational Model (1971)



Database: a handful of relations (tables) with fixed schema.

Takes (Student, Class)

Query with small # of operations:

- Selection (filter)
- Projection
- Join
- Union

Basically, an operational finite model theory. We will see the connection to logics!

# Some example highlights

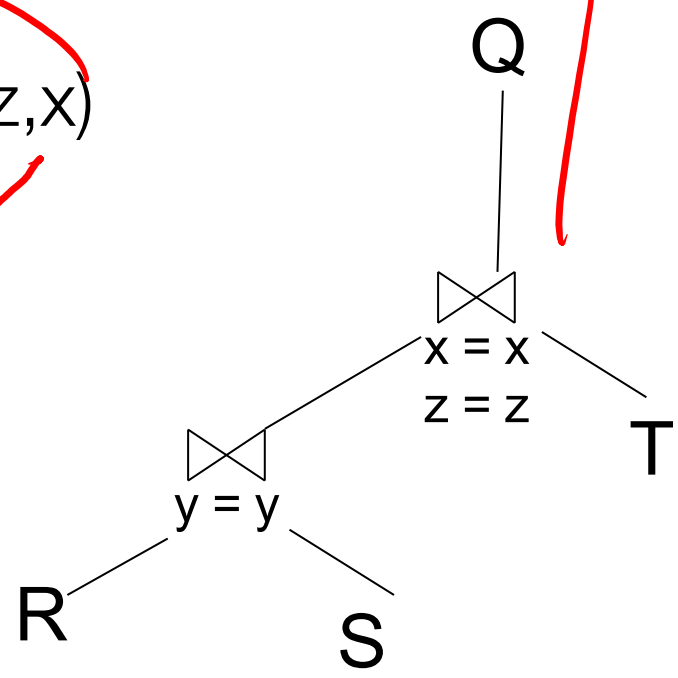
# Joins in databases

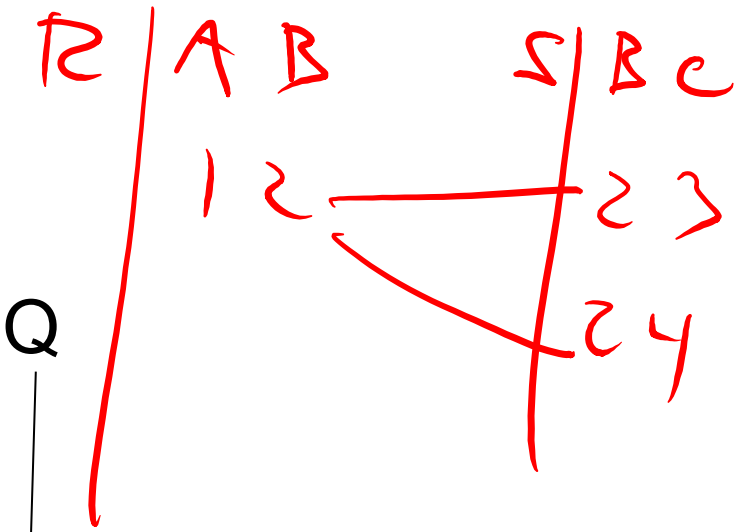
Efficient multi-way join processing

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

Three plans

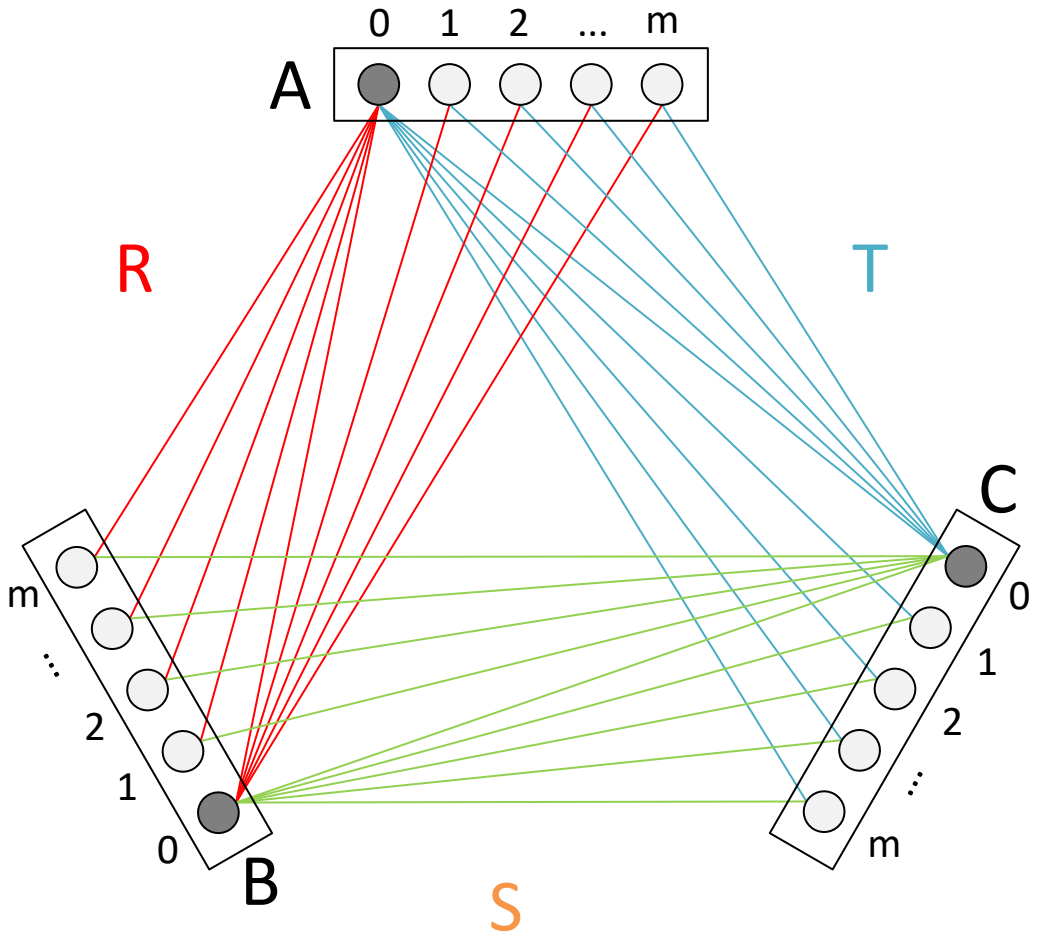
- $(R \bowtie S) \bowtie T$
- $(S \bowtie T) \bowtie R$
- $(T \bowtie R) \bowtie S$

