Large-Scale Similarity Joins With Guarantees

Rasmus Pagh
IT University of Copenhagen

SISAP
October 13, 2015

European Research Council
Established by the European Commission

SCALABLE SIMILARITY SEARCH

Talk outline

- In theory…
- The (approximate) similarity join problem
- Techniques for candidate set generation
  - Locality-sensitive hashing
  - Cache-efficiency via recursion
  - CoveringLSH: Achieving 100% recall
- In practice…

In theory…

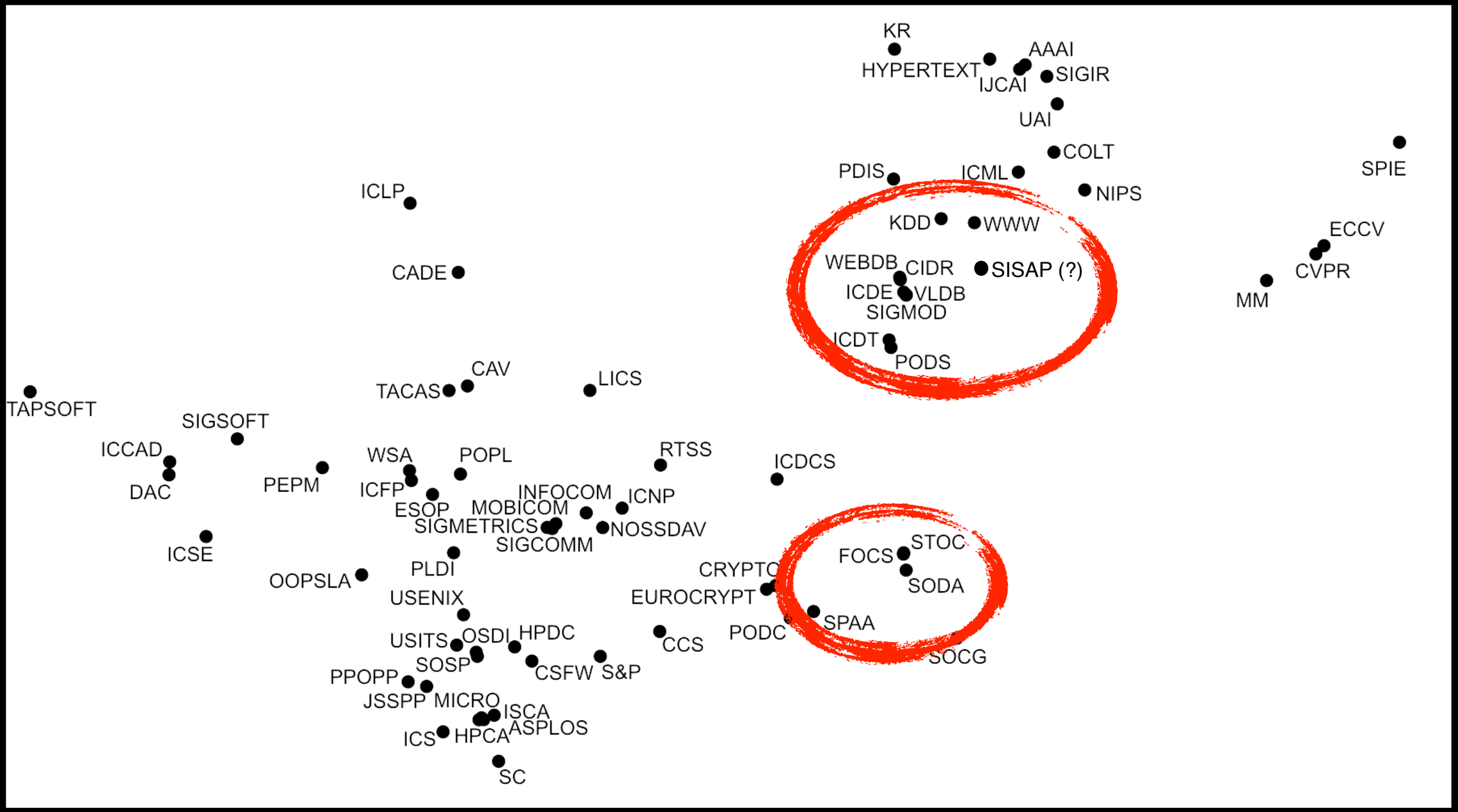
Confession

- I was trained as an algorithm theorist
  - Worst case assumptions on data
  - Big-O notation
  - Papers full of math, rarely experiments

Confession

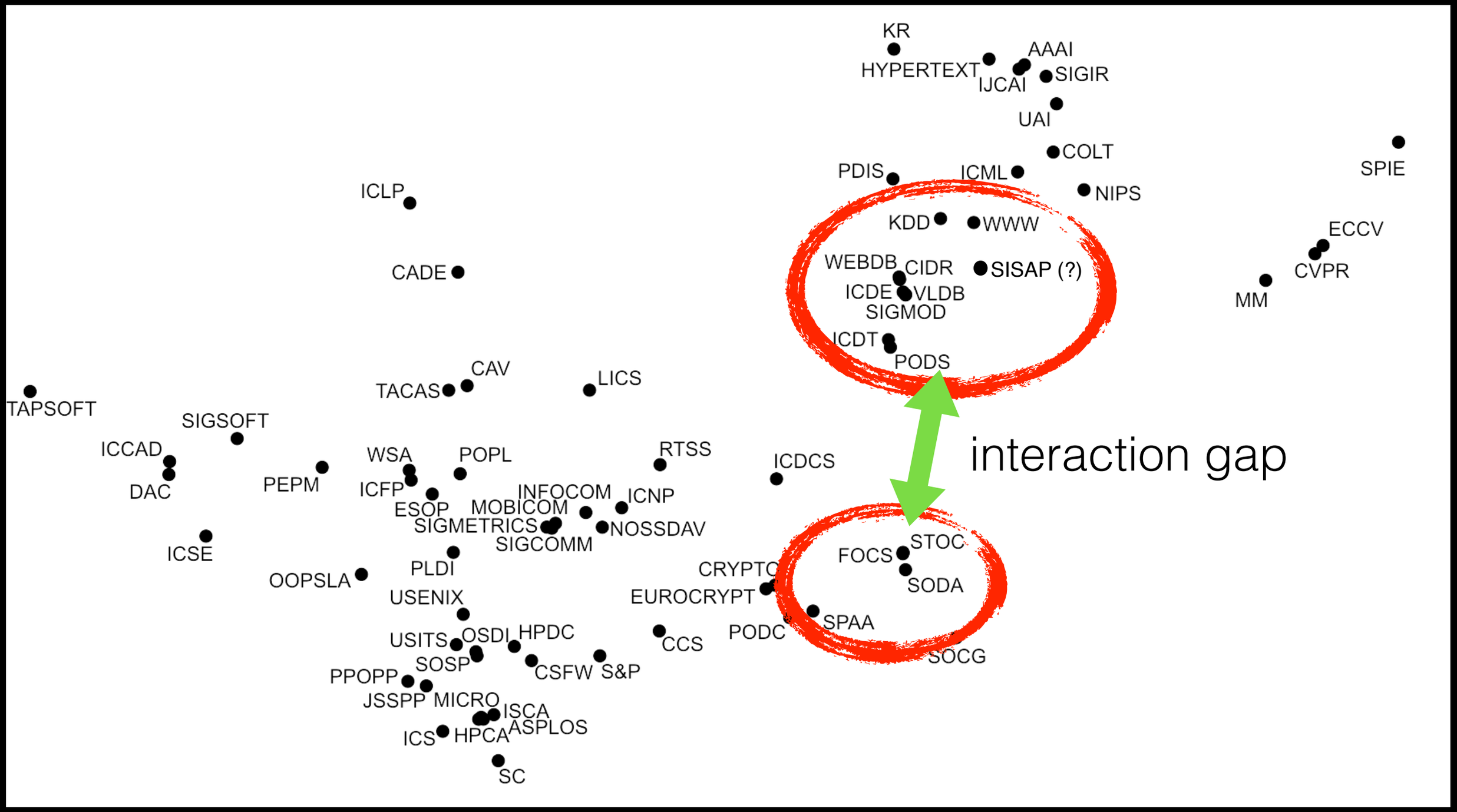
- I was trained as an algorithm theorist
  - Worst case assumptions on data
  - Big-O notation
  - Papers full of math, rarely experiments
- "In theory there is no difference between theory and practice… But in practice there is!"

Similarity map of CS conferences



Source: Kuhn & Wattenhofer. The Theoretic Center of Computer Science

Similarity map of CS conferences



Source: Kuhn & Wattenhofer. The Theoretic Center of Computer Science

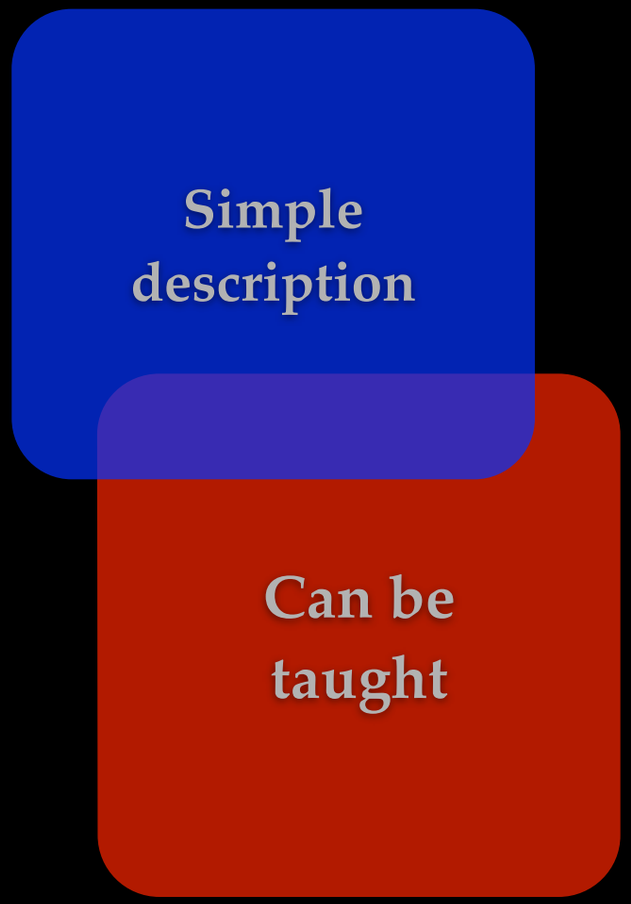
Theory with impact

- Almost always algorithms that are easy to describe, implement, and adapt

Simple description

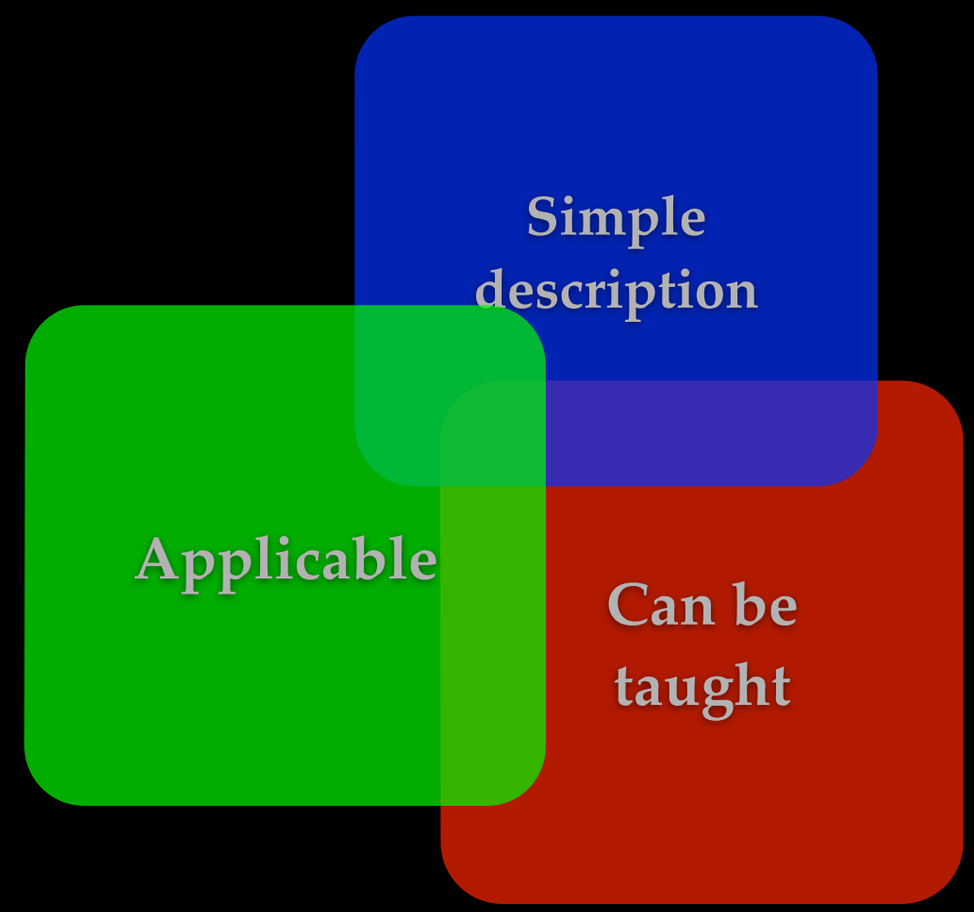
Theory with impact

- Almost always algorithms that are easy to describe, implement, and adapt
- Analysis often simple enough to be taught to undergraduates



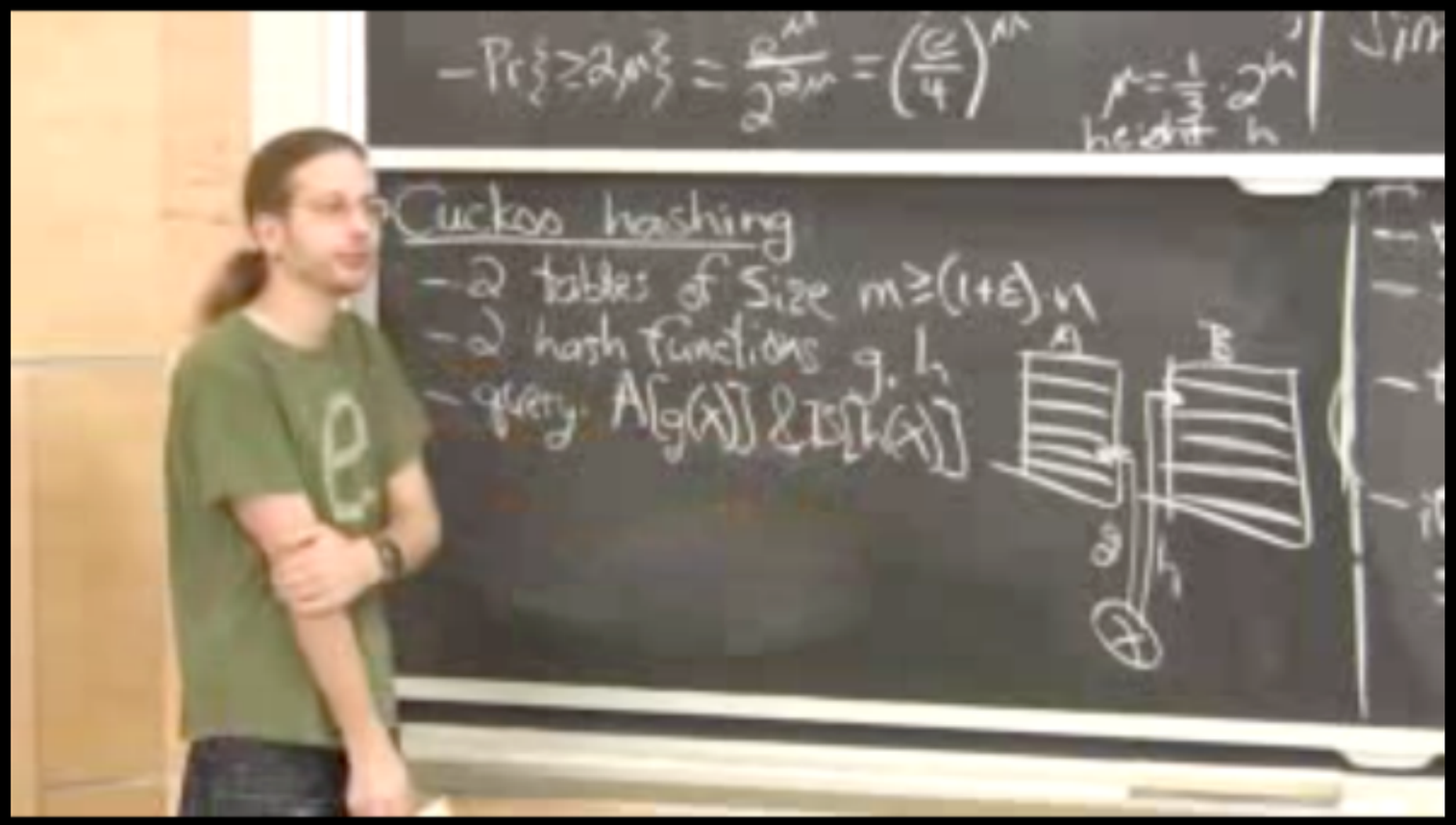
Theory with impact

- Almost always algorithms that are easy to describe, implement, and adapt
- Analysis often simple enough to be taught to undergraduates
- Solving a real problem

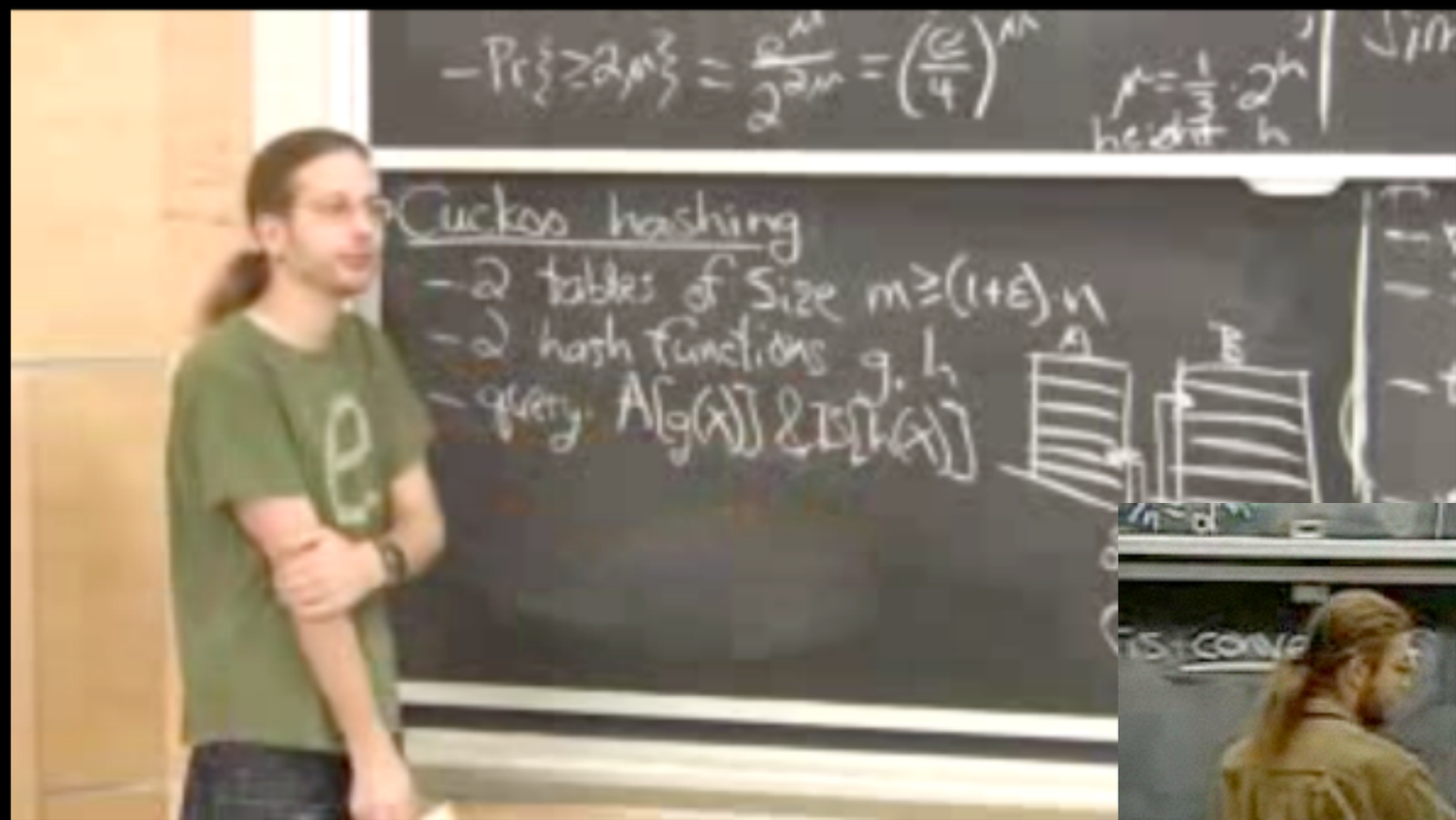


NEW YORK, NY, April 9, 2013—ACM (the Association for Computing Machinery) today announced the winners of six prestigious awards […]
Andrei Broder, Moses Charikar, and Piotr Indyk, recipients of the Paris Kanellakis Theory and Practice Award for algorithms that allow for quickly finding similar entries in large databases, known as locality-sensitive hashing (LSH). […]
The Kanellakis Award honors specific theoretical accomplishments that significantly affect the practice of computing.

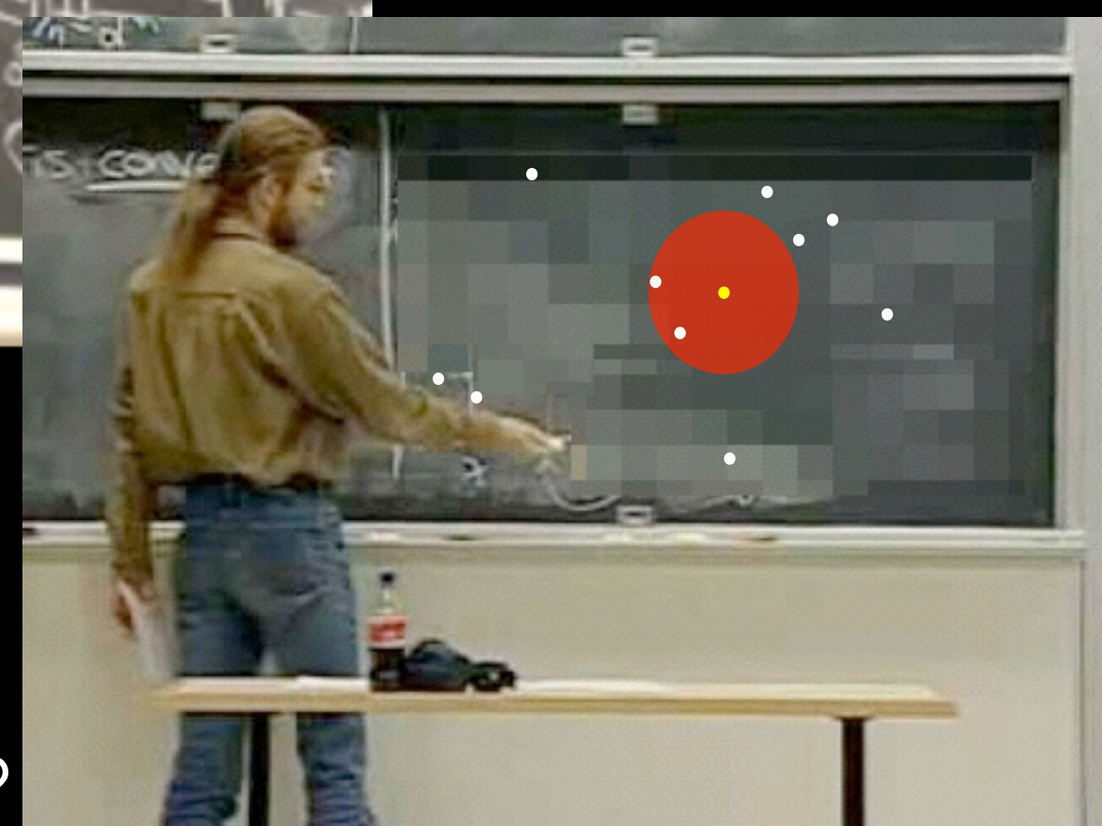
Main contributions in the years 1997-2002


Cuckoo hashing
- 2 tables of size m ≥ (1+ε)·n
- 2 hash functions g, h
- query A[g(x)] & B[h(x)]
A
B

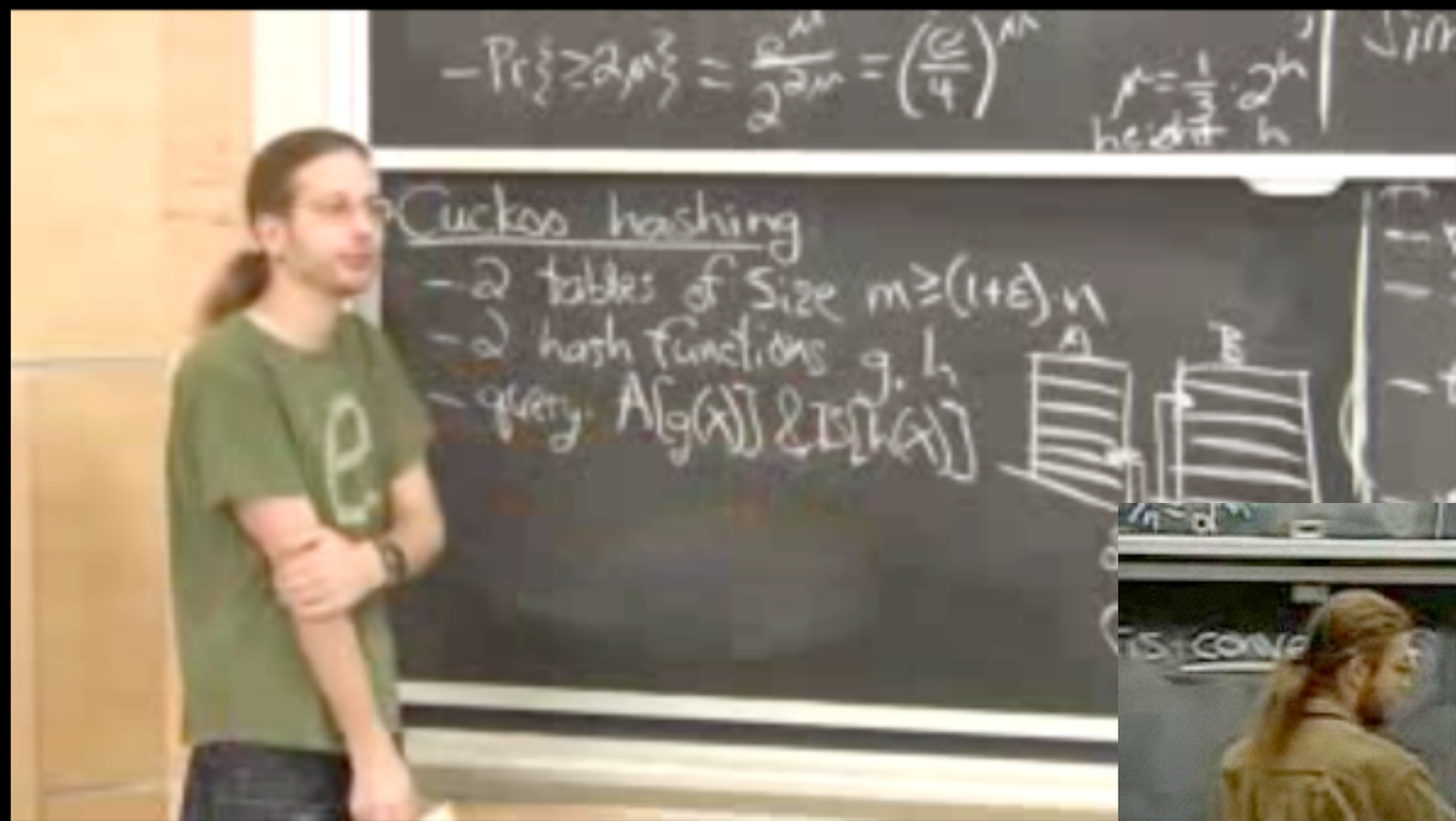
Now


Cuckoo hashing
- 2 tables of size m ≥ (1+ε)·n
- 2 hash functions g, h
- query A[g(x)] & B[h(x)]

Now



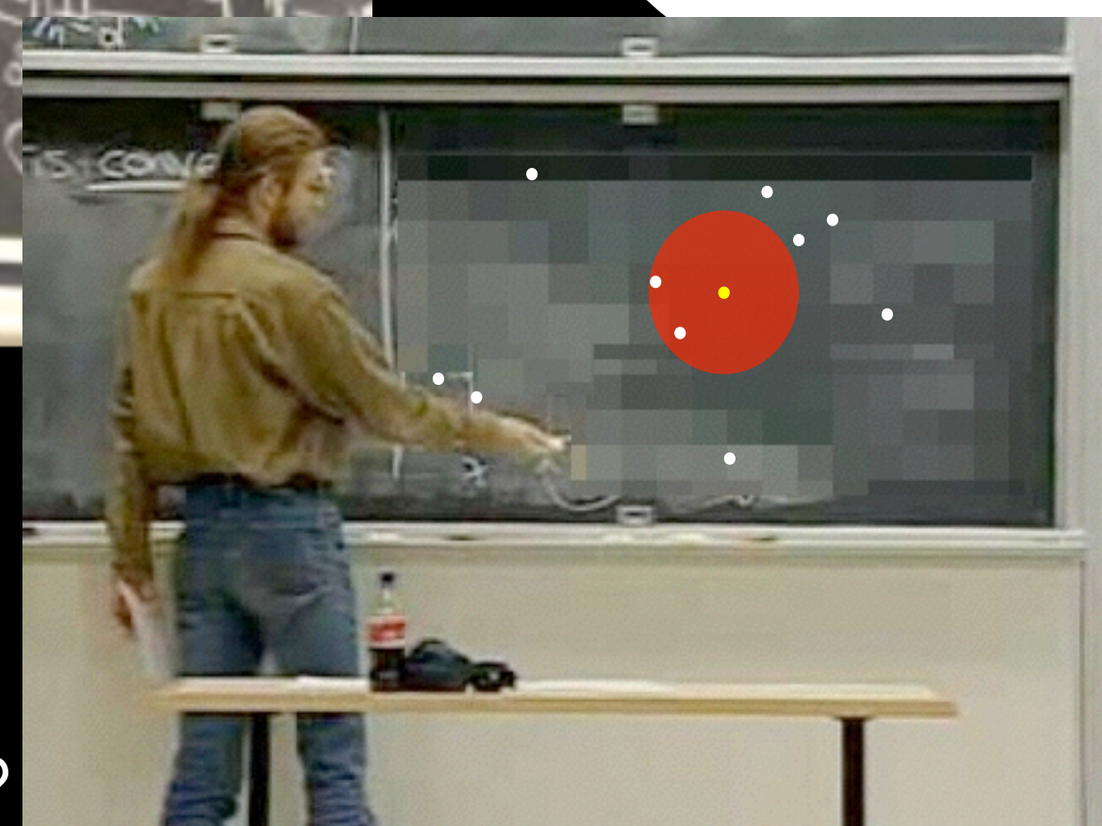
Soon?



Now

European Research Council

Established by the European Commission



Soon?

