Large-Scale Similarity Joins With Guarantees

Rasmus Pagh
IT University of Copenhagen

EDBT/ICDT
March 25, 2015

European Research Council
Established by the European Commission

Scalable Similarity Search

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

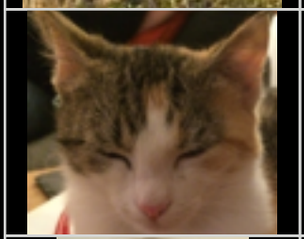
Similarity join example 1 (record linkage)

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)

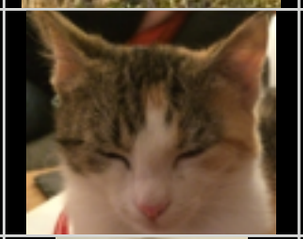
Class	Name
Mammalia	Cat
Mammalia	Dog
Reptilia	Snake
Aves	Parrot

Image	ID
	1
	2
	3
	4

Images by Bodlina and Marek Szczepanek

Similarity join example 2 (classification)

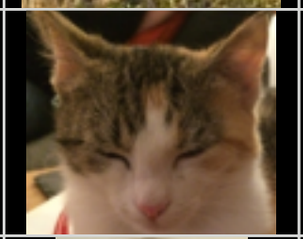
Class	Name
Mammalia	Cat
Mammalia	Dog
Reptilia	Snake
Aves	Parrot

Features	Image	ID
0101101		1
1001000		2
1101101		3
1101110		4

Images by Bodlina and Marek Szczepanek

Similarity join example 2 (classification)

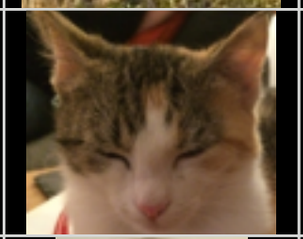
Class	Name	Features
Mammalia	Cat	1100101
Mammalia	Dog	1101101
Reptilia	Snake	1011000
Aves	Parrot	1010111

Features	Image	ID
0101101		1
1001000		2
1101101		3
1101110		4

Images by Bodlina and Marek Szczepanek

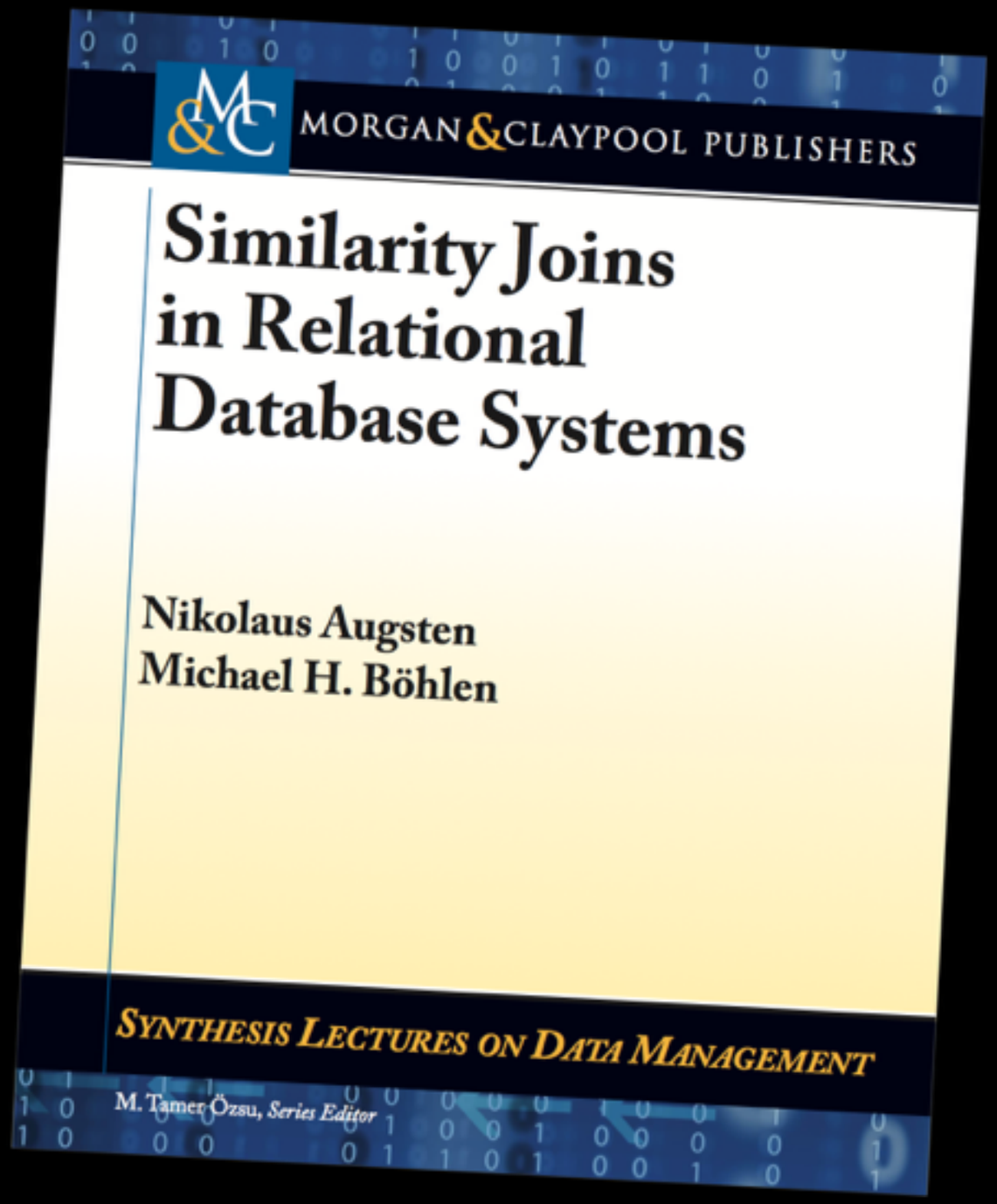
Similarity join example 2 (classification)

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

More examples



Similarity Search in High Dimensions via Hashing

Aristides Gionis * Piotr Indyk† Rajeev Motwani‡
Department of Computer Science
Stanford University
Stanford, CA 94305
{gionis,indyk,rajeev}@cs.stanford.edu

Abstract

The nearest- or near-neighbor query problems arise in a large variety of database applications, usually in the context of similarity searching. Of late, there has been increasing interest in building search/index structures for performing similarity search over high-dimensional data, e.g., image databases, document collections, time-series databases, and genome databases. Unfortunately, all known techniques for solving this problem fall prey to the "curse of dimensionality." That is, the data structures scale poorly with data dimensionality; in fact, if the number of dimensions exceeds 10 to 20, searching in k-d trees and related structures involves the inspection of a large fraction of the database, thereby doing no better than brute-force linear search. It has been suggested that since the selection of features and the choice of a distance metric in typical applications is rather heuristic, determining an approximate nearest neighbor should suffice for most practical purposes. In this paper, we examine a novel scheme for approximate similarity search based on hashing. The basic idea is to hash the points from the database so as to ensure that the probability of collision is much higher for objects that are close to each other than for those that are far apart. We provide experimental evidence that our method gives significant improvement in running time over other methods for searching in high-dimensional spaces based on hierarchical tree decomposition. Experimental results also indicate that our scheme scales well even for a relatively large number of dimensions (more than 50).

*Supported by NAVY N00014-96-1-1221 grant and NSF Grant IIS-9811904.

†Supported by Stanford Graduate Fellowship and NSF NYI Award CCR-9357849.

‡Supported by ARO MURI Grant DAAH04-96-1-0007, NSF Grant IIS-9811904, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

**Proceedings of the 25th VLDB Conference,
Edinburgh, Scotland, 1999.**

1 Introduction

A similarity search problem involves a collection of objects (e.g., documents, images) that are characterized by a collection of relevant features and represented as points in a high-dimensional attribute space; given queries in the form of points in this space, we are required to find the nearest (most similar) object to the query. The particularly interesting and well-studied case is the d-dimensional Euclidean space. The problem is of major importance to a variety of applications; some examples are: data compression [20]; databases and data mining [21]; information retrieval [11, 16, 38]; image and video databases [15, 17, 37, 42]; machine learning [7]; pattern recognition [9, 13]; and, statistics and data analysis [12, 27]. Typically, the features of the objects of interest are represented as points in $\Re^d$ and a distance metric is used to measure similarity of objects. The basic problem then is to perform indexing or similarity searching for query objects. The number of features (i.e., the dimensionality) ranges anywhere from tens to thousands. For example, in multimedia applications such as IBM's QBIC (Query by Image Content), the number of features could be several hundreds [15, 17]. In information retrieval for text documents, vector-space representations involve several thousands of dimensions, and it is considered to be a dramatic improvement that dimension-reduction techniques, such as the Karhunen-Loève transform [26, 30] (also known as principal components analysis [22] or latent semantic indexing [11]), can reduce the dimensionality to a mere few hundreds!

Similarity Search in High Dimensions via Hashing

Aristides Gionis * Piotr Indyk† Rajeev Motwani‡
Department of Computer Science
Stanford University
Stanford, CA 94305
{gionis,indyk,rajeev}@cs.stanford.edu

Abstract

from the database so as to ensure that the probability of collision is much higher for objects that are close to each other than for those that are far apart. We provide experimental evidence that our

[PDF] Similarity search in high dimensions via hashing

A Gionis, P Indyk, R Motwani - **VLDB**, 1999 - cs.princeton.edu

Abstract The nearest-or near-neighbor query problems arise in a large variety of database applications, usually in the context of similarity searching. Of late, there has been increasing interest in building search index structures for performing similarity search over high- ...

Cited by 1723 Related articles All 46 versions Cite Save More

*Supported by NAVY N00014-96-1-1221 grant and NSF Grant IIS-9811904.
†Supported by Stanford Graduate Fellowship and NSF NYI Award CCR-9357849.
‡Supported by ARO MURI Grant DAAH04-96-1-0007, NSF Grant IIS-9811904, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

Proceedings of the 25th VLDB Conference,
Edinburgh, Scotland, 1999.

than brute-force linear search of the whole database for moderate dimensionality. It has been suggested that since the selection of features and the choice of a distance metric in typical applications is rather heuristic, determining an approximate nearest neighbor should suffice for most practical purposes. In this paper, we examine a novel scheme for approximate similarity search based on hashing. The basic idea is to hash the points

some examples are: data compression [26]; and data mining [21]; information retrieval [11, 16, 38]; image and video databases [15, 17, 37, 42]; machine learning [7]; pattern recognition [9, 13]; and, statistics and data analysis [12, 27]. Typically, the features of the objects of interest are represented as points in $\Re^d$ and a distance metric is used to measure similarity of objects. The basic problem then is to perform indexing or similarity searching for query objects. The number of features (i.e., the dimensionality) ranges anywhere from tens to thousands. For example, in multimedia applications such as IBM's QBIC (Query by Image Content), the number of features could be several hundreds [15, 17]. In information retrieval for text documents, vector-space representations involve several thousands of dimensions, and it is considered to be a dramatic improvement that dimension-reduction techniques, such as the Karhunen-Loéve transform [26, 30] (also known as principal components analysis [22] or latent semantic indexing [11]), can reduce the dimensionality to a mere few hundreds!

Beyond databases

- Association rule mining
- Automation
- Bio-chemistry (finding motifs)
- Bio-informatics (homology search)
- Clustering
- Computer vision and pattern recognition
- Data cleaning
- Data stream computation
- Data privacy
- First story detection (with application to twitter)
- Identifying trends in time series
- Linear algebra
- Motion planning for robots
- Near-duplicate detection
- News personalization (collaborative filtering)
- Privacy preserving data mining
- Search engines for 3D models
- Sensor networks
- …

Talk outline

- The algorithmic problem.
- Curse of dimensionality and approximation.
- Techniques for candidate set generation

Talk outline

- The algorithmic problem.
- Curse of dimensionality and approximation.
- Techniques for candidate set generation
  A. Locality-sensitive hashing

Talk outline

- The algorithmic problem.
- Curse of dimensionality and approximation.
- Techniques for candidate set generation
  A. Locality-sensitive hashing
  B. Cache-efficiency via recursion

Talk outline

- The algorithmic problem.
- Curse of dimensionality and approximation.
- Techniques for candidate set generation
  A. Locality-sensitive hashing
  B. Cache-efficiency via recursion
  C. Exact similarity join: Achieving 100% recall

Algorithmic problem

- Given distance function d and tolerance r compute:

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

Algorithmic problem

- Given distance function d and tolerance r compute:

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

- **This talk**:

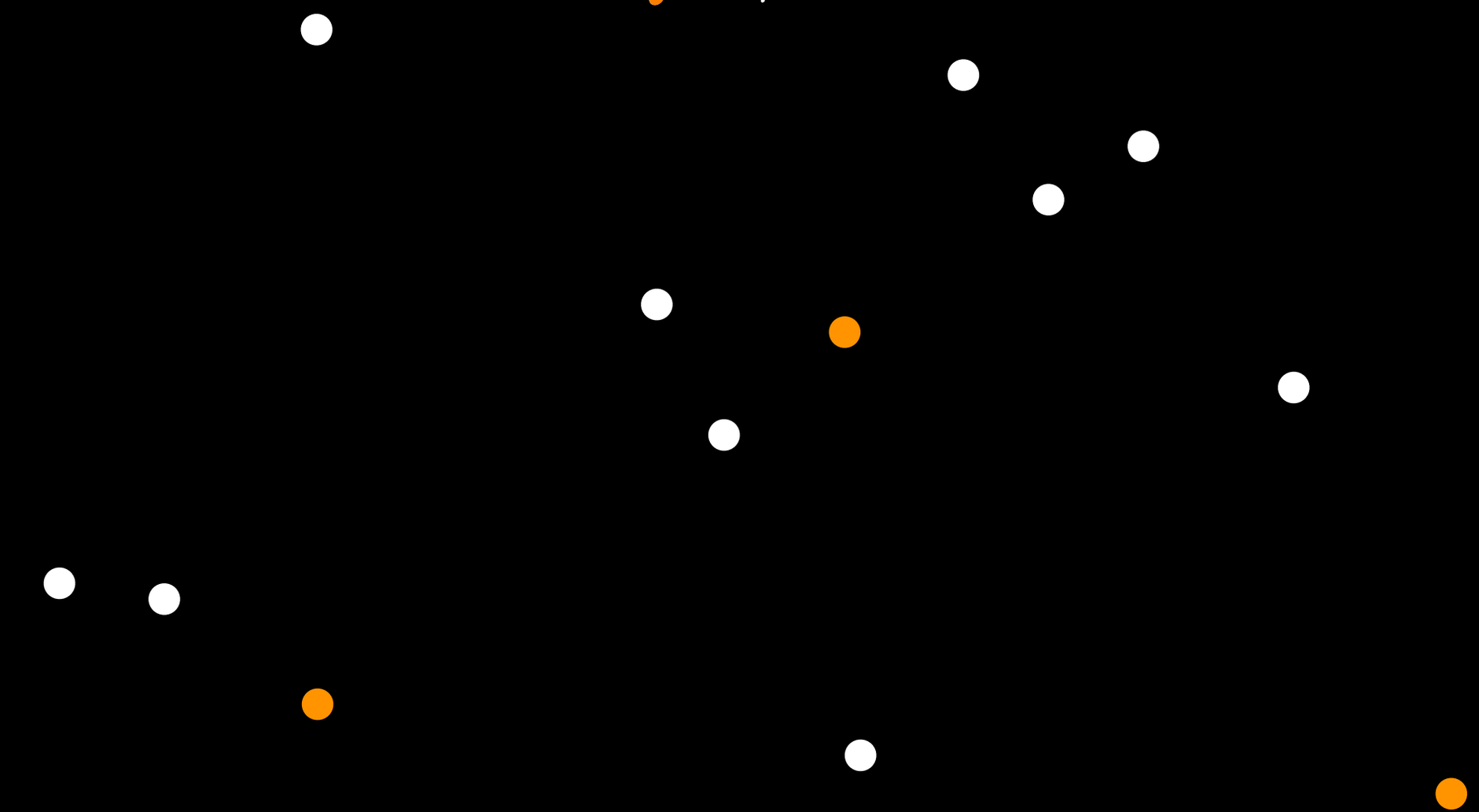
  Consider n vectors in $\{0,1\}^D$ and Hamming distance.