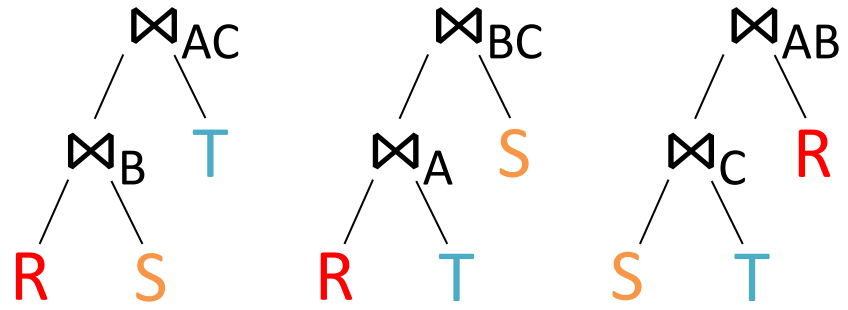


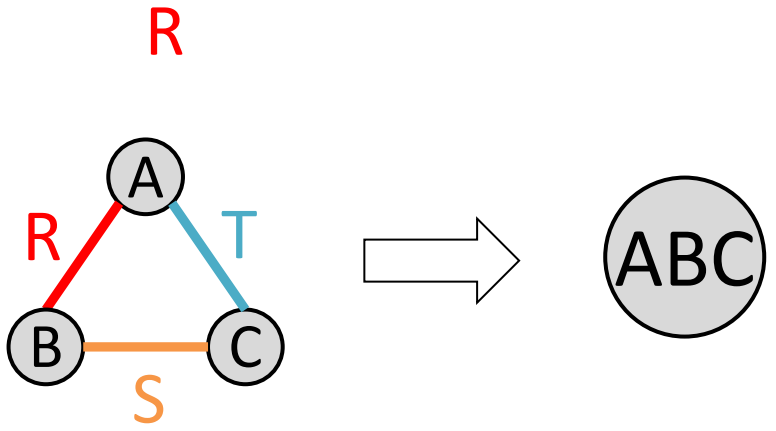

Can we do better? ☺

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



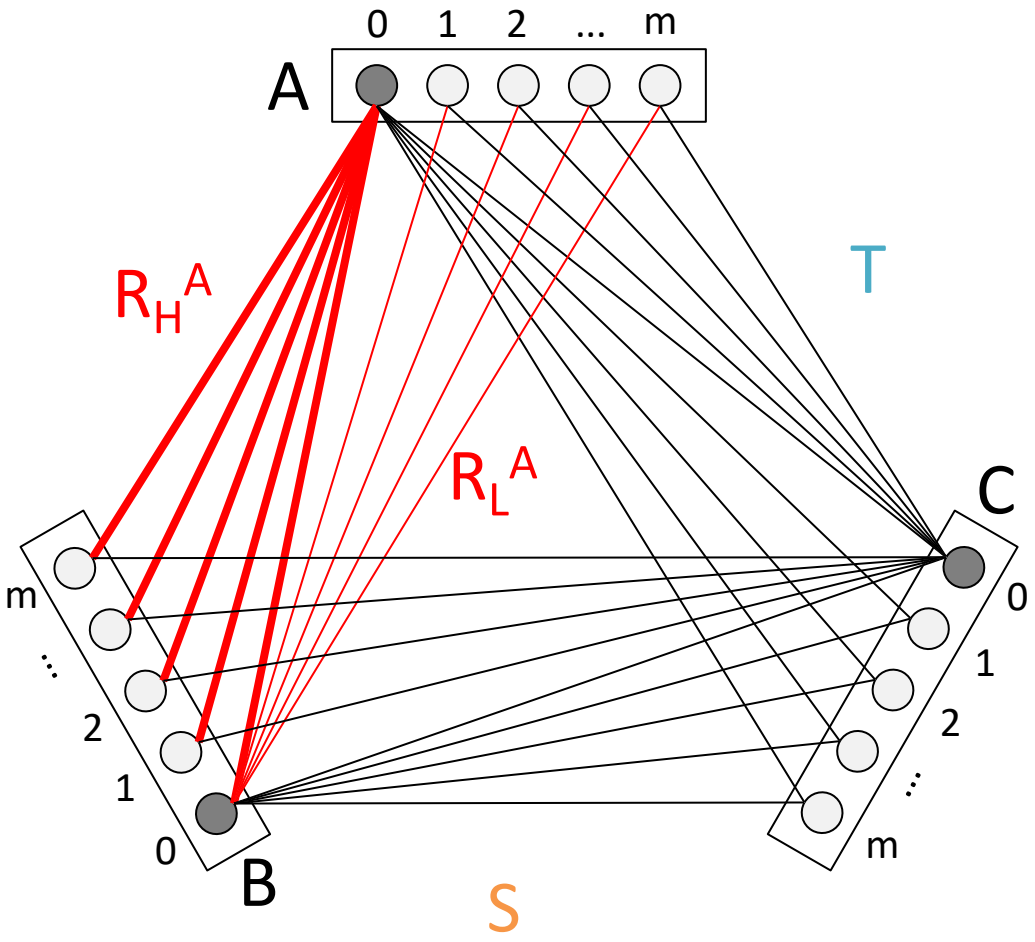

Query plans (correspond to variable elimination orders)