Similarity joins

Similarity join example 1 (record linkage)

Country	Name
USA	IBM
USA	Microsoft
Germany	SAP
China	Baidu

Token	ID
Mircosoft	1
SAP SE	2
I.B.M.	3
baidu.com	4

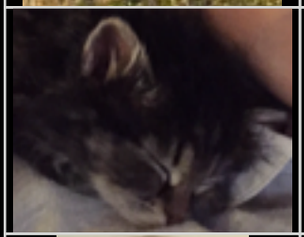
Similarity join example 1 (record linkage)

Country	Name
USA	IBM
USA	Microsoft
Germany	SAP
China	Baidu

Token	ID
Mircosoft	1
SAP SE	2
I.B.M.	3
baidu.com	4

Similarity join example 2 (classification)

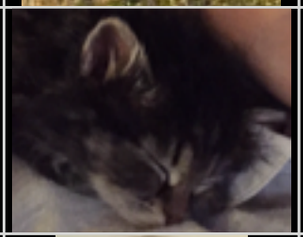
Class	Name
Mammalia	Cat
Mammalia	Dog
Reptilia	Snake
Aves	Parrot

Image	ID
	1
	2
	3
	4

Images by Bodlina and Marek Szczepanek

Similarity join example 2 (classification)

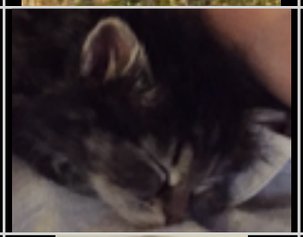
Class	Name
Mammalia	Cat
Mammalia	Dog
Reptilia	Snake
Aves	Parrot

Features	Image	ID
0101101		1
1001000		2
1101101		3
1101110		4

Images by Bodlina and Marek Szczepanek

Similarity join example 2 (classification)

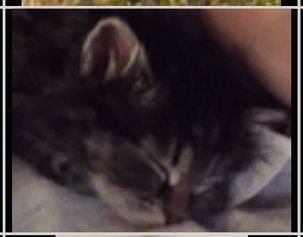
Class	Name	Features
Mammalia	Cat	1100101
Mammalia	Dog	1101101
Reptilia	Snake	1011000
Aves	Parrot	1010111

Features	Image	ID
0101101		1
1001000		2
1101101		3
1101110		4

Images by Bodlina and Marek Szczepanek

Similarity join example 2 (classification)

Class	Name	Features
Mammalia	Cat	1100101
Mammalia	Dog	1101101
Reptilia	Snake	1011000
Aves	Parrot	1010111

Features	Image	ID
0101101		1
1001000		2
1101101		3
1101110		4

Images by Bodlina and Marek Szczepanek

Similarity join example 3 (recommendation)

User	Follows
@rasmuspagh1	{@ERC_SSS, @NateSilver538, @NatureNews, @sapinker,@techreview,…}
@Reza_Zadeh	{@medialab,@techreview, @AndrewYNg,…}
…	…
@simMachines	{@neiltyson,@NatureNews, @RichardDawkins,…}

Algorithmic problem

- Given a tolerance *r* compute:

$$Q \bowtie_r S = \{(q, x) \in Q \times S \mid ||q - x|| \leq r\}$$

Algorithmic problem

- Given a tolerance *r* compute:

Distance measure

$$Q \bowtie_r S = \{(q, x) \in Q \times S \mid ||q - x|| \leq r\}$$

Algorithmic problem

- Given a tolerance r compute:

Distance measure

$$Q \bowtie_r S = \{(q, x) \in Q \times S \mid ||q - x|| \leq r\}$$

- <u>**This talk**</u>:

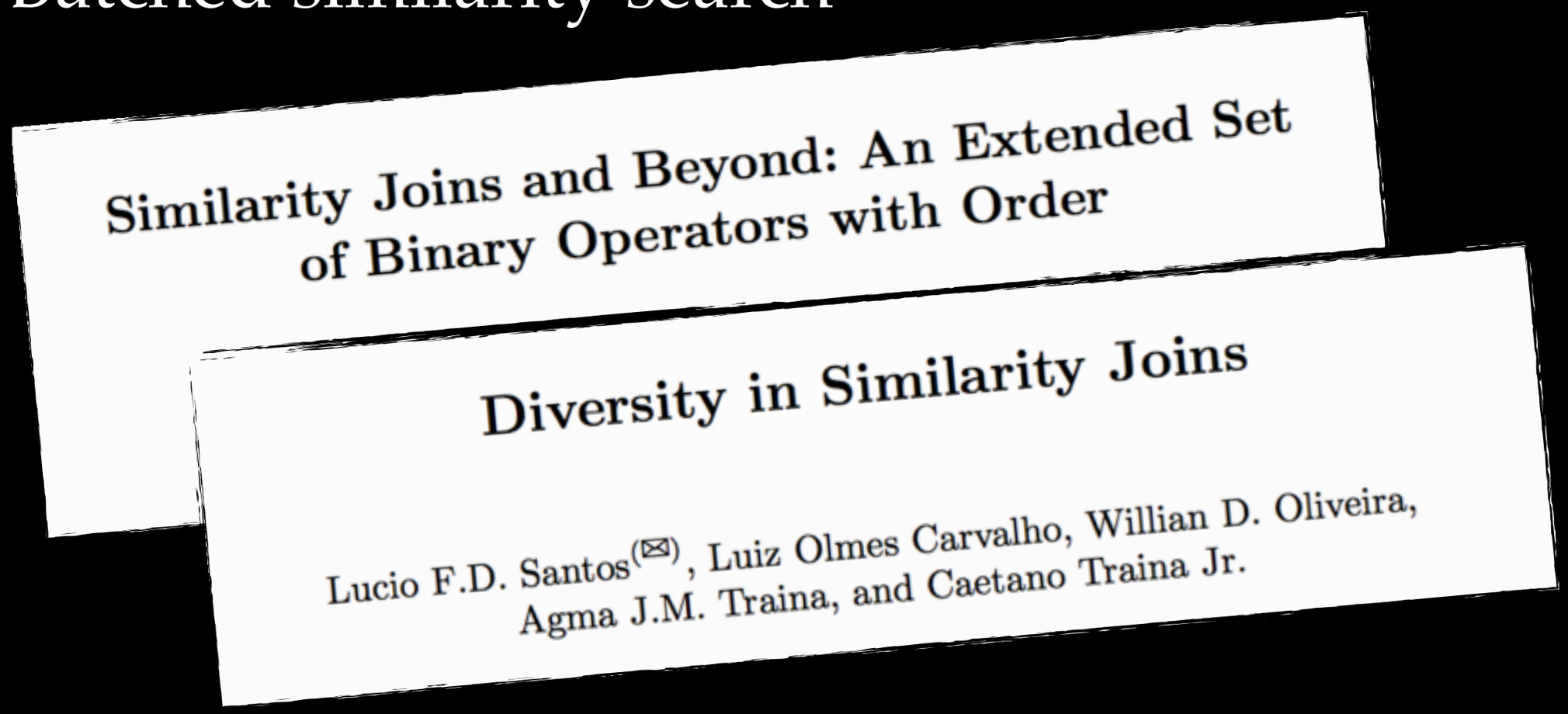
  Consider n vectors in $\{0,1\}^d$ and Hamming distance.

$$q = 1100101$$
$$x = 1101101$$

$$||q - x|| = 1$$

Variants

- kNN similarity join
- Batched similarity search


Similarity Joins and Beyond: An Extended Set of Binary Operators with Order

Diversity in Similarity Joins

Lucio F.D. Santos(✉), Luiz Olmes Carvalho, Willian D. Oliveira, Agma J.M. Traina, and Caetano Traina Jr.

A FASTER SUBQUADRATIC ALGORITHM FOR FINDING OUTLIER CORRELATIONS

MATTI KARPPA, PETTERI KASKI, AND JUKKA KOHONEN

LEMP:
Fast Retrieval of Large Entries in a Matrix Product

Christina Teflioudi
Max Planck Institute for

Rainer Gemulla
University of Mannheim

Olga Mykytiuk
Sulzer GmbH
München, Germany
olga.mykytiuk@gmail.com

07 / Track: Data Mining

Session: Simi

Scaling Up All Pairs Similarity Search

Roberto J. Bayardo

Yiming Ma*

Ramakrishnan Srikant

Many names…

ba

Finding Correlations in Subquadratic Time, with Applications to Learning Parities and the Closest Pair Problem

GREGORY VALIANT,

Probabilistic Polynomials and Hamming Nearest Neighbors

BATCH HAMMING NEAREST NEIGHBOR problem

Josh Alman*

Ryan Williams†

A FASTER SUBQUADRATIC ALGORITHM FOR FINDING OUTLIER CORRELATIONS

MATTI KARPPA, PETTERI KASKI, AND JUKKA KOHONEN

LEMP:
Fast Retrieval of Large Entries in a Matrix Product

Christina Teflioudi
Max Planck Institute for Informatics
Saarbrücken, Germany
chteflio@mpi-inf.mpg.de

Rainer Gemulla
University of Mannheim
Mannheim, Germany
rgem
ma

Olga Mykytiuk
Sulzer GmbH
München, Germany
olga.mykytiuk@gmail.com

2015

Many names…

Finding Correlations in Subquadratic Time, with Applications to Learning Parities and the Closest Pair Problem

GREGORY VALIANT,

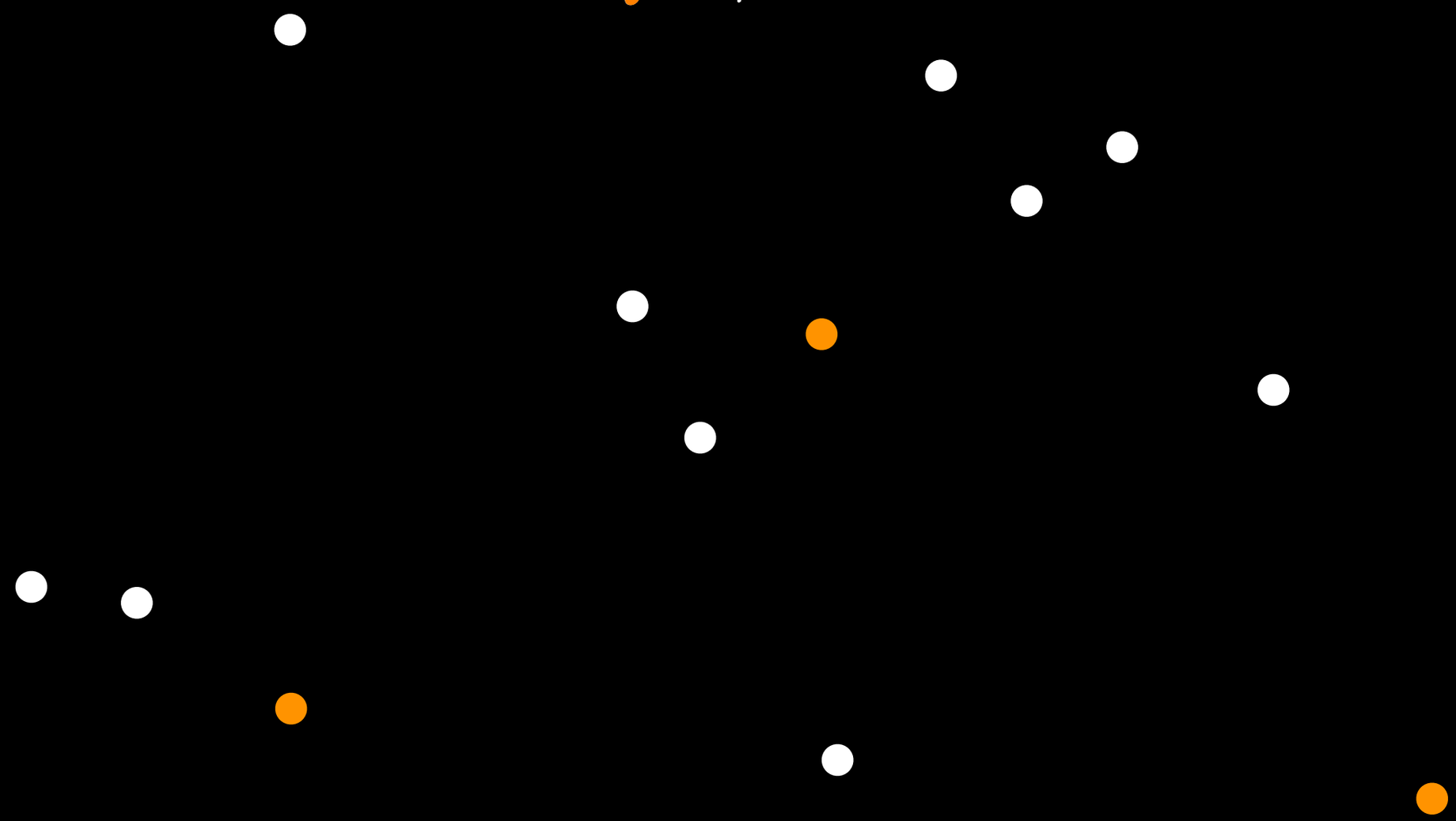
Probabilistic Polynomials and Hamming Nearest Neighbors

BATCH HAMMING NEAREST NEIGHBOR problem

Josh Alman* Ryan Williams†

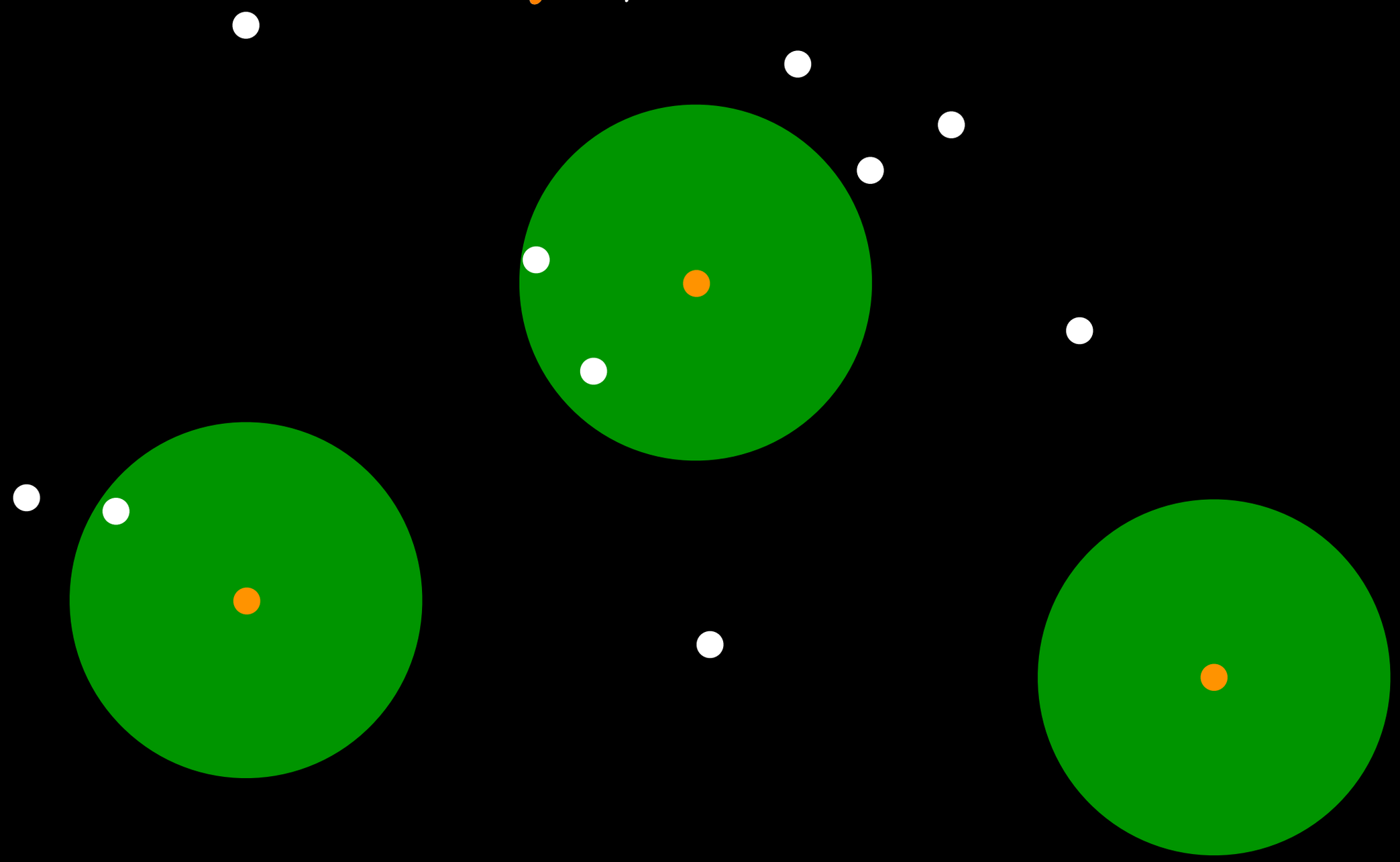
Similarity join in a picture

$$R = Q \bowtie_r S$$



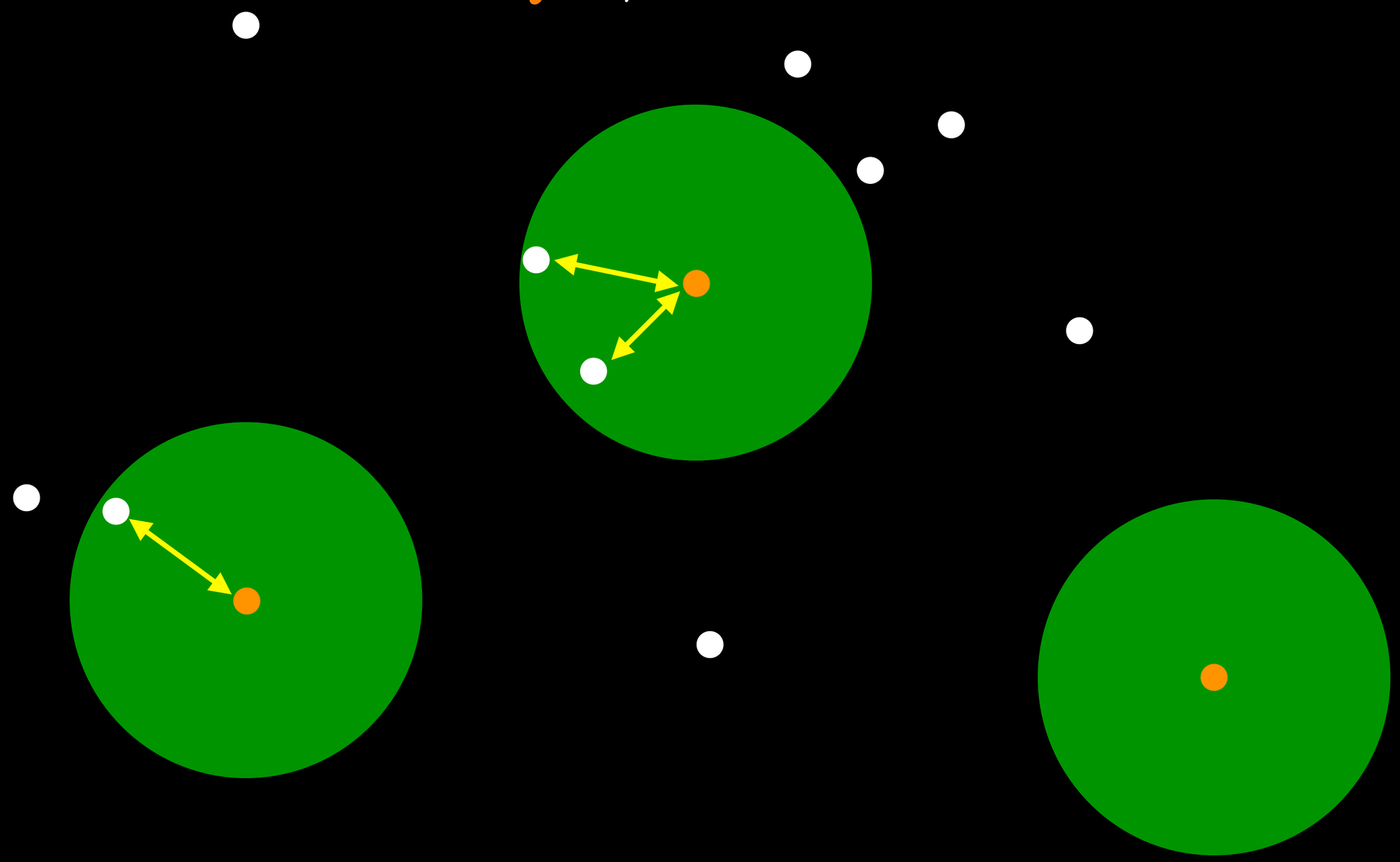
Similarity join in a picture

$$R = Q \bowtie_r S$$



Similarity join in a picture

$$R = Q \bowtie_r S$$



Similarity join in a picture

in high dimensional space (d=80)

```
1 0 0 0 1 1 1 0 0 0 0 1 0 0 1 1 0 1 1 0 1 1 0 0 1 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 0 1 0 1 0 0 0
1 0 1 1 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 0 0
0 1 0 1 1 0 0 1 0 0 1 1 0 1 1 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 0 0 1 1 1 1 0 1 1 1 0 1 0 0 1 1 0 0 1 0 1 1
0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 1 1 0 0 1 1 1 1 1 0 1 0 1 1 0 0 0 1 0 1 1 0 1 0 0 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 0 1 0 1
1 1 1 1 0 1 0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 0 0 1 1 0 1 0 1 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 1 0
0 0 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 1 1
0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 1 1 0 1 1 1 1 0 0 0 0 1 1 0 1 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 0
0 0 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0
1 1 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0
0 1 1 1 1 1 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 1 1 1
0 0 1 1 0 0 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 0 1 0 0
0 0 0 1 1 1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 0 1 0 1
1 1 1 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0
1 1 1 1 1 0 1 1 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0
1 0 1 1 0 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 0 1 1 1 0 1 1 0 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 1
0 0 1 0 1 1 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 0 0 0 1 0 1 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1
0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 1 1 1 1 0 0 1 1 0 0 1 0 1 1 1 1 0 0 1 1
0 1 0 1 1 1 0 1 0 0 0 0 0 1 1 1 1 0 1 0 0 1 1 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 0 1 1 1 0 1
0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 0 1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 1 1 1 0 1 1 0 0 1 1 0
1 0 1 1 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 0 1 1 1 0 0 0 0 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0
1 1 0 1 1 0 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 1 0 0 1 0 0 1 1
1 1 1 0 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 1 0 1 1 0 0 1 1 1 1 0 1 0 1
0 1 1 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 0 0 1 0 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0 0 1 0 1
1 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 1 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 1 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 0 1 0 1 1 0 0 0 0 1 0 1 0 1 1 0 1 1 0 0
```

```
0 1 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 1 0 1 0 0 1 0 0 1 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0 1 1
1 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0 1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 0 1 1 1 0 1 0 1 0 1 0 0 0 1 0 0 1 1 0 1 1 0 0 0 1 0
1 1 0 0 0 0 1 1 0 0 1 0 1 0 0 0 1 1 1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 1 0
```

Why is this hard?

Why is this hard?

Because of the CURSE of dimensionality!

Why is this hard?

- [Williams '04], [Alman & Williams '15]: Hamming similarity search in time $n^{0.99}\, 2^{o(d)} \Longrightarrow$ k-SAT w. n variables can be solved in time c^n, $c < 2$

Why is this hard?

- [Williams '04], [Alman & Williams '15]: Hamming similarity search in time $n^{0.99}\, 2^{o(d)} \Longrightarrow$ k-SAT w. n variables can be solved in time c^n, $c < 2$

Strong ETH states that this is not possible

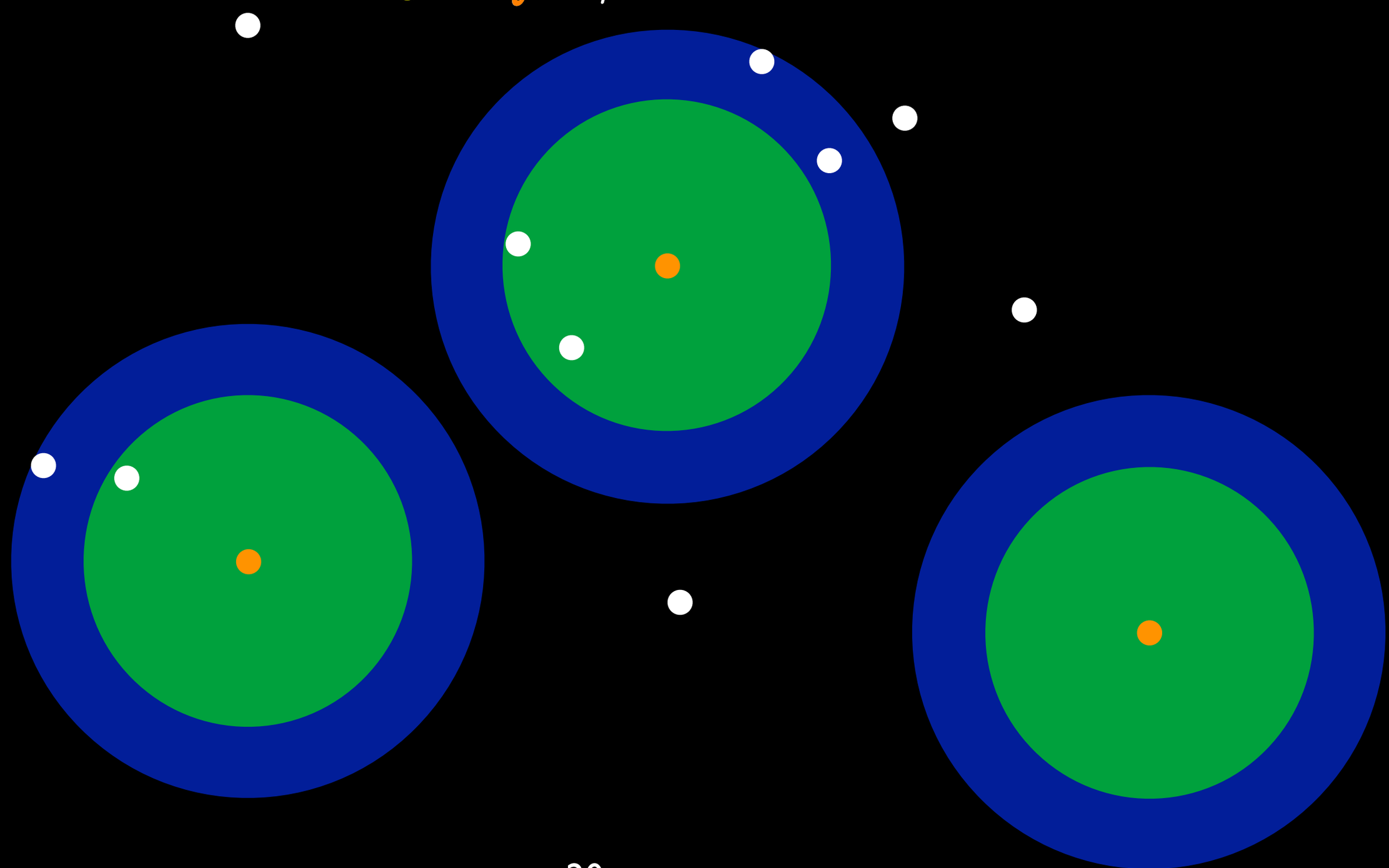
WIRED **Major Advance Reveals the Limits of Computation**

Williams' attempts at disproving SETH have borne considerable fruit. For example, in October he will present a new algorithm for solving the "nearest neighbors" problem. The advance grew out of a failed attempt to disprove SETH.