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

$$d(q, x) = 1$$

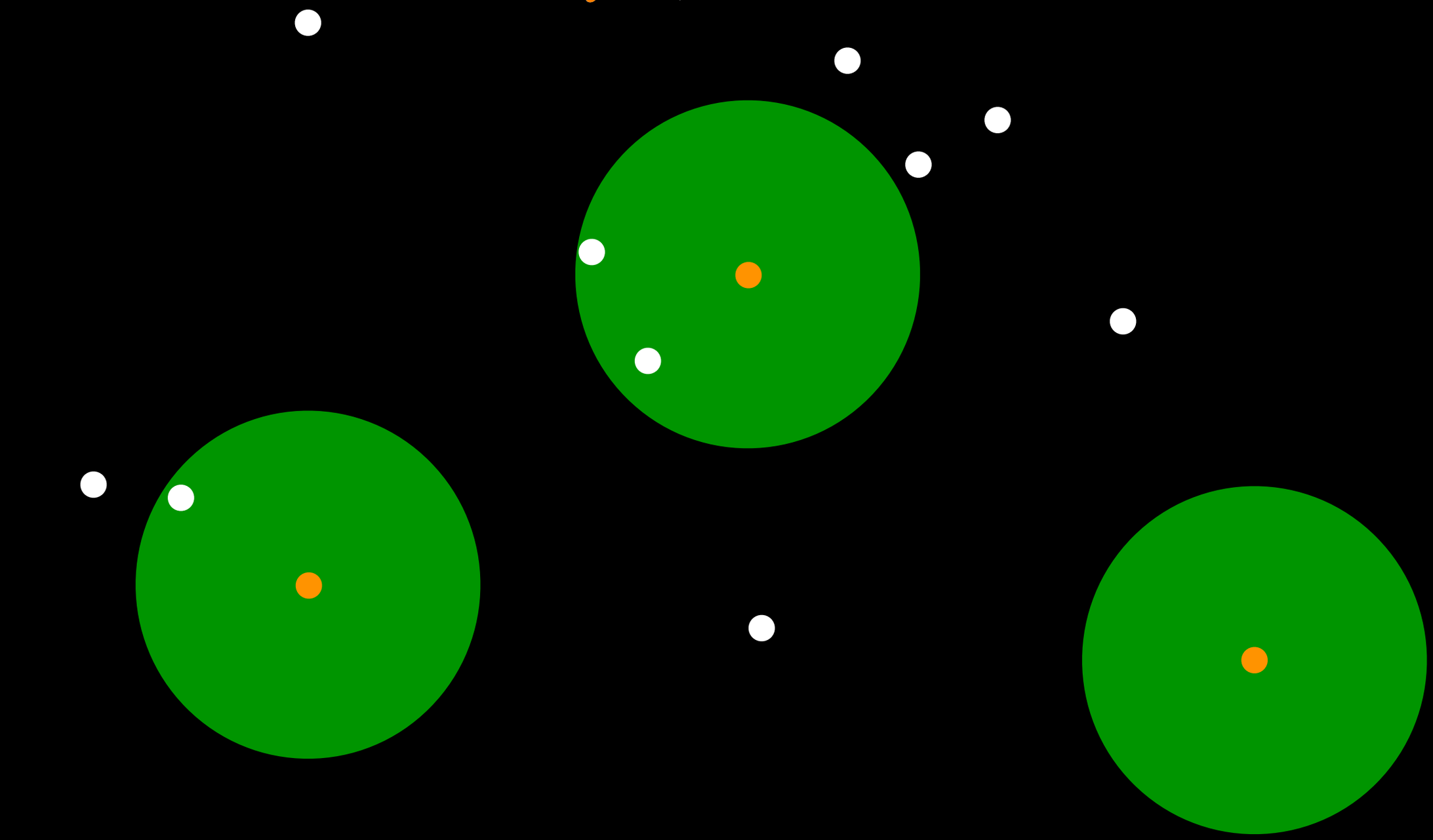
Similarity join in a picture

$$R = Q \bowtie_r S$$



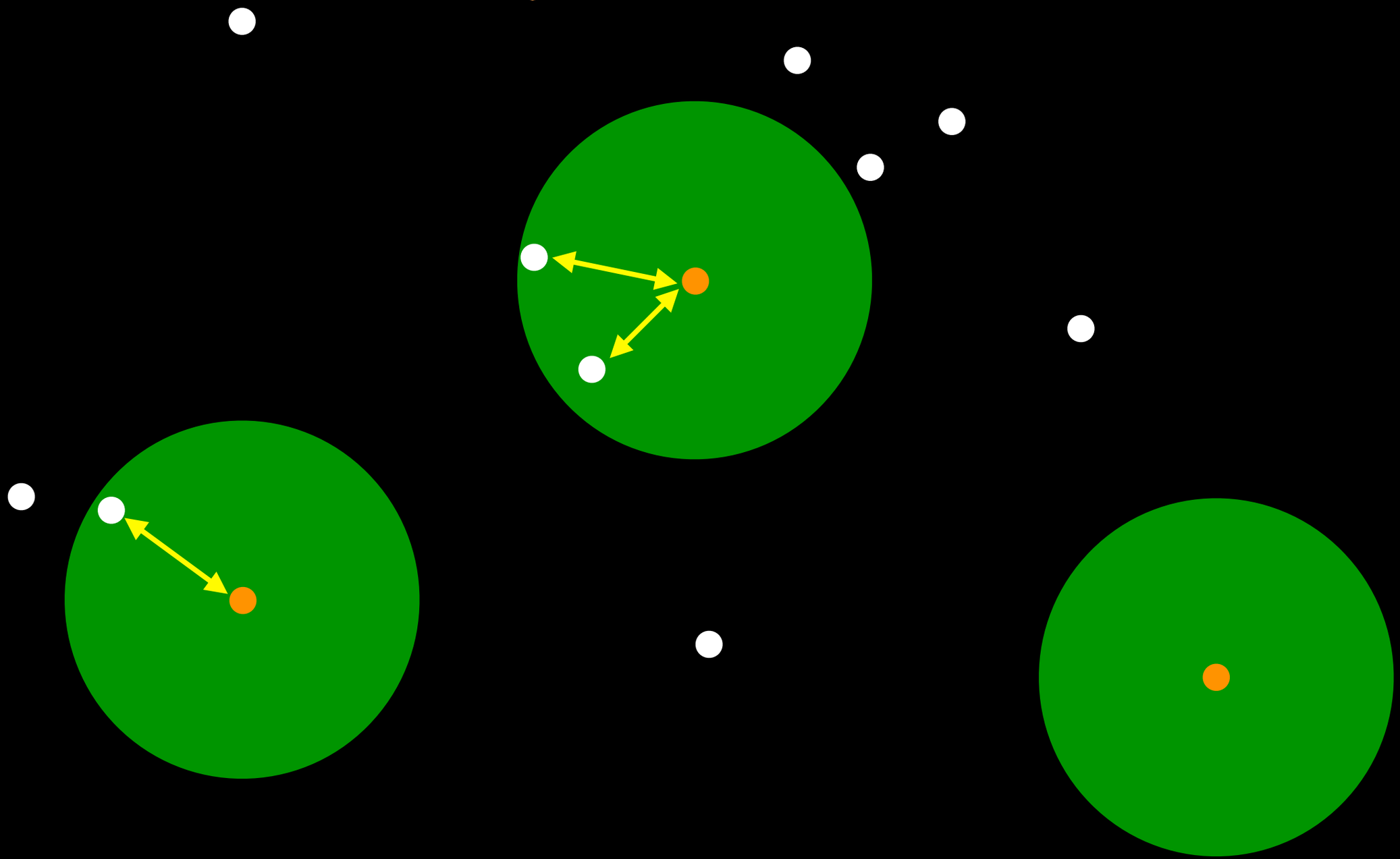
Similarity join in a picture

$$R = Q \bowtie_r S$$



Similarity join in a picture

$$R = Q \bowtie_r S$$

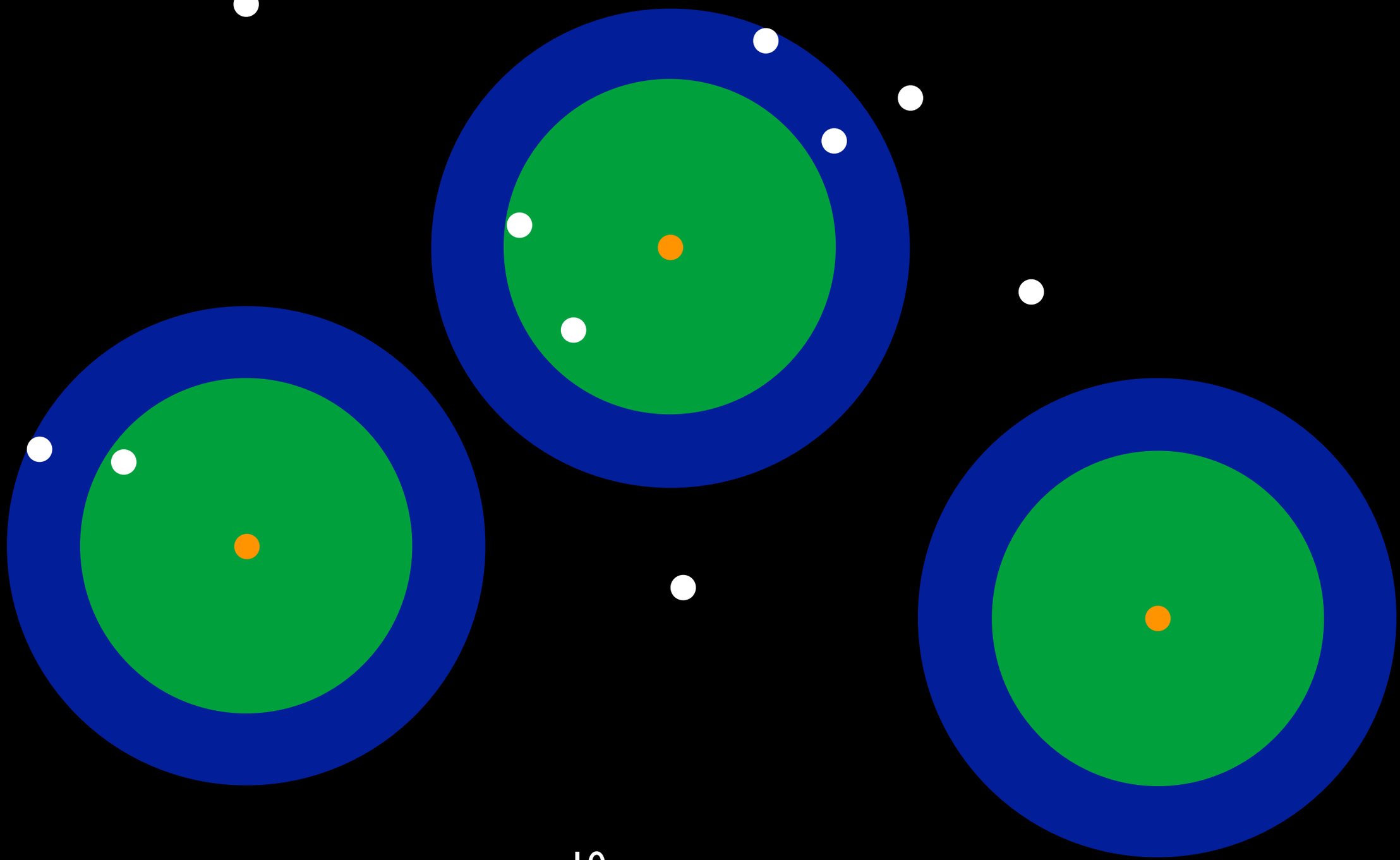


Similarity join in a picture

in high dimensional space (D=80)