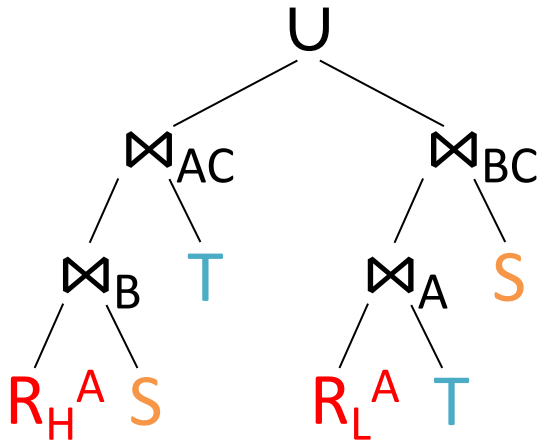



Query (hyper)graph Tree decomposition

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

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




Using multiple query plans

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

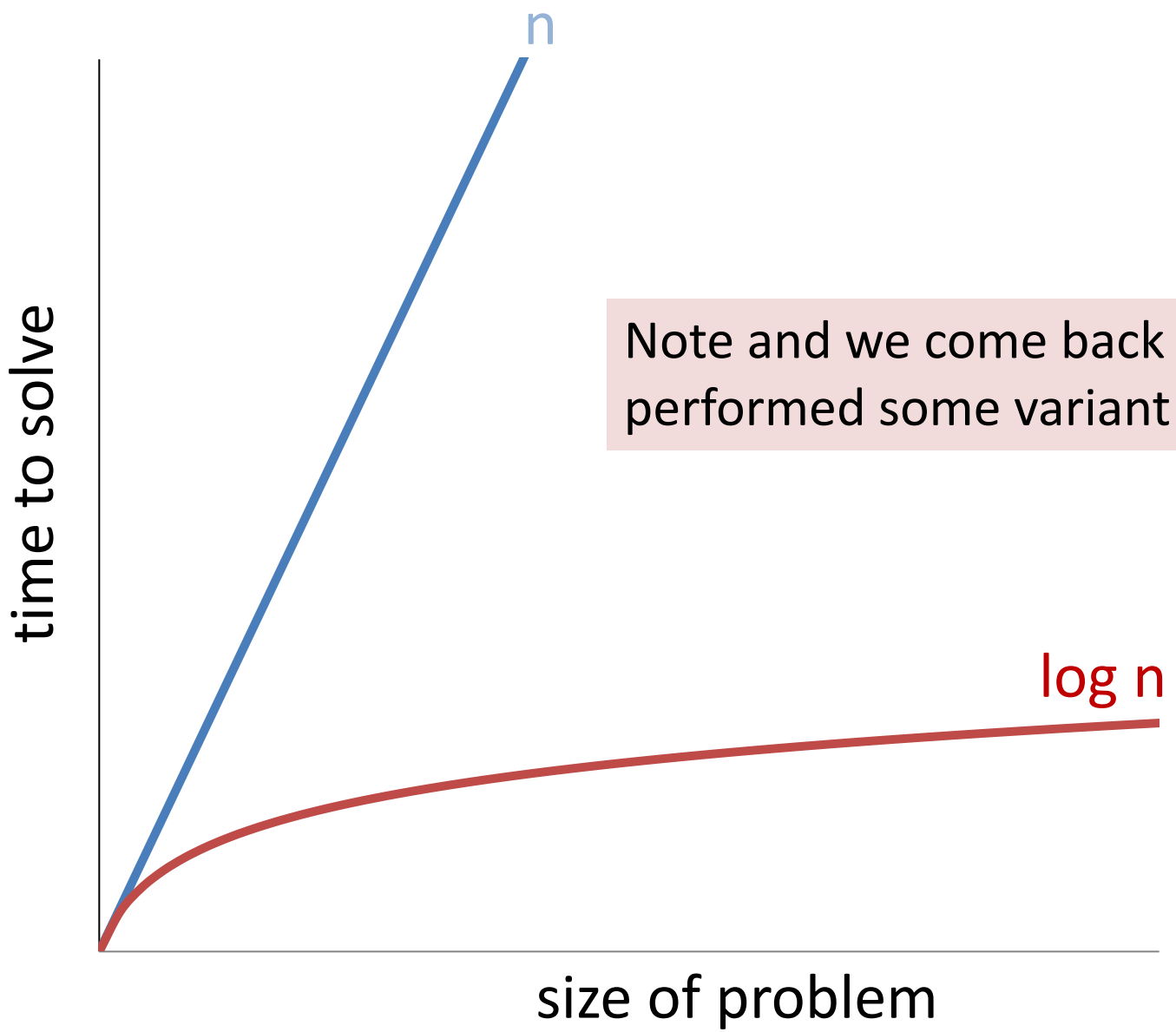
# Generalizing Dynamic Programming

- 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

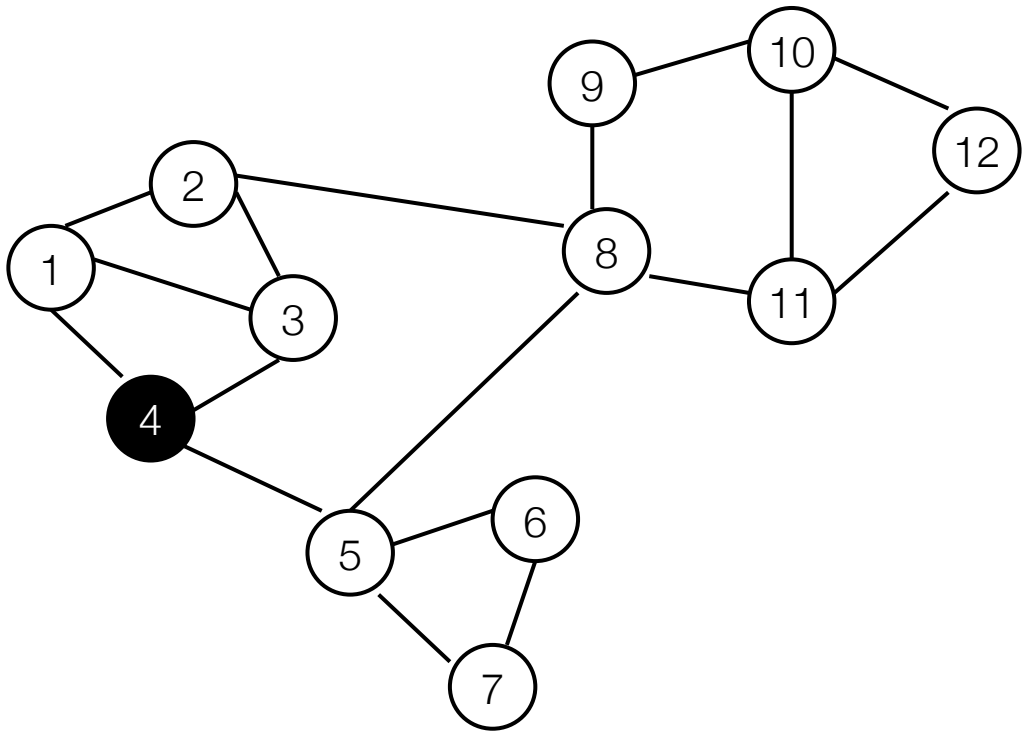
- Stand up and think of the number 1
- Pair off with someone standing, add your numbers together, and take the sum as your new number
- One of you should sit down; the other should go back to step 2