JOHN PAVLUS SCIENCE 10.11.15 7:00 AM

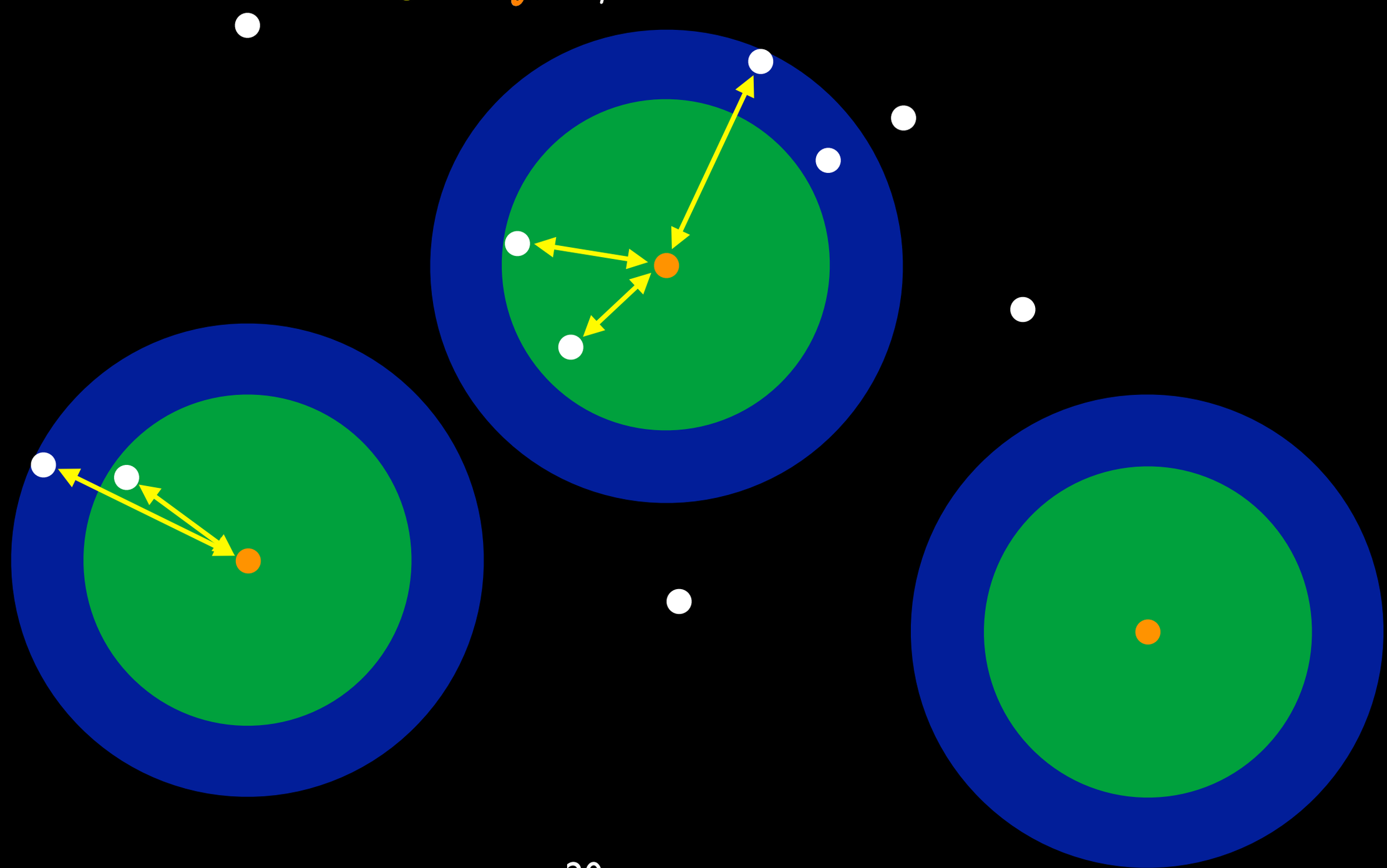
c-approximate similarity join

$C \supseteq Q \bowtie_r S$

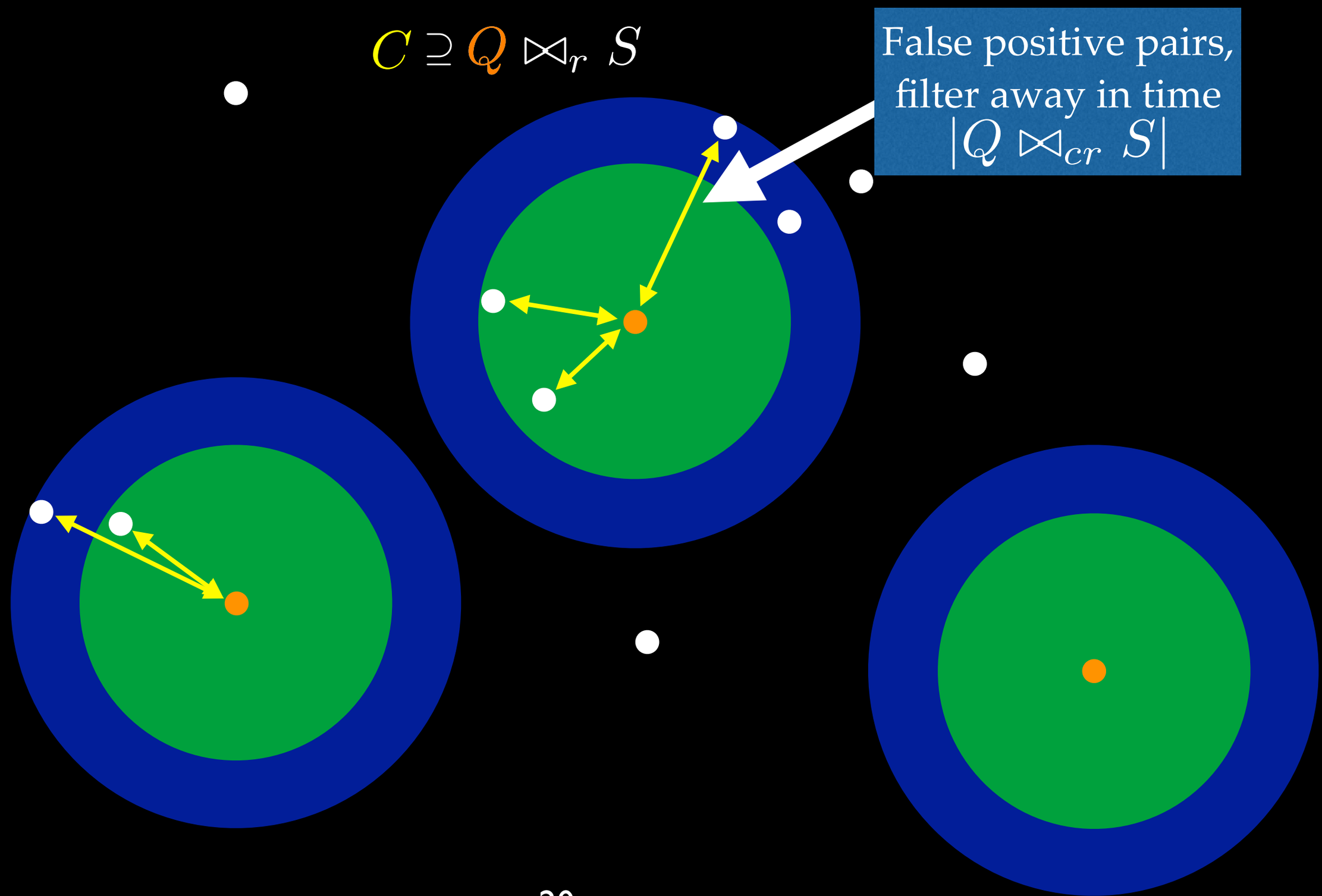


c-approximate similarity join

$C \supseteq Q \bowtie_r S$

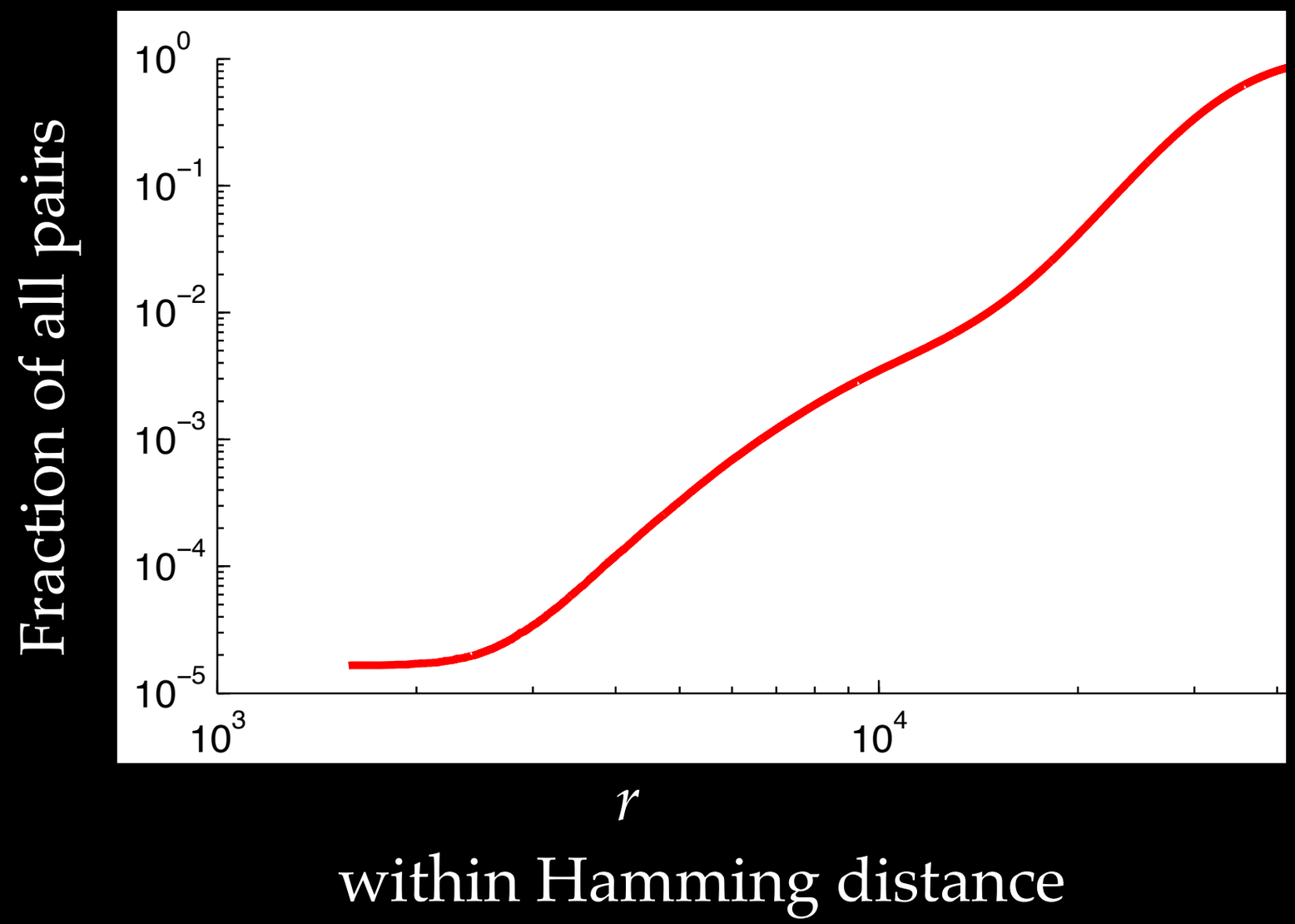


c-approximate similarity join



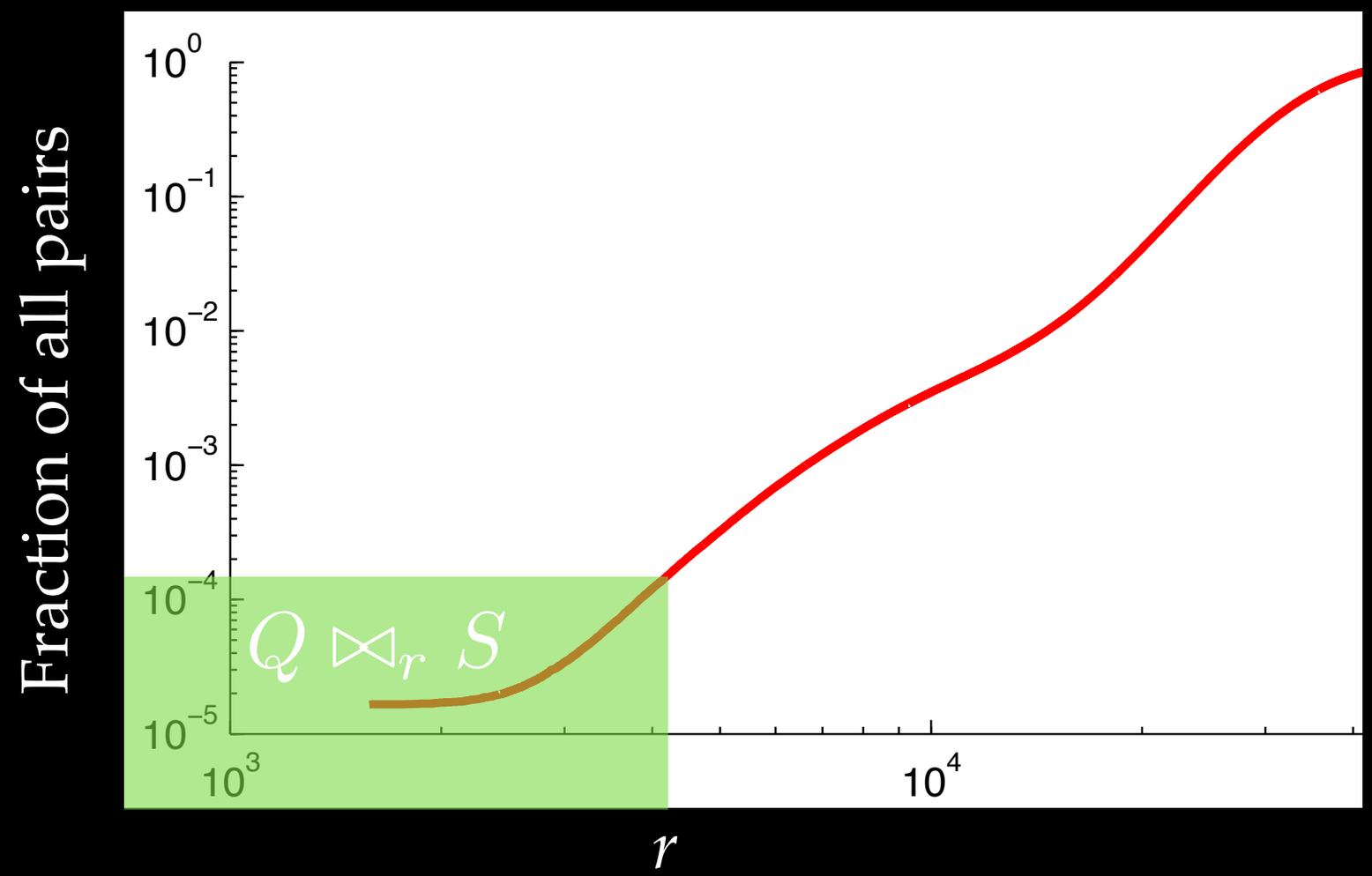
Effect of approximation

MNIST data set (unary encoding)



Effect of approximation

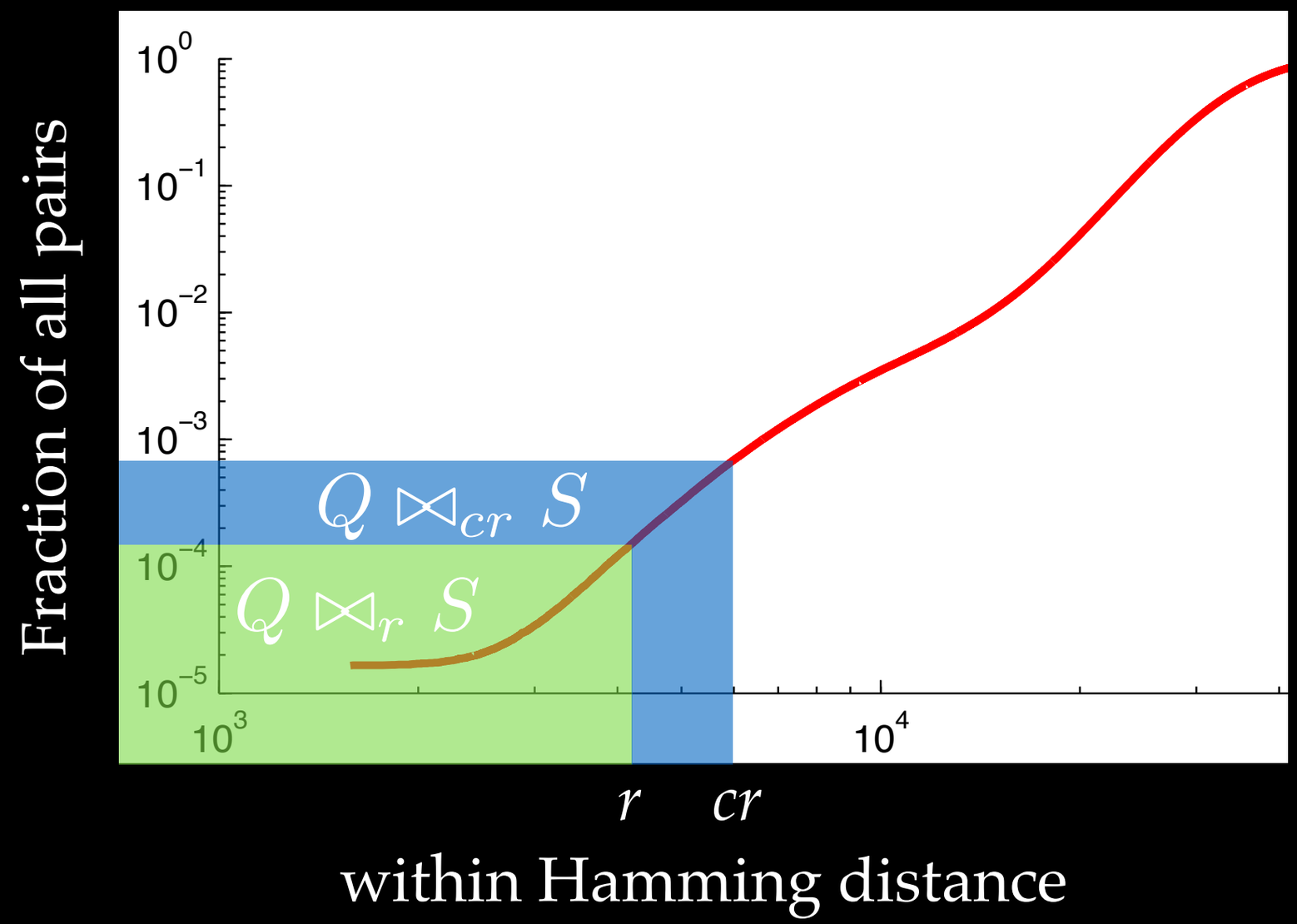
MNIST data set (unary encoding)



within Hamming distance

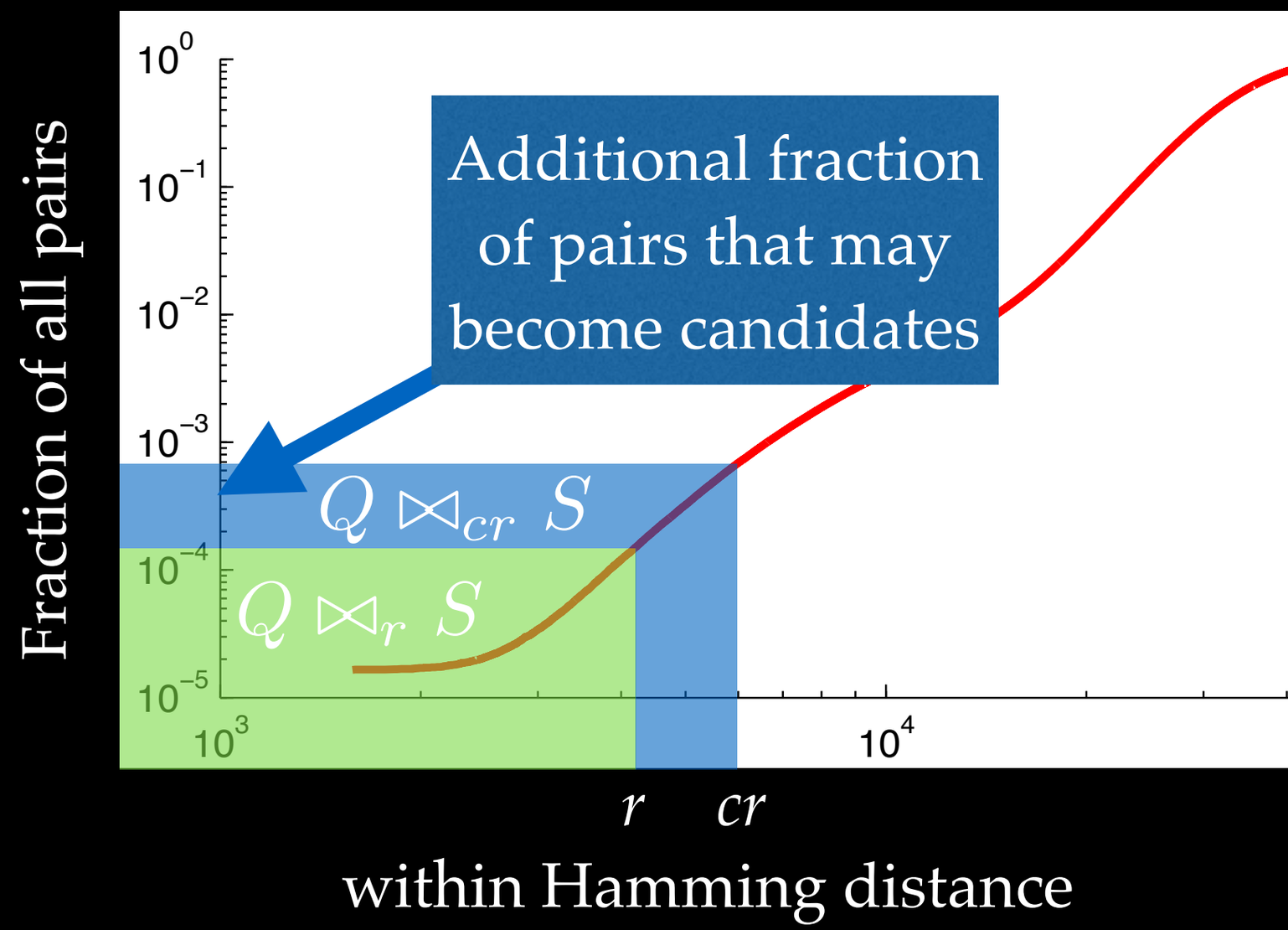
Effect of approximation

MNIST data set (unary encoding)



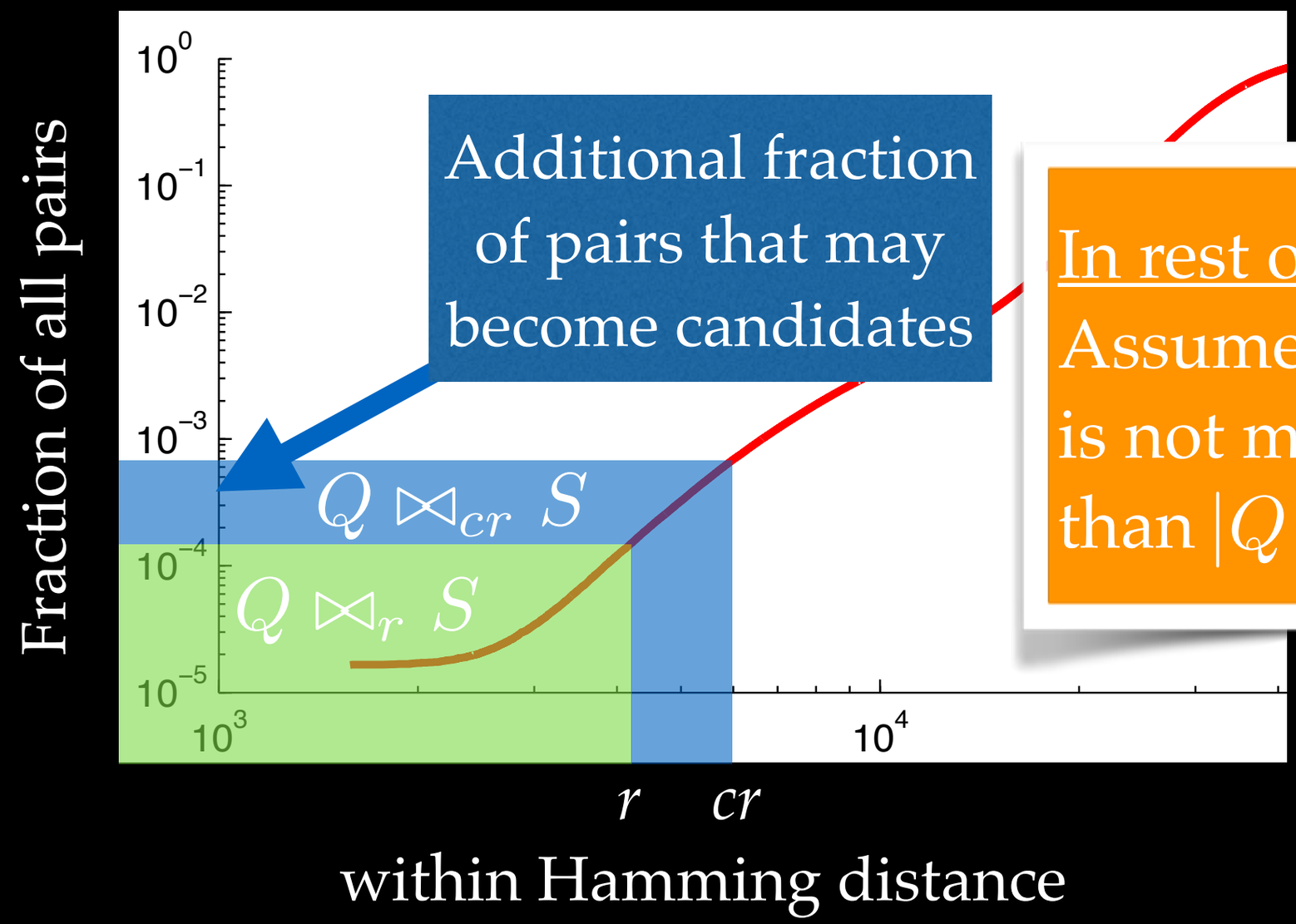
Effect of approximation

MNIST data set (unary encoding)



Effect of approximation

MNIST data set (unary encoding)



Additional fraction of pairs that may become candidates

In rest of the talk: Assume $|Q \bowtie_{cr} S|$ is not much larger than $|Q \bowtie_r S|$

Candidate set generation
- Locality-sensitive hashing (LSH)

[Indyk & Motwani ’98]

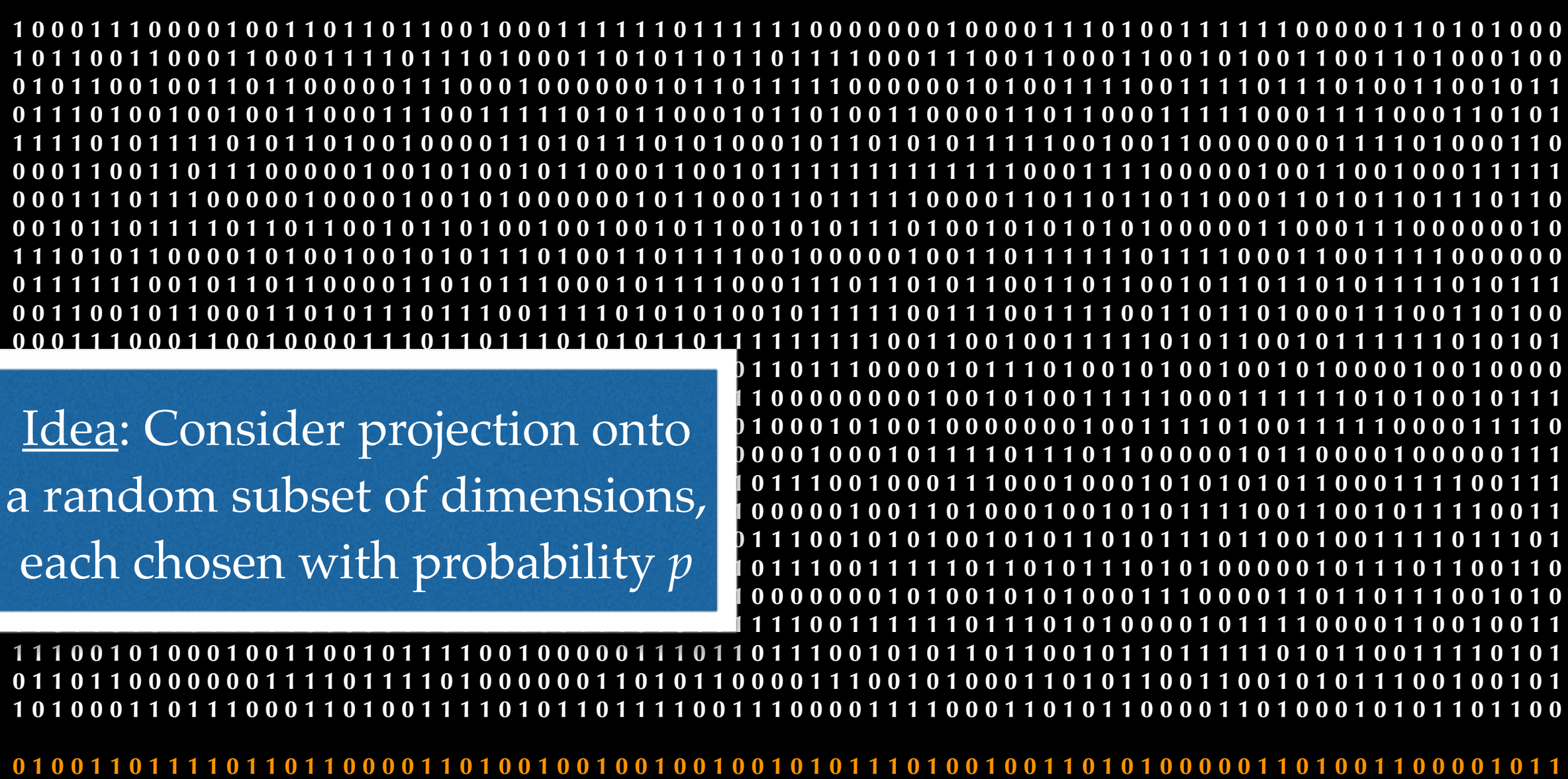
Locality-sensitive hashing

1 0 0 0 1 1 1 0 0 0 0 1 0 0 1 1 0 1 1 0 1 1 0 0 1 0 0 0 1 1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 0 1 0 1 0 0 0
1 0 1 1 0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 1 1 1 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 0 0
0 1 0 1 1 0 0 1 0 0 1 1 0 1 1 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 1 1 0 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1 1 1 0 0 1 1 1 1 0 1 1 1 0 1 0 0 1 1 0 0 1 0 1 1
0 1 1 1 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 1 1 0 0 1 1 1 1 1 0 1 0 1 1 0 0 0 1 0 1 1 0 1 0 0 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 0 1 0 1
1 1 1 1 0 1 0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 0 0 1 1 0 1 0 1 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 1 1 1 0
0 0 0 1 1 0 0 1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 1 1
0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 1 0 0 0 1 1 0 1 1 1 1 1 0 0 0 0 1 1 0 1 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 1 0 1 1 0
0 0 1 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 0 1 0 1 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0
1 1 1 0 1 0 1 1 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 0 0 0 0 0 0
0 1 1 1 1 1 1 0 0 1 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 0 0 0 1 0 1 1 1 1 0 0 0 1 1 1 0 1 1 0 1 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 1 1 1
0 0 1 1 0 0 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 1 0 0 0 1 1 1 0 0 1 1 0 1 0 0
0 0 0 1 1 1 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1 1 1 0 1 0 1 0 1
1 1 1 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 1 1 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 0 0 0
1 1 1 1 1 0 1 1 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 1 1 0 1 0 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0
1 0 1 1 0 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 0 1 1 1 0 1 1 0 0 0 0 0 1 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1 1
0 0 1 0 1 1 1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 0 1 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1
0 1 1 0 1 0 1 0 0 1 0 0 0 0 0 1 0 1 0 0 1 1 1 1 0 0 1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 1 1 1 1 0 0 1 1 0 0 1 0 1 1 1 1 0 0 1 1
0 1 0 1 1 1 0 1 0 0 0 0 0 1 1 1 1 0 1 0 0 1 1 0 0 0 1 1 1 0 1 0 0 1 1 1 1 0 1 1 1 0 0 1 0 1 0 1 0 0 1 0 1 0 1 1 0 1 0 1 1 1 0 1 1 0 0 1 0 0 1 1 1 1 0 1 1 1 0 1
0 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1 0 1 0 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 1 0 1 1 1 0 1 1 0 0 1 1 0
1 0 1 1 1 1 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 1 0 0 0 1 1 1 0 0 0 0 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0
1 1 0 1 1 0 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 0 1 0 1 0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 1 0 0 1 0 0 1 1
1 1 1 0 0 1 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 1 1 0 1 0 1 1 0 0 1 1 1 1 0 1 0 1
0 1 1 0 1 1 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 0 1 1 1 0 0 1 0 1 0 0 0 1 1 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0 0 1 0 1
1 0 1 0 0 0 1 1 0 1 1 1 0 0 0 1 1 0 1 0 0 1 1 1 1 0 1 0 1 1 0 1 1 1 1 0 0 1 1 1 0 0 0 0 1 1 1 1 0 0 0 1 1 0 1 0 1 1 0 0 0 0 1 1 0 1 0 0 0 1 0 1 0 1 1 0 1 1 0 0

0 1 0 0 1 1 0 1 1 1 1 0 1 1 0 1 1 0 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 1 0 1 1 1 0 1 0 0 1 0 0 1 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1 0 0 0 0 1 0 1 1

[Indyk & Motwani '98]

Locality-sensitive hashing



Idea: Consider projection onto a random subset of dimensions, each chosen with probability p

[Indyk & Motwani '98]

Locality-sensitive hashing

Idea: Consider projection onto a random subset of dimensions, each chosen with probability p

[Indyk & Motwani '98]

Locality-sensitive hashing

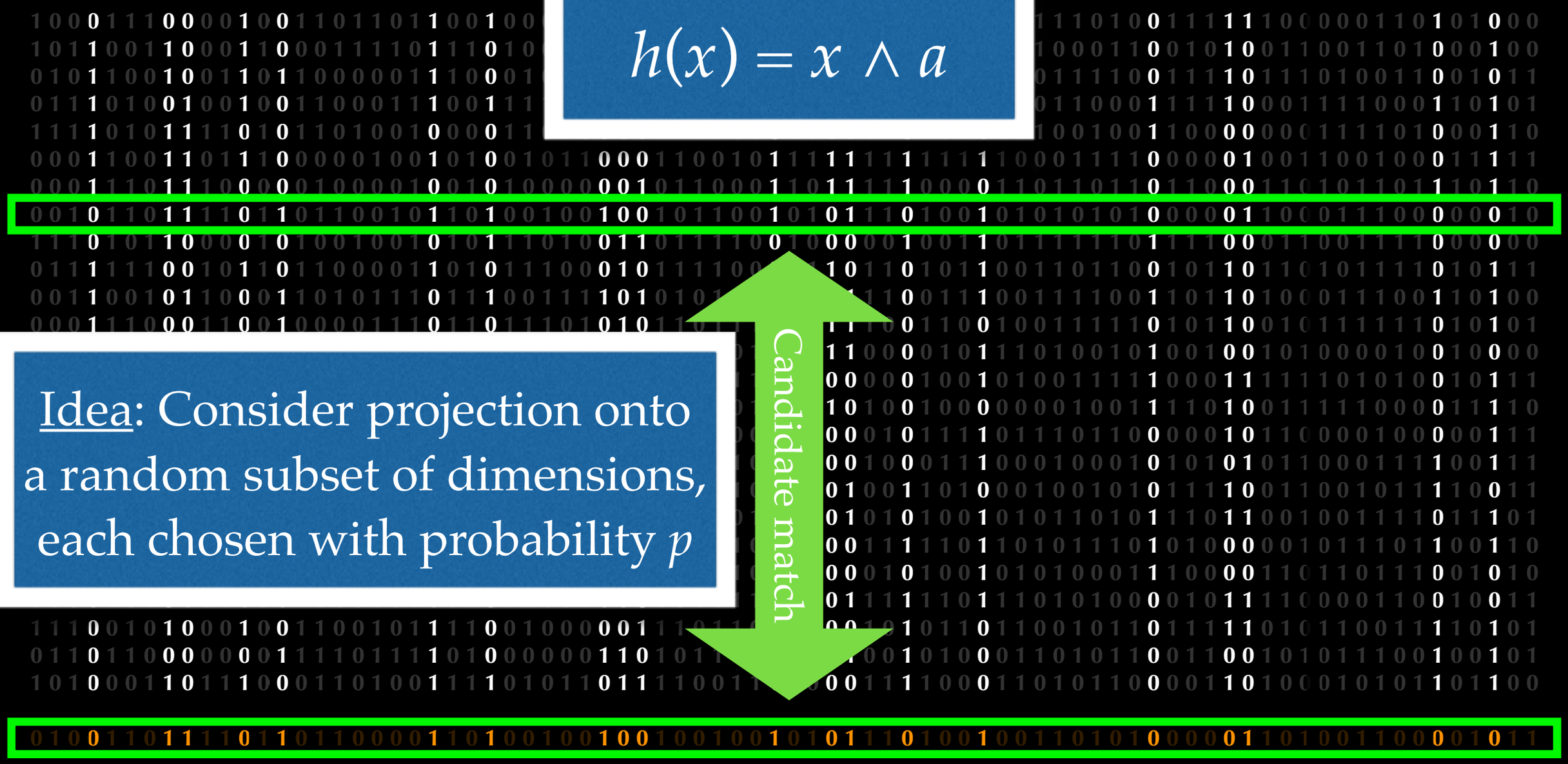
$$h(x) = x \wedge a$$

Idea: Consider projection onto a random subset of dimensions, each chosen with probability p

[Indyk & Motwani '98]

Locality-sensitive hashing

$$h(x) = x \wedge a$$



Idea: Consider projection onto a random subset of dimensions, each chosen with probability p

[Indyk & Motwani '98]

Locality-sensitive hashing

$$h(x) = x \wedge a$$

Candidate match

Idea: Consider projection onto a random subset of dimensions, each chosen with probability p

Repeat enough times that vectors at distance r produce at least one collision

[Indyk & Motwani '98]

Locality-sensitive hashing

$$h_i(x) = x \wedge a_i$$

Candidate match

Idea: Consider projection onto a random subset of dimensions, each chosen with probability p

Repeat enough times that vectors at distance r produce at least one collision

[Indyk & Motwani '98]

Probability of collision?