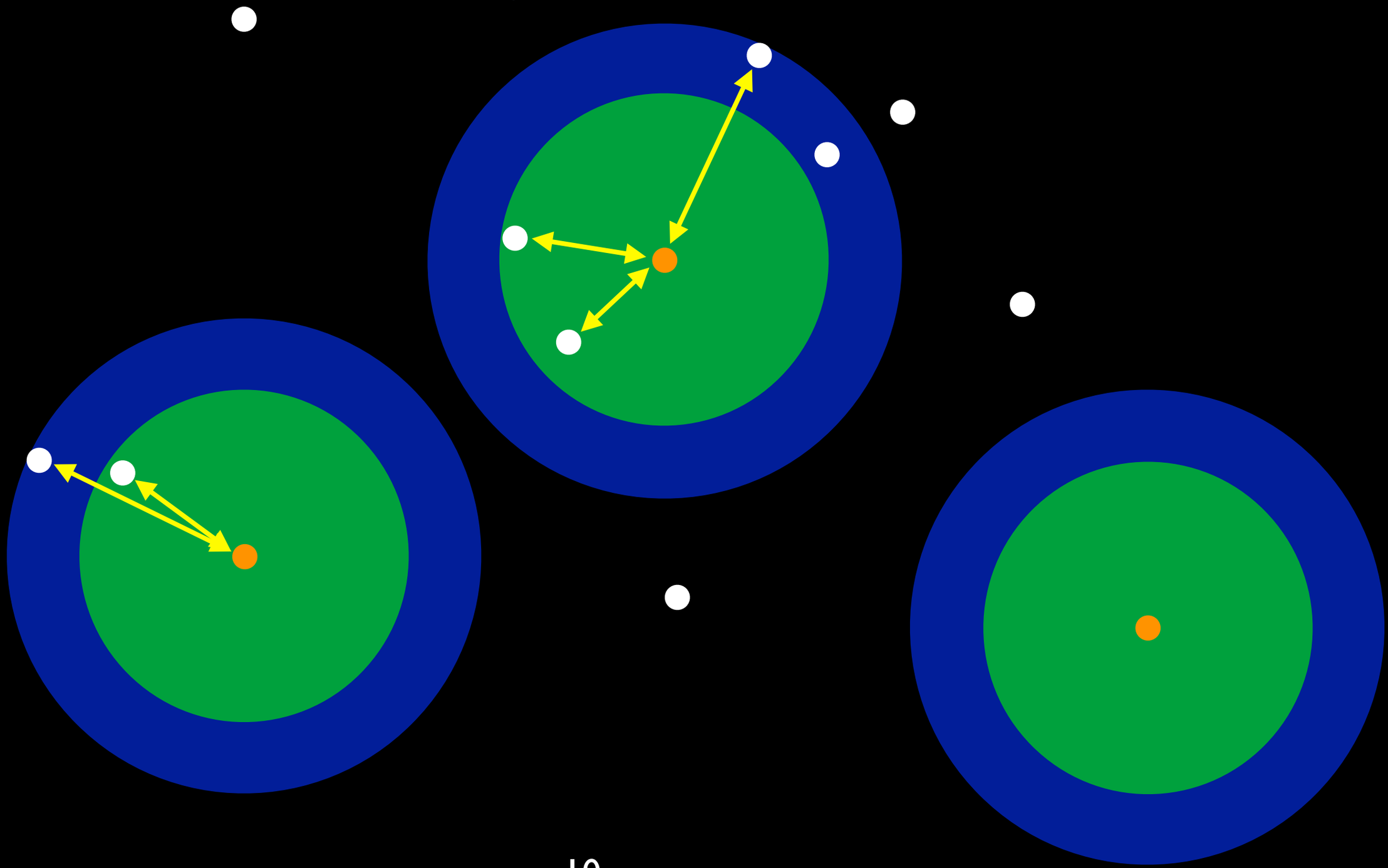
c-approximate similarity join

$$C \supseteq Q \bowtie_r S$$



c-approximate similarity join

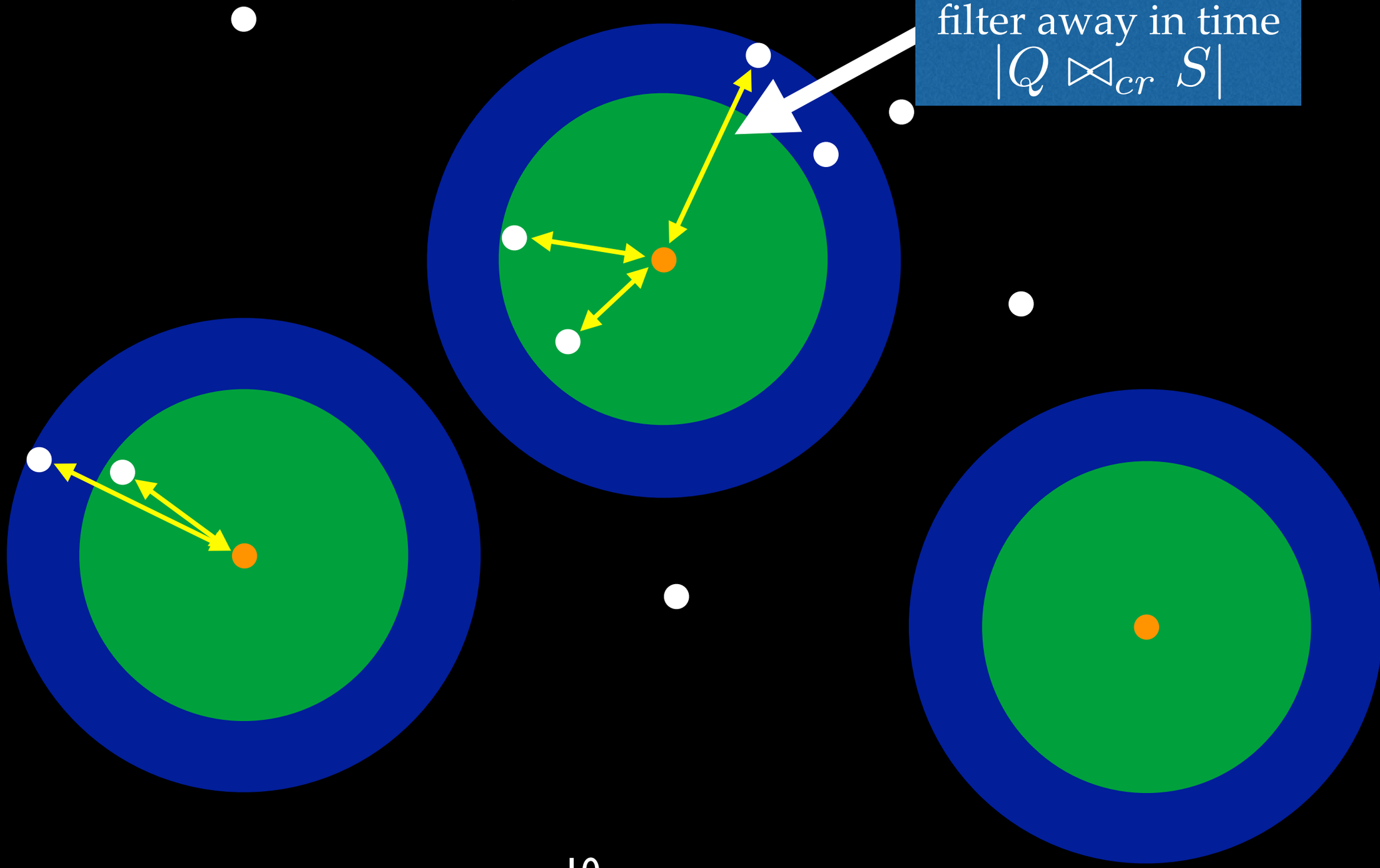
$$C \supseteq Q \bowtie_r S$$



c-approximate similarity join

$$C \supseteq Q \bowtie_r S$$

False positive pairs, filter away in time $|Q \bowtie_{cr} S|$



Why approximation?

Why approximation?

Because of the CURSE of dimensionality!

Why approximation?

- Silvestri et al. 2015:

  $\exists$ similarity join algorithm using time $n^{1+o(1)}\, 2^{o(D)}$

  $\Longrightarrow$

  k-SAT w. n variables can be solved in time $2^{n/2+o(1)}$

Why approximation?

- Silvestri et al. 2015:

∃ similarity join algorithm using time $n^{1+o(1)}\, 2^{o(D)}$

$\Rightarrow$

k-SAT w. n variables can be solved in time $2^{n/2+o(1)}$

Huge research effort has been devoted to k-SAT, but so far failed to even reach 1.99^n for general k.

Why approximation?

Conclusion: A highly efficient algorithm for similarity join with no approximation of distances would be very surprising.

- Silvestri et al. 2015:

∃ similarity join algorithm using time $n^{1+o(1)}\, 2^{o(D)}$

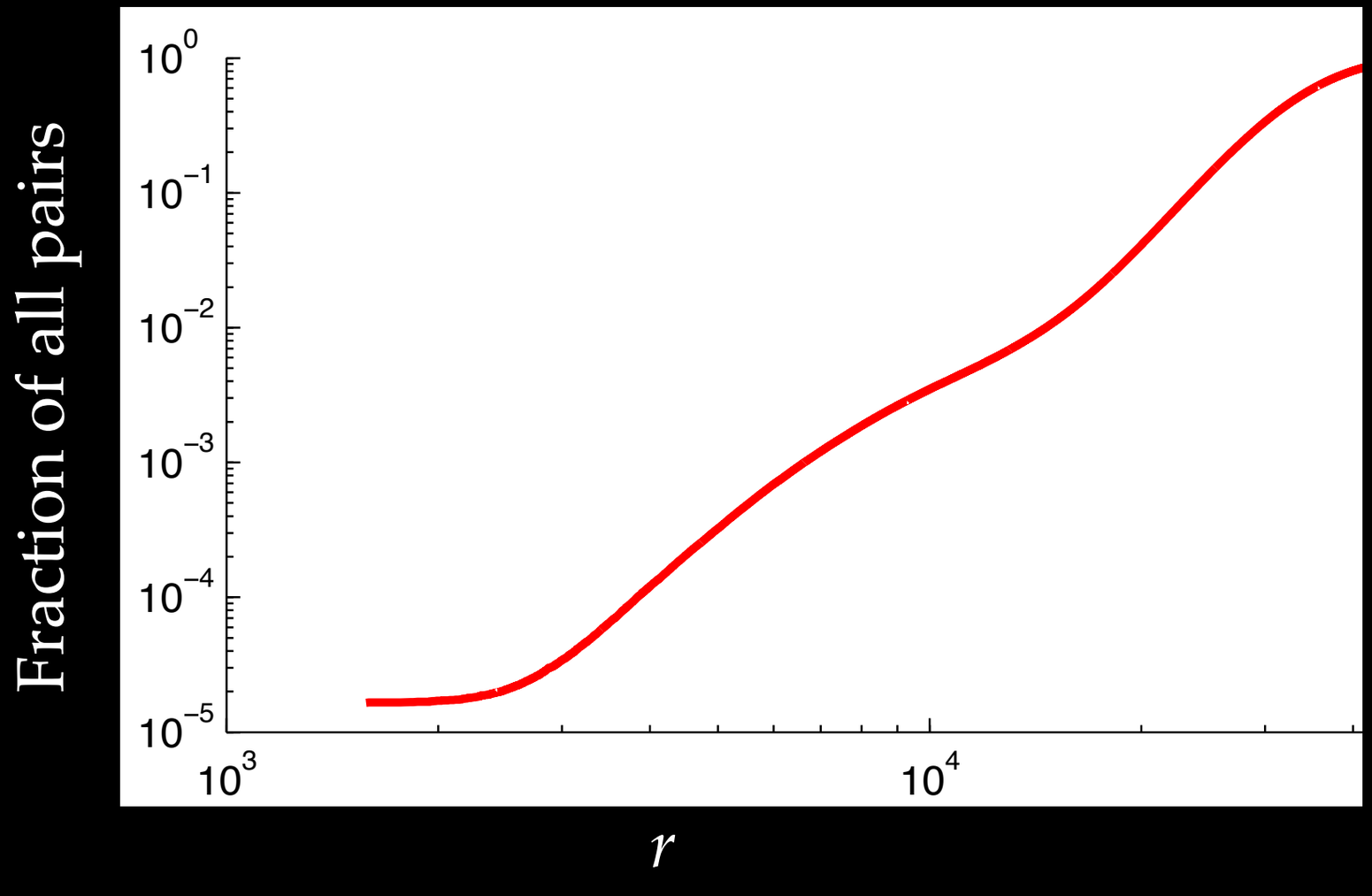
$\Rightarrow$

k-SAT w. n variables can be solved in time $2^{n/2+o(1)}$

Huge research effort has been devoted to k-SAT, but so far failed to even reach 1.99^n for general k.

Effect of approximation

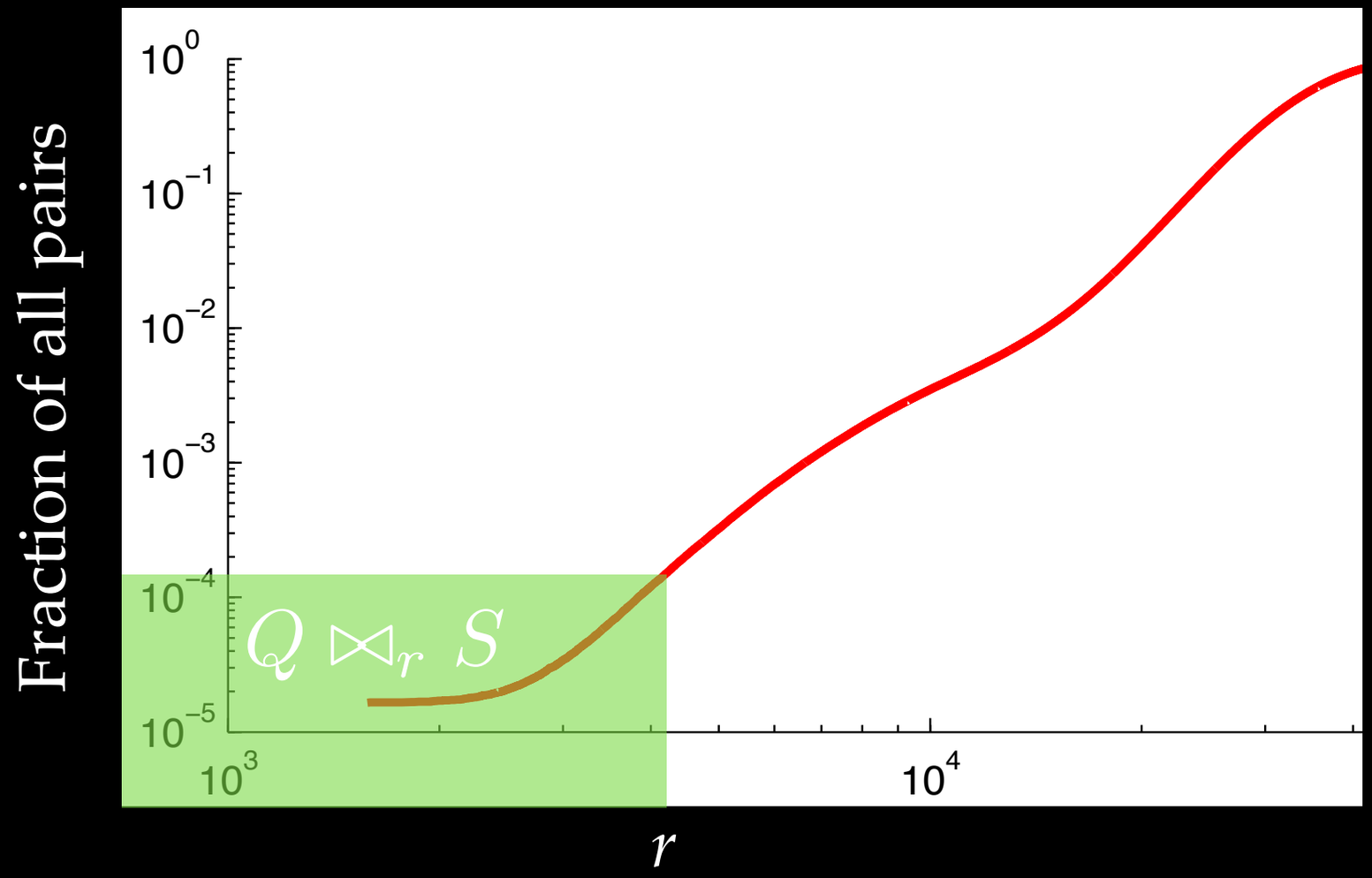
MNIST data set (unary encoding)