# 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

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

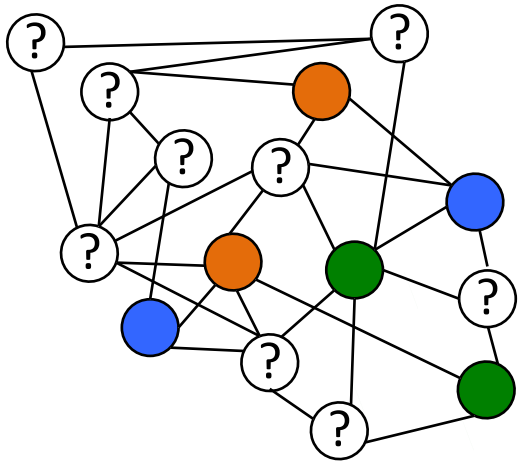
"How close" is each node to node 4?

-> "Personalized PageRank"

Linear algebra ☺

# A problem where algebra give surprising speed-ups

**Graph & seed labels**



**Compatibilities**

$$\mathbf{H} = \begin{bmatrix} 0.2 & 0.6 & 0.2 \\ 0.6 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix} \quad \Sigma=1$$

**Prob 1: Propagating arbitrary compatibilities**

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

Find:
- remaining labels

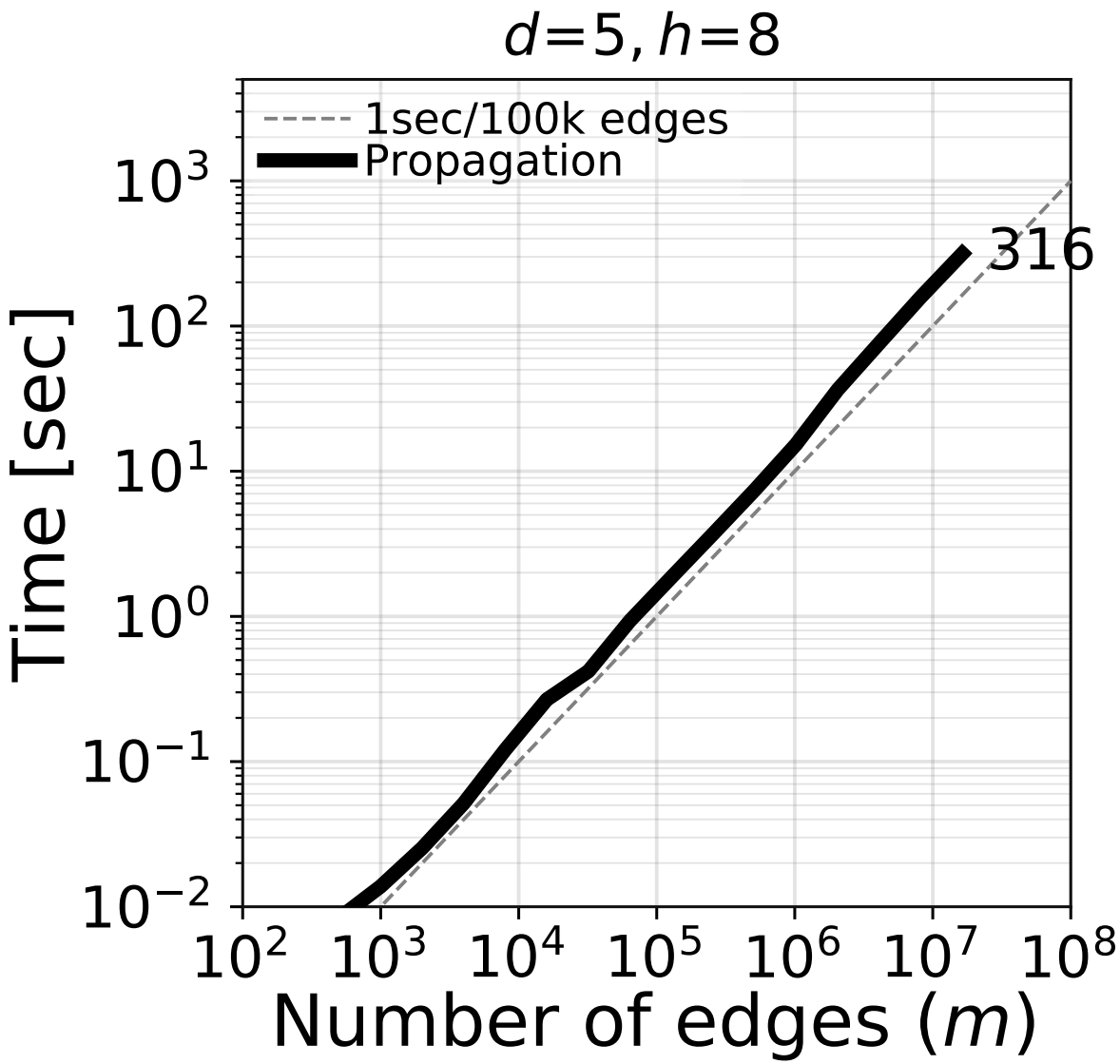
**Prob 2: Learning & prop. compatibilities**