Collision probability for ith hash table:

$$\Pr[x \wedge a_i = q \wedge a_i] = (1-p)^{||x-q||} \approx e^{-p||x-q||}$$

[Indyk & Motwani '98]

Probability of collision?

Collision probability for ith hash table:

$$\Pr[x \wedge a_i = q \wedge a_i] = (1-p)^{||x-q||} \approx e^{-p||x-q||}$$

With $p = \frac{\ln(n)}{cr}$, probability

$\approx 1/n$ at distance cr and

$\approx 1/n^{1/c}$ at distance r

[Indyk & Motwani '98]

Probability of collision?

Collision probability for ith hash table:

$$\Pr[x \wedge a_i = q \wedge a_i] = (1-p)^{||x-q||} \approx e^{-p||x-q||}$$

$t = n^{1/c}$ repetitions ensure constant success probability

$$x \wedge a_1 \stackrel{?}{=} q \wedge a_1$$

$$x \wedge a_2 \stackrel{?}{=} q \wedge a_2$$

$$x \wedge a_3 \stackrel{?}{=} q \wedge a_3$$

$$\vdots \qquad \vdots$$

$$x \wedge a_t \stackrel{?}{=} q \wedge a_t$$

With $p = \frac{\ln(n)}{cr}$, probability $\approx 1/n$ at distance cr and $\approx 1/n^{1/c}$ at distance r

[Indyk & Motwani '98]

Probability of collision?

Collision probability for ith hash table:

$$\Pr[x \wedge a_i = q \wedge a_i] = (1-p)^{||x-q||} \approx e^{-p||x-q||}$$

$t = n^{1/c}$ repetitions ensure constant success probability

$$x \wedge a_1 \stackrel{?}{=} q \wedge a_1$$
$$x \wedge a_2 \stackrel{?}{=} q \wedge a_2$$
$$x \wedge a_3 \stackrel{?}{=} q \wedge a_3$$
$$\vdots \qquad \vdots$$
$$x \wedge a_t \stackrel{?}{=} q \wedge a_t$$

With $p = \frac{\ln(n)}{cr}$, probability $\approx 1/n$ at distance cr and $\approx 1/n^{1/c}$ at distance r

Possibility of *false negatives*: Saying 'No' when 'Yes' is required

Summary of analysis

Analysis (GIM '99): Each bit of a_i is 1 with probability p.

- Expect $c_p \leq n^2(1\text{-}p)^{cr}$ candidate pairs outside $Q \bowtie_{cr} S$
- Probability of $h(q)=h(x)$ when $d(q,x) = r$ is $(1\text{-}p)^r$, so need to hash $t_p=(1\text{-}p)^{-r}$ times to achieve constant recall.
- Choose p to balance cost of hashing t_p times and checking $t_p\, c_p$ candidate pairs not in $Q \bowtie_{cr} S$

Summary of analysis

Analysis (GIM '99): Each bit of a_i is 1 with probability p.

- Expect $c_p \leq n^2(1\text{-}p)^{cr}$ candidate pairs outside $Q \bowtie_{cr} S$
- Probability of $h(q)=h(x)$ when $d(q,x) = r$ is $(1\text{-}p)^r$, so need to hash $t_p=(1\text{-}p)^{-r}$ times to achieve constant recall.
- Choose p to balance cost of hashing t_p times and checking $t_p\, c_p$ candidate pairs not in $Q \bowtie_{cr} S$

Summary of analysis

Analysis (GIM '99): Each bit of a_i is 1 with probability p.

- Expect $c_p \leq n^2(1\text{-}p)^{cr}$ candidate pairs outside $Q \bowtie_{cr} S$
- Probability of $h(q)=h(x)$ when $d(q,x) = r$ is $(1\text{-}p)^r$, so need to hash $t_p=(1\text{-}p)^{-r}$ times to achieve constant recall.
- Choose p to balance cost of hashing t_p times and checking $t_p\, c_p$ candidate pairs not in $Q \bowtie_{cr} S$

Locality-sensitive hashing

Analysis (GIM '99): Each bit of a_i is 1 with probability p.

- Expect $c_p \leq n^2(1\text{-}p)^{cr}$ candidate pairs outside $Q \bowtie_{cr} S$
- Probability of $h(q)=h(x)$ when $d(q,x) = r$ is $(1\text{-}p)^r$, so need to hash $t_p=(1\text{-}p)^{-r}$ times to achieve constant recall.
- Choose p to balance cost of hashing t_p times and checking $t_p\, c_p$ candidate pairs not in $Q \bowtie_{cr} S$

Locality-sensitive hashing

Analysis (GIM '99): Each bit of a_i is 1 with probability p.

- Expect $c_p \leq n^2(1\text{-}p)^{cr}$ candidate pairs outside $Q \bowtie_{cr} S$
- Probability of $h(q)=h(x)$ when $d(q,x) = r$ is $(1\text{-}p)^r$, so need to hash $t_p=(1\text{-}p)^{-r}$ times to achieve constant recall.
- Choose p to balance cost of hashing t_p times and checking $t_p\, c_p$ candidate pairs not in $Q \bowtie_{cr} S$

Number of operations is $O(dn^{1+1/c})$, expected;

Newer theory developments*

Reference	Exponent, search time	Comment
Linear search	1	
Indyk & Motwani STOC '98	$1/c$	Worst case upper bound

* Focus on subquadratic space; lower order terms ignored.

Newer theory developments*

Reference	Exponent, search time	Comment
Linear search	1	
Indyk & Motwani STOC '98	$1/c$	Worst case upper bound
O'Donnell, Wu & Zhou ITCS '10	$1/c$	Data indep. LSH, worst case lower bound

* Focus on subquadratic space; lower order terms ignored.

Newer theory developments*

Reference	Exponent, search time	Comment
Linear search	1	
Indyk & Motwani STOC '98	$1/c$	Worst case upper bound
O'Donnell, Wu & Zhou ITCS '10	$1/c$	Data indep. LSH, worst case lower bound
Dubiner Trans. Inf. Theory '10	$1/(2c-1)$	Random data upper bound

* Focus on subquadratic space; lower order terms ignored.

Newer theory developments*

Reference	Exponent, search time	Comment
Linear search	1	
Indyk & Motwani STOC '98	$1/c$	Worst case upper bound
O'Donnell, Wu & Zhou ITCS '10	$1/c$	Data indep. LSH, worst case lower bound
Dubiner Trans. Inf. Theory '10	$1/(2c-1)$	Random data upper bound
Andoni & Razenshteyn STOC '15	$1/(2c-1)$	Data dep. LSH, worst case upper bound

* Focus on subquadratic space; lower order terms ignored.

Newer theory developments*

Reference	Exponent, search time	Comment
Linear search	1	
Indyk & Motwani STOC '98	$1/c$	Worst case upper bound
O'Donnell, Wu & Zhou ITCS '10	$1/c$	Data indep. LSH, worst case lower bound
Dubiner Trans. Inf. Theory '10	$1/(2c-1)$	Random data upper bound
Andoni & Razenshteyn STOC '15	$1/(2c-1)$	Data dep. LSH, worst case upper bound and lower bound

* Focus on subquadratic space; lower order terms ignored.

Newer theory developments*

Reference	Exponent, search time	Comment
Linear search	1	
Indyk & Motwani STOC '98	$1/c$	Worst case upper bound
O'Donnell, Wu & Zhou ITCS '10	$1/c$	Data indep. LSH, worst case lower bound
Dubiner Trans. Inf. Theory '10	$1/(2c-1)$	Random data upper bound
Andoni & Razenshteyn STOC '15	$1/(2c-1)$	Data dep. LSH, worst case upper bound and lower bound
Kapralov PODS '15	$4/(c+1)$	Linear space upper bound

* Focus on subquadratic space; lower order terms ignored.

Newer theory developments*

Reference	Exponent, search time	Comment
Linear search	1	
Indyk & Motwani STOC '98	$1/c$	Worst case upper bound
O'Donnell, Wu & Zhou ITCS '10	$1/c$	Data indep. LSH, worst case lower bound
Dubiner Trans. Inf. Theory '10	$1/(2c-1)$	Random data upper bound
Andoni & Razenshteyn STOC '15	$1/(2c-1)$	Data dep. LSH, worst case upper bound and lower bound
Kapralov PODS '15	$4/(c+1)$	Linear space upper bound
Valiant FOCS '12	$\frac{1}{4-\omega} + O\left(\frac{1-1/c}{\log d}\right)$	Batched search [random data]

* Focus on subquadratic space; lower order terms ignored. $\omega < 2.38$

Newer theory developments*

Reference	Exponent, search time	Comment
Linear search	1	
Indyk & Motwani STOC '98	$1/c$	Worst case upper bound
O'Donnell, Wu & Zhou ITCS '10	$1/c$	Data indep. LSH, worst case lower bound
Dubiner Trans. Inf. Theory '10	$1/(2c-1)$	Random data upper bound
Andoni & Razenshteyn STOC '15	$1/(2c-1)$	Data dep. LSH, worst case upper bound and lower bound
Kapralov PODS '15	$4/(c+1)$	Linear space upper bound
Valiant FOCS '12	$\frac{1}{4-\omega} + O\left(\frac{1-1/c}{\log d}\right)$	Batched search [random data]
Karppa, Kaski & Kohonen SODA '16	$\frac{2\omega-3}{3}$	Batched search [c>1, random data, $\omega > 2.25$]

* Focus on subquadratic space; lower order terms ignored. $\omega < 2.38$

Newer theory developments*

Reference	Exponent, search time	Comment
Linear search	1	
Indyk & Motwani STOC '98	$1/c$	Worst case upper bound
O'Donnell, Wu & Zhou ITCS '10	$1/c$	Data indep. LSH, worst case lower bound
Dubiner Trans. Inf. Theory '10	$1/(2c-1)$	Random data upper bound
Andoni & Razenshteyn STOC '15	$1/(2c-1)$	Data dep. LSH, worst case upper bound and lower bound
Kapralov PODS '15	$4/(c+1)$	Linear space upper bound
Valiant FOCS '12	$\frac{1}{4-\omega} + O\left(\frac{1-1/c}{\log d}\right)$	Batched search [random data]
Karppa, Kaski & Kohonen SODA '16	$\frac{2\omega-3}{3}$	Batched search [c>1, random data, $\omega > 2.25$]
Alman & Williams FOCS '15	$1 - \tilde{\Omega}(\log(n)/d)$	Batched search, c=1

* Focus on subquadratic space; lower order terms ignored. $\omega < 2.38$

Can LSH work for large data?

- Quote from a database paper: *"LSH needs large memory space and long processing time to achieve good performance when searching a massive dataset"*.

Can LSH work for large data?

- Quote from a database paper: *"LSH needs large memory space and long processing time to achieve good performance when searching a massive dataset"*.

For similarity join, space is linear: Process *one hash function at a time.*

Can LSH work for large data?

- Quote from a database paper: *"LSH needs large memory space and long processing time to achieve good performance when searching a massive dataset"*.

For similarity join, space is linear: Process *one hash function at a time.*

- <u>Other issues</u>:
  - LSH parameters are pessimistic; chosen to work for worst-case data set.

Can LSH work for large data?

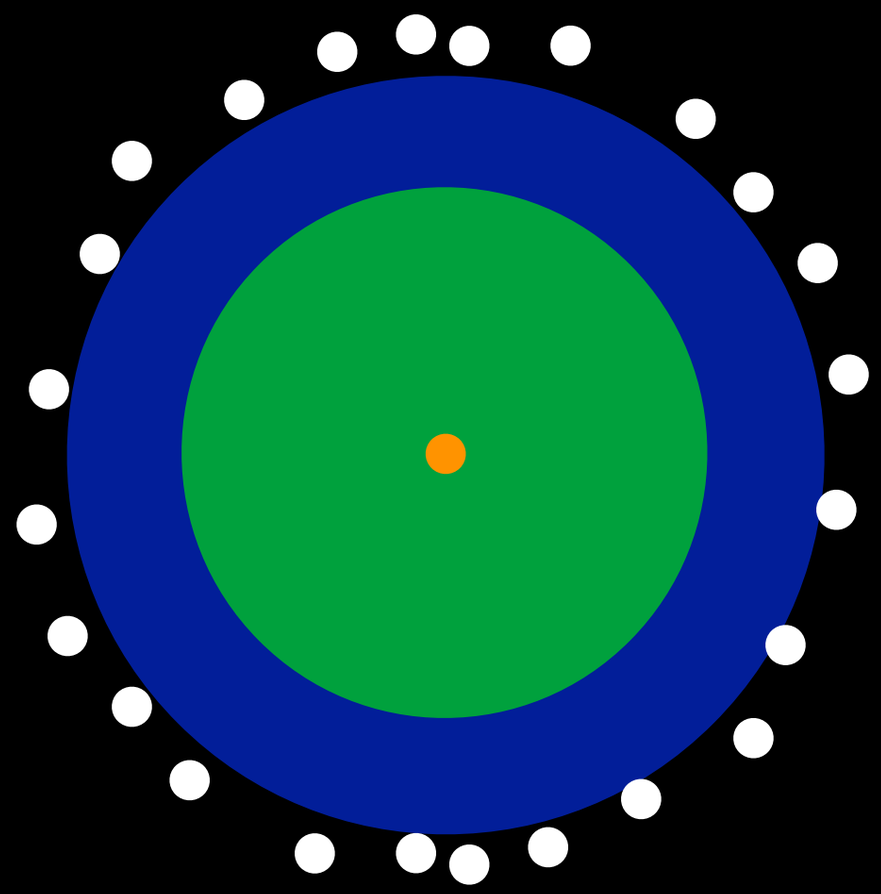
- Quote from a database paper: *"LSH needs large memory space and long processing time to achieve good performance when searching a massive dataset"*.

For similarity join, space is linear: Process *one hash function at a time.*

- <u>Other issues</u>:
  - LSH parameters are pessimistic; chosen to work for worst-case data set.
  - Poor use of internal memory: A simple nested loop join often has better I/O complexity.

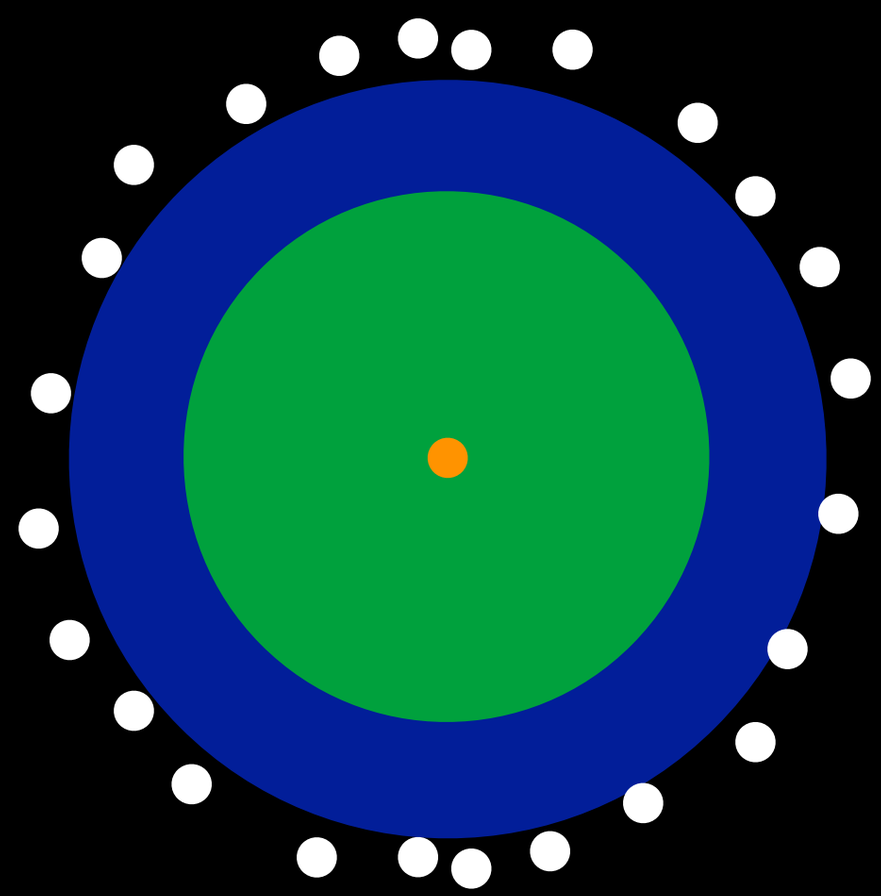
On pessimism

- LSH chosen to ensure few collisions at distance *cr*, even in worst-case scenarios:



On pessimism

- LSH chosen to ensure few collisions at distance *cr*, even in worst-case scenarios:



Price paid:
Also small collision probability at distance *r*, so need many repetitions.

Candidate set generation

- Cache-efficiency via recursion

I/O model

- Data is stored on an external storage device, with B vectors / storage block
- Internal memory can hold M vectors
- Count the number of block transfers between internal memory and external storage (I/Os)

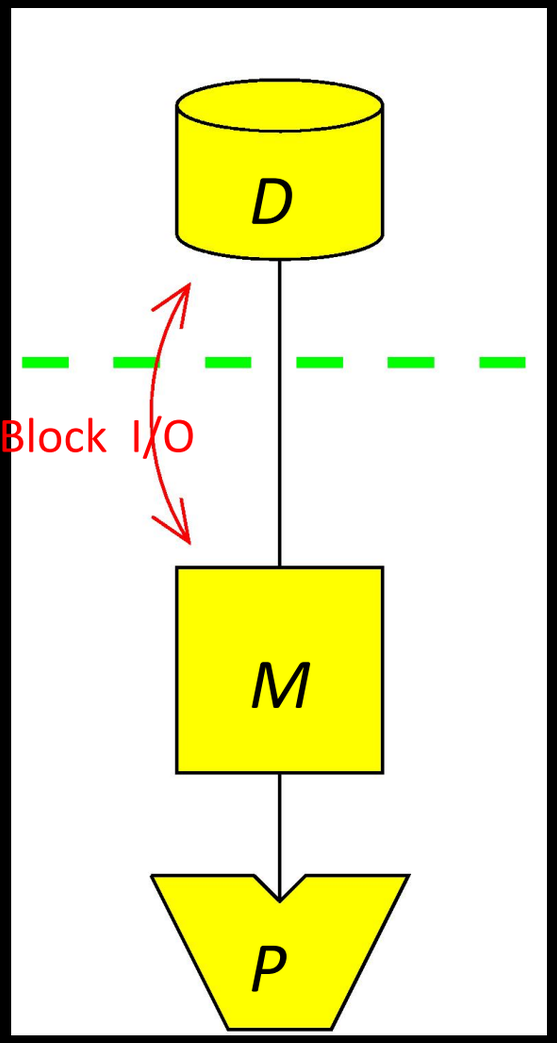


Figure courtesy of Lars Arge

Cautious LSH

Joint work with Pham, Silvestri, and Stöckel

<u>Idea</u>: Apply a weak LSH with constant collision probability.

Cautious LSH

Joint work with Pham, Silvestri, and Stöckel

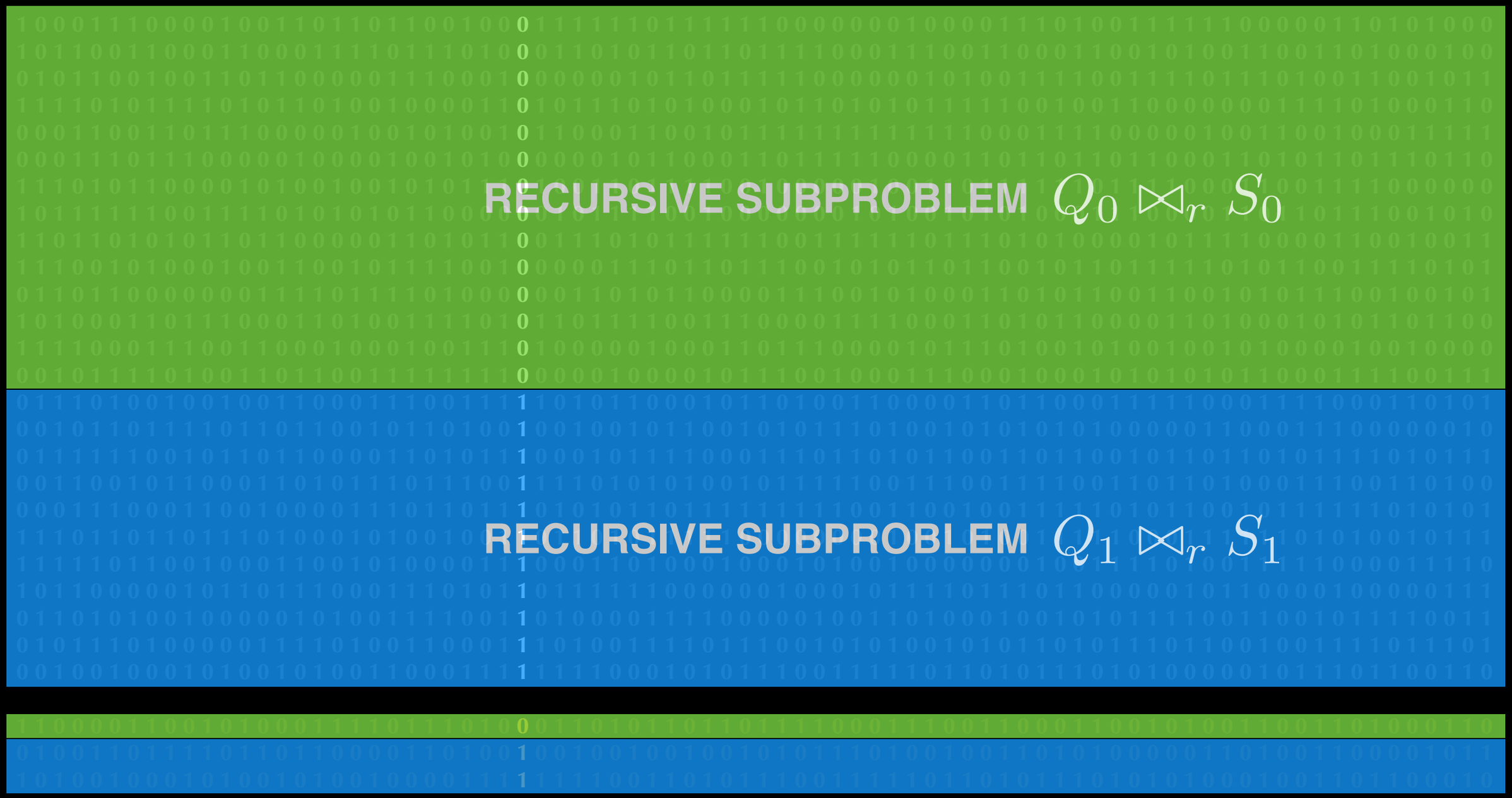
Idea: Apply a weak LSH with constant collision probability.

Cautious LSH

Joint work with Pham, Silvestri, and Stöckel

Cautious LSH

Joint work with Pham, Silvestri, and Stöckel



Cautious LSH

Joint work with Pham, Silvestri, and Stöckel

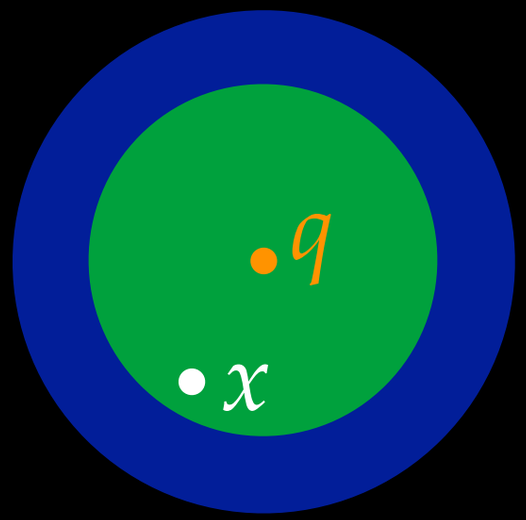
RECURSIVE SUBPROBLEM $Q_0 \bowtie_r S_0$

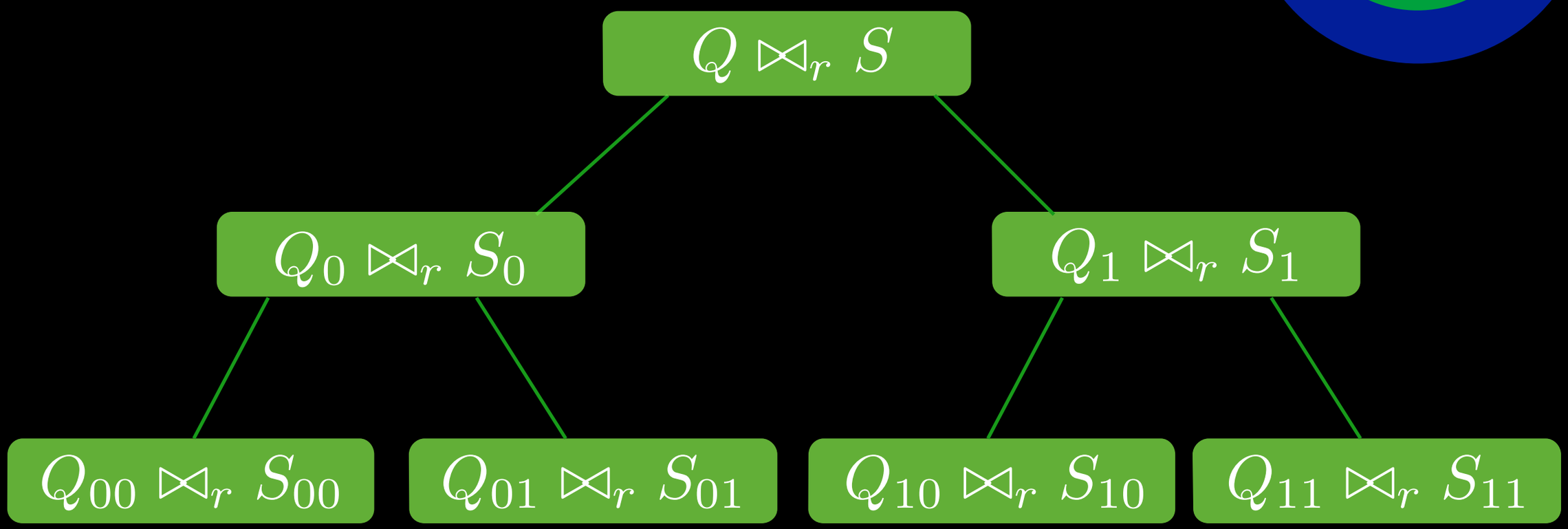
Subproblems of size M or less require no further I/Os

RECURSIVE SUBPROBLEM $Q_1 \bowtie_r S_1$

Example, cautious LSH

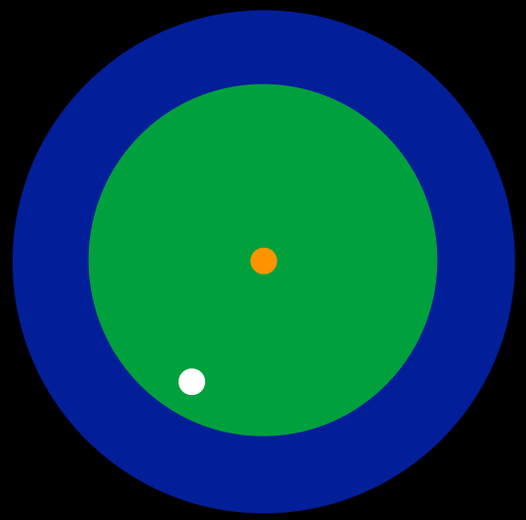
- Assume weak LSH collision prob. 0.9.
- Prob. that q and x collide i times is 0.9^i.
- If depth is log n, success probability is $\approx n^{-0.152}$.

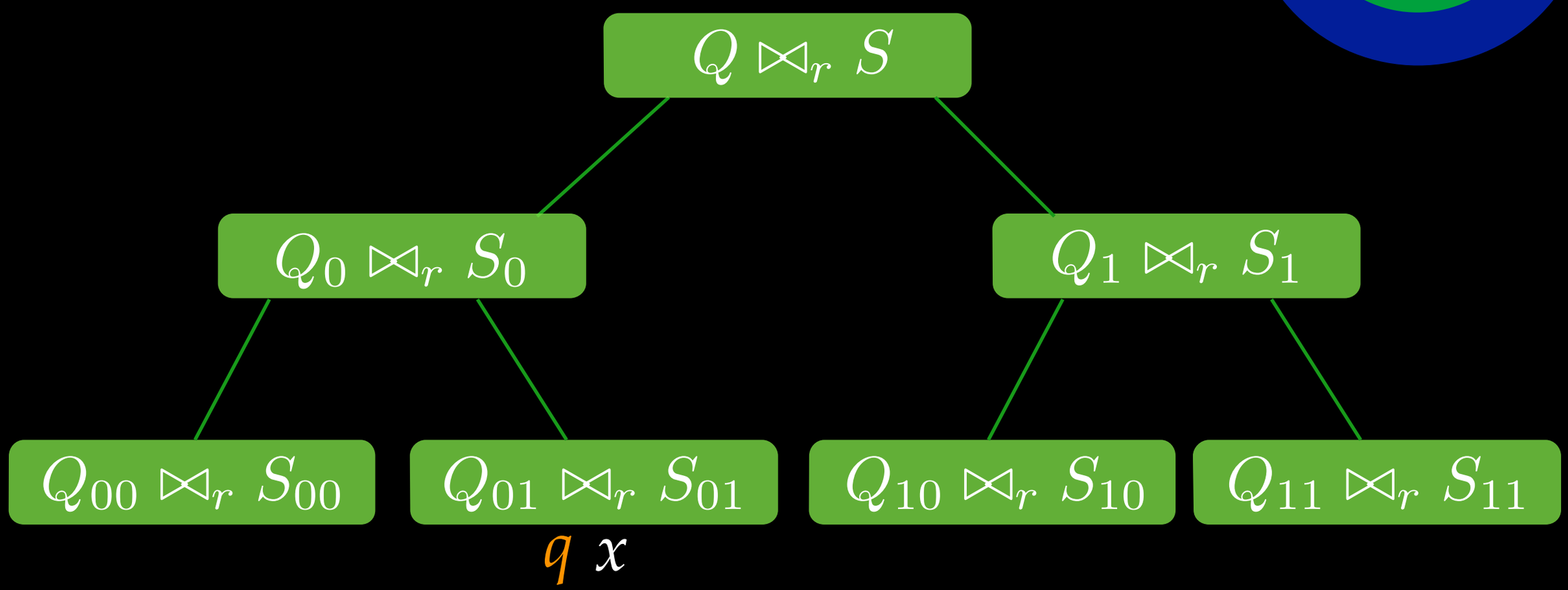




Example, cautious LSH

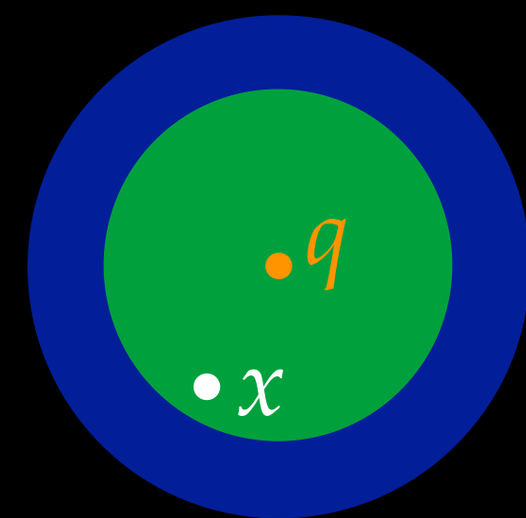
- Assume weak LSH collision prob. 0.9.
- Prob. that q and x collide i times is 0.9^i.
- If depth is log n, success probability is $\approx n^{-0.152}$.





Example 2, cautious LSH

- Assume weak LSH collision prob. 0.5.
- Prob. that q and x collide i times is 0.5^i.