within Hamming distance

Effect of approximation

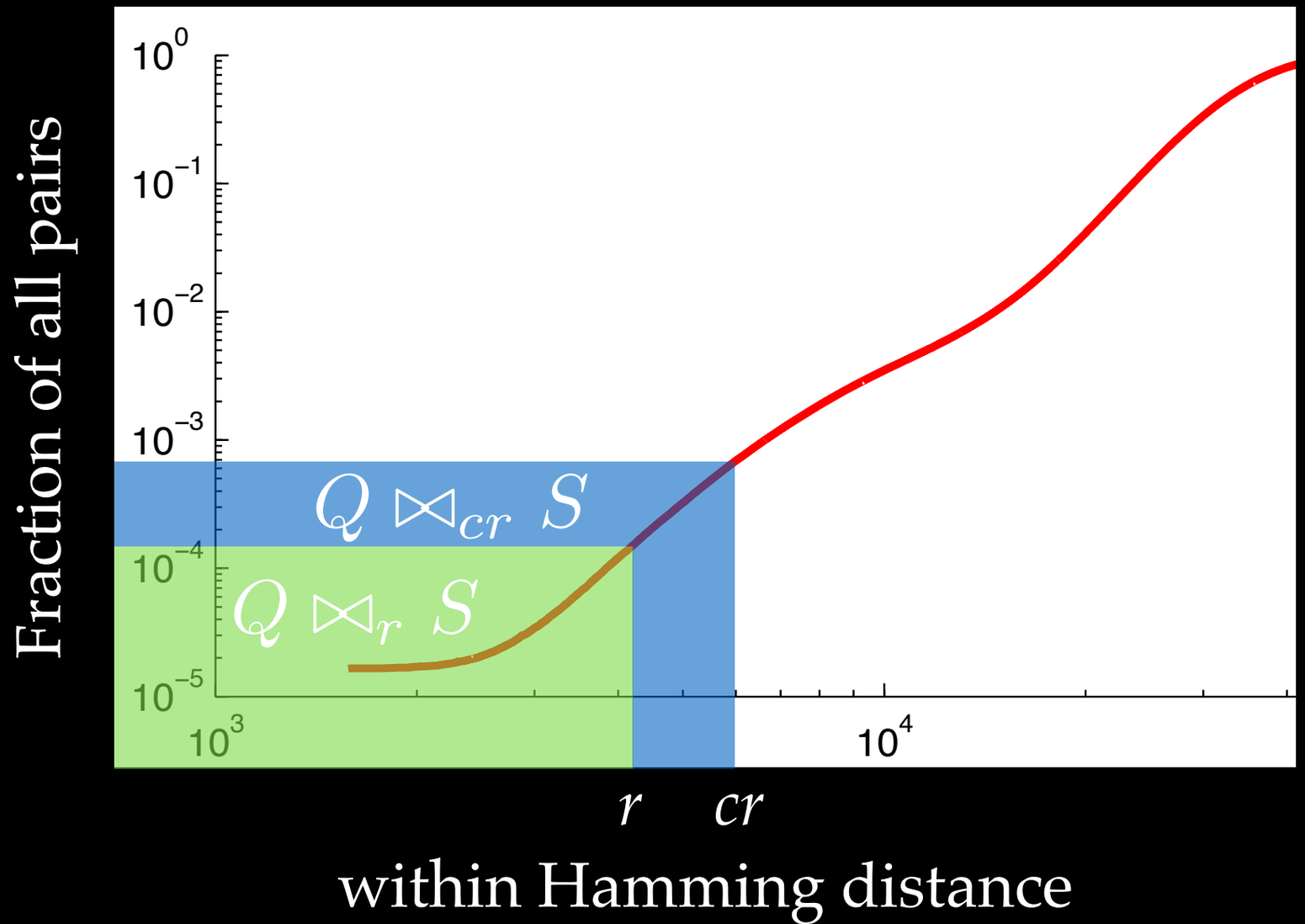
MNIST data set (unary encoding)



within Hamming distance

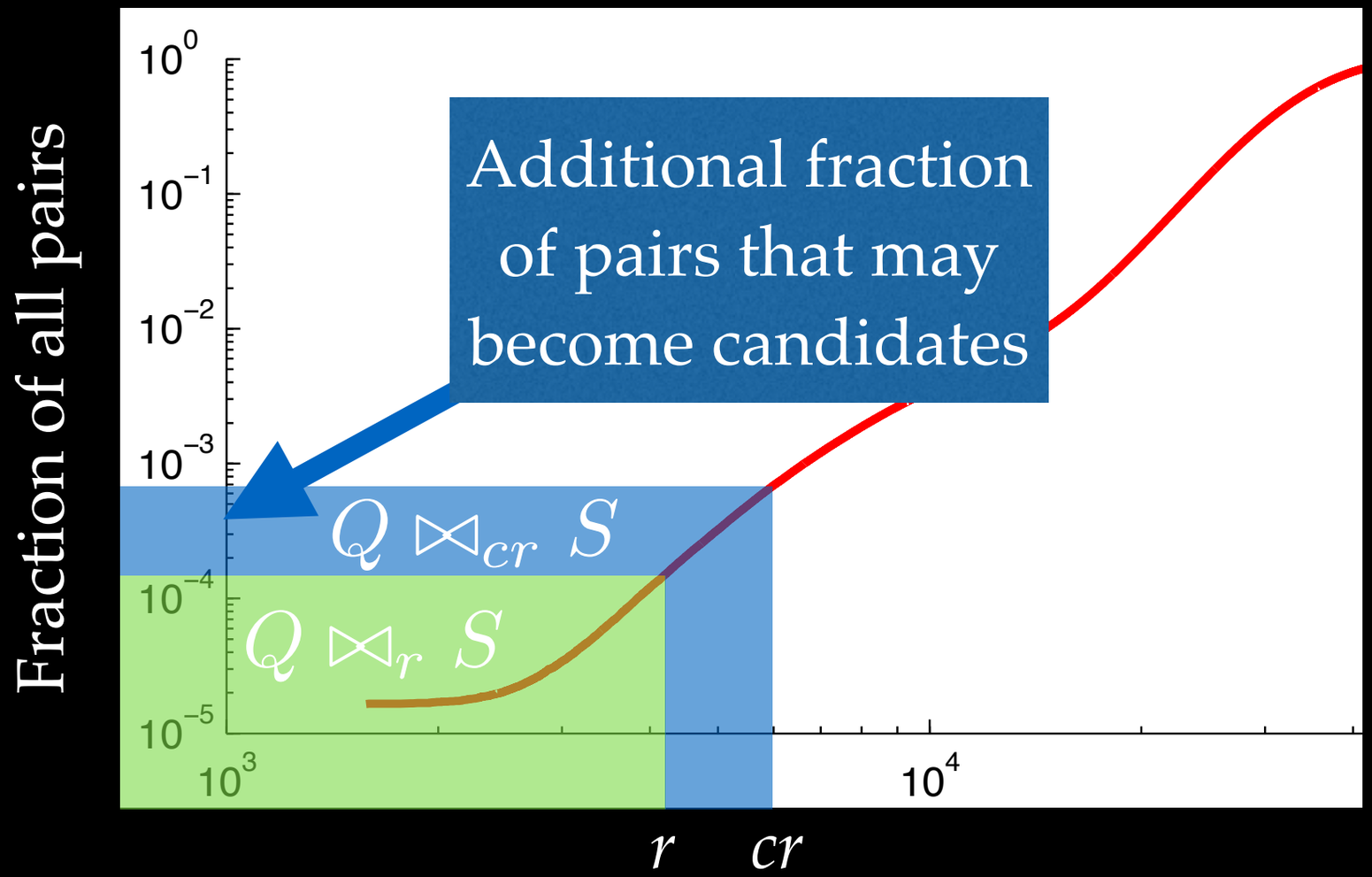
Effect of approximation

MNIST data set (unary encoding)



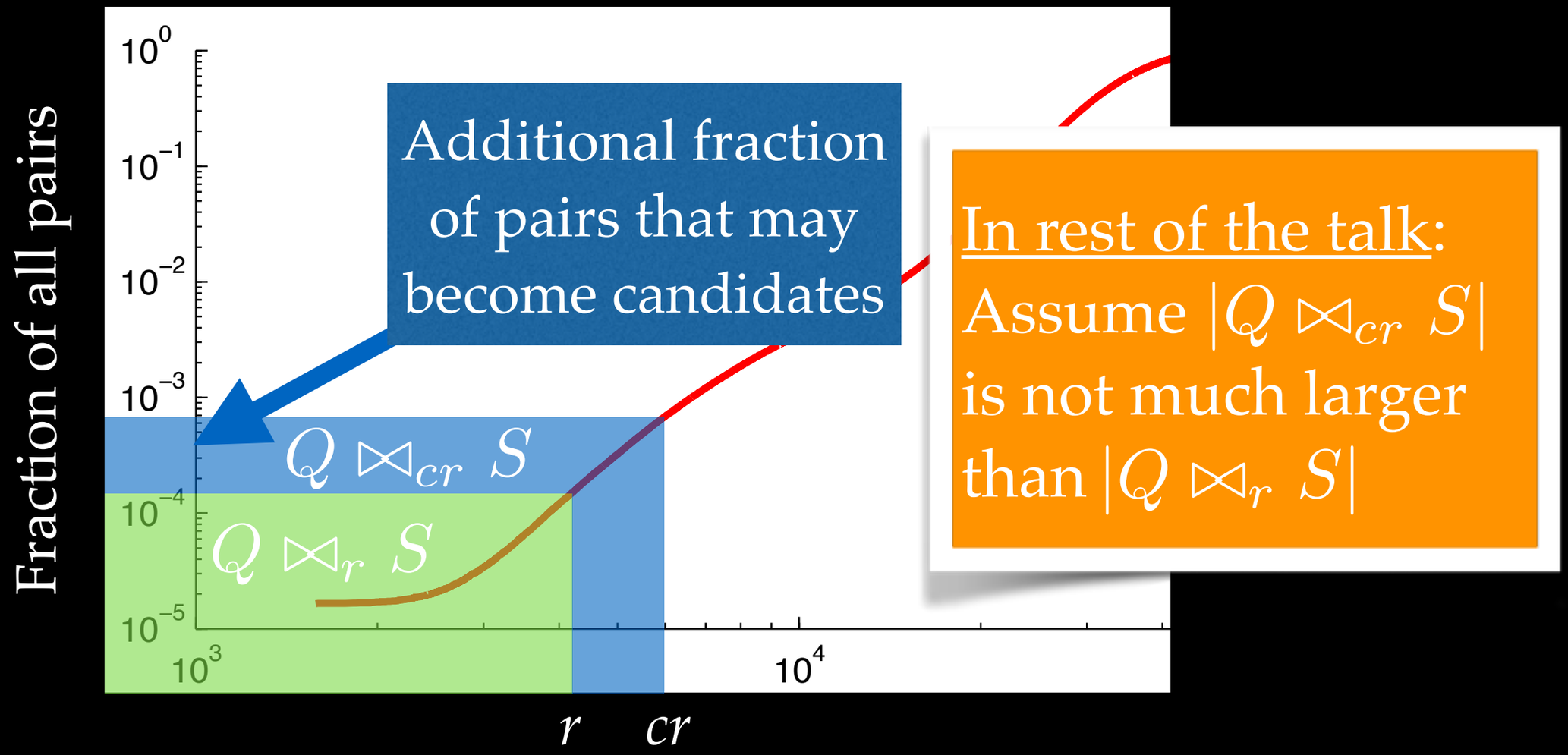
Effect of approximation

MNIST data set (unary encoding)



Effect of approximation

MNIST data set (unary encoding)

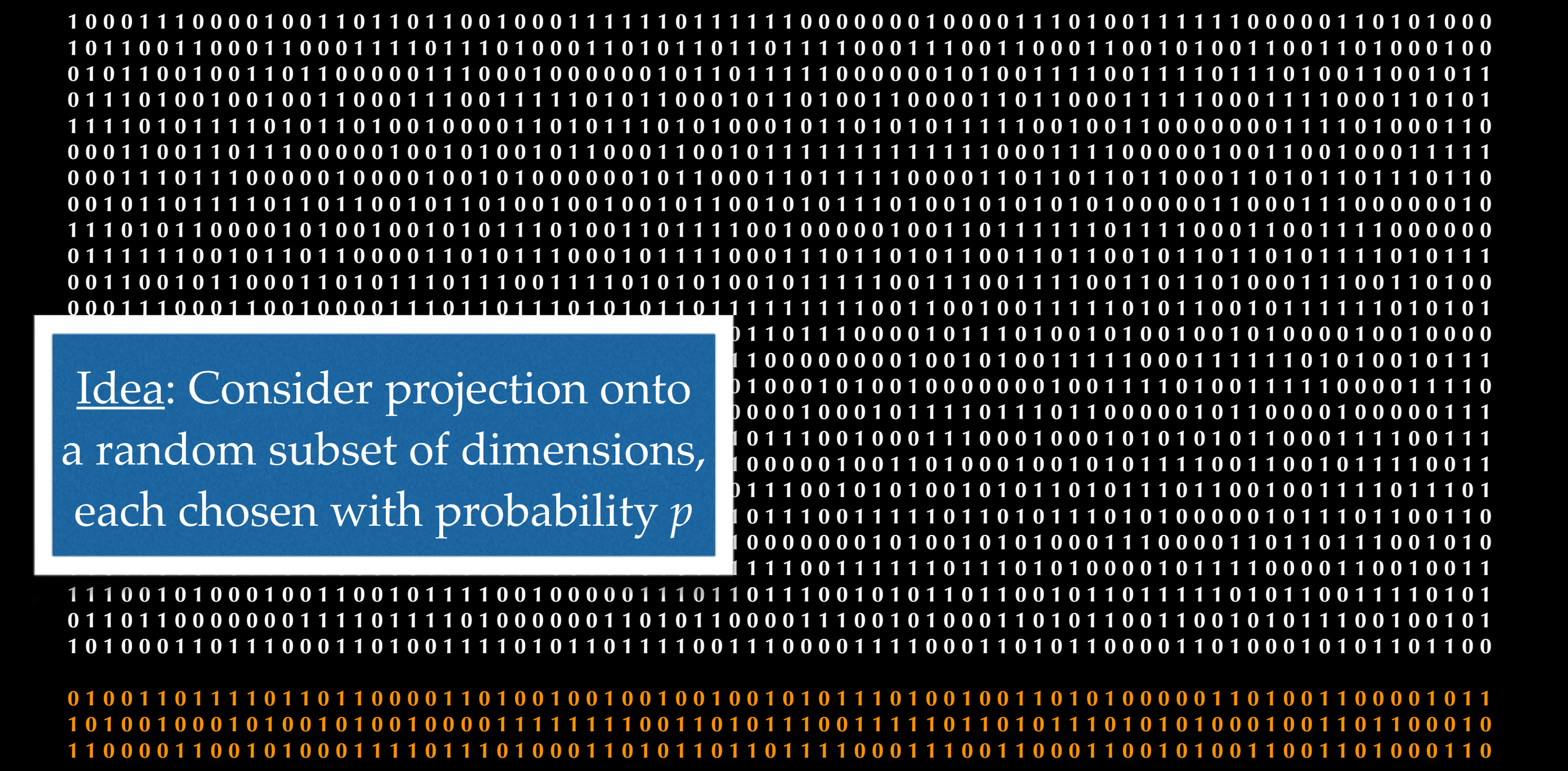


Part A.