Given

Find

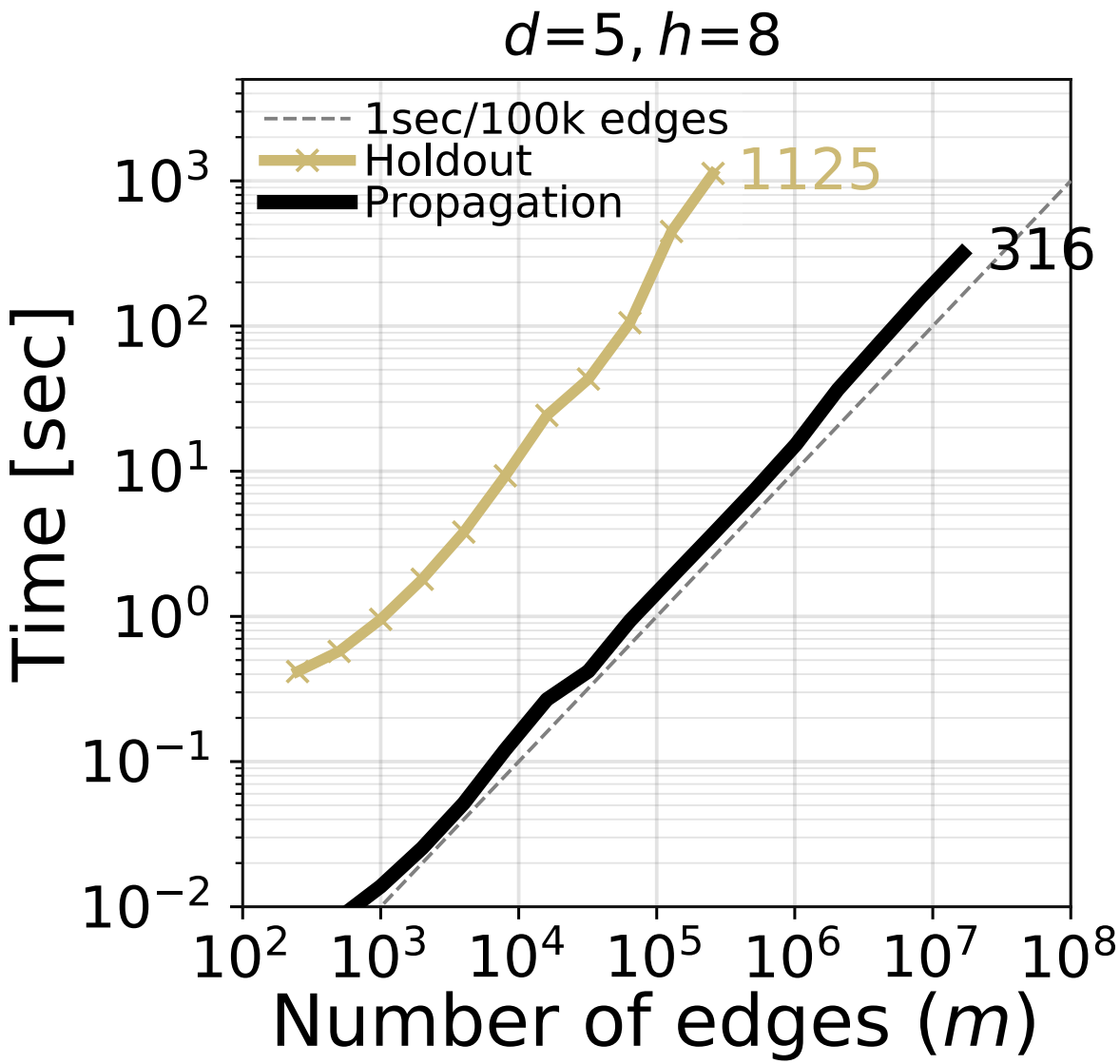
*No more heuristics!*

# What is the overhead of compatibility estimation?




Propagation scales linearly with graph size (number of edges)

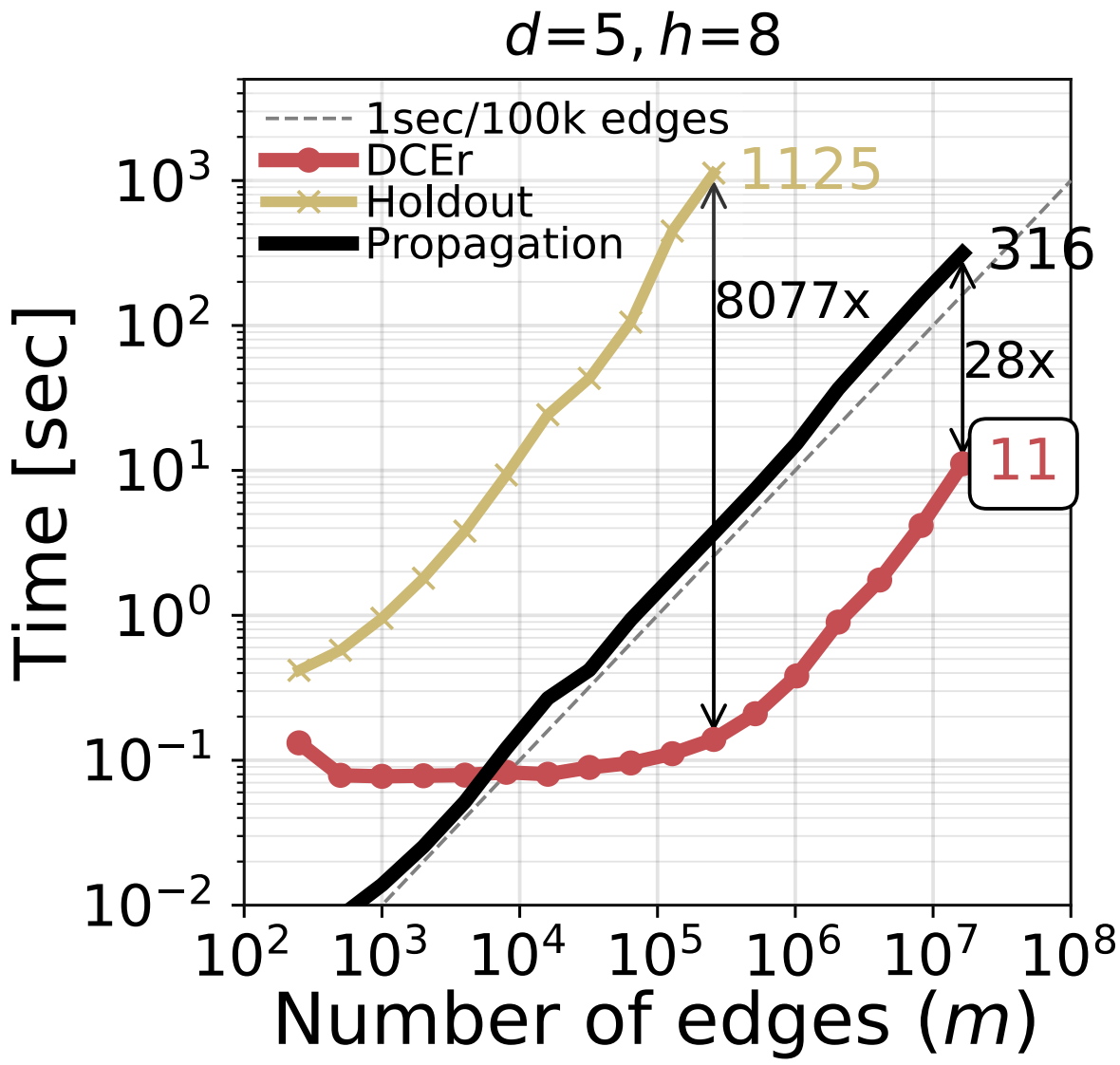
# What is the overhead of compatibility estimation?