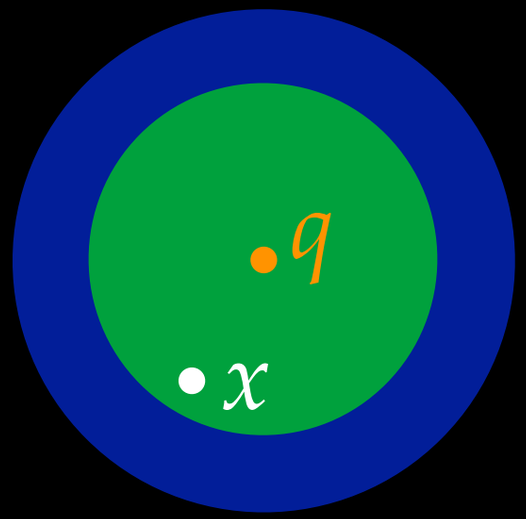


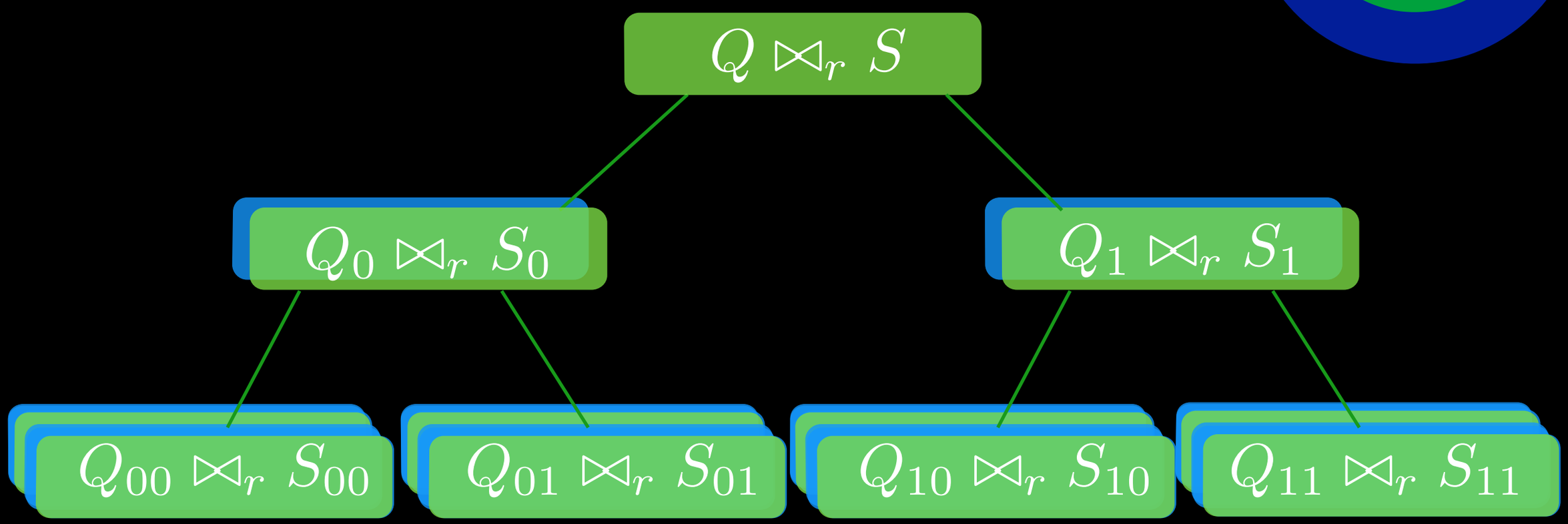
$$Q \bowtie_r S$$

$$Q_0 \bowtie_r S_0 \qquad Q_1 \bowtie_r S_1$$

Example 2, cautious LSH

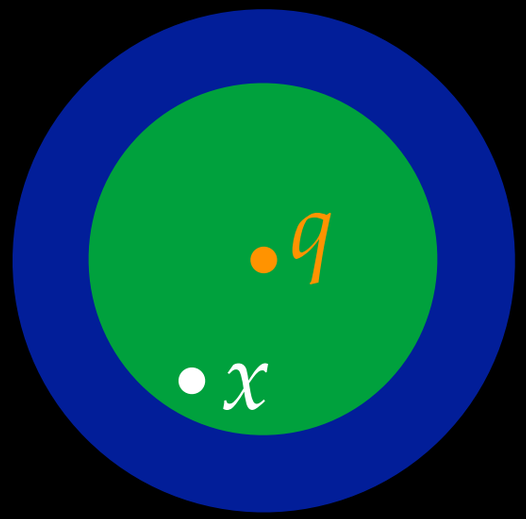
- Assume weak LSH collision prob. 0.5.
- Prob. that q and x collide i times is 0.5^i.
- <u>Idea</u>: Recurse *twice* at each node.



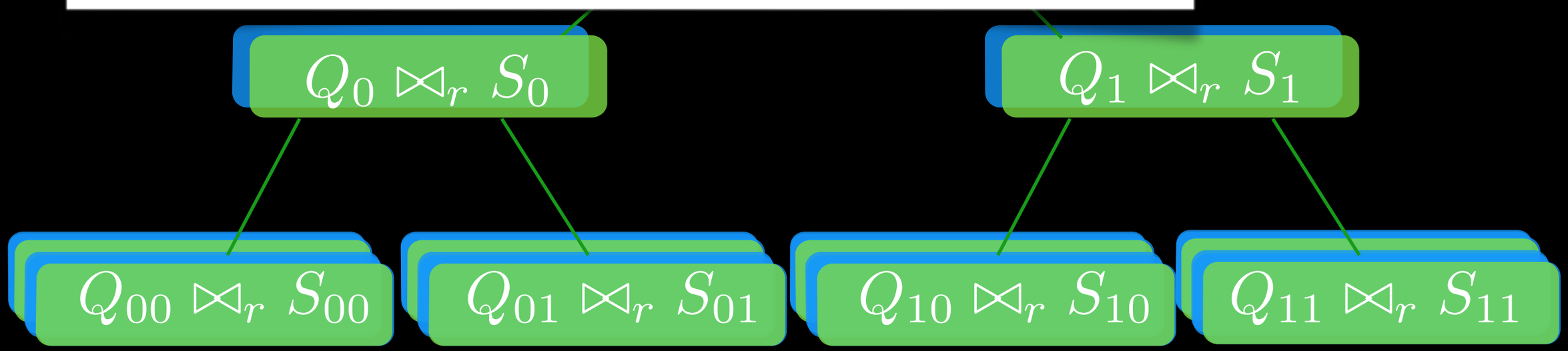


Example 2, cautious LSH

- Assume weak LSH collision prob. 0.5.
- Prob. that q and x collide i times is 0.5^i.
- <u>Idea</u>: Recurse *twice* at each node.



#subproblems containing q **and** x:
1 at each level, expected!

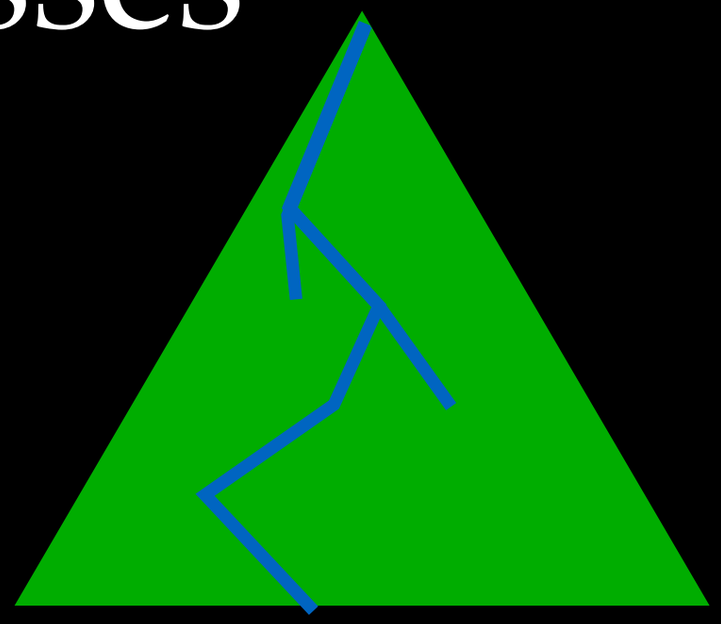


Aside:
Don’t be fooled by great expectations

Aside:
Don't be fooled by great expectations

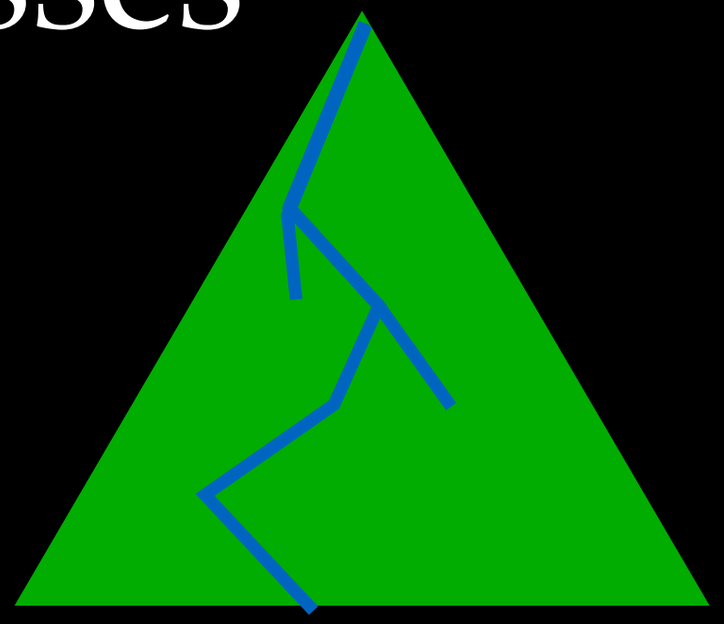
What is the expected TNT equivalent of asteroid impacts during this talk?

Branching processes



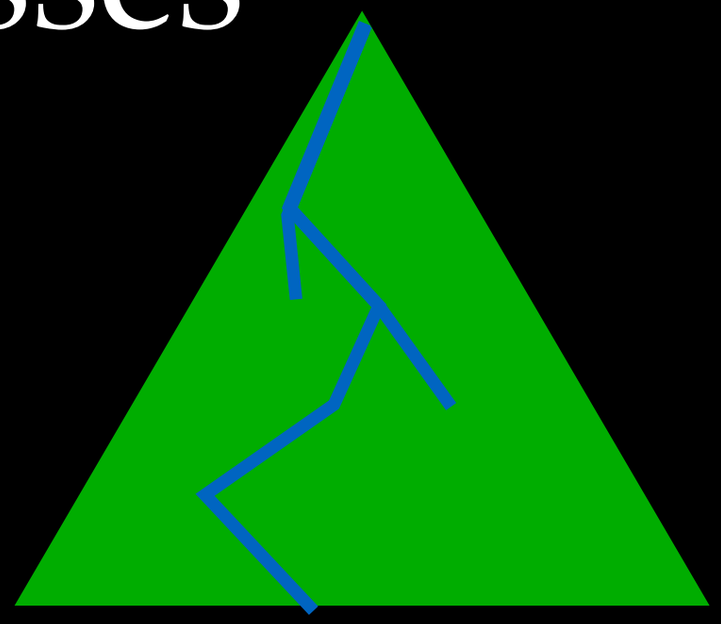
- <u>Abstract setting</u>:
  - One pair (q,x) at root problem.
  - Generate t subproblems such that (q,x) is "reproduced" in each with probability $\geq 1/t$.

Branching processes



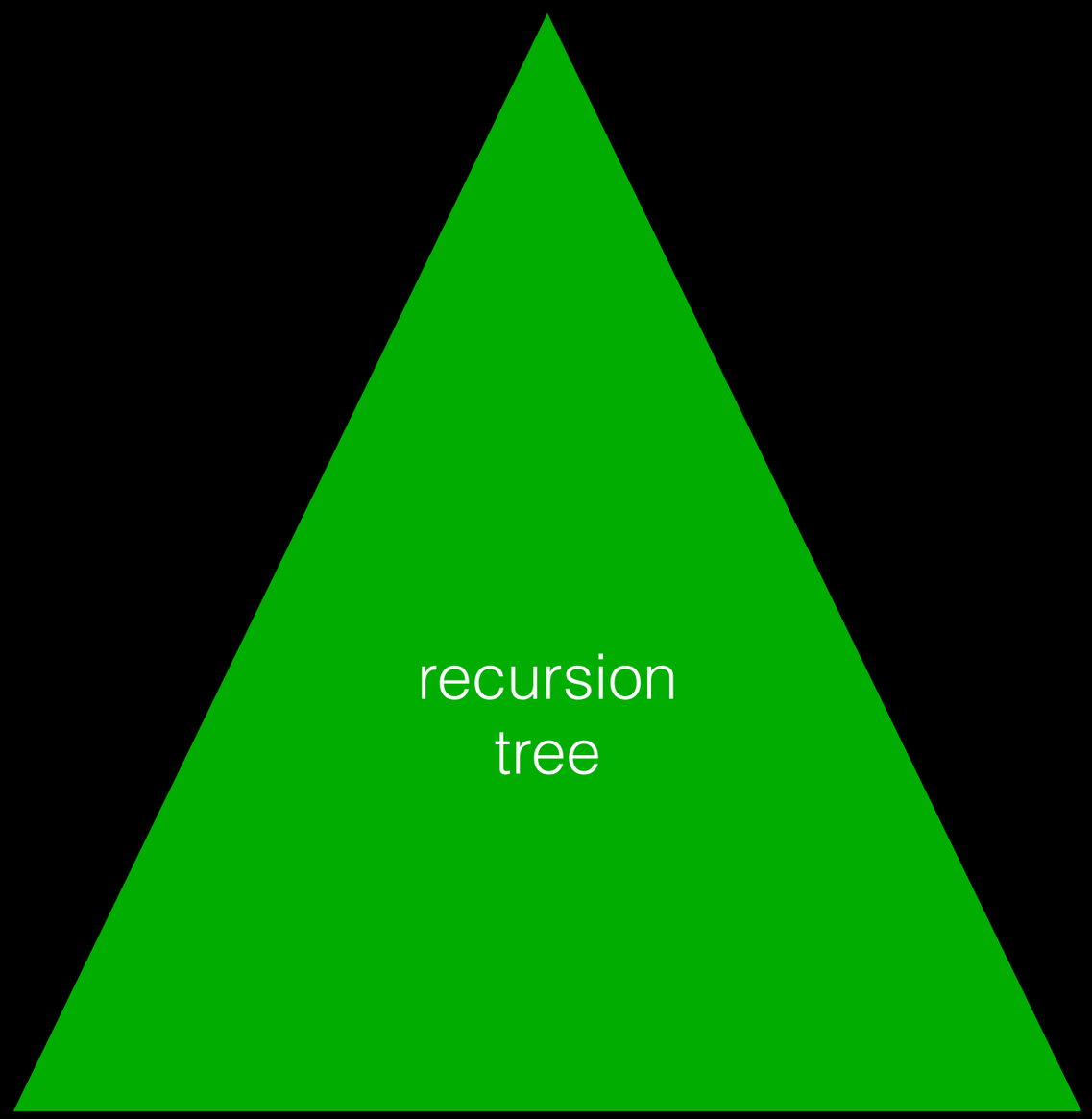
- <u>Abstract setting</u>:
  - One pair (q,x) at root problem.
  - Generate t subproblems such that (q,x) is "reproduced" in each with probability $\geq 1/t$.
- What is the probability that (q,x) is *extinct* at recursive level i?

Branching processes

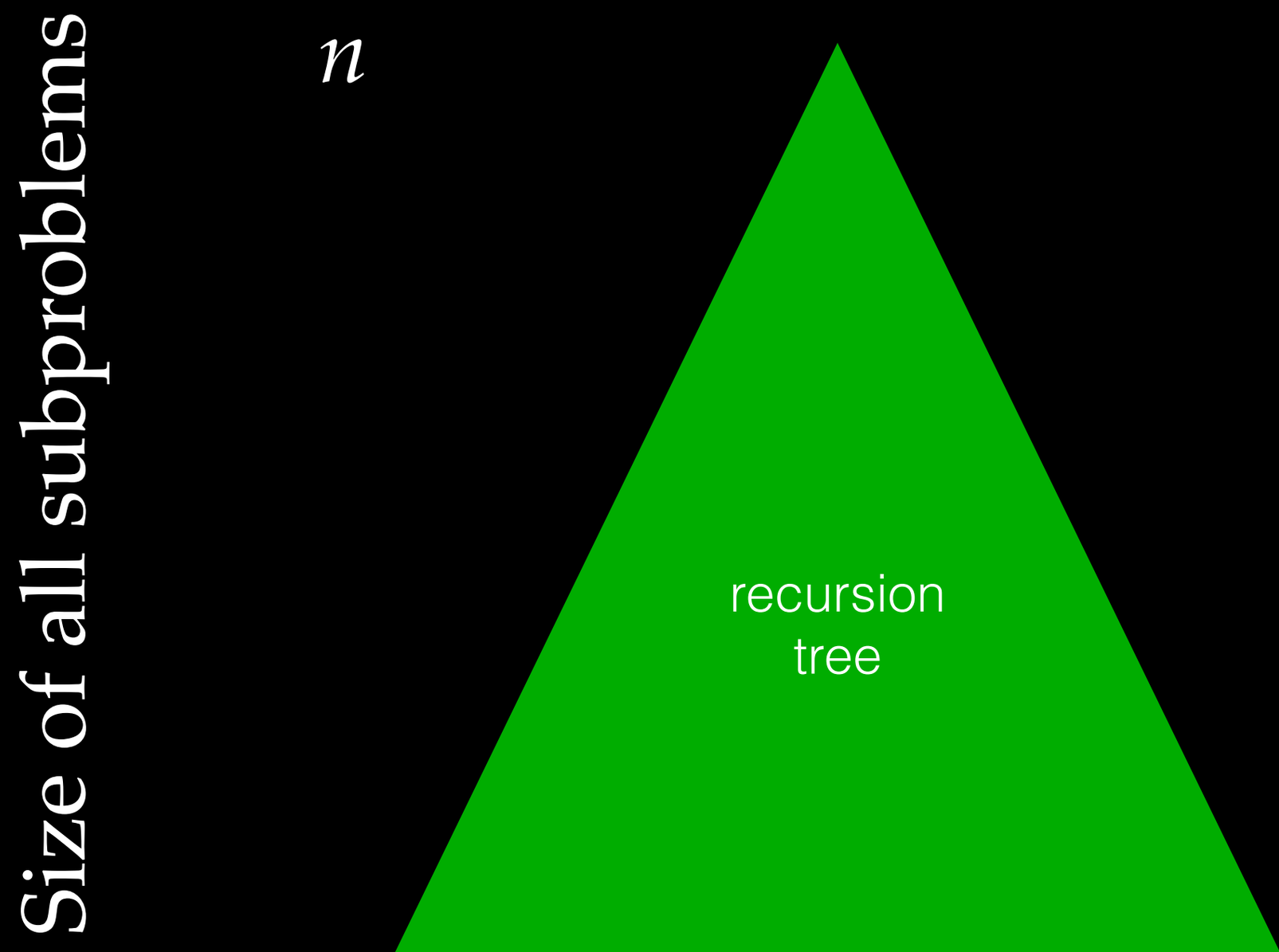


- <u>Abstract setting</u>:
  - One pair (q,x) at root problem.
  - Generate t subproblems such that (q,x) is "reproduced" in each with probability $\geq 1/t$.
- What is the probability that (q,x) is *extinct* at recursive level i?
  - From theory of branching processes: $\Omega(1/\sqrt{i})$

I/O complexity, sketch



I/O complexity, sketch

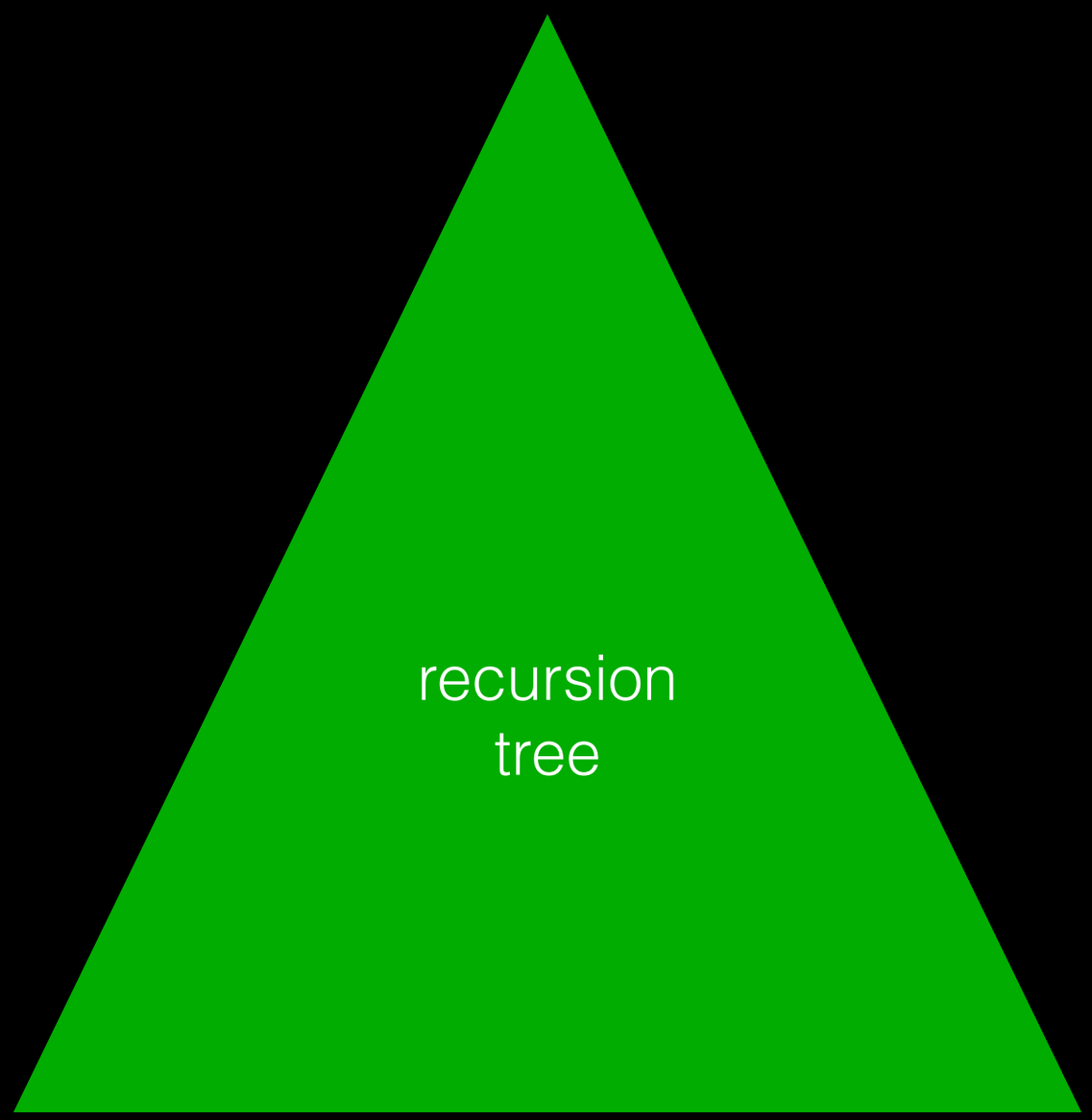


I/O complexity, sketch

Size of all subproblems

n

nt



I/O complexity, sketch

Size of all subproblems

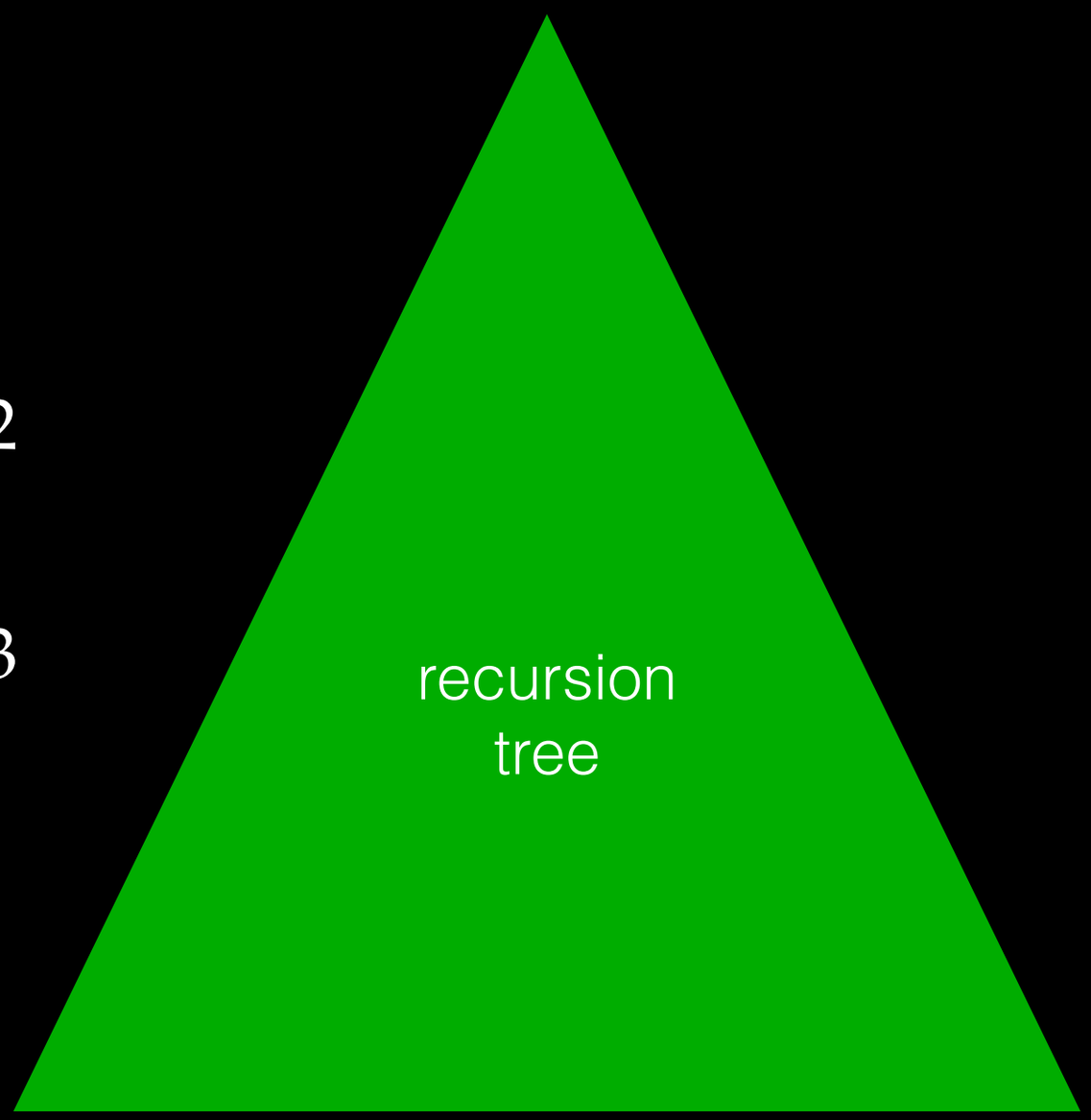
n

nt

nt^2

nt^3

$\vdots$



I/O complexity, sketch

Size of all subproblems

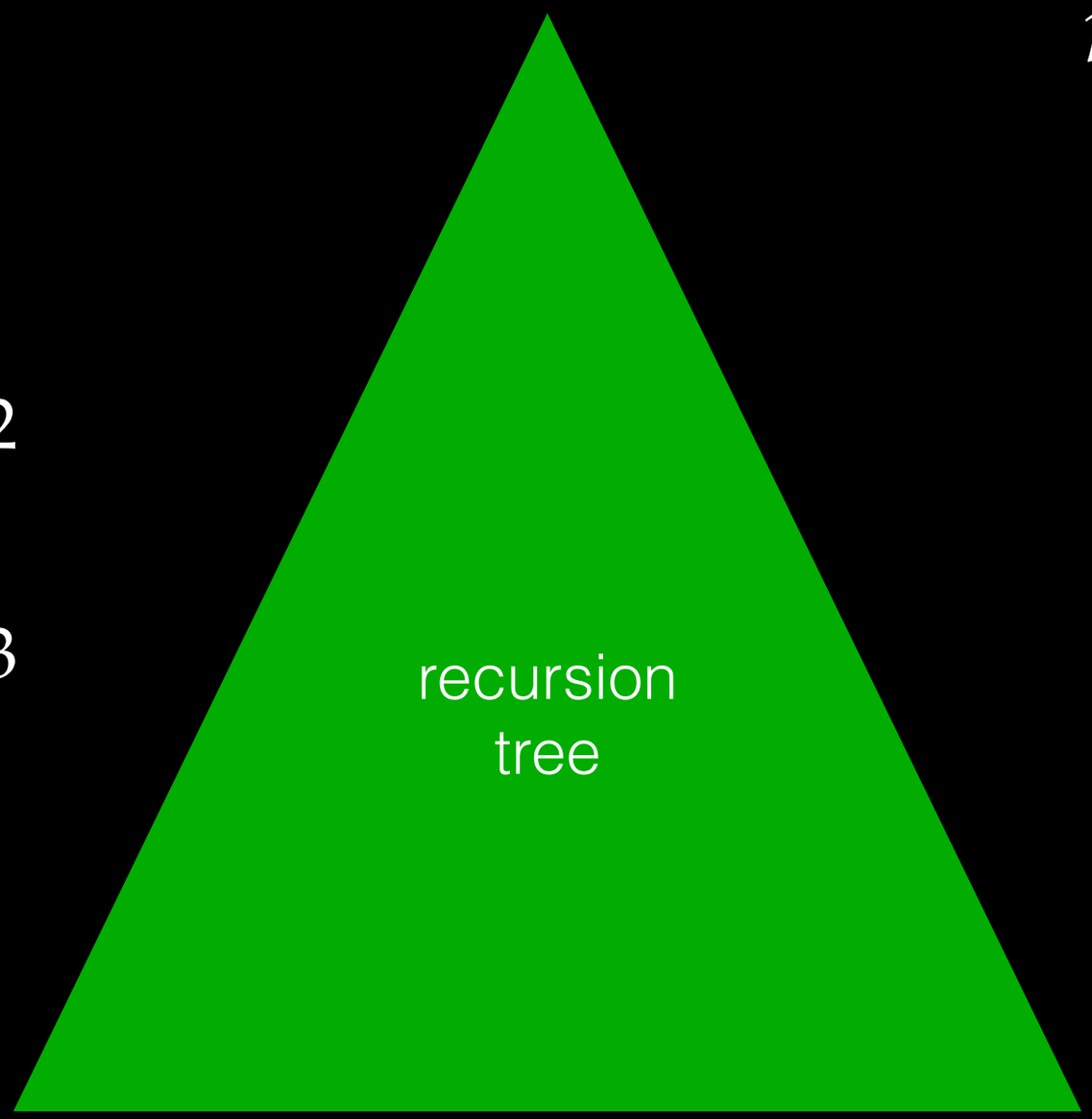
n

nt

nt^2

nt^3

$\vdots$



n

Expected average
points at distance $> cr$.