Locality-sensitive hashing (LSH)

Gionis et al.
VLDB 1999

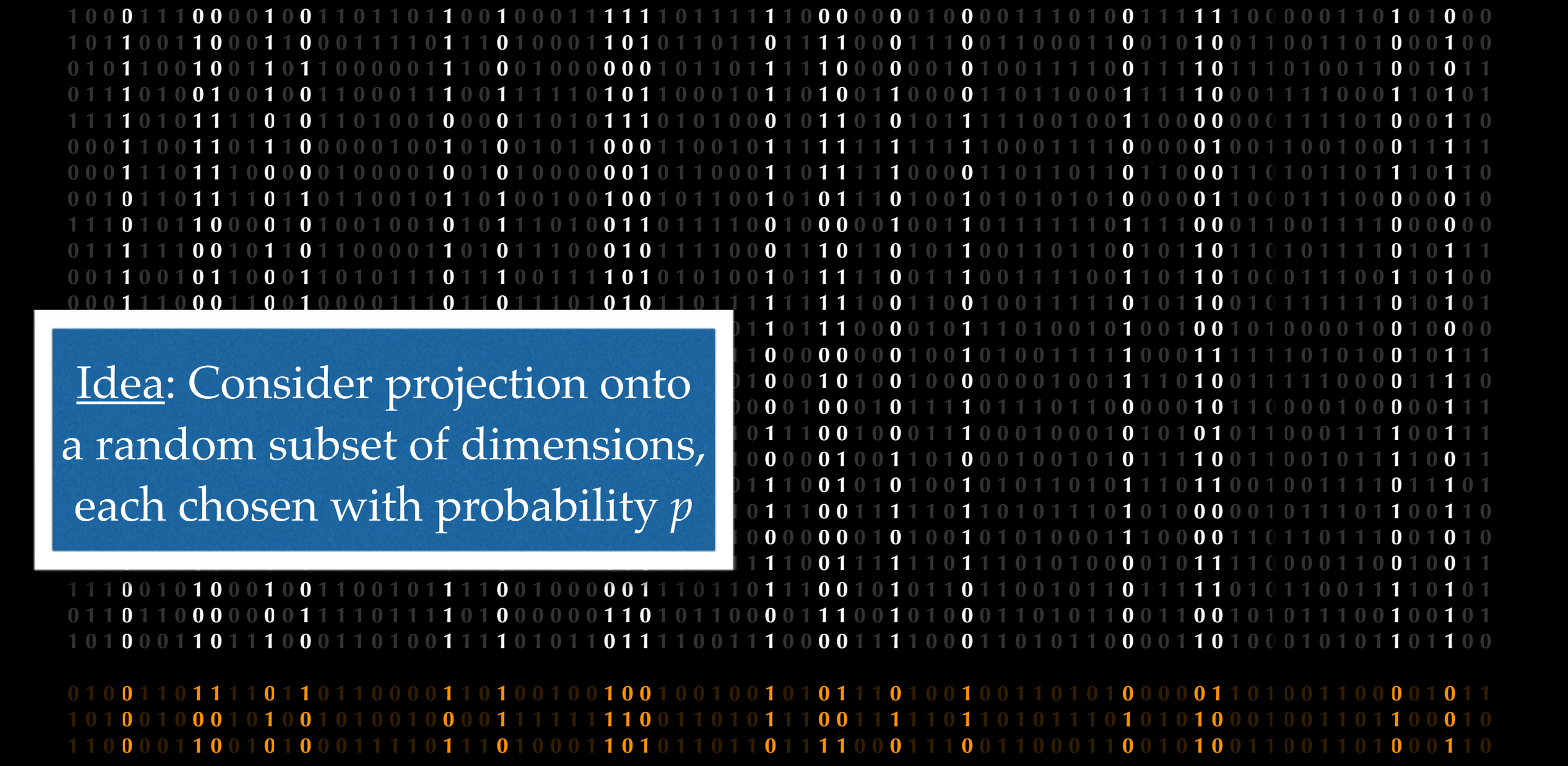
Locality-sensitive hashing



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

Gionis et al.
VLDB 1999

Locality-sensitive hashing



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

Gionis et al.
VLDB 1999

Locality-sensitive hashing

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

```
int lsh(int x) {return x & a}
```

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

Gionis et al.
VLDB 1999

Locality-sensitive hashing

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

```
int lsh(int x) {return x & a}
```

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

Candidate pair

Gionis et al.
VLDB 1999

Locality-sensitive hashing

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

```
int lsh(int x) {return x & a}
```

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

Candidate pair

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

Gionis et al.
VLDB 1999

Locality-sensitive hashing

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

```
int lsh(int x, int i) {return x & a[i]}
```

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

Candidate pair

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

Gionis et al.
VLDB 1999

Chance of collision?

$$
\begin{array}{ccc}
x \wedge a_1 & \stackrel{?}{=} & q \wedge a_1 \\
x \wedge a_2 & \stackrel{?}{=} & q \wedge a_2 \\
x \wedge a_3 & \stackrel{?}{=} & q \wedge a_3 \\
\vdots & & \vdots \\
x \wedge a_t & \stackrel{?}{=} & q \wedge a_t
\end{array}
$$

Gionis et al.
VLDB 1999

Chance of collision?

$$x \wedge a_1 \stackrel{?}{=} q \wedge a_1 \qquad \text{prob. } (1\text{-}p)^{d(q,x)}$$

$$x \wedge a_2 \stackrel{?}{=} q \wedge a_2 \qquad \text{prob. } (1\text{-}p)^{d(q,x)}$$

$$x \wedge a_3 \stackrel{?}{=} q \wedge a_3 \qquad \text{prob. } (1\text{-}p)^{d(q,x)}$$

$$\vdots \qquad\qquad \vdots$$

$$x \wedge a_t \stackrel{?}{=} q \wedge a_t \qquad \text{prob. } (1\text{-}p)^{d(q,x)}$$

Gionis et al.
VLDB 1999

Chance of collision?

$x \wedge a_1 \stackrel{?}{=} q \wedge a_1$ prob. $(1\text{-}p)^{d(q,x)}$

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

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

$\vdots \qquad\qquad \vdots$

$x \wedge a_t \stackrel{?}{=} q \wedge a_t$ prob. $(1\text{-}p)^{d(q,x)}$

Probability of no collision:

$(1\text{-}(1\text{-}p)^{d(q,x)})^t$

Gionis et al.
VLDB 1999

Chance of collision?

$x \wedge a_1 \stackrel{?}{=} q \wedge a_1$ prob. $(1\text{-}p)^{d(q,x)}$

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

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

$\vdots \qquad \vdots$

Probability of no collision:

$\left(1\text{-}(1\text{-}p)^{d(q,x)}\right)^t$

From math book: $(1\text{-}x)^{1/x} \longrightarrow 1/e$ as $x \longrightarrow 0$

Gionis et al.
VLDB 1999

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 $Q \bowtie_{cr} S$

Gionis et al.
VLDB 1999

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 $Q \bowtie_{cr} S$

Gionis et al.
VLDB 1999

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 $O(n^{1+1/c}D)$, expected;
Can be implemented in $O(n^{1+1/c}D/B)$ I/Os.

Can LSH work for large data?

- Quote from an ICDT 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 an ICDT 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 an ICDT 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?

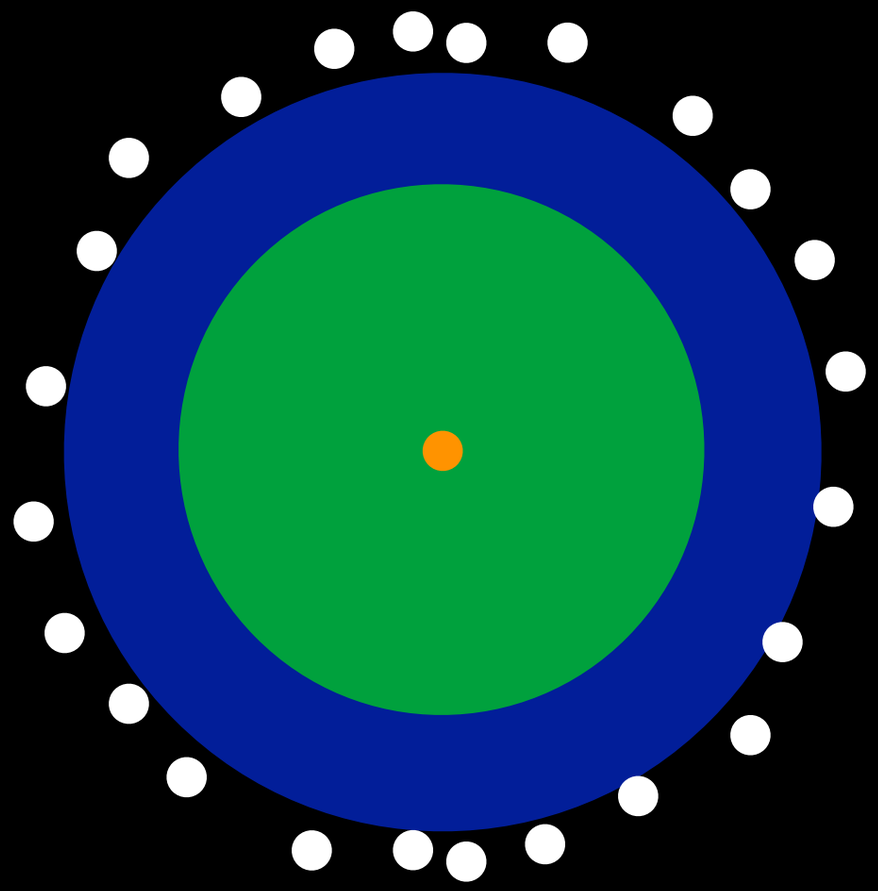
- Quote from an ICDT paper:
  "LSH needs large memory space and long processing time to achieve good performance when searching a massive dataset".

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 that uses $O(D^2n^2/(MB))$ I/Os is often better.

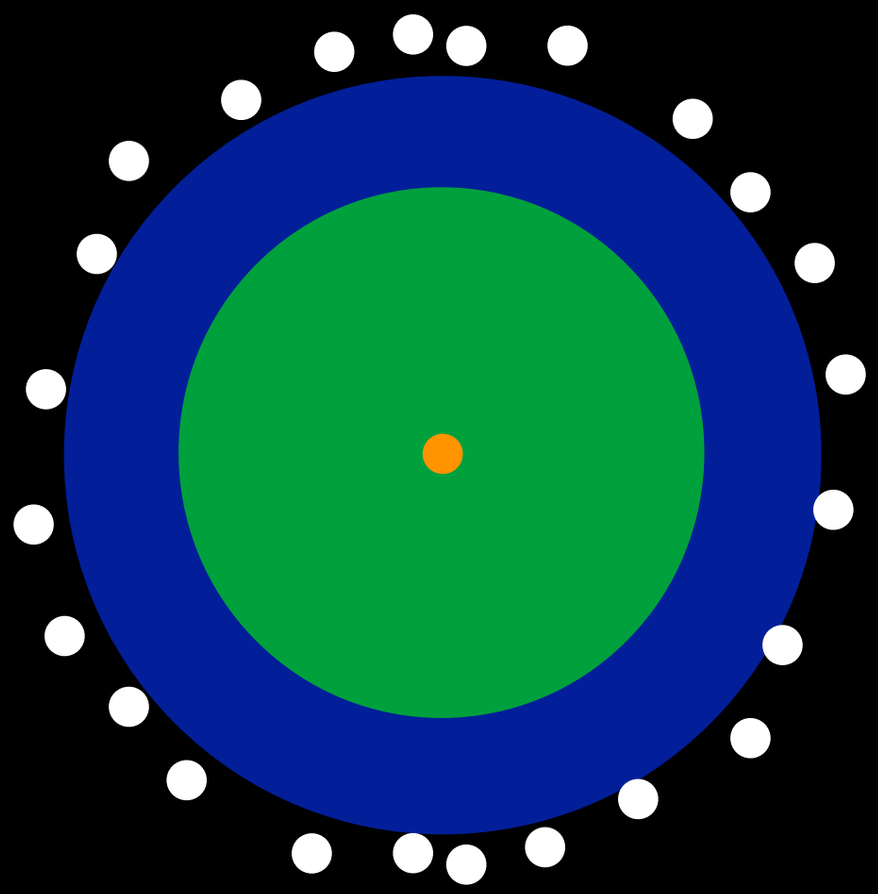
On pessimism

- 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:



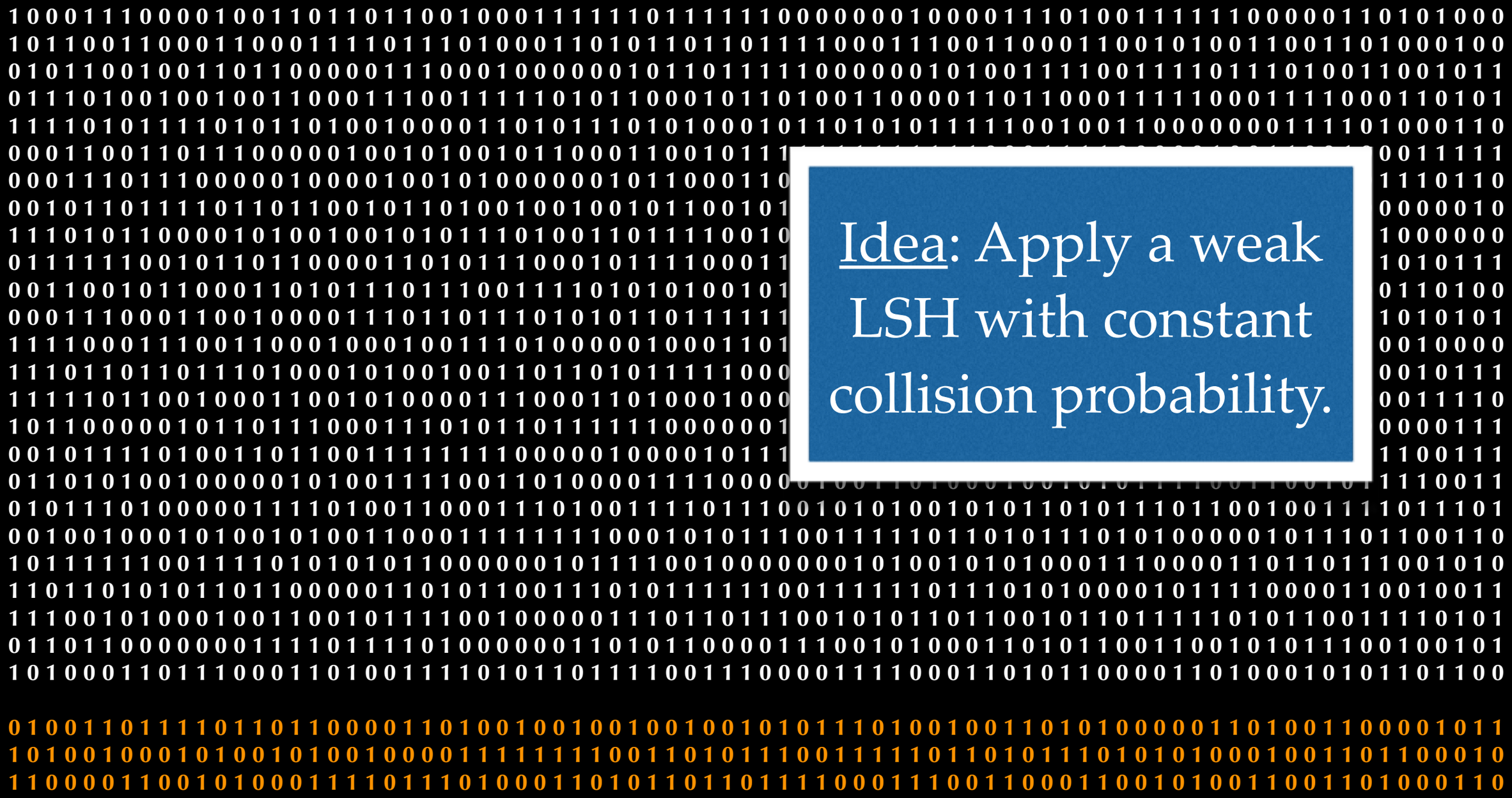
Price paid:
Also small collision probability at distance *r*, so need many repetitions.

Part B.