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?

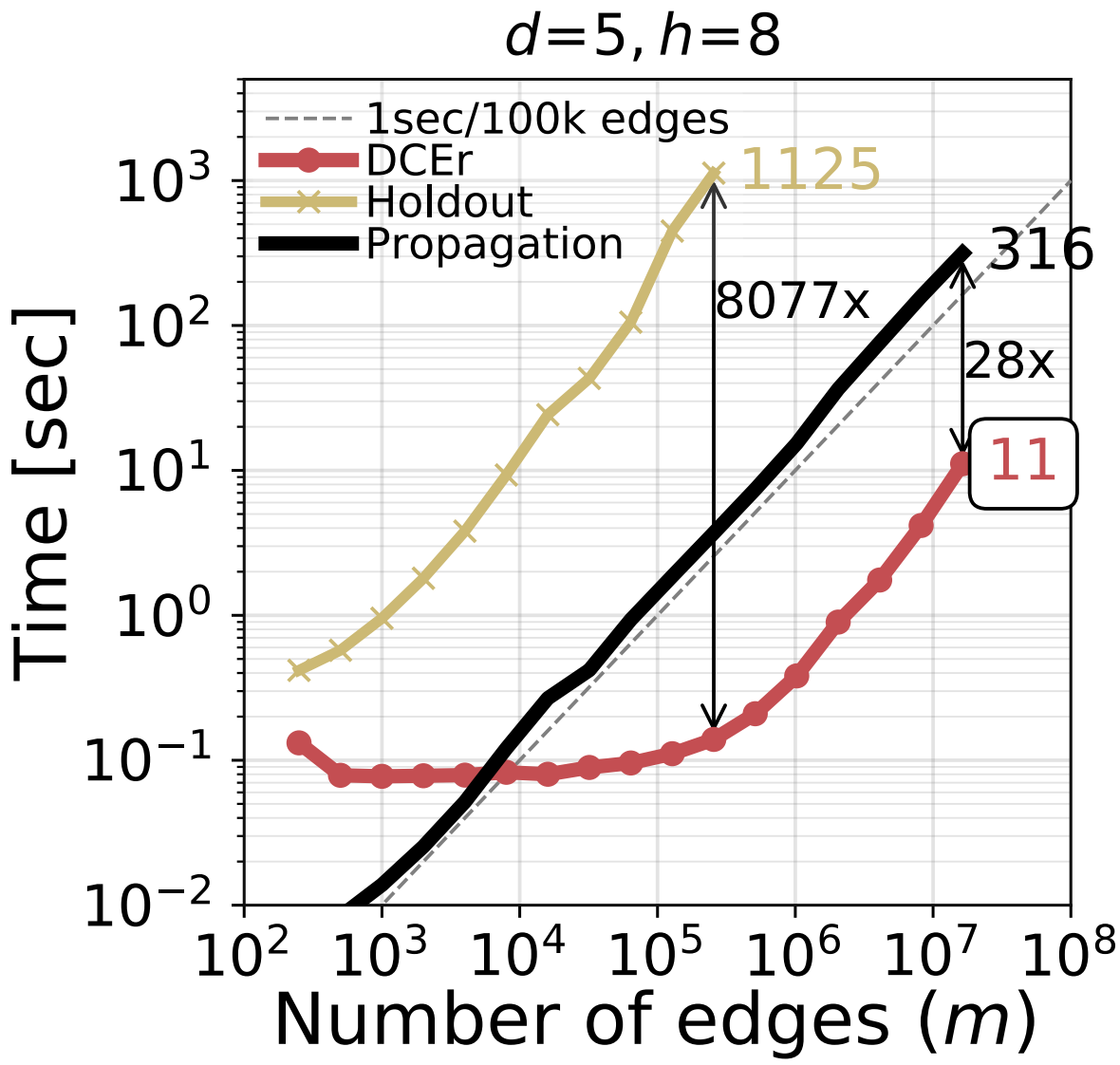



Propagation scales linearly with graph size (number of edges)

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"