I/O complexity, sketch

Size of all subproblems

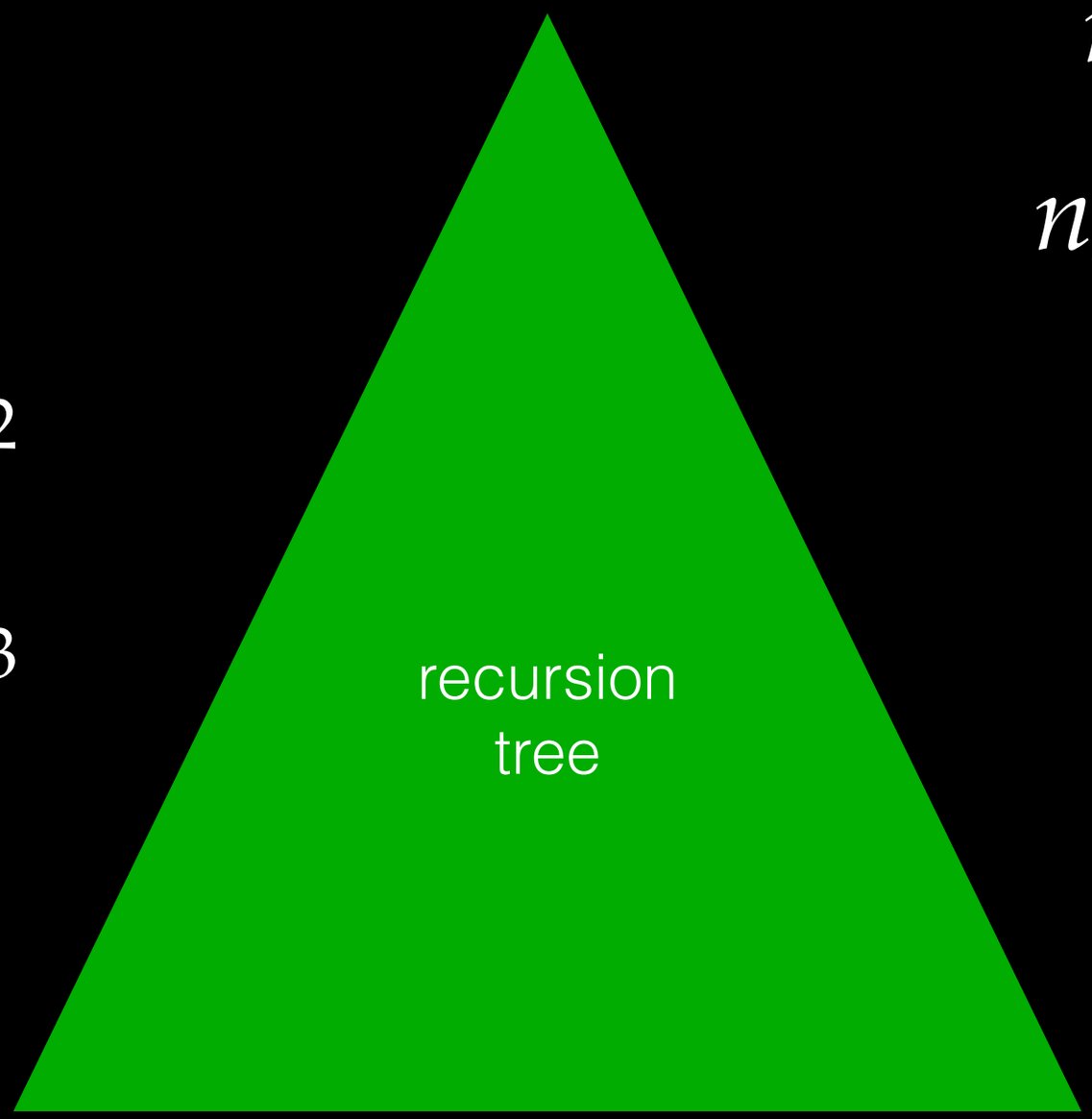
n

nt

nt^2

nt^3

$\vdots$



n

n/t^c

Expected average
points at distance $> cr$.

I/O complexity, sketch

Size of all subproblems

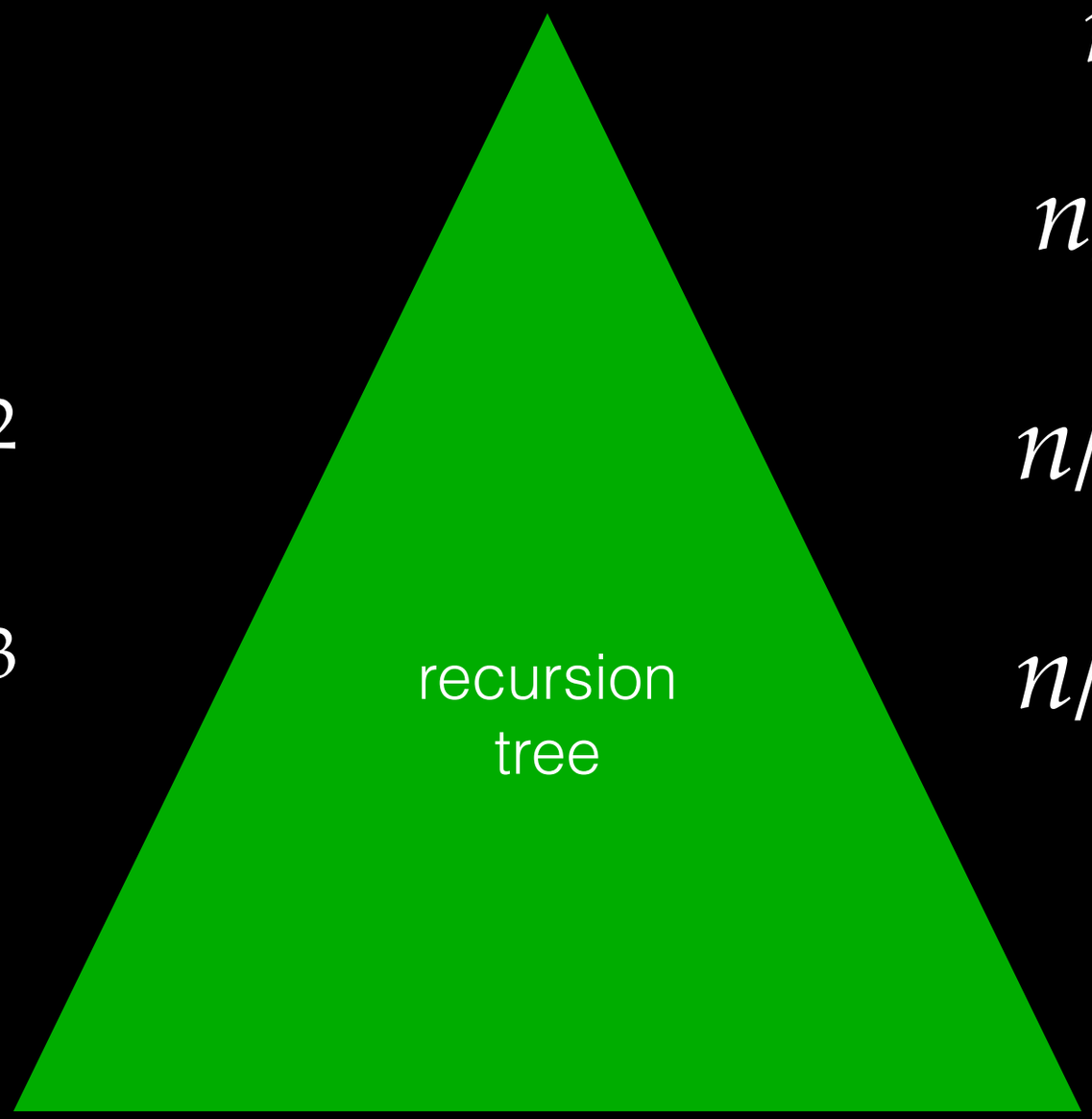
n

nt

nt^2

nt^3

$\vdots$



n

n/t^c

n/t^{2c}

n/t^{3c}

$\vdots$

Expected average
points at distance $> cr$.

I/O complexity, sketch

Size of all subproblems

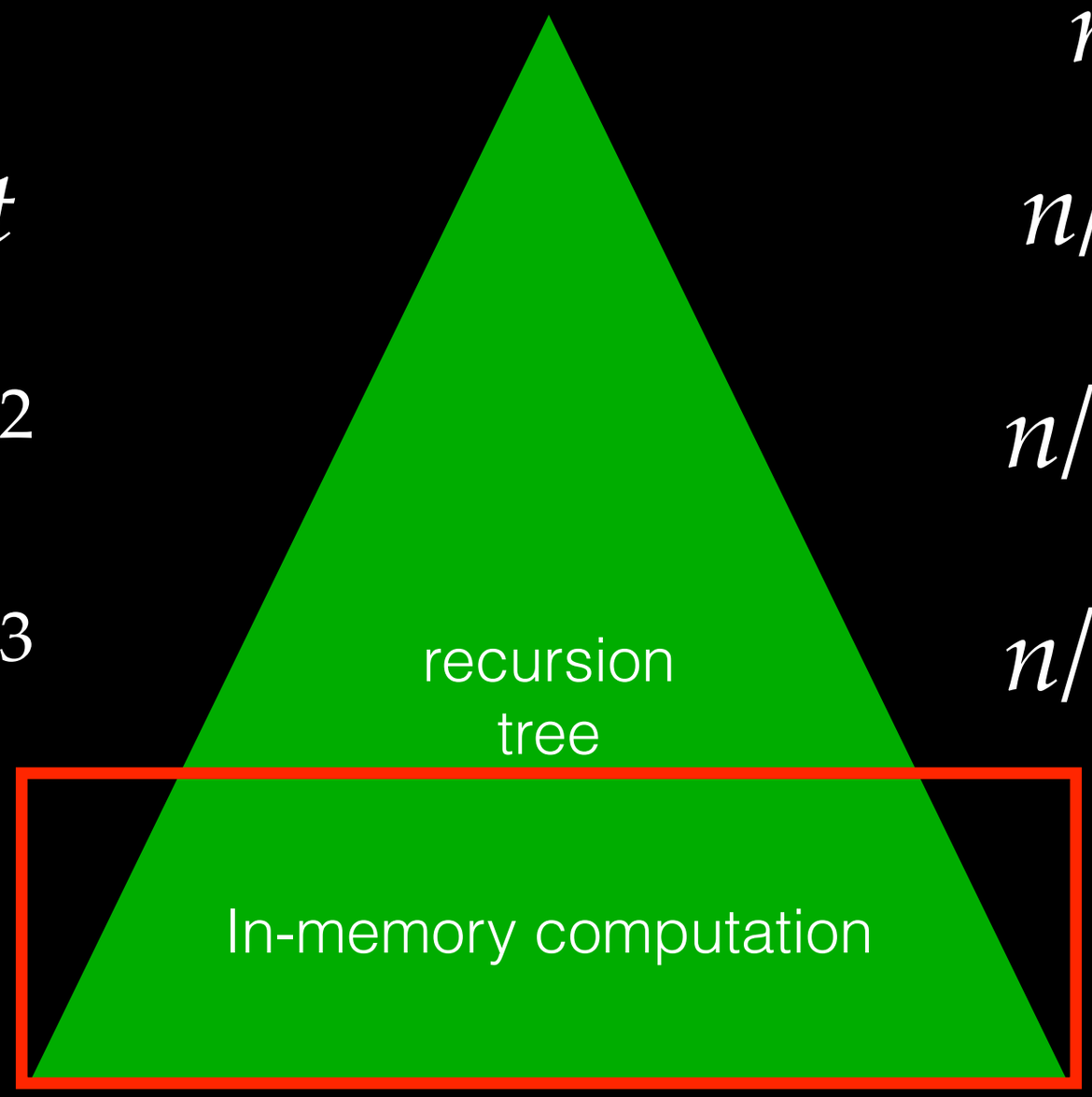
n

nt

nt^2

nt^3

⋮



n

n/t^c

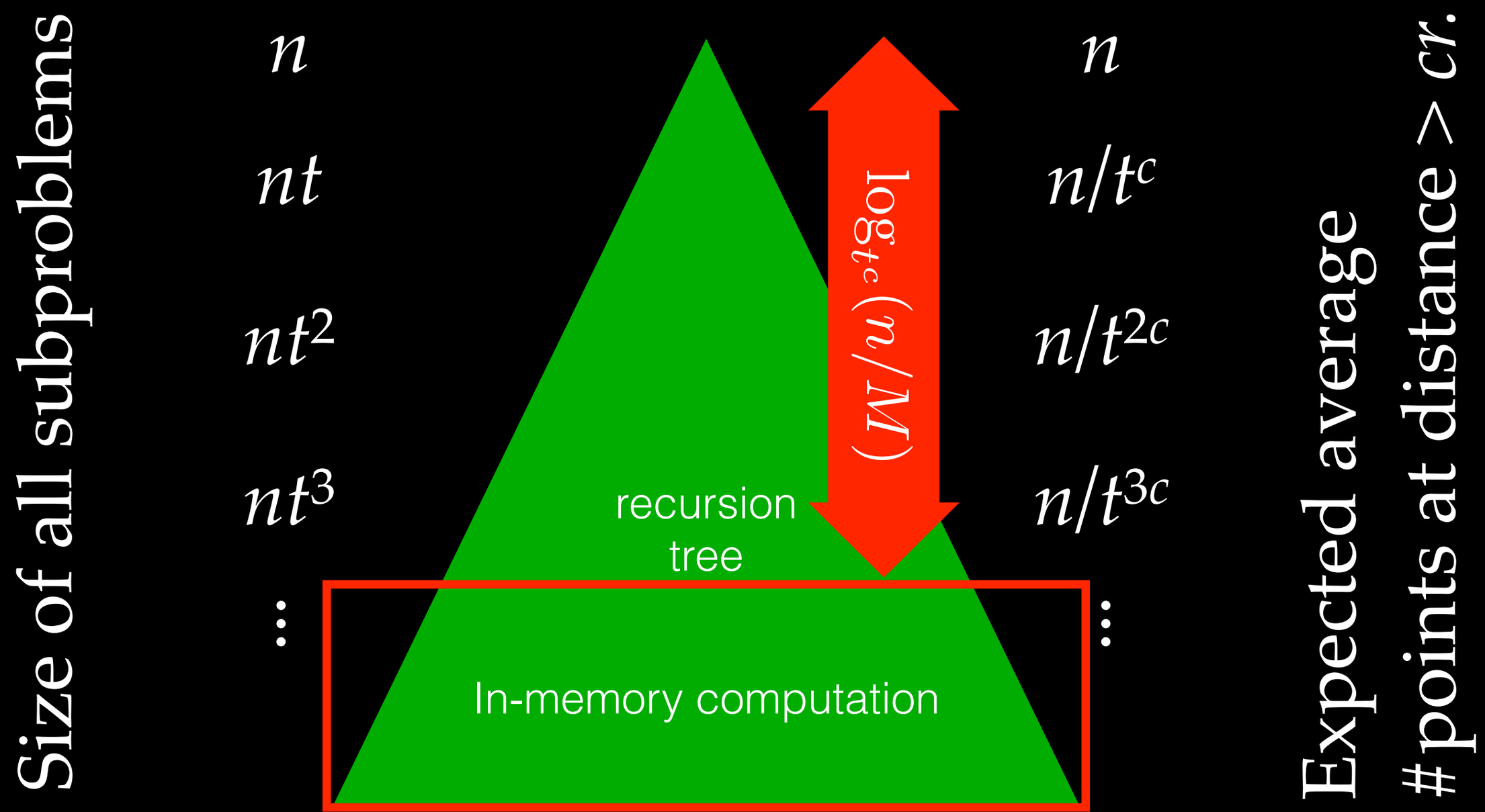
n/t^{2c}

n/t^{3c}

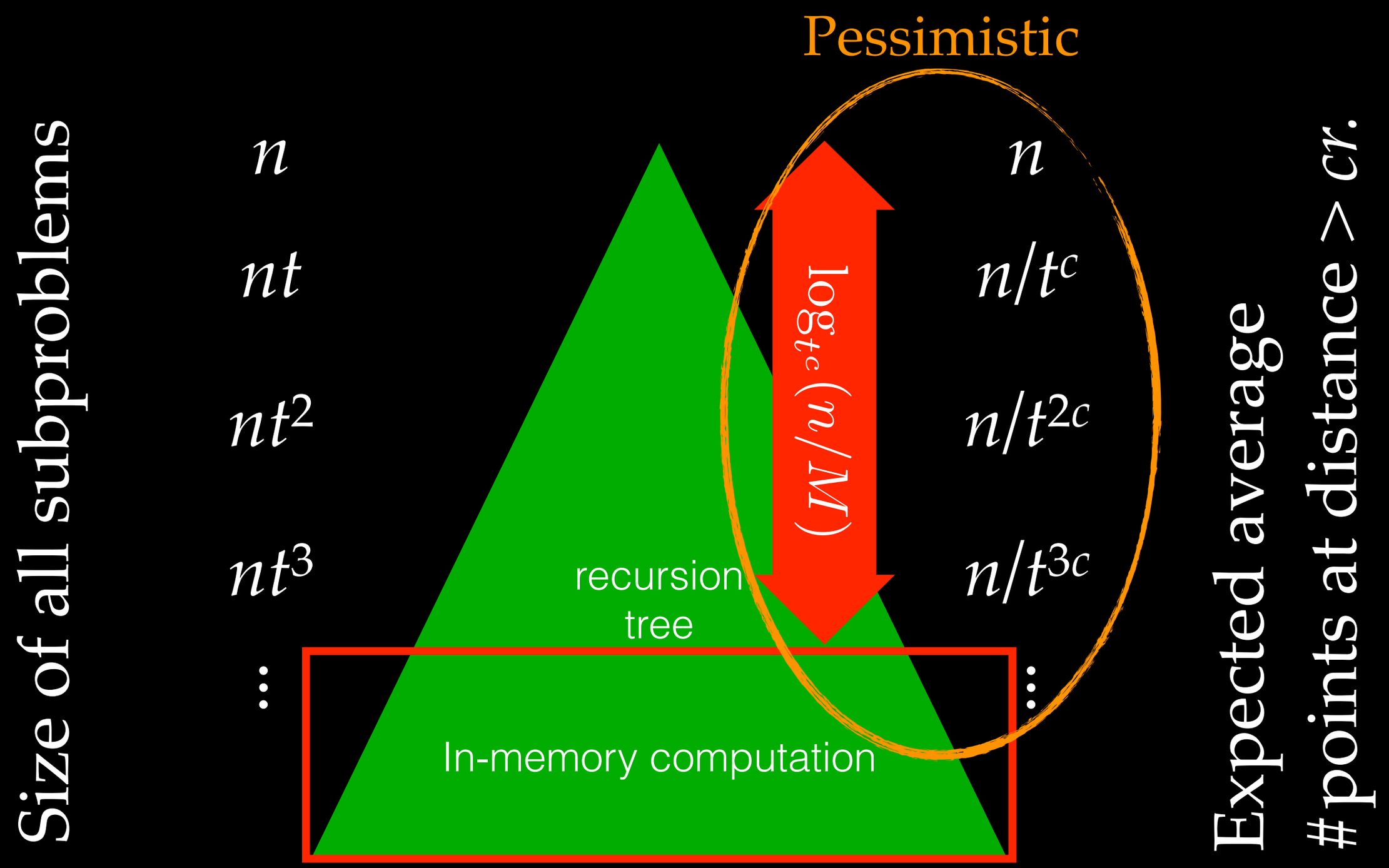
⋮

Expected average
points at distance $> cr$.

I/O complexity, sketch



I/O complexity, sketch



I/O complexity

- Simplified:

$$O\left(\left(\frac{n}{M}\right)^{1/c}\left(\frac{n}{B}+\frac{|Q \underset{\leq r}{\bowtie} S|}{MB}\right)\right) \text{ I/Os}$$

Assuming $|Q \bowtie_{cr} S|$ is not much larger than $|Q \bowtie_{r} S|$

I/O complexity

Cost of reading input

- Simplified:

$$O\left(\left(\frac{n}{M}\right)^{1/c}\left(\frac{n}{B}+\frac{|Q \underset{\leq r}{\bowtie} S|}{MB}\right)\right) \text{ I/Os}$$

Assuming $|Q \bowtie_{cr} S|$ is not much larger than $|Q \bowtie_{r} S|$

I/O complexity

Cost of reading input

Cost of generating output

- Simplified:

$$O\left(\left(\frac{n}{M}\right)^{1/c}\left(\frac{n}{B}+\frac{|Q \underset{\leq r}{\bowtie} S|}{MB}\right)\right) \text{ I/Os}$$

Assuming $|Q \bowtie_{cr} S|$ is not much larger than $|Q \bowtie_{r} S|$

I/O complexity

Overhead

Cost of reading input

Cost of generating output

- Simplified:

$$O\left(\left(\frac{n}{M}\right)^{1/c}\left(\frac{n}{B}+\frac{|Q \underset{\leq r}{\bowtie} S|}{MB}\right)\right) \text{ I/Os}$$

Assuming $|Q \bowtie_{cr} S|$ is not much larger than $|Q \bowtie_{r} S|$

I/O complexity

- Simplified:

$$O\left(\left(\frac{n}{M}\right)^{1/c}\left(\frac{n}{B}+\frac{|Q \underset{\leq r}{\bowtie} S|}{MB}\right)\right) \text{ I/Os}$$

- In general:

$$\tilde{O}\left(\left(\frac{n}{M}\right)^{1/c}\left(\frac{n}{B}+\frac{|Q \underset{\leq r}{\bowtie} S|}{MB}\right)+\frac{|Q \underset{\leq cr}{\bowtie} S|}{MB}\right) \text{ I/Os}$$

Candidate set generation

- CoveringLSH: Achieving 100% recall

Efficient Exact Set-Similarity Joins

Arvind Arasu
Microsoft Research
One Microsoft Way
Redmond, WA 98052
arvinda@microsoft.com

Venkatesh Ganti
Microsoft Research
One Microsoft Way
Redmond, WA 98052
vganti@microsoft.com

Raghav Kaushik
Microsoft Research
One Microsoft Way
Redmond, WA 98052
skaushi@microsoft.com

ABSTRACT

Given two input collections of sets, a *set-similarity join* (SSJoin) identifies all pairs of sets, one from each collection, that have high similarity. Recent work has identified SSJoin as a useful primitive operator in data cleaning. In this paper, we

City	*State*
Los Angeles	CA
Palo Alto	CA
San Diego	CA
Santa Barbara	CA

City	*State*
Los Angeles	California
San Diego	California
Santa Barbara	California

rithms [6, 22] do not provide any such guarantee. Previous work [8, 15] has proposed probabilistic algorithms based on the idea of *locality-sensitive hashing (LSH)* [13] that have similar guarantees. However, these algorithms are *approximate* since they can miss some output set pairs; in contrast, all of our algorithms are *exact*, and never produce a wrong output. We believe our algorithms are the first exact ones with such performance guarantees.

collection, for example, the same address could be encoded using different strings in different records in the collection. Multiple representations arise due to a variety of reasons such as misspellings caused by typographic errors and different formatting conventions

(*SSJoin*) as a powerful primitive for supporting (string-)similarity joins involving many common similarity functions. In other words, we can express these similarity joins using SSJoin as an sub-operator, and thereby avoid separate implementations for them from scratch.

Informally, SSJoin is defined as follows: Given two input collec-

Approximation

<u>Two kinds of approximation:</u>

- Approximate distances
- Allow false positives and negatives (precision and recall below 100%)

Approximation

<u>Two kinds of approximation:</u>

- Approximate distances

Inherent to LSH

- Allow false positives and negatives (precision and recall below 100%)

Approximation

<u>Two kinds of approximation:</u>

- Approximate distances

Inherent to LSH

- Allow false positives and negatives (precision and recall below 100%)

Total recall possible!

[Arasu, Ganti & Kaushik '06]

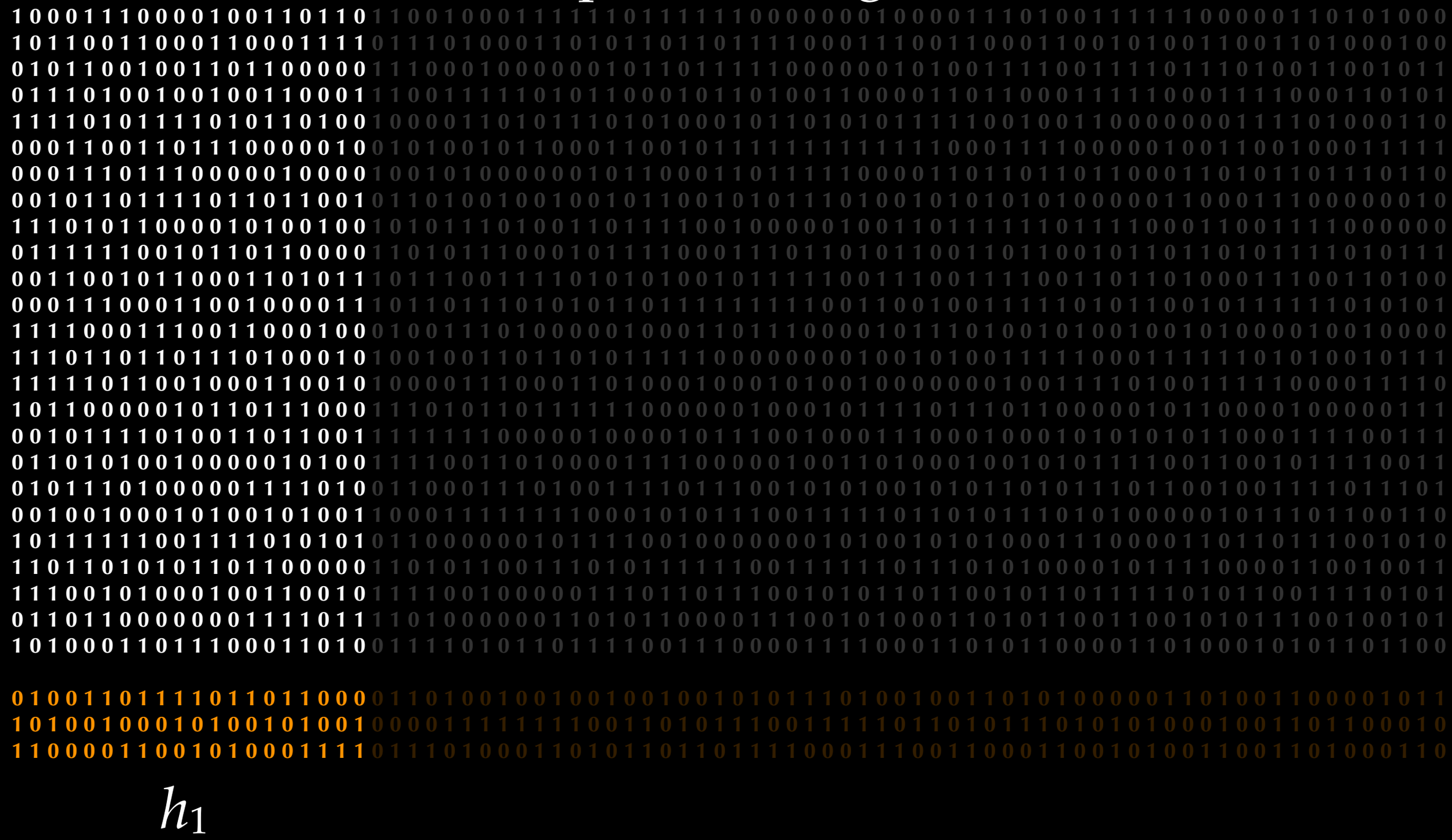
Basic correlated LSH

partitioning

[Arasu, Ganti & Kaushik '06]

Basic correlated LSH

partitioning

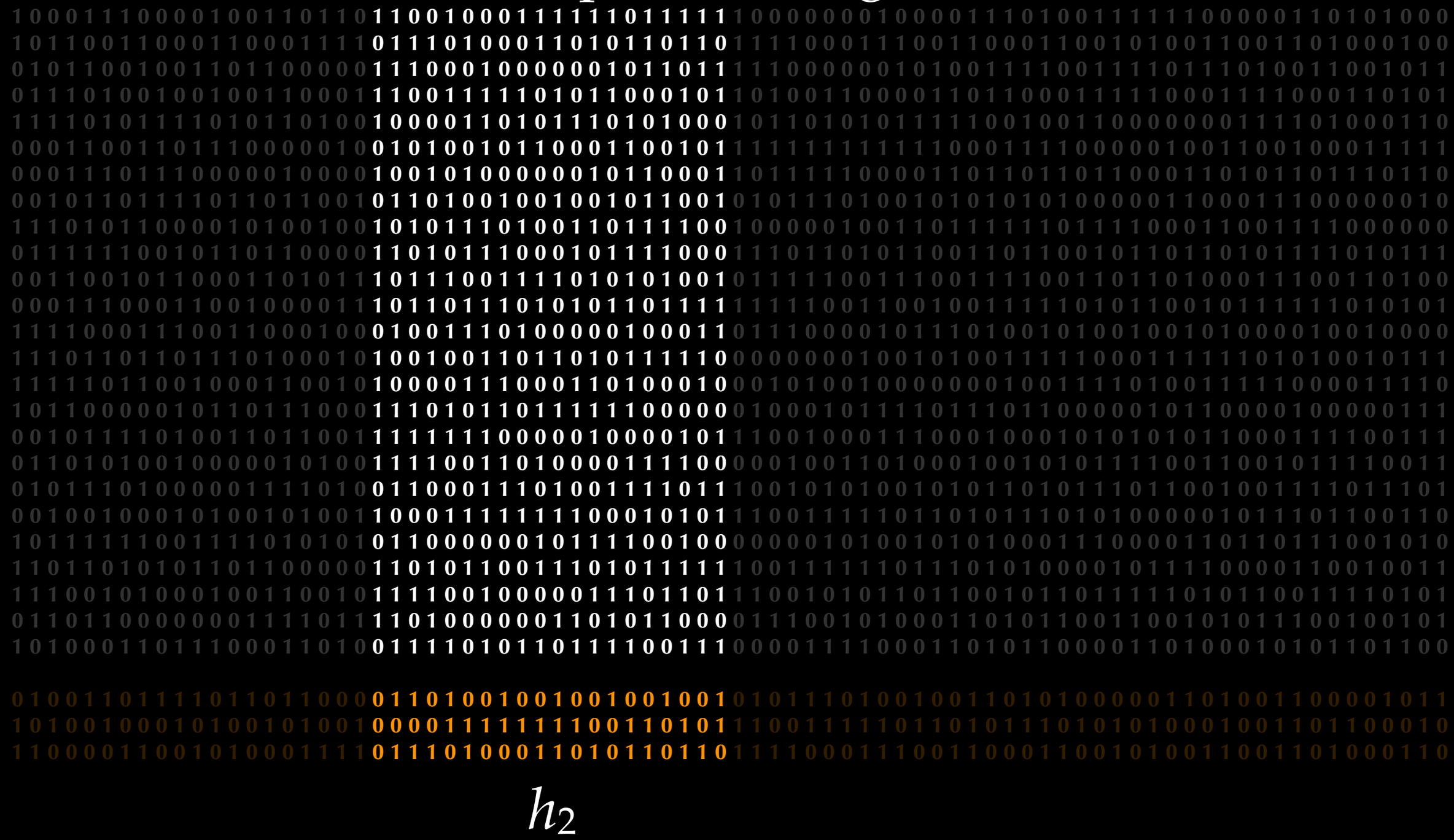


h_1

[Arasu, Ganti & Kaushik '06]

Basic correlated LSH

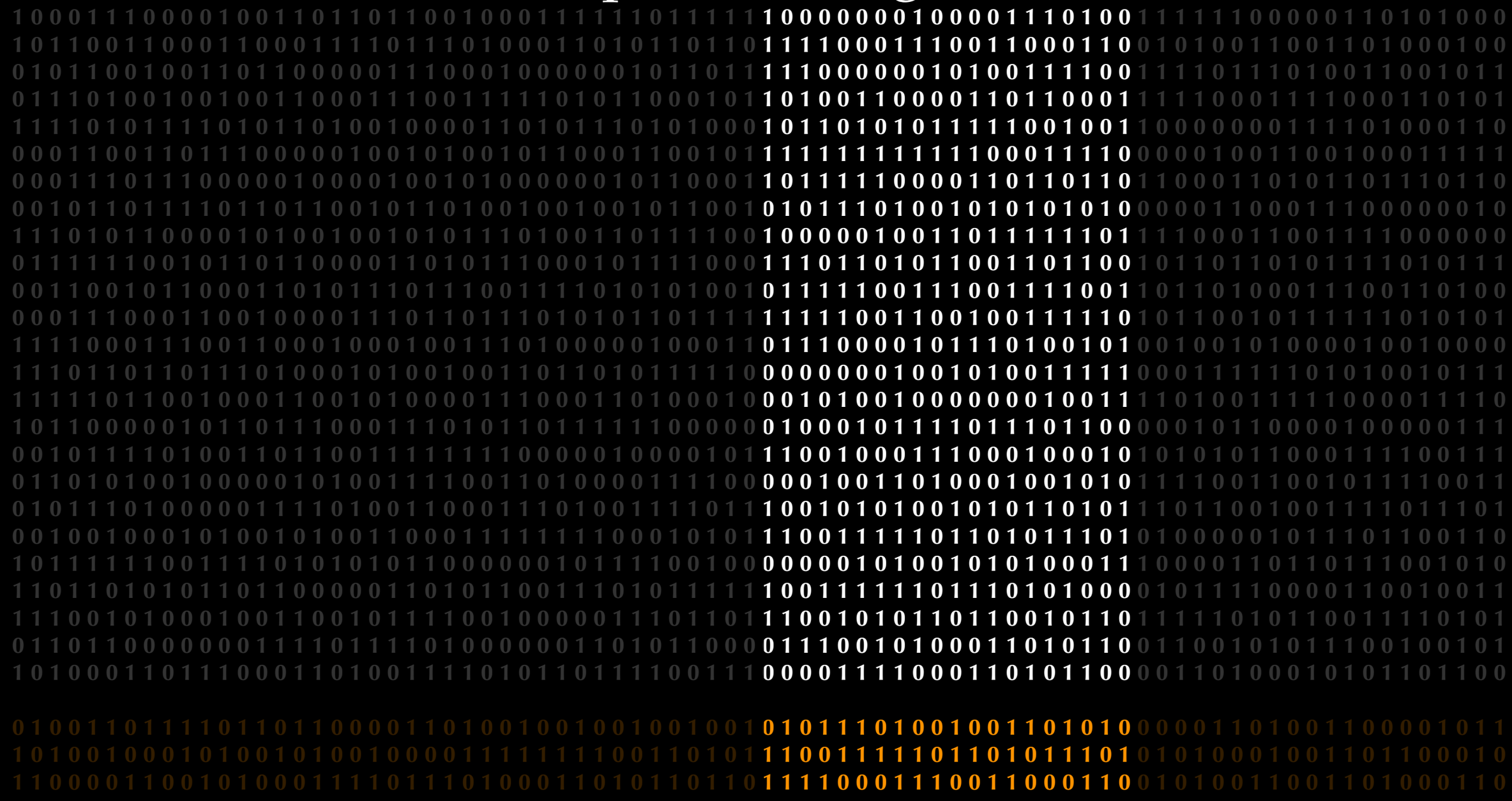
partitioning



[Arasu, Ganti & Kaushik '06]