Cache-efficiency via recursion

Cautious LSH

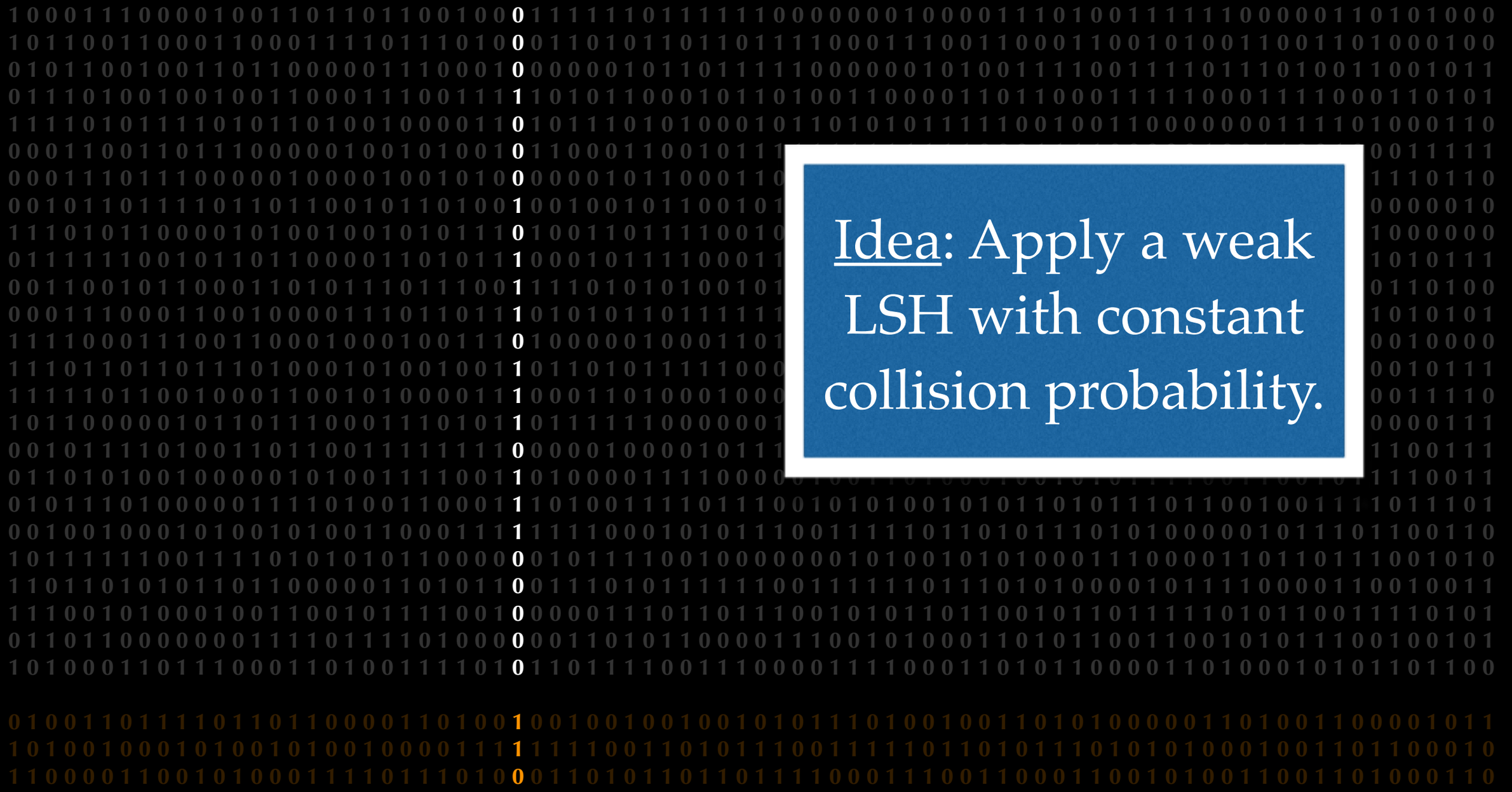
Joint work with Pham, Silvestri, and Stöckel



Idea: Apply a weak LSH with constant collision probability.

Cautious LSH

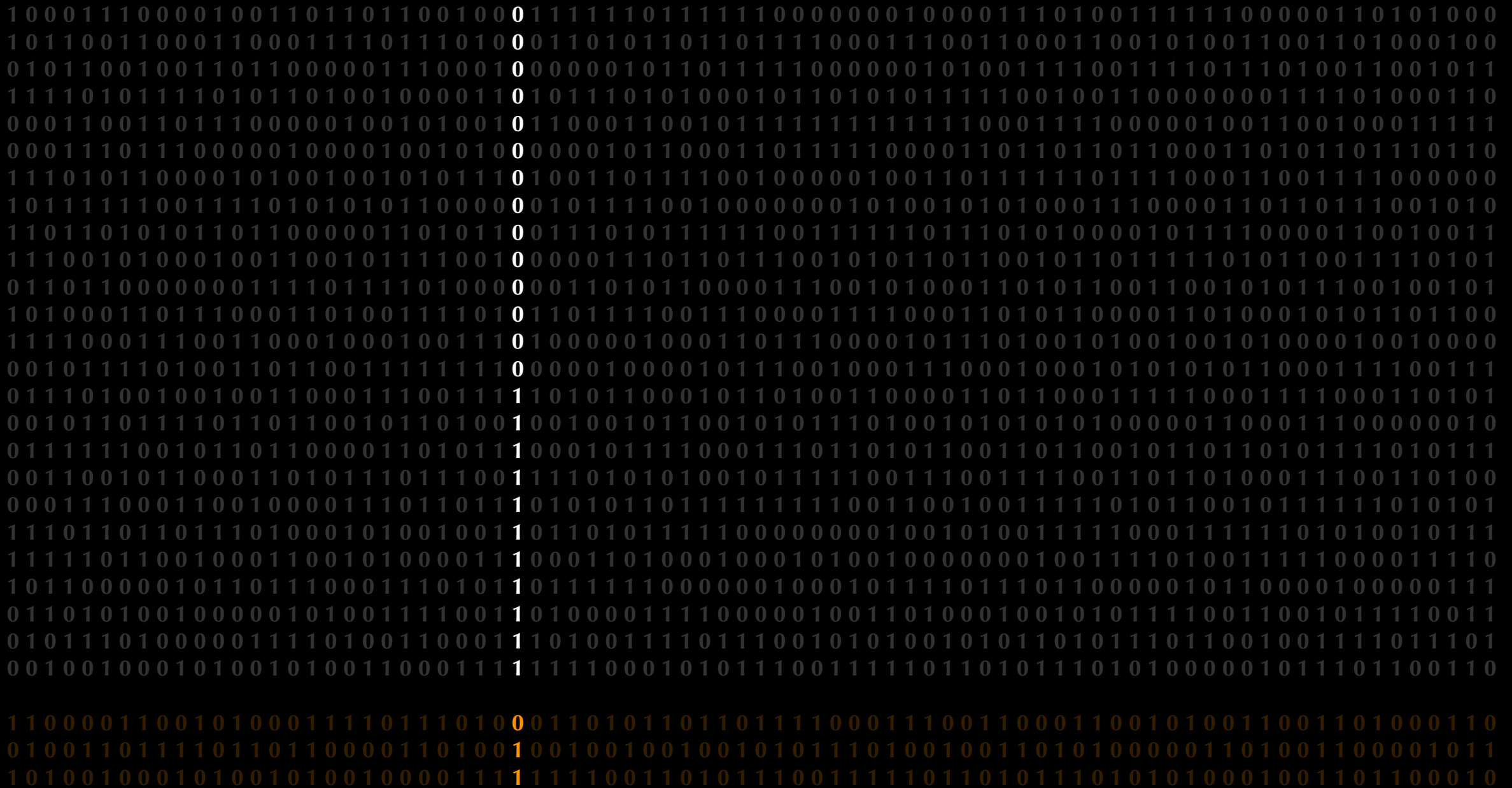
Joint work with Pham, Silvestri, and Stöckel



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

RECURSIVE SUBPROBLEM $Q_0 \bowtie_r S_0$

RECURSIVE SUBPROBLEM $Q_1 \bowtie_r S_1$

Cautious LSH

Joint work with Pham, Silvestri, and Stöckel

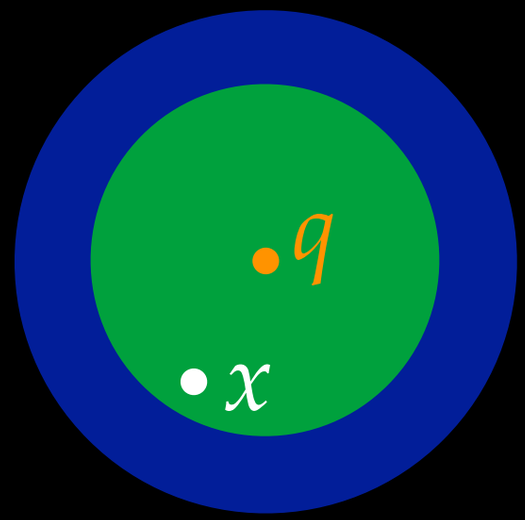
RECURSIVE SUBPROBLEM $Q_0 \bowtie_r S_0$

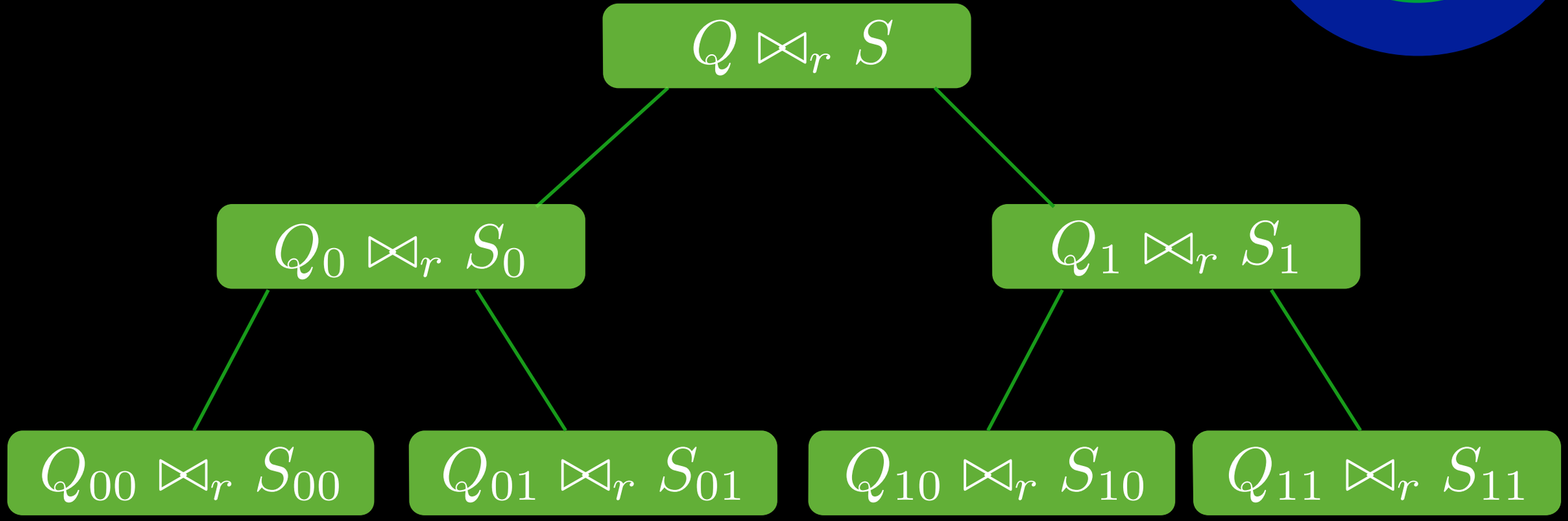
Subproblems of size M or less require no further I/Os

RECURSIVE SUBPROBLEM $Q_1 \bowtie_r S_1$

Example, cautious LSH

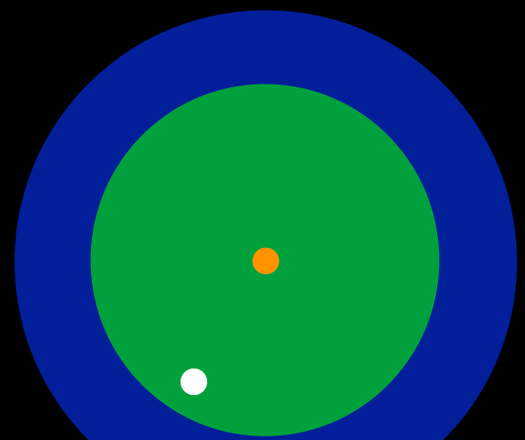
- Assume weak LSH collision prob. 0.9.
- Prob. that q and x collide i times is 0.9^i.
- If depth is log n, success probability is $\approx n^{-0.152}$.

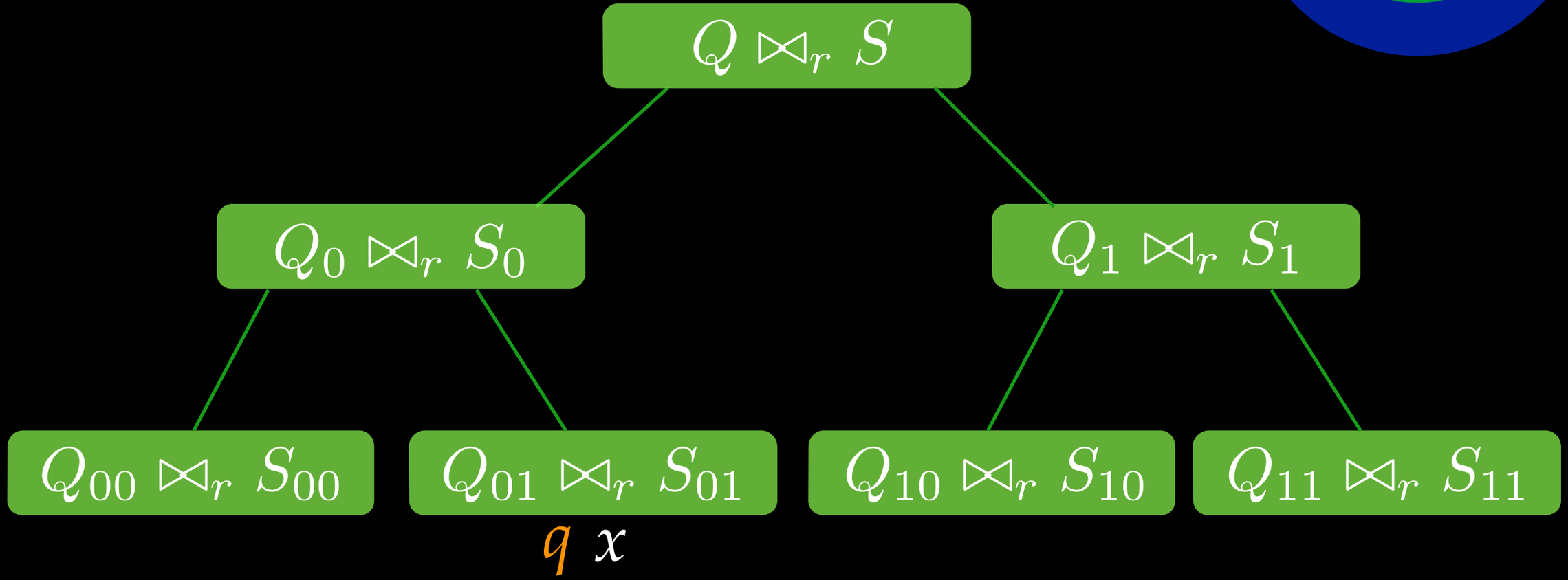




Example, cautious LSH

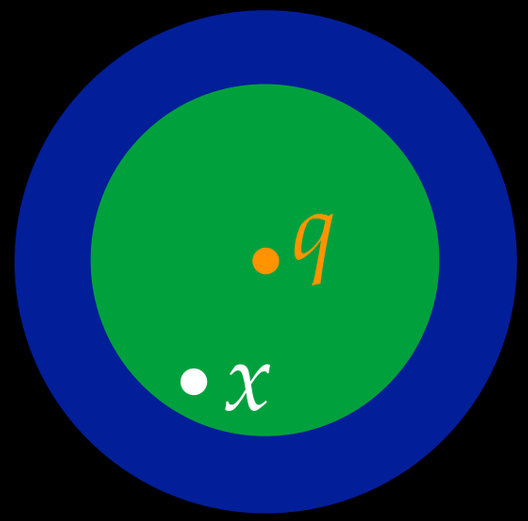
- Assume weak LSH collision prob. 0.9.
- Prob. that q and x collide i times is 0.9^i.
- If depth is log n, success probability is $\approx n^{-0.152}$.