$(a+b)(c+d) = ac+ad+bc+bd$

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 |

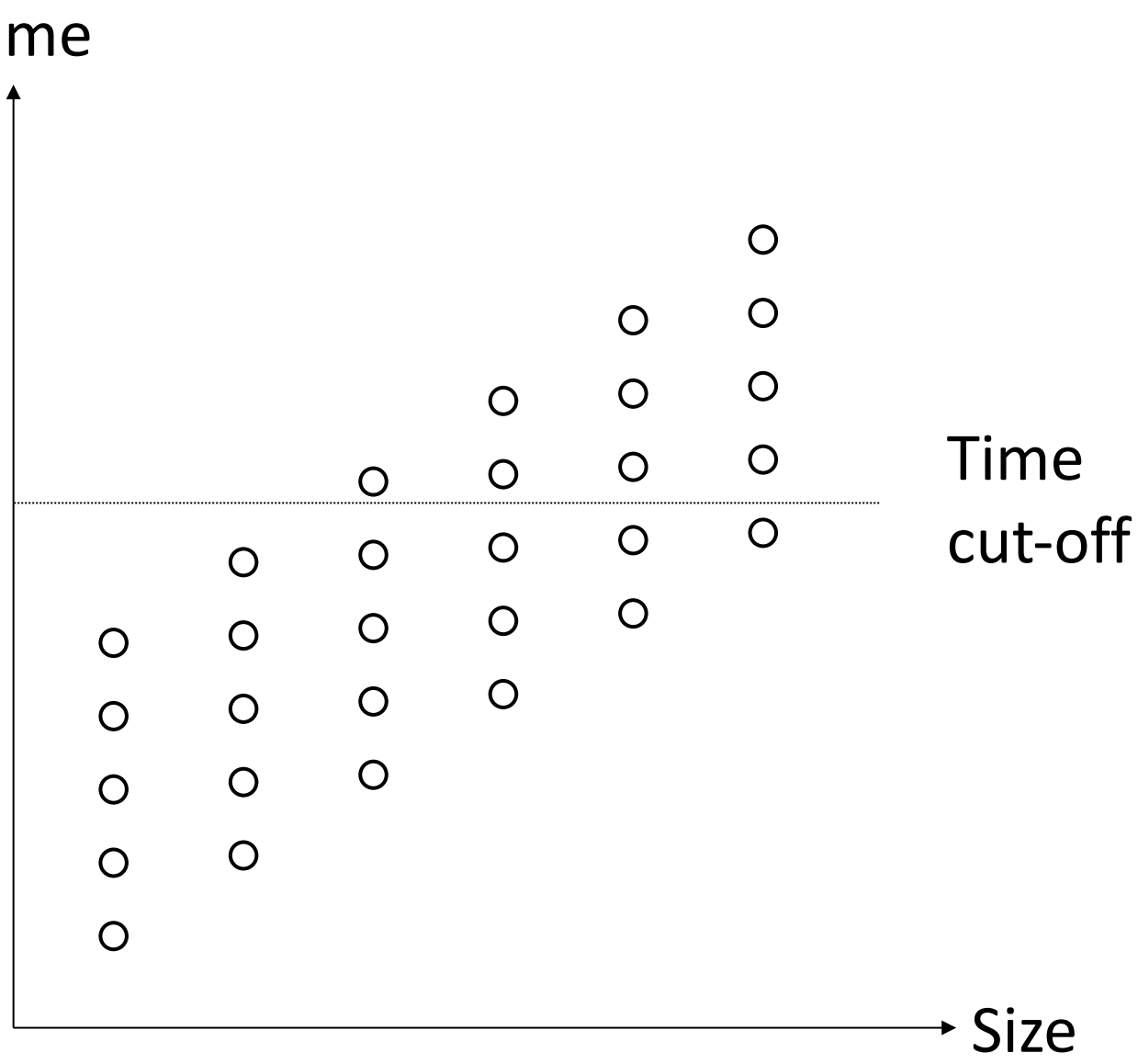
⋈

**Book**

| Course | Book |
|---|---|
| cs3200 | Complete book |
| cs3200 | Cow book |

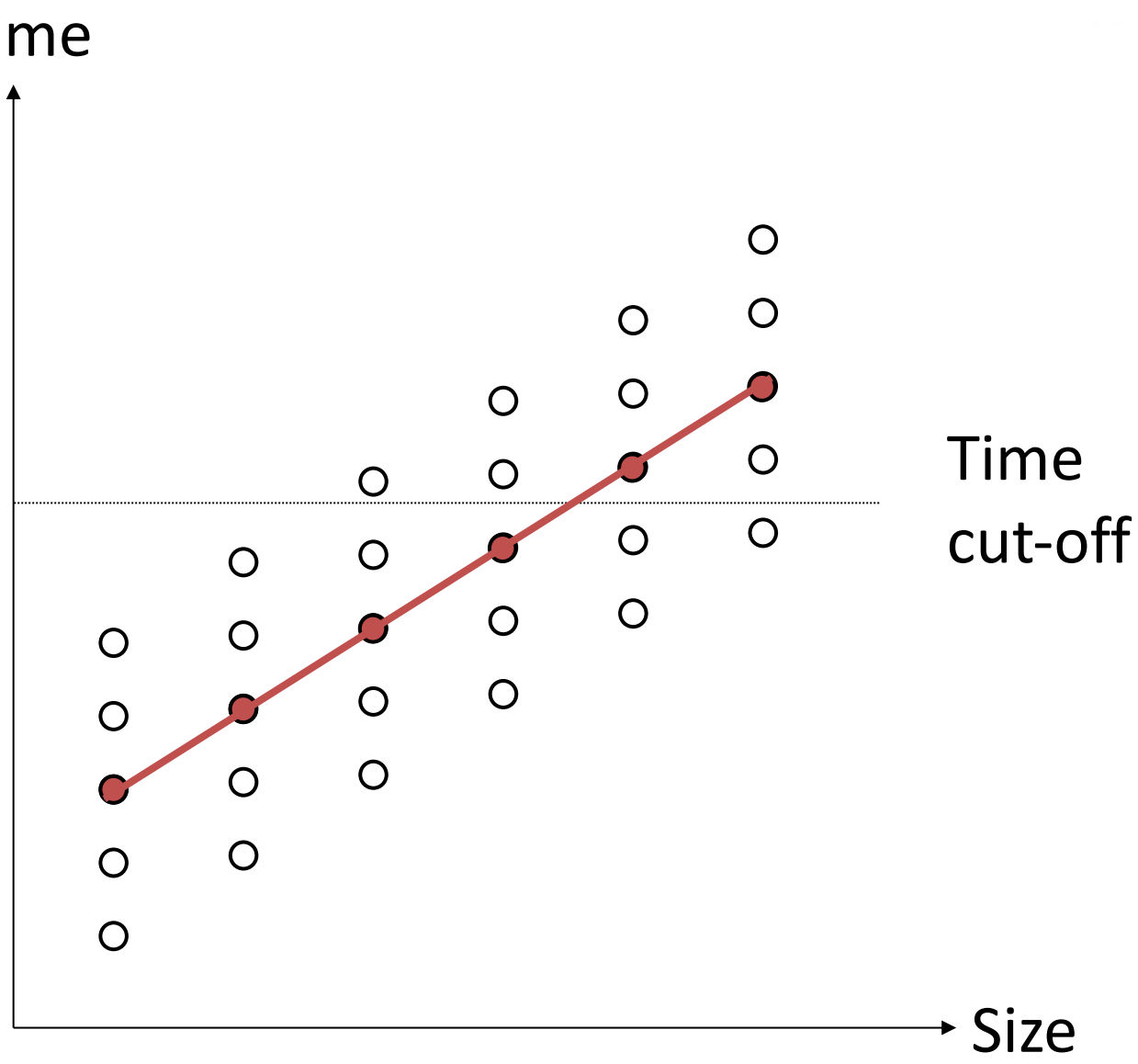
# How to deal with cut-offs when binning