Basic correlated LSH

partitioning

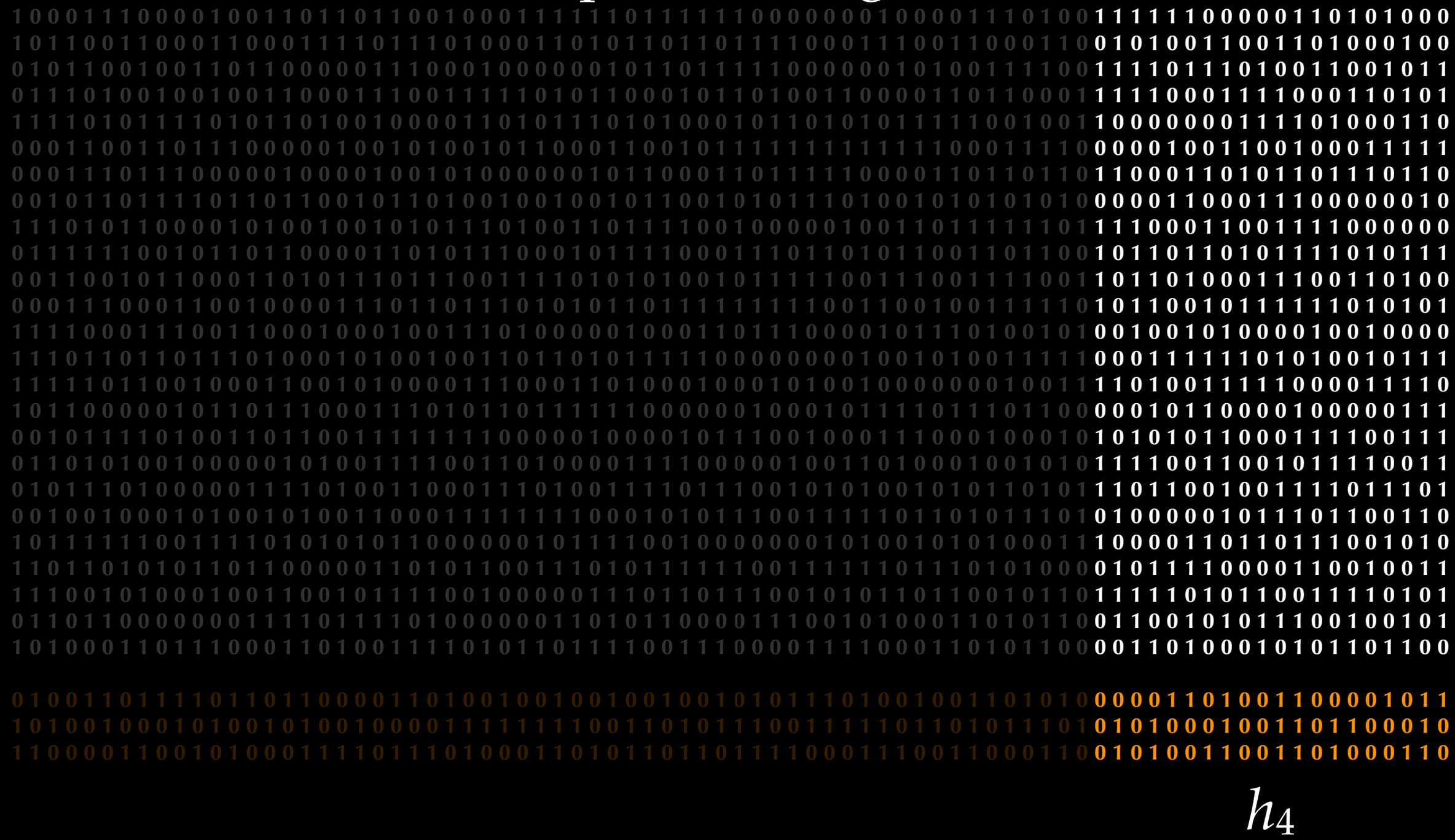


h_3

[Arasu, Ganti & Kaushik ’06]

Basic correlated LSH

partitioning



[Arasu, Ganti & Kaushik '06]

Basic correlated LSH

partitioning

For Hamming distance ≤ 3, a collision is guaranteed!

[Arasu, Ganti & Kaushik '06]

Basic correlated LSH

partitioning

For Hamming distance ≤ 3, a collision is guaranteed!

[Arasu et al. '06]:
To bound probability of collision for distance > 3 *randomly permute* the dimensions

[Arasu, Ganti & Kaushik ’06]

Basic correlated LSH

enumeration

h_{12}

[Arasu, Ganti & Kaushik ’06]

Basic correlated LSH

enumeration

h_{13}

[Arasu, Ganti & Kaushik '06]

Basic correlated LSH

enumeration

```
1000111000010011011011001000111111011111100000000100001110100111111100000110101000
1011001100011000111101110100011010110110111100011100110001100101001100110100010 0
0101100100110110000011100010000000101101111100000001010011111001111101100110010 11
0111010010010011000111001111101011000101101001100001101100011111000111100011010 1
1111010111101011010010000110101110101000101101010111111001001100000001111010001 10
0001100110111000000100101001011000110010111111111111110001111100000100110010001 1111
0001110111000000100001001010000000101100011011111100001101101101100011010110111 0110
0010110111110110110010101101001001001011001010111101001010101010100000110001110 00000010
1110101100001010010010110111010110110010000010011011111110111000011001111000000
0111111001011011000011101011100011011010110011011001011101110101110
0011001011000110101110110101001011111001110011111001101101000111001101001
0001110001100100001110110110110110110110110110110110110110110110110110110110110
1110110110111010001010100100110111100100000100110111111101111000111100111110101
1111110110011000100110000100110111111000000000100101001111111000011110
1011000010110111000111010111000111000111000111000111000111000111000111000111000
0010110110001101011101110110110110110110110110110110110110110110110110110110110
0001110001100100001110110110110110110110110110110110110110110110110110110110110
1110011000100110001000100100100100100100100100100100100100100100100100100100100
0111011011110110110000110100100100100100100100100100100100100100100100100100100
1010010001010010100110100100100100100100100100100100100100100100100100100100100
1100001100101000111101110100011101101101111000111001100011010010110110110110110
```

h_{14}

[Arasu, Ganti & Kaushik '06]

Basic correlated LSH

enumeration

For Hamming distance ≤ 2, a collision is guaranteed in

$h_{12}, h_{13}, h_{14}, h_{23}, h_{24}, h_{34}$

h_{14}

Result in LSH framework

- The bit sampling LSH achieves:

$$\Pr[h(q) = h(x) \mid ||x - q|| = cr] \leq \Pr[h(q) = h(x) \mid ||x - q|| = r]^c$$

Result in LSH framework

- The bit sampling LSH achieves:

$$\Pr[h(q) = h(x) \mid ||x - q|| = cr] \leq \Pr[h(q) = h(x) \mid ||x - q|| = r]^c$$

- Bound on *PartEnum* construction with partitioning + enumeration + permutation:

$$\Pr[h(q) = h(x) \mid ||x - q|| = cr] \leq \Pr[h(q) = h(x) \mid ||x - q|| = r]^{0.36\,c}$$

Result in LSH framework

- The bit sampling LSH achieves:

$$\Pr[h(q) = h(x) \mid ||x - q|| = cr] \leq \Pr[h(q) = h(x) \mid ||x - q|| = r]^{c}$$

- Bound on *PartEnum* construction with partitioning + enumeration + permutation:

Need ≈ 3× larger c

$$\Pr[h(q) = h(x) \mid ||x - q|| = cr] \leq \Pr[h(q) = h(x) \mid ||x - q|| = r]^{0.36\,c}$$

Result in LSH framework

- The bit sampling LSH achieves:

$$\Pr[h(q) = h(x) \mid ||x - q|| = cr] \leq \Pr[h(q) = h(x) \mid ||x - q|| = r]^{c}$$

- Bound on *PartEnum* construction with partitioning + enumeration + permutation:

Need ≈ 3× larger c

$$\Pr[h(q) = h(x) \mid ||x - q|| = cr] \leq \Pr[h(q) = h(x) \mid ||x - q|| = r]^{0.36\,c}$$

Probabilistic argument suggests that it is possible to do much better. But how?

Mathematical question

- A (p,r)-***covering*** matrix of dim. d satisfies:
  - Has $(1-p)d$ zeros and pd ones in each row;
  - for every set of r columns there exists a row with 0s in all of them.

Mathematical question

- A (p,r)-***covering*** matrix of dim. d satisfies:
  - Has $(1-p)d$ zeros and pd ones in each row;
  - for every set of r columns there exists a row with 0s in all of them.

Small collision probability

Mathematical question

- A (p,r)-***covering*** matrix of dim. d satisfies:
  - Has $(1-p)d$ zeros and pd ones in each row;
  - for every set of r columns there exists a row with 0s in all of them.

Small collision probability

Collision guarantee

Mathematical question

- A (p,r)-***covering*** matrix of dim. d satisfies:
  - Has $(1-p)d$ zeros and pd ones in each row;
  - for every set of r columns there exists a row with 0s in all of them.

Small collision probability

Collision guarantee

- <u>Question</u>:
  How few rows can such a matrix have?

Mathematical question

- A (p,r)-***covering*** matrix of dim. d satisfies:
  - Has $(1-p)d$ zeros and pd ones in each row;
  - for every set of r columns there exists a row with 0s in all of them.

Small collision probability

Collision guarantee

- <u>Question</u>:
  How few rows can such a matrix have?

Number of hash functions

Mathematical question

- A (p,r)-***covering*** matrix of dim. d satisfies:
  - Has $(1-p)d$ zeros and pd ones in each row;
  - for every set of r columns there exists a row with 0s in all of them.

Small collision probability

Collision guarantee

- <u>Question</u>:
  How few rows can such a matrix have?

Number of hash functions

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$p = 4/7$
$r = 2$

Mathematical question

- A (p,r)-***covering*** matrix of dim. d satisfies:
  - Has $(1-p)d$ zeros and pd ones in each row;
  - for every set of r columns there exists a row with 0s in all of them.

Small collision probability

Collision guarantee

- <u>Question</u>:
  How few rows can such a matrix have?

Number of hash functions

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$p = 4/7$
$r = 2$

Mathematical question

- A (p,r)-***covering*** matrix of dim. d satisfies:
  - Has $(1-p)d$ zeros and pd ones in each row;
  - for every set of r columns there exists a row with 0s in all of them.

Small collision probability

Collision guarantee

- <u>Question</u>:
  How few rows can such a matrix have?

Number of hash functions

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$p = 4/7$
$r = 2$

Mathematical question

- A (p,r)-***covering*** matrix of dim. d satisfies:
  - Has $(1-p)d$ zeros and pd ones in each row;
  - for every set of r columns there exists a row with 0s in all of them.

Small collision probability

Collision guarantee

- <u>Question</u>:
  How few rows can such a matrix have?

Number of hash functions

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$p = 4/7$
$r = 2$

Mathematical question

- A (p,r)-***covering*** matrix of dim. d satisfies:
  - Has $(1-p)d$ zeros and pd ones in each row;
  - for every set of r columns there exists a row with 0s in all of them.
- <u>Question</u>:
  How few rows can such a matrix have?

Related to covering problems in extremal combinatorics, but good constructions have been known only for $p=O(1/d)$

[P., SODA '16]

CoveringLSH

- <u>Next</u>: Answer for $d = 2^{r+1}$-1, $pd = 2^r+1 \approx d/2$

[P., SODA '16]

CoveringLSH

- <u>Next</u>: Answer for $d = 2^{r+1}$-1, $pd = 2^r+1 \approx d/2$

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} \mathbf{001} & \mathbf{010} & \mathbf{011} & \mathbf{100} & \mathbf{101} & \mathbf{110} & \mathbf{111} \end{array} \\
\text{Index (binary)} \begin{array}{c} \mathbf{001} \\ \mathbf{010} \\ \mathbf{011} \\ \mathbf{100} \\ \mathbf{101} \\ \mathbf{110} \\ \mathbf{111} \end{array} & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}
\end{array}
$$

[P., SODA '16]

CoveringLSH

- <u>Next</u>: Answer for $d = 2^{r+1}-1$, $pd = 2^r+1 \approx d/2$

Idea: Entry is dot product (mod 2) of row / column ID vectors (Hadamard code)

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} \mathtt{001} & \mathtt{010} & \mathtt{011} & \mathtt{100} & \mathtt{101} & \mathtt{110} & \mathtt{111} \end{array} \\
\text{Index (binary)} \begin{array}{c} \mathtt{001} \\ \mathtt{010} \\ \mathtt{011} \\ \mathtt{100} \\ \mathtt{101} \\ \mathtt{110} \\ \mathtt{111} \end{array} & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}
\end{array}
$$

[P., SODA '16]

CoveringLSH

- <u>Next</u>: Answer for $d = 2^{r+1}-1$, $pd = 2^r+1 \approx d/2$

Idea: Entry is dot product (mod 2) of row / column ID vectors (Hadamard code)

Index (binary)	001	010	011	100	101	110	111
001	1	0	1	0	1	0	1
010	0	1	1	0	0	1	1
011	1	$\boxed{1}$	0	0	1	1	0
100	0	0	0	1	1	1	1
101	1	0	1	1	0	1	0
110	0	1	1	1	1	0	0
111	1	1	0	1	0	0	1

$(0,1,1) \cdot (0,1,0)=1$

[P., SODA '16]

CoveringLSH

- <u>Next</u>: Answer for $d = 2^{r+1}-1$, $pd = 2^r+1 \approx d/2$

Idea: Entry is dot product (mod 2) of row / column ID vectors (Hadamard code)

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} \mathtt{001} & \mathtt{010} & \mathtt{011} & \mathtt{100} & \mathtt{101} & \mathtt{110} & \mathtt{111} \end{array} \\
\text{Index (binary)} \begin{array}{c} \mathtt{001} \\ \mathtt{010} \\ \mathtt{011} \\ \mathtt{100} \\ \mathtt{101} \\ \mathtt{110} \\ \mathtt{111} \end{array} &
\begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & \boxed{0} & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}
\end{array}
$$

$(0,1,1) \cdot (0,1,1) = 0$

[P., SODA '16]

CoveringLSH

- <u>Next</u>: Answer for $d = 2^{r+1}-1$, $pd = 2^r+1 \approx d/2$

Idea: Entry is dot product (mod 2) of row / column ID vectors (Hadamard code)

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} \mathtt{001} & \mathtt{010} & \mathtt{011} & \mathtt{100} & \mathtt{101} & \mathtt{110} & \mathtt{111} \end{array} \\
\text{Index (binary)} \begin{array}{c} \mathtt{001} \\ \mathtt{010} \\ \mathtt{011} \\ \mathtt{100} \\ \mathtt{101} \\ \mathtt{110} \\ \mathtt{111} \end{array} & \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}
\end{array}
$$

[P., SODA '16]

CoveringLSH

- <u>Next</u>: Answer for $d = 2^{r+1}-1$, $pd = 2^r+1 \approx d/2$

Idea: Entry is dot product (mod 2) of row / column ID vectors (Hadamard code)

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} \mathtt{001} & \mathtt{010} & \mathtt{011} & \mathtt{100} & \mathtt{101} & \mathtt{110} & \mathtt{111} \end{array} \\
\text{Index (binary)} \begin{array}{c} \mathtt{001} \\ \mathtt{010} \\ \mathtt{011} \\ \mathtt{100} \\ \mathtt{101} \\ \mathtt{110} \\ \mathtt{111} \end{array} &
\begin{pmatrix}
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 1
\end{pmatrix}
\end{array}
$$

[P., SODA '16]

CoveringLSH

- <u>Next</u>: Answer for $d = 2^{r+1}-1$, $pd = 2^r+1 \approx d/2$

Idea: Entry is dot product (mod 2) of row / column ID vectors (Hadamard code)

Index (binary)	001	010	011	100	101	110	111
001	1	0	1	0	1	0	1
010	0	1	1	0	0	1	1
011	1	1	0	0	1	1	0
100	0	0	0	1	1	1	1
101	1	0	1	1	0	1	0
110	0	1	1	1	1	0	0
111	1	1	0	1	0	0	1

Lemma:
For every set of r vectors in $\{0,1\}^{r+1}$ there exists a nonzero vector that is orthogonal to all

[P., SODA '16]

CoveringLSH

- <u>Next</u>: Answer for $d = 2^{r+1}-1$, $pd = 2^r+1 \approx d/2$

Idea: Entry is dot product (mod 2) of row / column ID vectors (Hadamard code)

Index (binary)	001	010	011	100	101	110	111
001	1	0	1	0	1	0	1
010	0	1	1	0	0	1	1
011	1	1	0	0	1	1	0
100	0	0	0	1	1	1	1
101	1	0	1	1	0	1	0
110	0	1	1	1	1	0	0
111	1	1	0	1	0	0	1

Lemma:
For every set of r vectors in $\{0,1\}^{r+1}$ there exists a nonzero vector that is orthogonal to all

Optimality?

- Could there be a smaller (4/7,2)-covering?
- We need to "cover" $\binom{7}{2} = 21$ sets of two columns

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Optimality?

- Could there be a smaller (4/7,2)-covering?
- We need to "cover" $\binom{7}{2} = 21$ sets of two columns
- Each vector can cover $\binom{3}{2} = 3$ sets of two columns

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Optimality?

- Could there be a smaller (4/7,2)-covering?
- We need to "cover" $\binom{7}{2} = 21$ sets of two columns
- Each vector can cover $\binom{3}{2} = 3$ sets of two columns

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

7 vectors is optimal!

Optimality?

- Could there be a smaller (4/7,2)-covering?
- We need to "cover" $\binom{7}{2} = 21$ sets of two columns
- Each vector can cover $\binom{3}{2} = 3$ sets of two columns

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

7 vectors is optimal!

For r>2: Within factor 2 of optimal

Optimality?

- Could there be a smaller $(1/2,r)$-covering?
- We need to cover $\binom{d}{r}$ sets of r columns

$$2^{r+1}-1$$

$$2^{r+1}-1$$

$$\begin{pmatrix}
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\
0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0\\
1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0\\
0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0\\
1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0\\
0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1\\
1&0&1&0&1&0&1&1&0&1&0&1&0&1&0&0&1&0&1&0&1&0&1&0&1&1&0&1&0&1&0&1\\
0&1&1&0&0&1&1&1&1&0&0&1&1&0&0&0&0&1&1&0&0&1&1&0&0&0&0&1&1&0&0&1\\
1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&1\\
0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0\\
1&0&1&1&0&1&0&1&0&1&0&0&1&0&1&0&1&0&1&1&0&1&0&1&0&1&0&0&1&0&1&0\\
0&1&1&1&1&0&0&1&1&0&0&0&0&1&1&0&0&1&1&0&0&1&1&0&0&0&0&1&1&0&0&1\\
1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&1&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\
0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1\\
0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1\\
1&0&1&1&0&1&0&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&0&1&0&1&1&0&1&0&1\\
0&1&1&1&1&0&0&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&0&0&1&1&1&1&0&0&1\\
1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1&1
\end{pmatrix}$$

Optimality?

- Could there be a smaller $(1/2,r)$-covering?
- We need to cover $\binom{d}{r}$ sets of r columns
- Each vector can cover $\binom{d/2}{r}$ sets of r columns

$$2^{r+1}-1$$

$$2^{r+1}-1$$

$$\begin{pmatrix}
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\
0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\
1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0\\
0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0\\
1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0\\
0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0\\
1&0&1&0&1&0&1&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&0&1&0&1&0&1&0\\
0&1&1&0&0&1&1&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&0&0&1&1&0&0&1&1\\
1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&0\\
0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0\\
1&0&1&1&0&1&0&1&0&1&0&0&1&0&1&0&1&0&1&1&0&1&0&1&0&0&1&0&1&0&1&0\\
0&1&1&1&1&0&0&1&1&0&0&0&0&1&1&0&0&1&1&0&0&1&1&0&0&0&0&1&1&0&0&1\\
1&1&0&1&0&0&1&1&0&0&1&1&0&0&1&0&1&1&0&0&1&1&0&0&1&0&1&1&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\
0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1\\
0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0&1&1&1&1\\
1&0&1&1&0&1&0&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&0&1&0&1&1&0&1&0\\
0&1&1&1&1&0&0&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&0&0&1&1&1&1&0&0\\
1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1
\end{pmatrix}$$

Optimality?

- Could there be a smaller $(1/2,r)$-covering?
- We need to cover $\binom{d}{r}$ sets of r columns
- Each vector can cover $\binom{d/2}{r}$ sets of r columns

$$2^{r+1}-1$$

$$2^{r+1}-1$$

$$\begin{pmatrix}
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\
0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1\\
1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0\\
0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0\\
1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&0&1&0\\
0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0\\
1&0&1&0&1&0&1&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&0&1&0&1&0&1&0\\
0&1&1&0&0&1&1&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&0&0&1&1&0&0&1&1\\
1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&0&1&1&0&0&1&1&0\\
0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&0&0&0&1&1&1&1&0&0\\
1&0&1&1&0&1&0&1&0&1&0&0&1&0&1&1&0&1&0&1&0&1&0&1&0&0&1&0&1&0&1\\
0&1&1&1&1&0&0&1&1&0&0&0&0&1&1&1&1&0&0&1&1&0&0&0&0&1&1&0&0&1&1\\
1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1\\
0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1\\
0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&1&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0\\
1&0&1&1&0&1&0&0&1&0&1&1&0&1&0&1&1&0&1&0&0&1&0&1&0&1&0&1&0&1&0\\
0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&1&1&0&0&0&0&1&1&0&0&1&1&1&1&0&0\\
1&1&0&1&0&0&1&1&0&0&1&0&1&1&0&1&0&0&1&0&1&1&0&0&1&1&0&1&0&0&1
\end{pmatrix}$$

$$\binom{d}{r}\Big/\binom{d/2}{r} > 2^r$$

Within factor 2 of optimal!

Use in similarity search

- Combine with random permutation trick:
  Each bit sampled w. prob. 1/2 in each hash value.

Use in similarity search

- Combine with random permutation trick:
  Each bit sampled w. prob. 1/2 in each hash value.

- "Sweet spot" is for $r = \log(n)/c$:
  - $2^{r+1} = 2n^{1/c}$ hash functions
  - Collision probability $1/n$ at distance $cr = \log(n)$, so number of "far" collisions insignificant

Use in similarity search

- Combine with random permutation trick:
  Each bit sampled w. prob. 1/2 in each hash value.

- "Sweet spot" is for $r = \log(n)/c$:
  - $2^{r+1} = 2n^{1/c}$ hash functions
  - Collision probability $1/n$ at distance $cr = \log(n)$, so number of "far" collisions insignificant

- Matches bound of Indyk and Motwani.

Smaller radius?

- Can map vectors from $\{0,1\}^d$ to $\{0,1\}^{td}$, increasing all distances by an integer factor t.

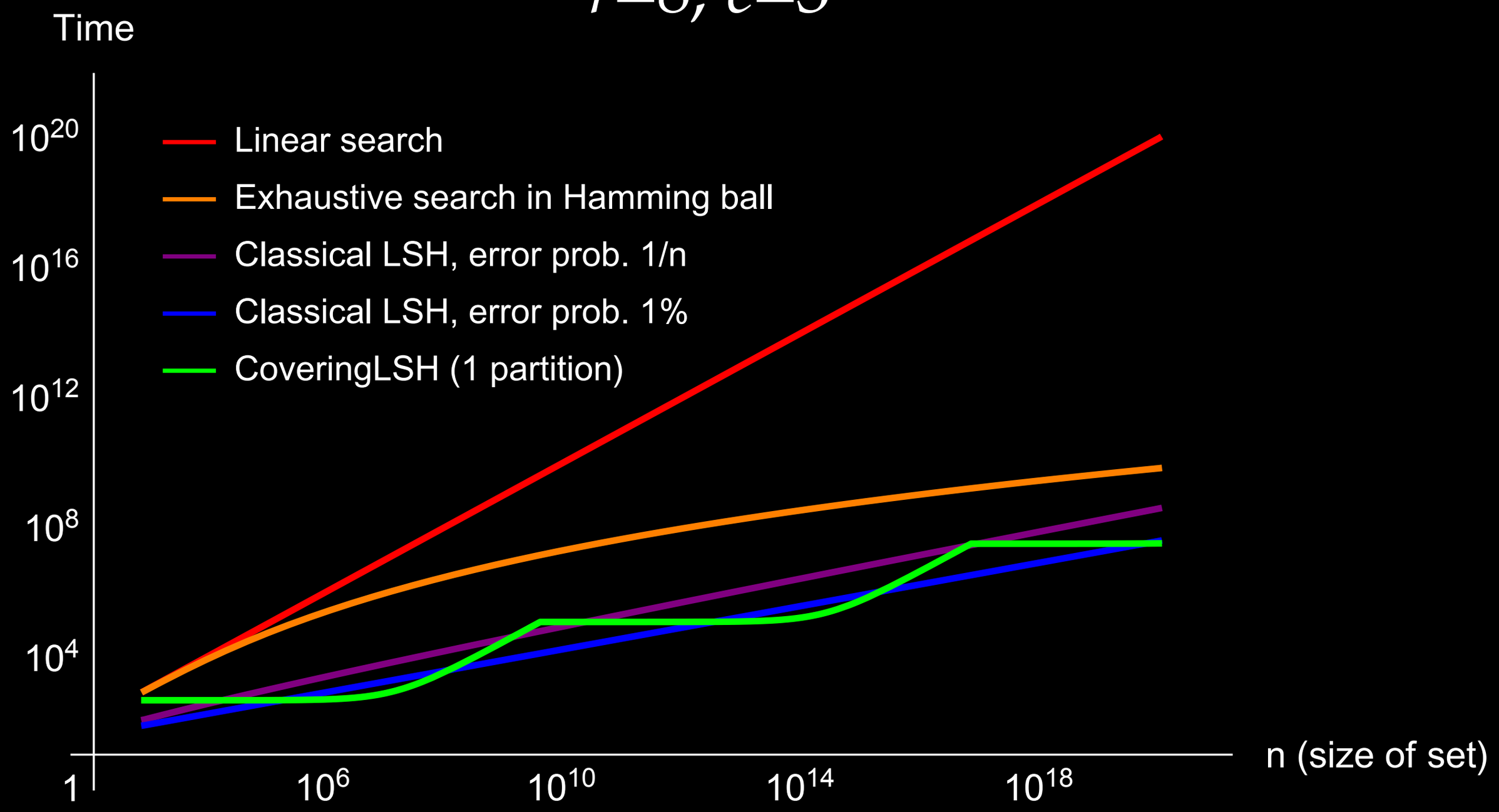
$$q^t = 110\mathbf{0}1011110\mathbf{0}1011110\mathbf{0}1011110\mathbf{0}101$$

$$x^t = 110\mathbf{1}1011110\mathbf{1}1011110\mathbf{1}1011110\mathbf{1}101$$

- Try to "hit" sweet spot $tr = \log(n)/c$
  - Details: arXiv:1507.03225 [cs.DS]

Example

$r=8,\ c=3$



Larger radius?

- Partition to reduce to $pd < d/2$ sampled bits:
  - Iterate over $1/(2p)$ parts

Part 1	Part 2	$\cdots$	Part $1/(2p)$

$$\begin{pmatrix}
1&0&1&0&1&0&1&0&0&0&0&0&0&0\\
0&1&1&0&0&1&1&0&0&0&0&0&0&0\\
1&1&0&0&1&1&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&1&0&0&0&0&0&0&0\\
1&0&1&1&0&1&0&0&0&0&0&0&0&0\\
0&1&1&1&1&0&0&0&0&0&0&0&0&0\\
1&1&0&1&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&1&0&1&0&1\\
0&0&0&0&0&0&0&0&1&1&0&0&1&1\\
0&0&0&0&0&0&0&1&1&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&1&0&1&1&0&1&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0\\
0&0&0&0&0&0&0&1&1&0&1&0&0&1
\end{pmatrix}$$

Larger radius?