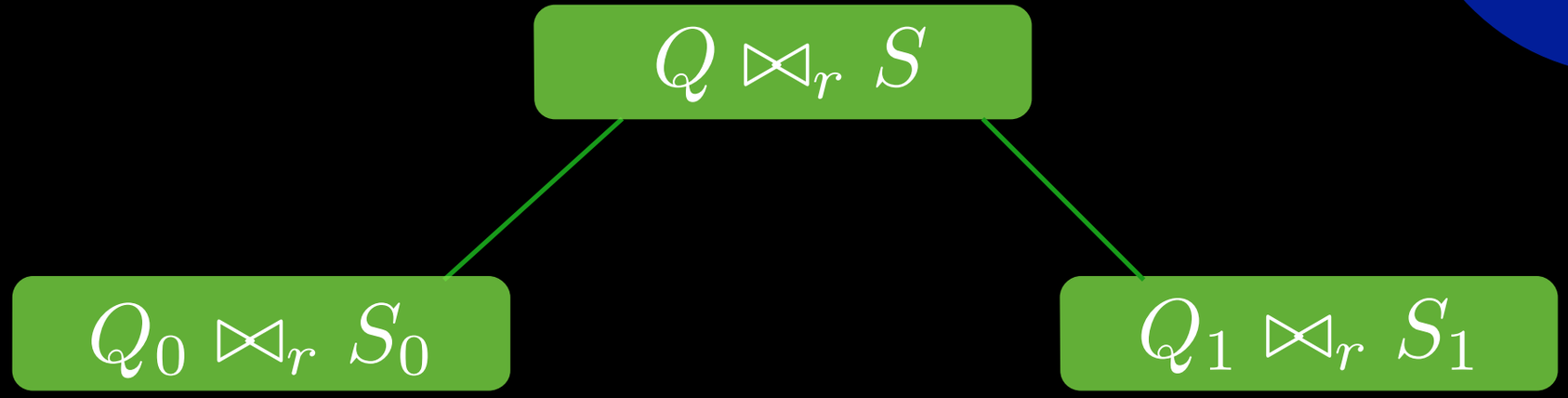




Example 2, cautious LSH

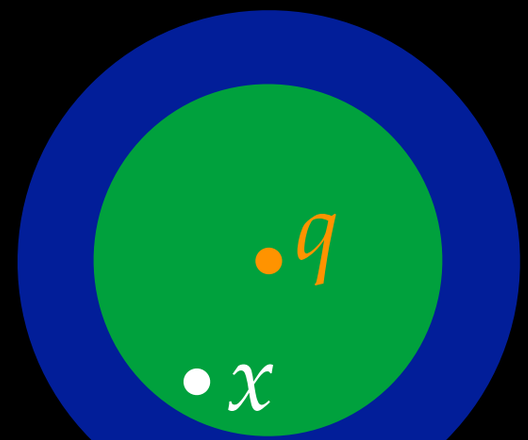
- Assume weak LSH collision prob. 0.5.
- Prob. that q and x collide i times is 0.5^i.

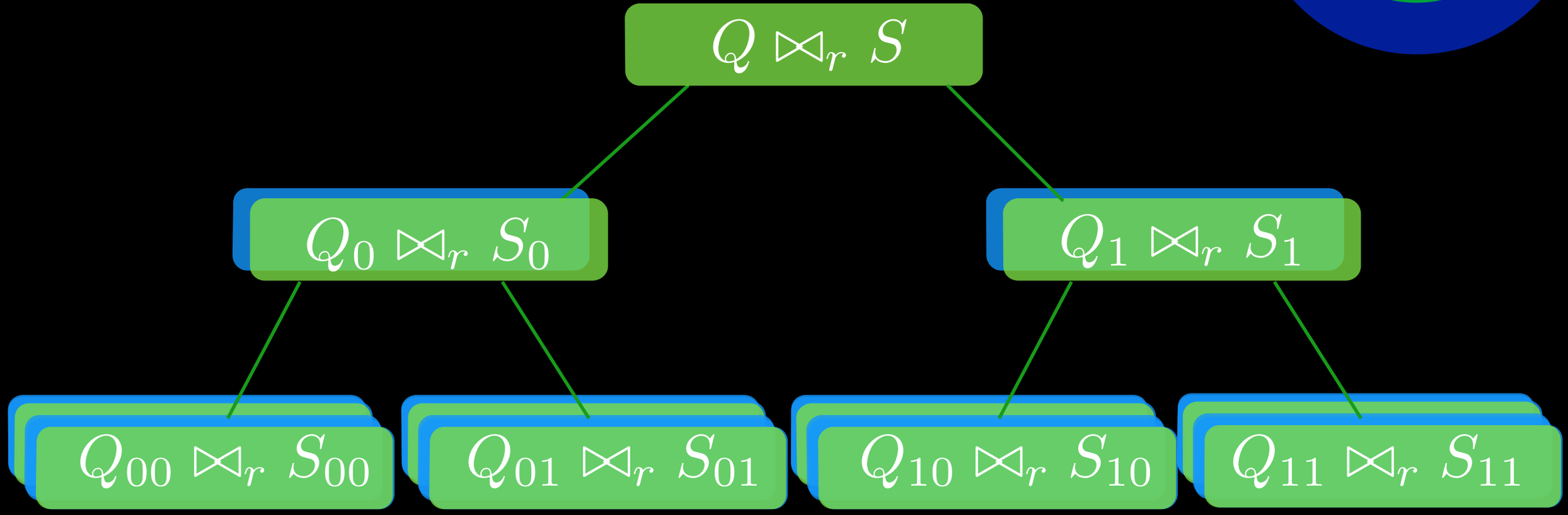




Example 2, cautious LSH

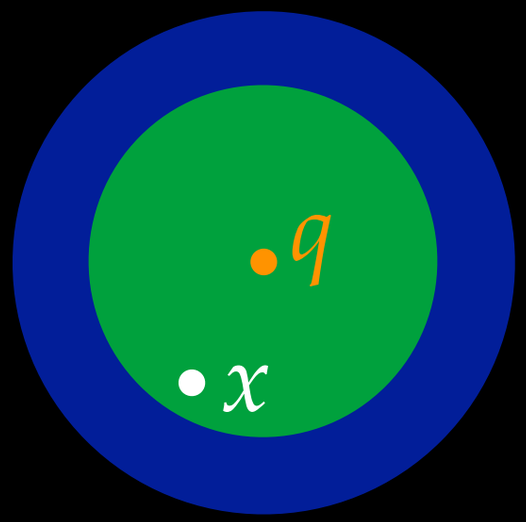
- Assume weak LSH collision prob. 0.5.
- Prob. that q and x collide i times is 0.5^i.
- Idea: Recurse *twice* at each node.





Example 2, cautious LSH

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

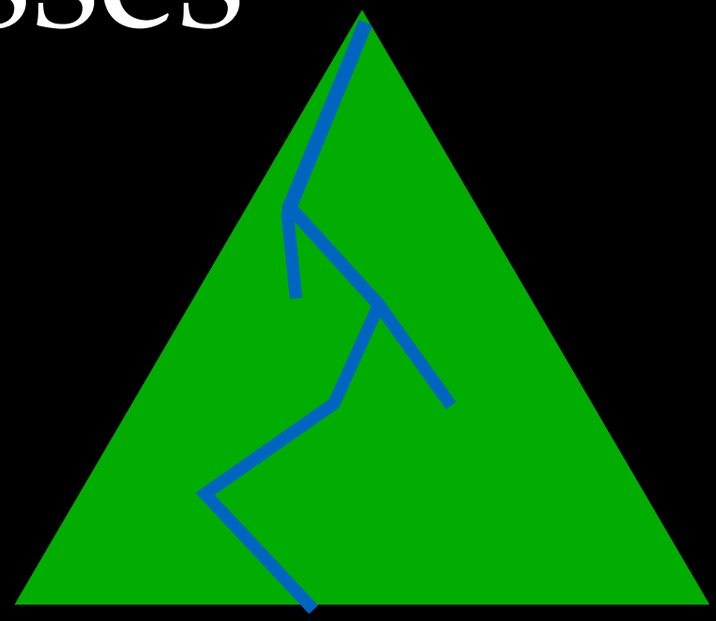
What is the expected TNT equivalent
of asteroid impacts during this talk?

Aside:
Don’t be fooled by great expectations

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

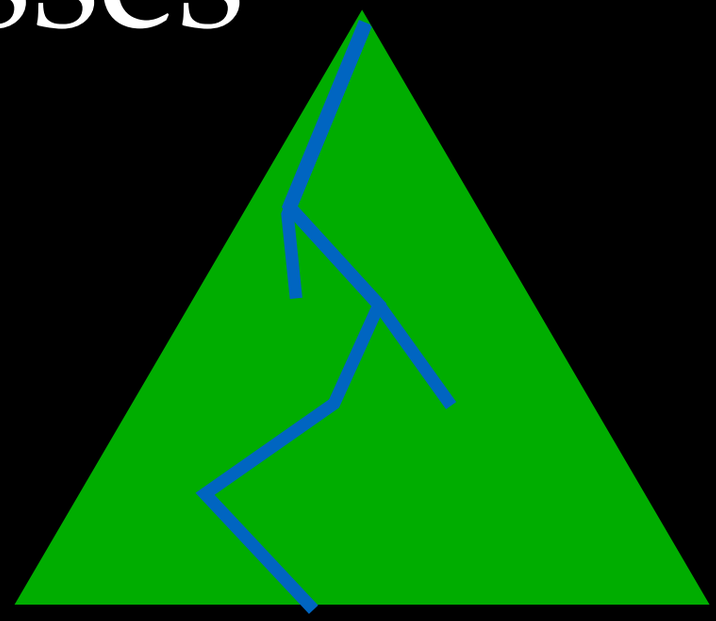
Branching processes

- Abstract setting:
  - 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

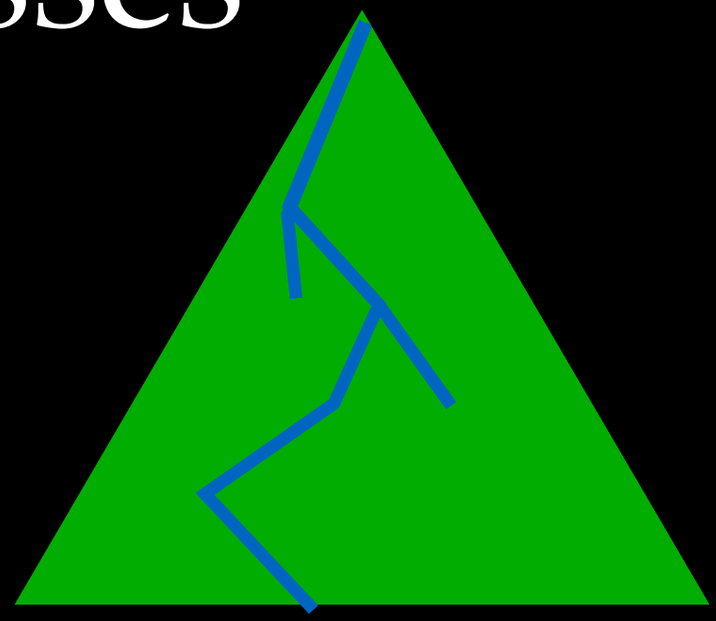
- Abstract setting:
  - One pair (q,x) at root problem.
  - Generate t subproblems such that (q,x) is "reproduced" in each with probability $\geq 1/t$.



- What is the probability that (q,x) is *extinct* at recursive level i?