- 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

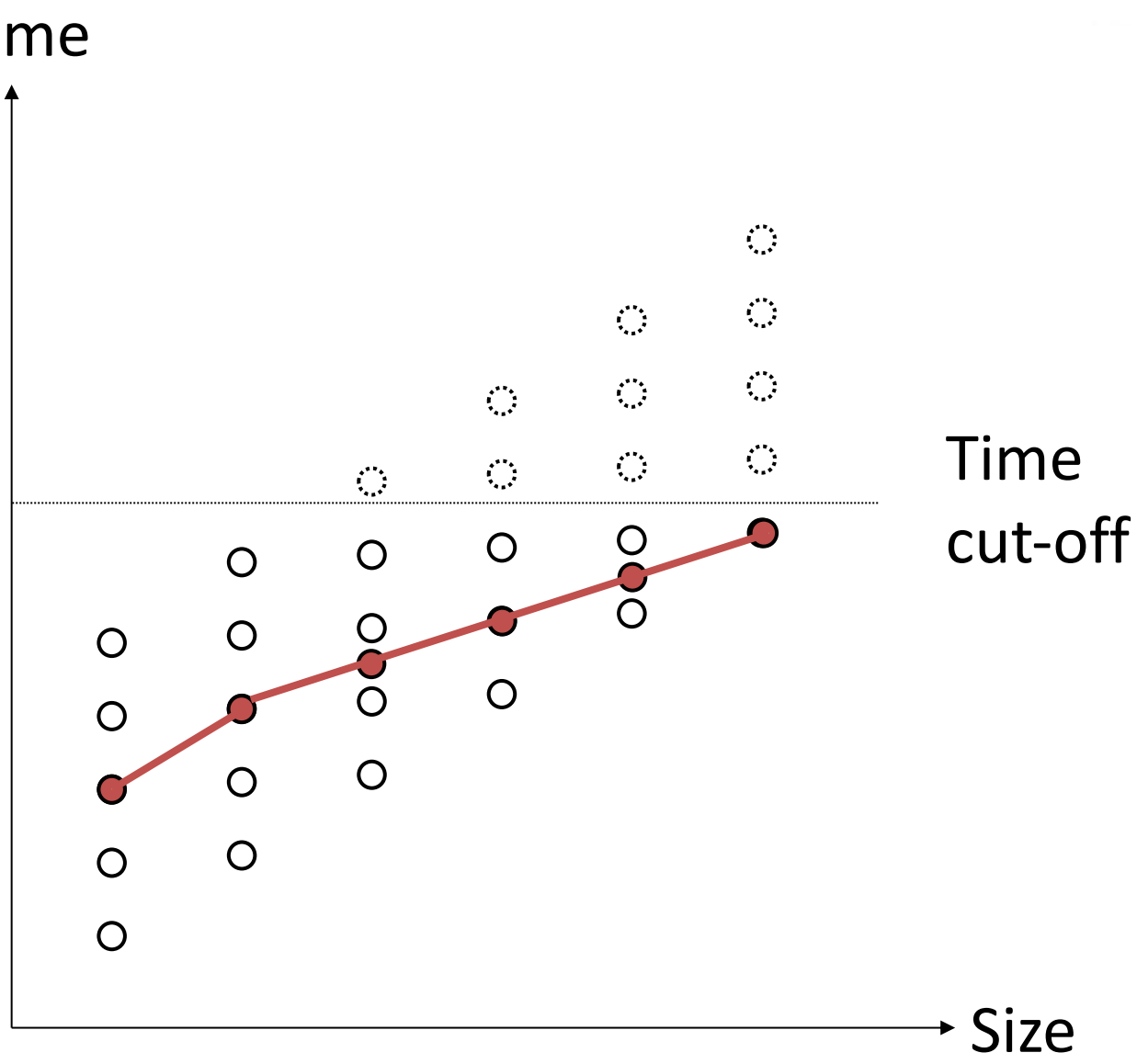
- Here is what the aggregate would look like like if we could get all points and then aggregated for each size



# 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*

**40%: 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.

**30%: 3 assignments**: We have three assignments in the first part of this class. Like research projects and scribe notes, assignment solutions must be typeset in LaTeX.

**15%: Midterm exam (Fri 3/13)**: The midterm exam tests fundamental concepts we saw in the first part of the class and will be "in class" (i.e. not take home).

**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 in-class exercises). The class provides extensive readings for those interested yet those are optional unless otherwise stated in class. Students also take turns in scribing 3-5 lectures. Scribes are due two days after class at midnight (Thu for Tue classes, Sun for Fri classes) and can be done individually or in teams of two. We have a signup sheet in Piazza. Please use our Overleaf latex template and share your write-up with me on Overleaf.

Source: https://northeastern-datalab.github.io/cs7240/sp20/

# Tools

- Website: calendar, links to optional readings
- Piazza: discussions, follow-up instructions beyond web page
- Overleaf: used for project deliverables
- Blackboard: copies of some papers, assignment submission

Let's look through the website