- Partition to reduce to $pd < d/2$ sampled bits:
  - Iterate over $1/(2p)$ parts

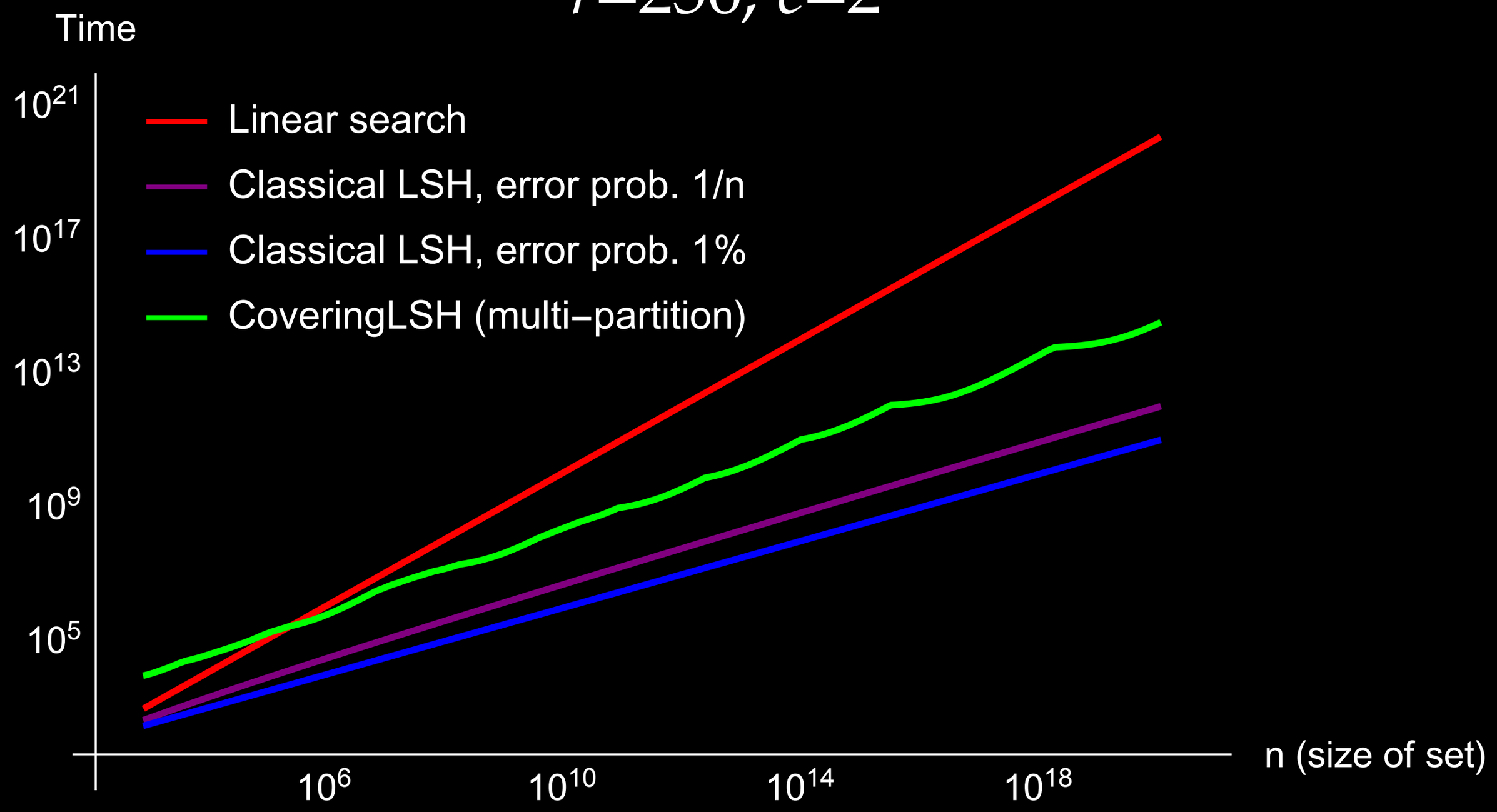
Part 1	Part 2	$\cdots$	Part $1/(2p)$

$$\begin{pmatrix}
1&0&1&0&1&0&1&0&0&0&0&0&0&0\\
0&1&1&0&0&1&1&0&0&0&0&0&0&0\\
1&1&0&0&1&1&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&1&0&0&0&0&0&0&0\\
1&0&1&1&0&1&0&0&0&0&0&0&0&0\\
0&1&1&1&1&0&0&0&0&0&0&0&0&0\\
1&1&0&1&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&1&0&1&0&1\\
0&0&0&0&0&0&0&0&1&1&0&0&1&1\\
0&0&0&0&0&0&0&1&1&0&0&1&1&0\\
0&0&0&0&0&0&0&0&0&0&1&1&1&1\\
0&0&0&0&0&0&0&1&0&1&1&0&1&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0\\
0&0&0&0&0&0&0&1&1&0&1&0&0&1
\end{pmatrix}$$

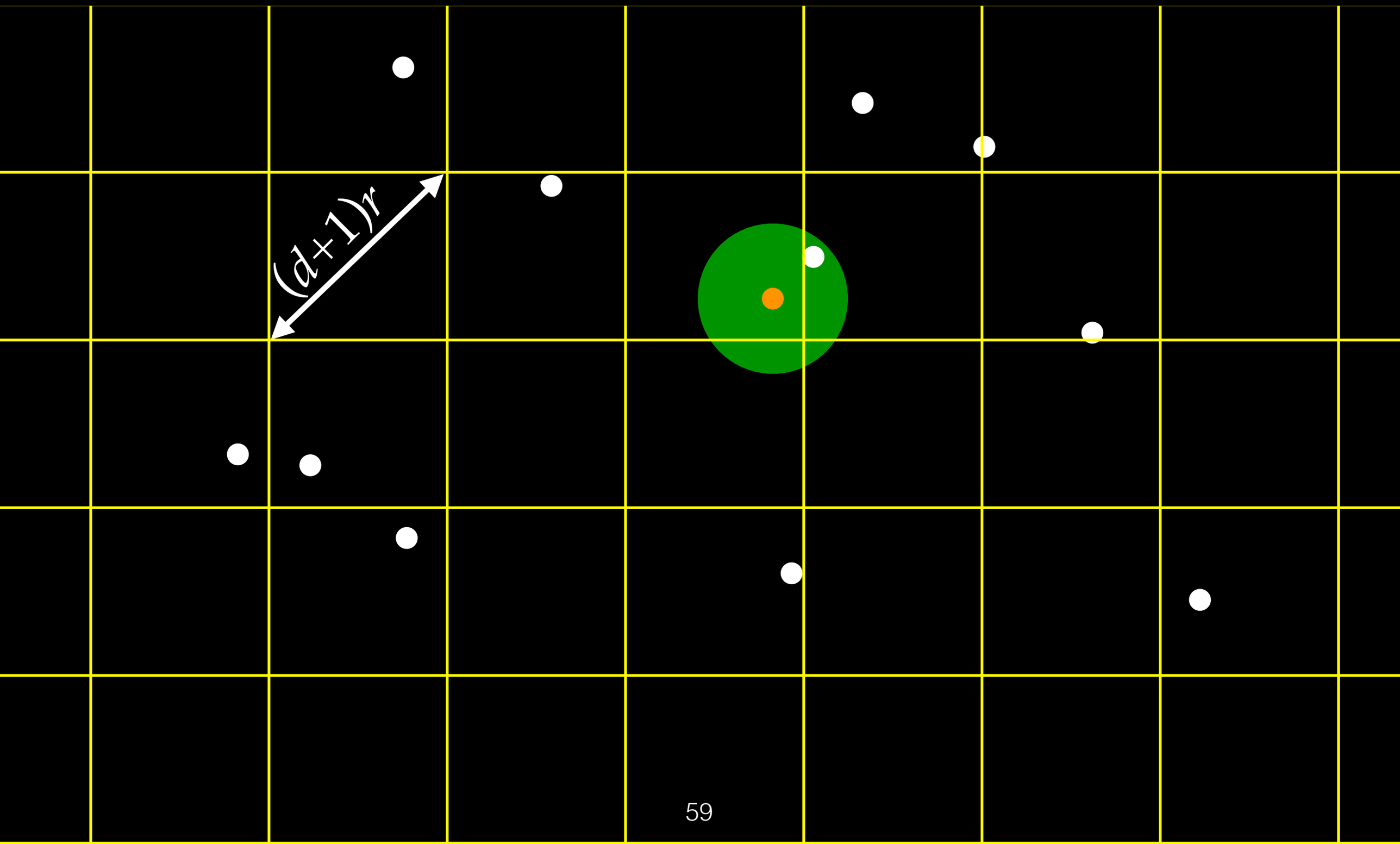
- Distance in *some* part will be $\leq 2pr$
  - Use CoveringLSH with radius $2pr$ on each part
  - Details: arXiv:1507.03225 [cs.DS]

Example

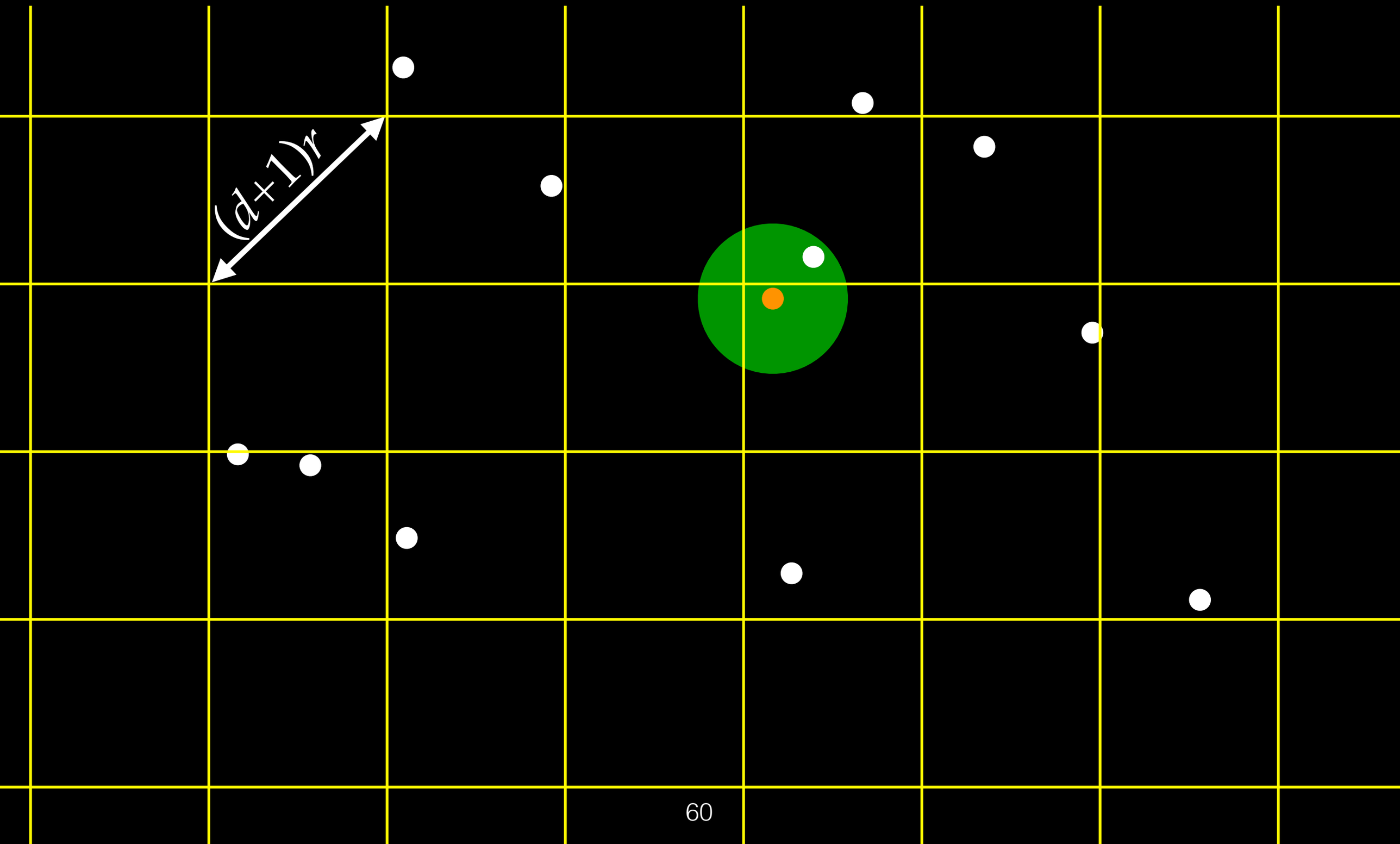
r=256, c=2



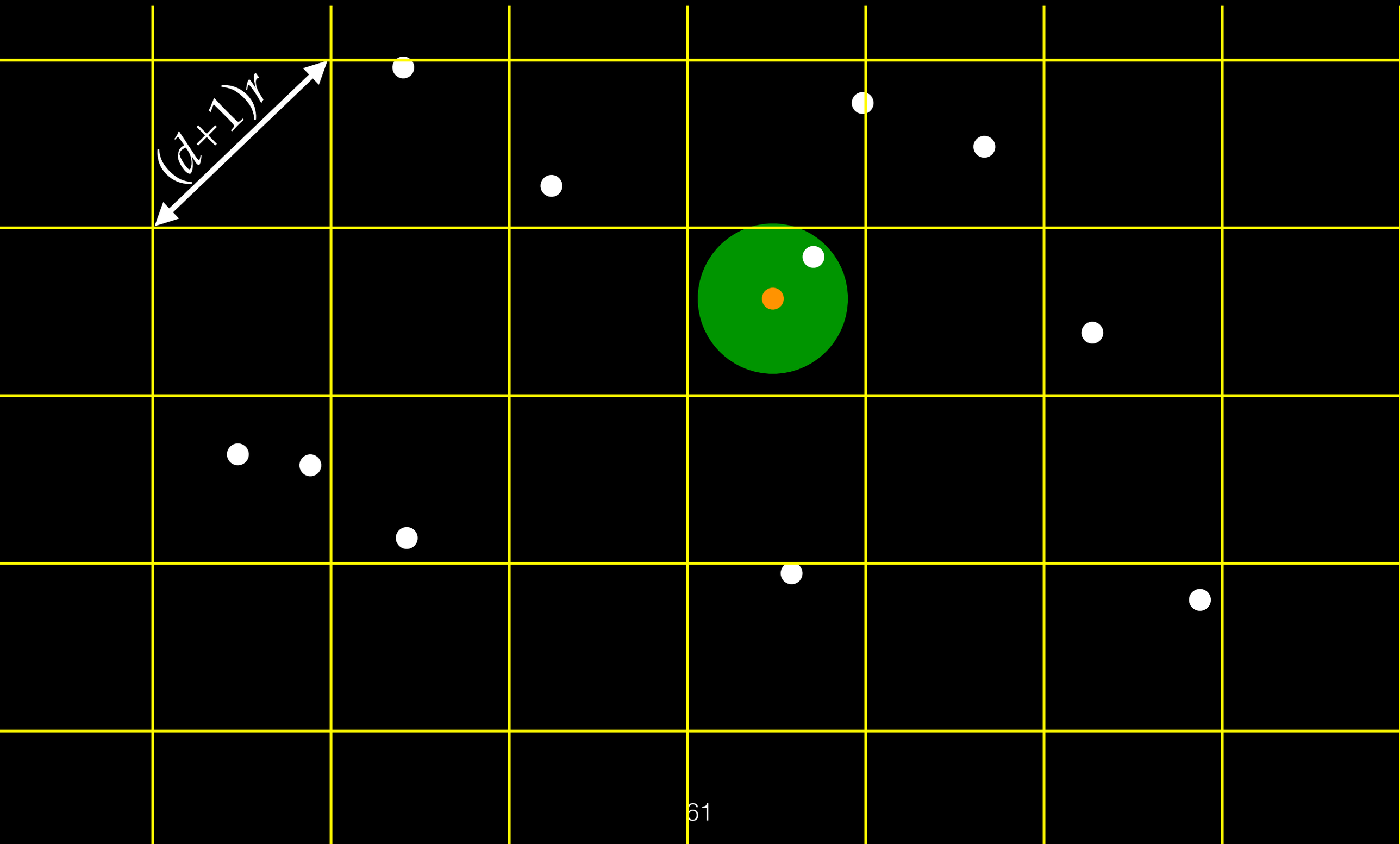
Euclidean space - shifted grids



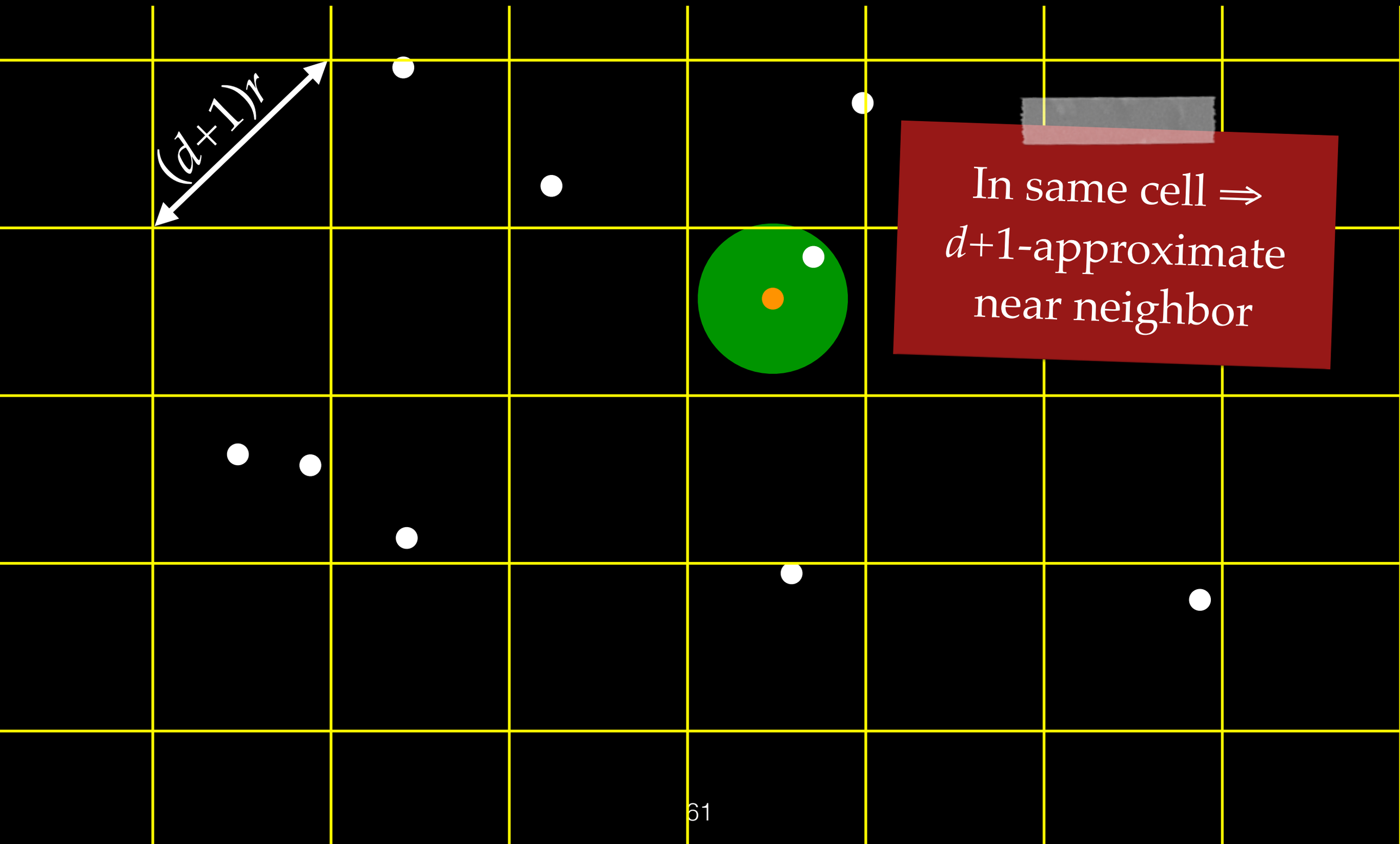
Euclidean space - shifted grids



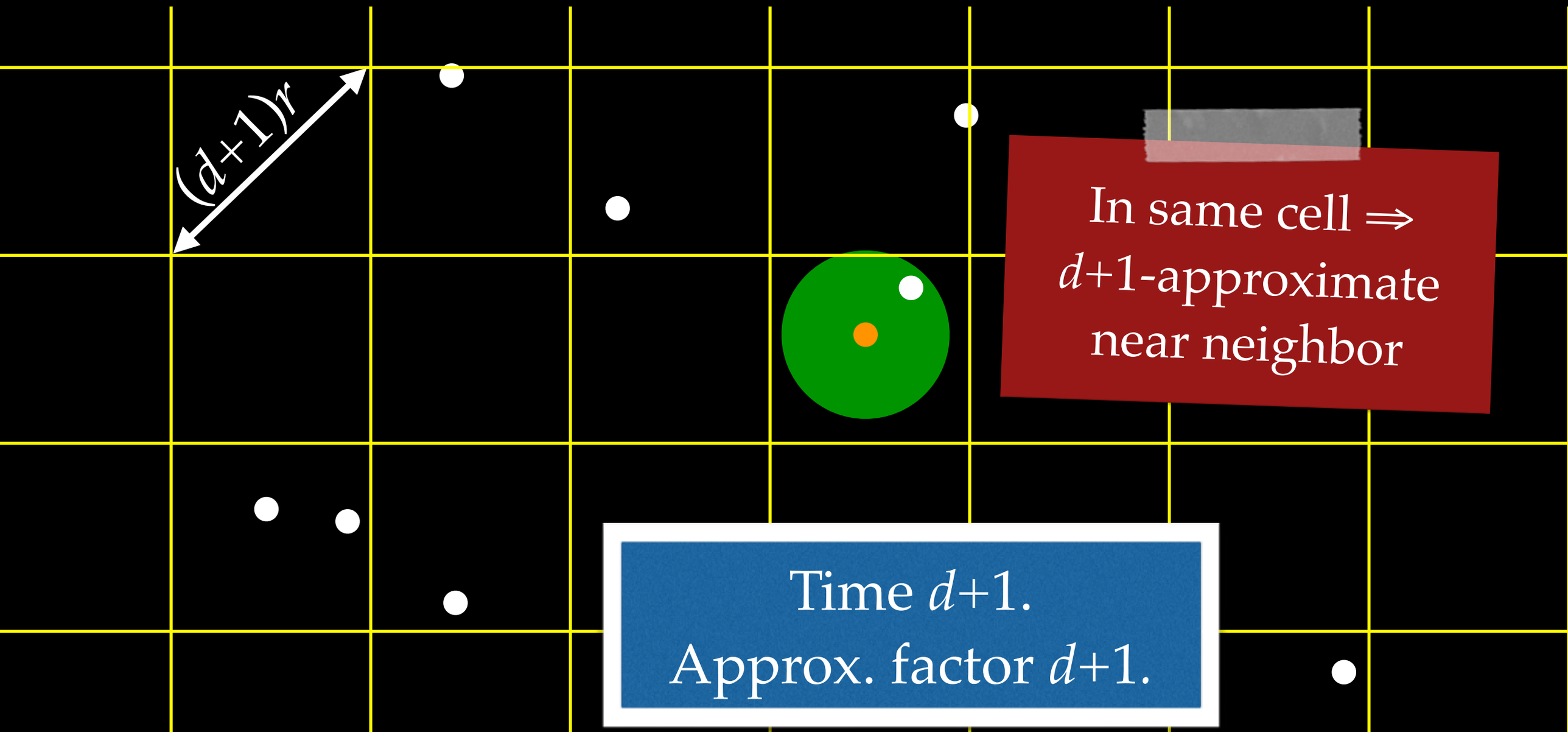
Euclidean space - shifted grids



Euclidean space - shifted grids



Euclidean space - shifted grids



JL dimension reduction

Euclidean vector x

random linear mapping

Projection

Length concentrated around $||x||_2$

JL dimension reduction

Euclidean vector x

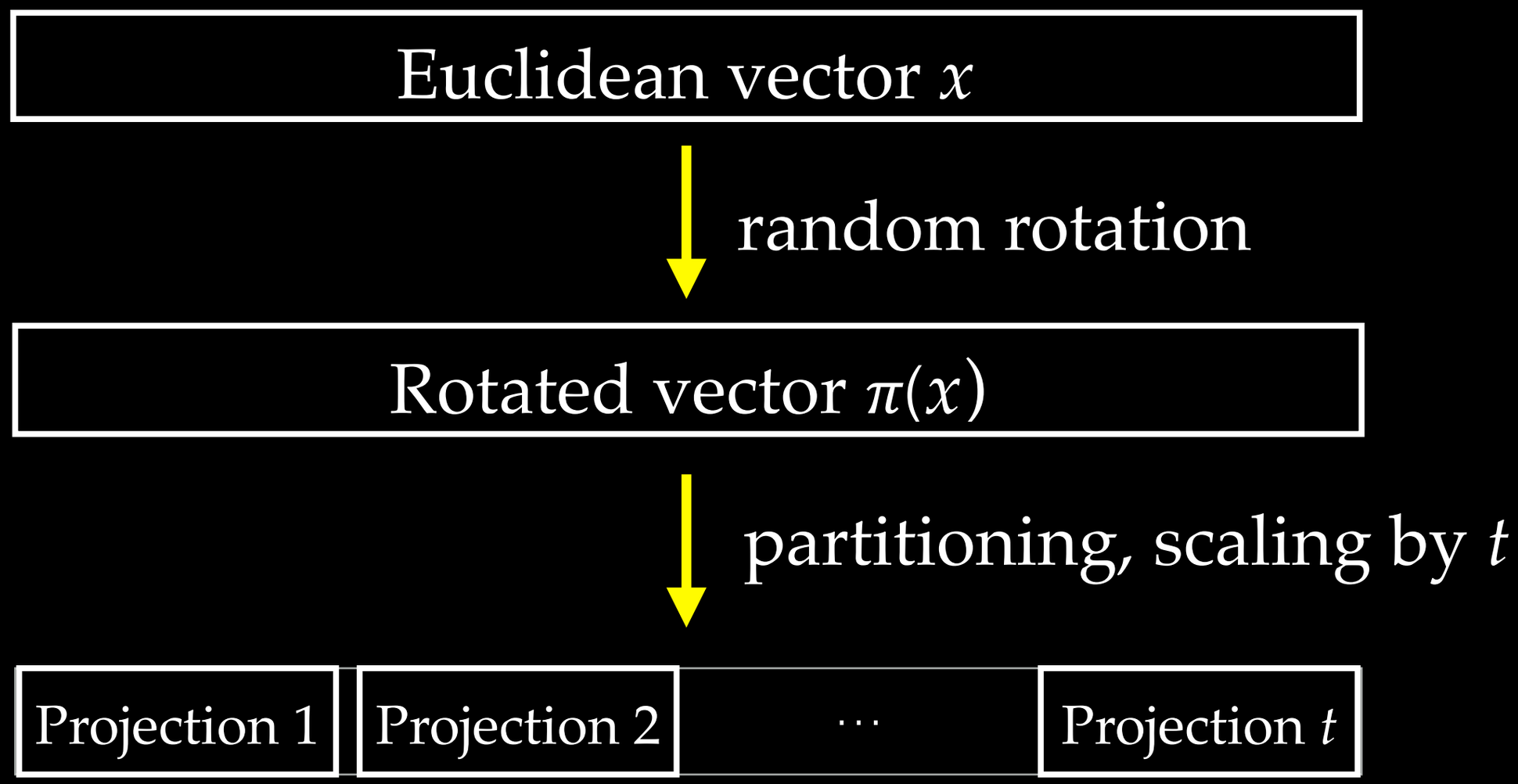
random linear mapping

Projection

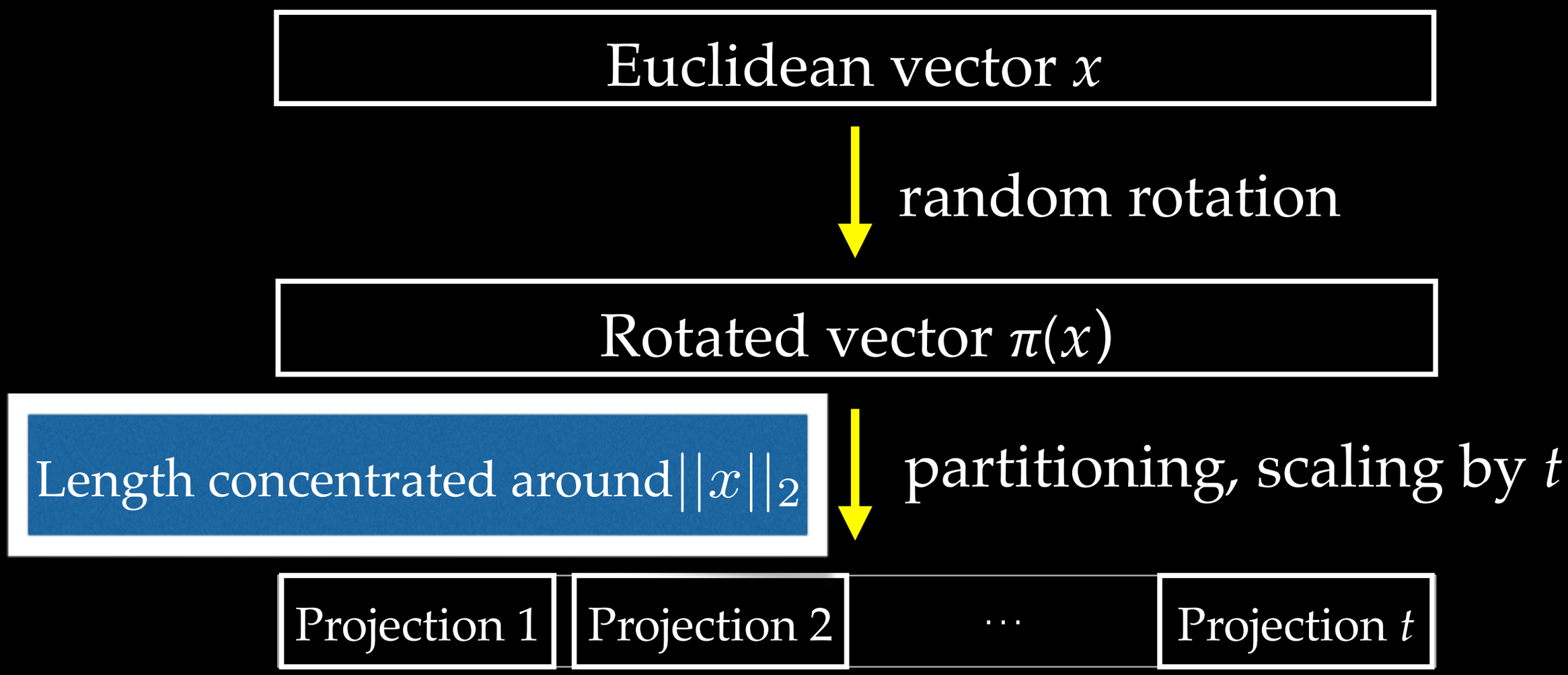
Length concentrated around $||x||_2$

Lengths can increase $\Rightarrow$
false negatives possible

Correlated dimension reduction



Correlated dimension reduction



Correlated dimension reduction

Euclidean vector x

random rotation

Rotated vector $\pi(x)$

Length concentrated around $||x||_2$

partitioning, scaling by t

Projection 1	Projection 2	$\cdots$	Projection t

Some projection will have length at most $||x||_2$

Correlated Euclidean LSH

(joint work with Matthew Skala)

- $dn^{\tilde{O}(1/c)}$ hash functions
- Collision guaranteed within distance r
- Collision probability $1/n$ at distance cr

In practice…

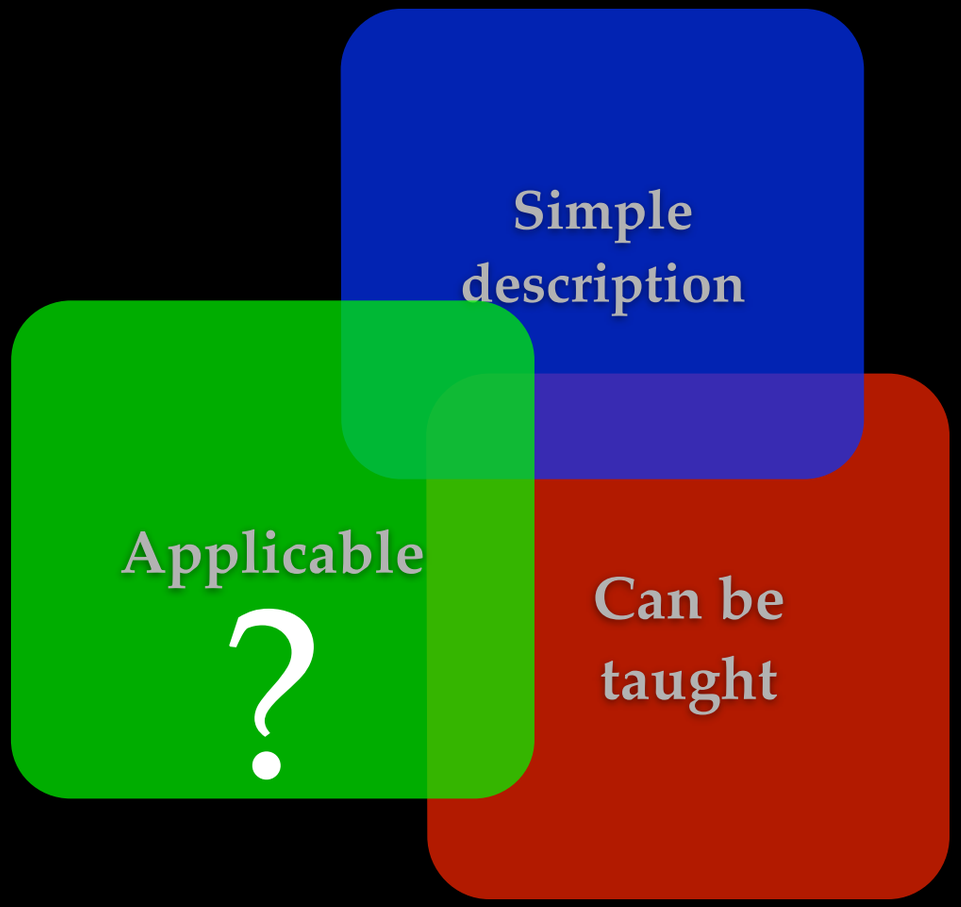
In practice…

- What are the good use cases for similarity join?
- When is 100% recall of particular value?

?

In practice…

- What are the good use cases for similarity join?
- When is 100% recall of particular value?



Linking new theory to practice?

(SODA '16)

Match performance of classical (or data dep.) LSH without false neg.?

Linking new theory to practice?

(SODA '16)

Match performance of classical (or data dep.) LSH without false neg.?

(PODS '15)

Indexing: When is sublinear query time possible with linear space

Linking new theory to practice?

(SODA '16)

Match performance of classical (or data dep.) LSH without false neg.?

(PODS '15)

Indexing: When is sublinear query time possible with linear space

(NIPS '15)

Making new LSH constructions truly practical?

Linking new theory to practice?

(SODA '16)

Match performance of classical (or data dep.) LSH without false neg.?

(PODS '15)

Indexing: When is sublinear query time possible with linear space

(NIPS '15)

Making new LSH constructions truly practical?

(STOC '15)

Data dep. LSH that works in theory *and* in practice?

Thank you

To people with whom I have discussed material of this talk: Annalisa De Bonis, Francesco Silvestri, Ilya Razenshteyn, Johan von Tangen Sivertsen, Matthew Skala, Ninh Pham, Riko Jacob, Thomas Dybdahl Ahle, Tobias Christiani, Ugo Vaccaro, and more.

For economic support:

erc

European Research Council

Established by the European Commission

Thank **you**

for your

attention!