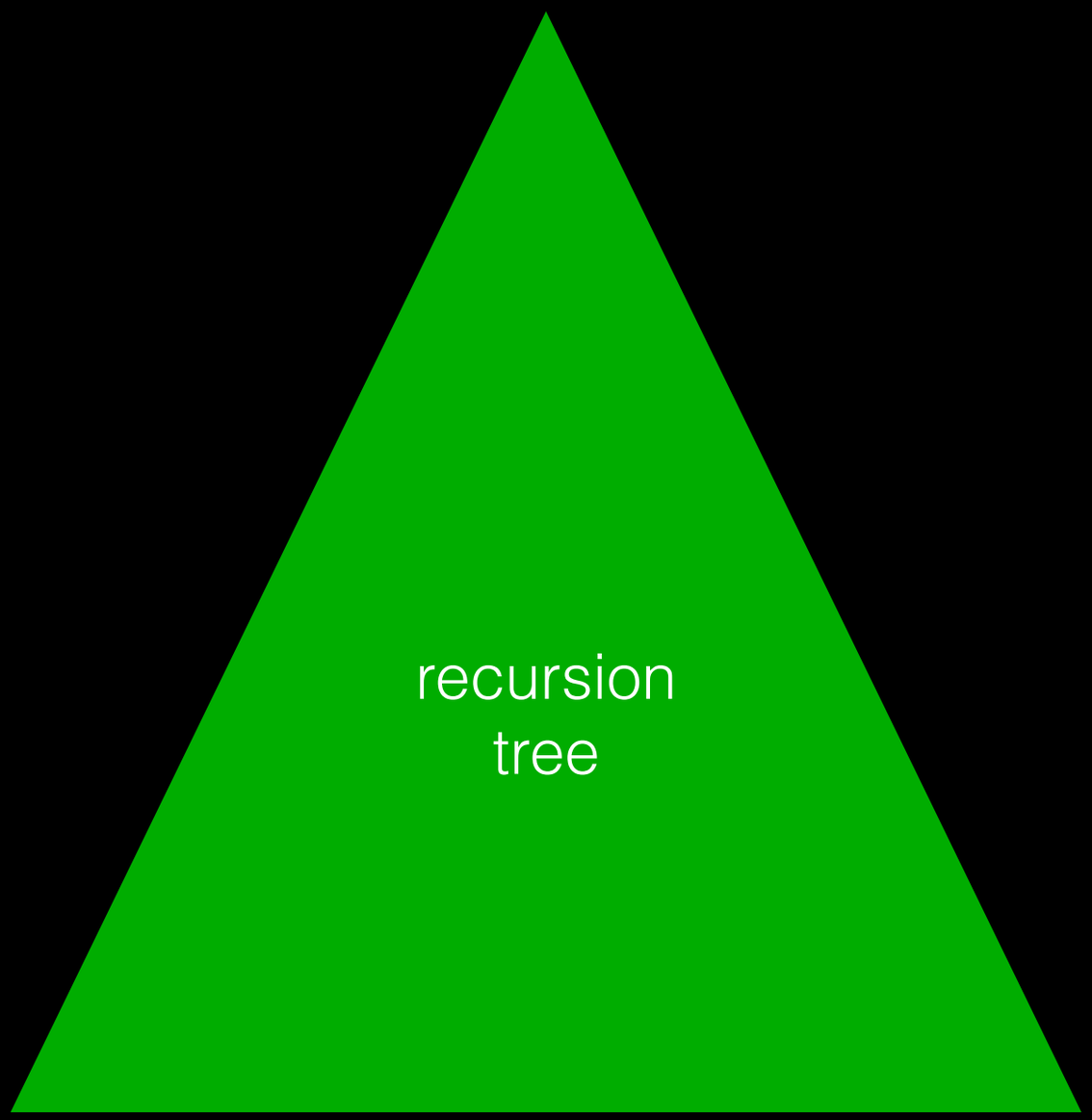
Branching processes

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

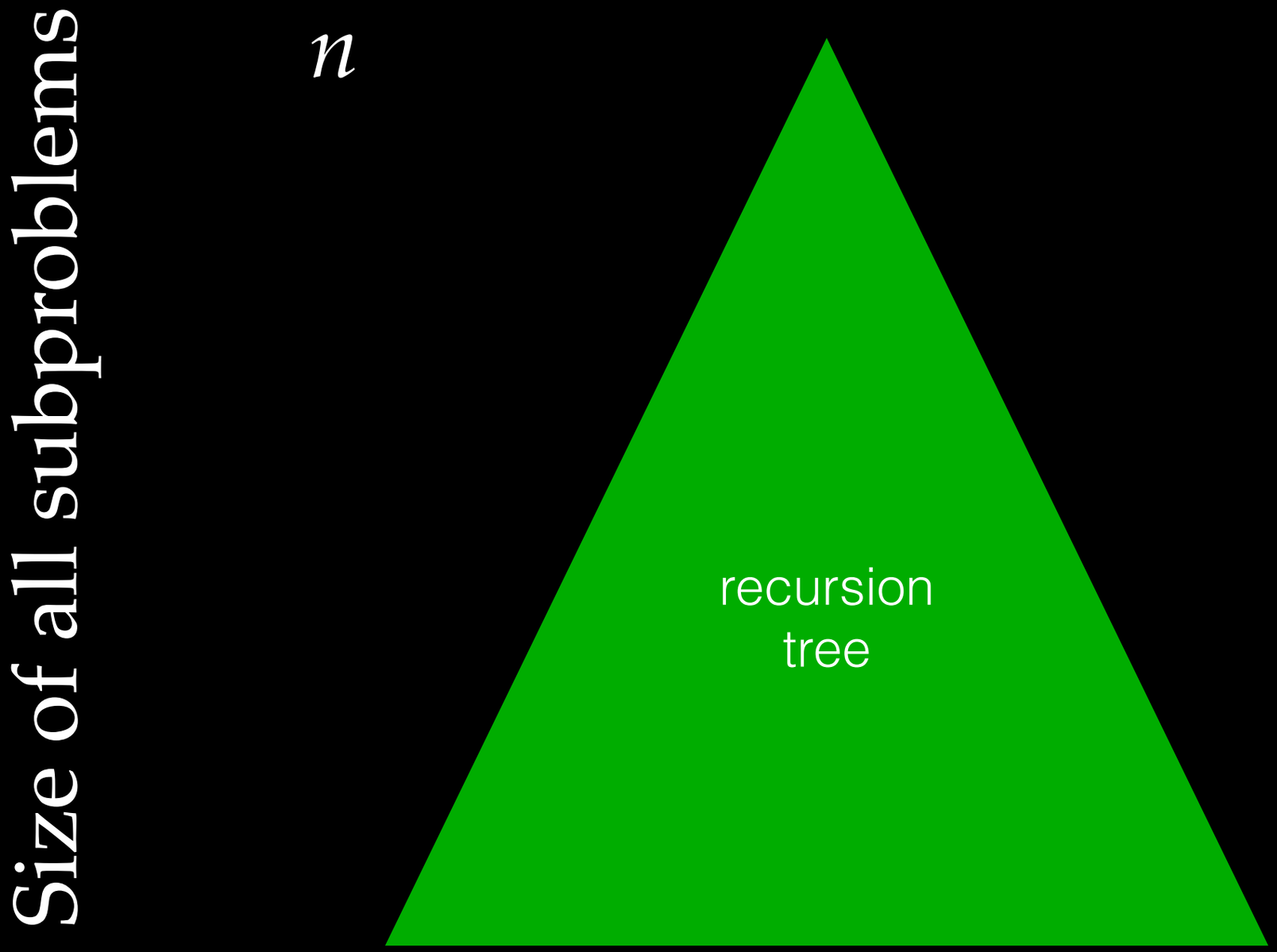


- What is the probability that (q,x) is *extinct* at recursive level i?
  - From theory of branching processes: $\Omega(1/\sqrt{i})$

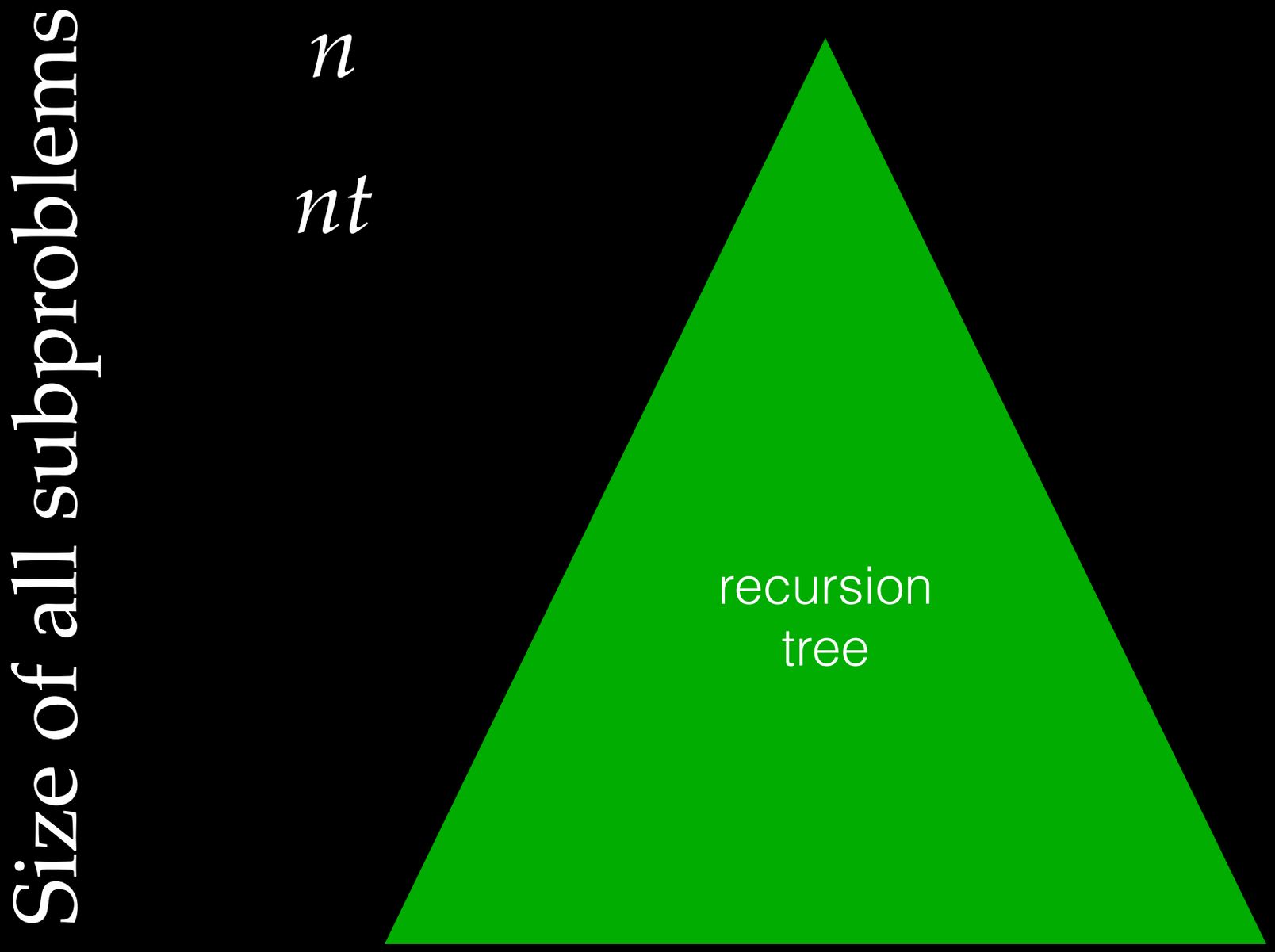
I/O complexity, sketch



I/O complexity, sketch



I/O complexity, sketch



I/O complexity, sketch

Size of all subproblems

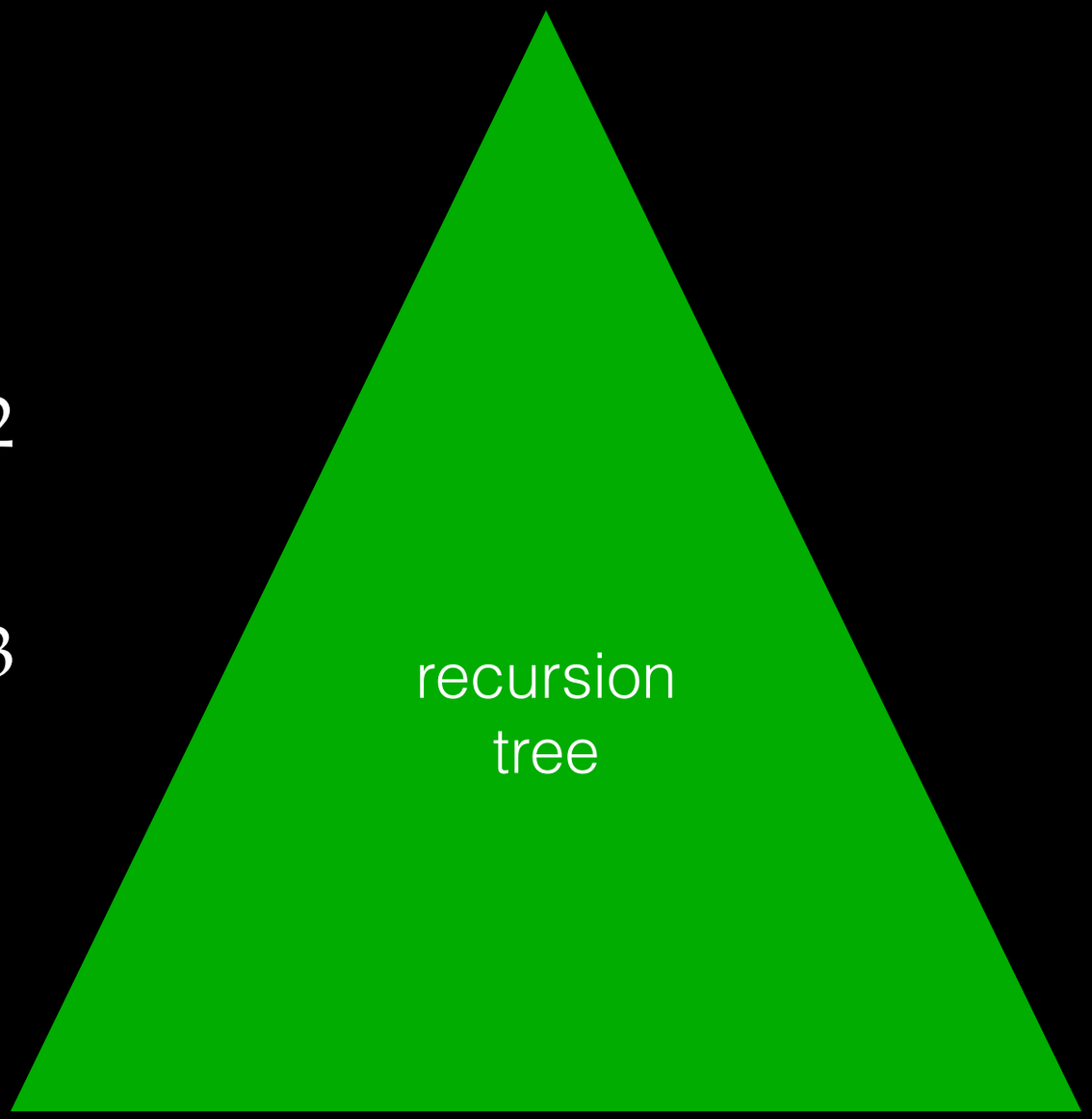
n

nt

nt^2

nt^3

⋮



I/O complexity, sketch

Size of all subproblems

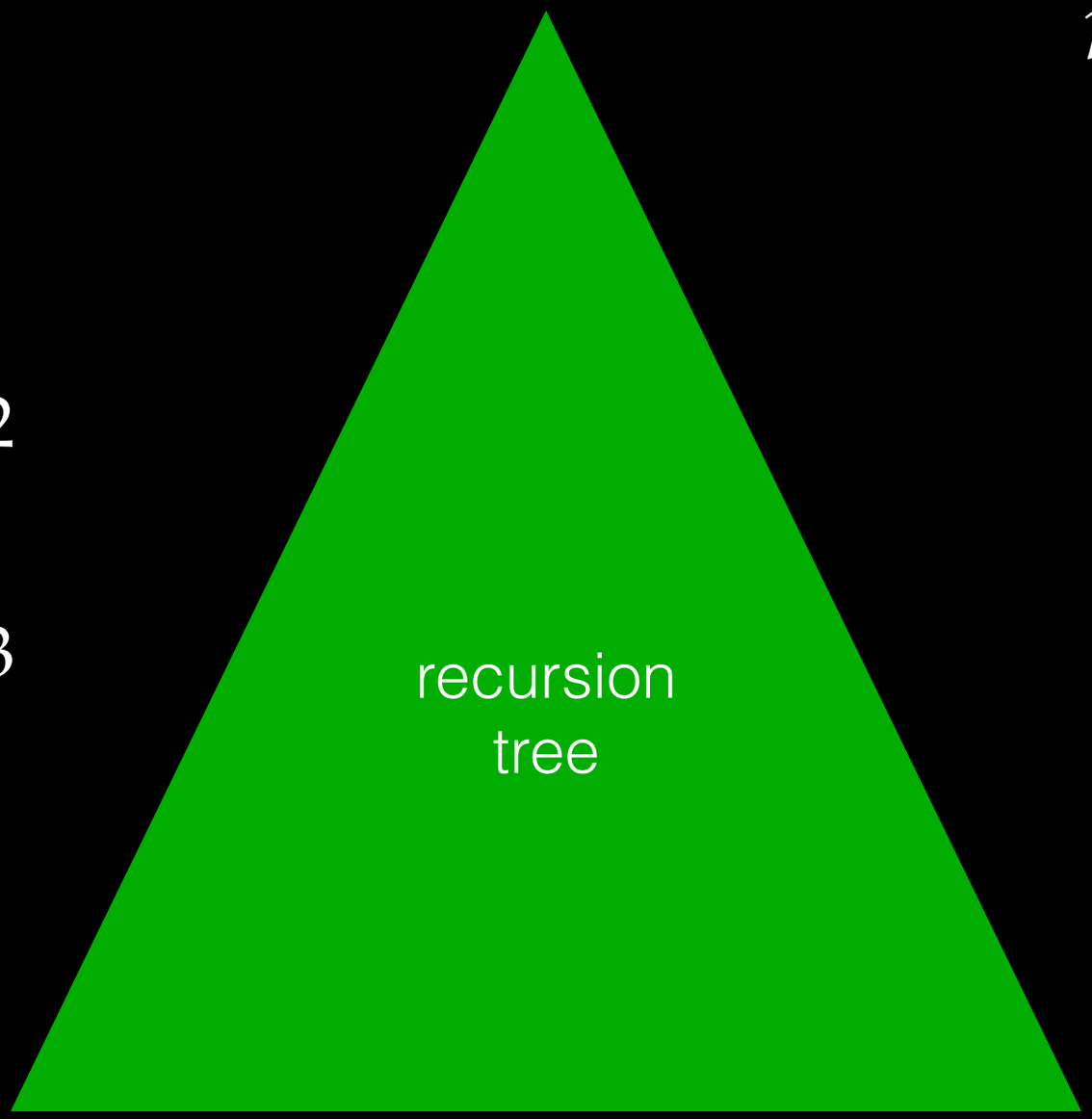
n

nt

nt^2

nt^3

⋮



n

Expected average
points at distance > cr.

I/O complexity, sketch

Size of all subproblems

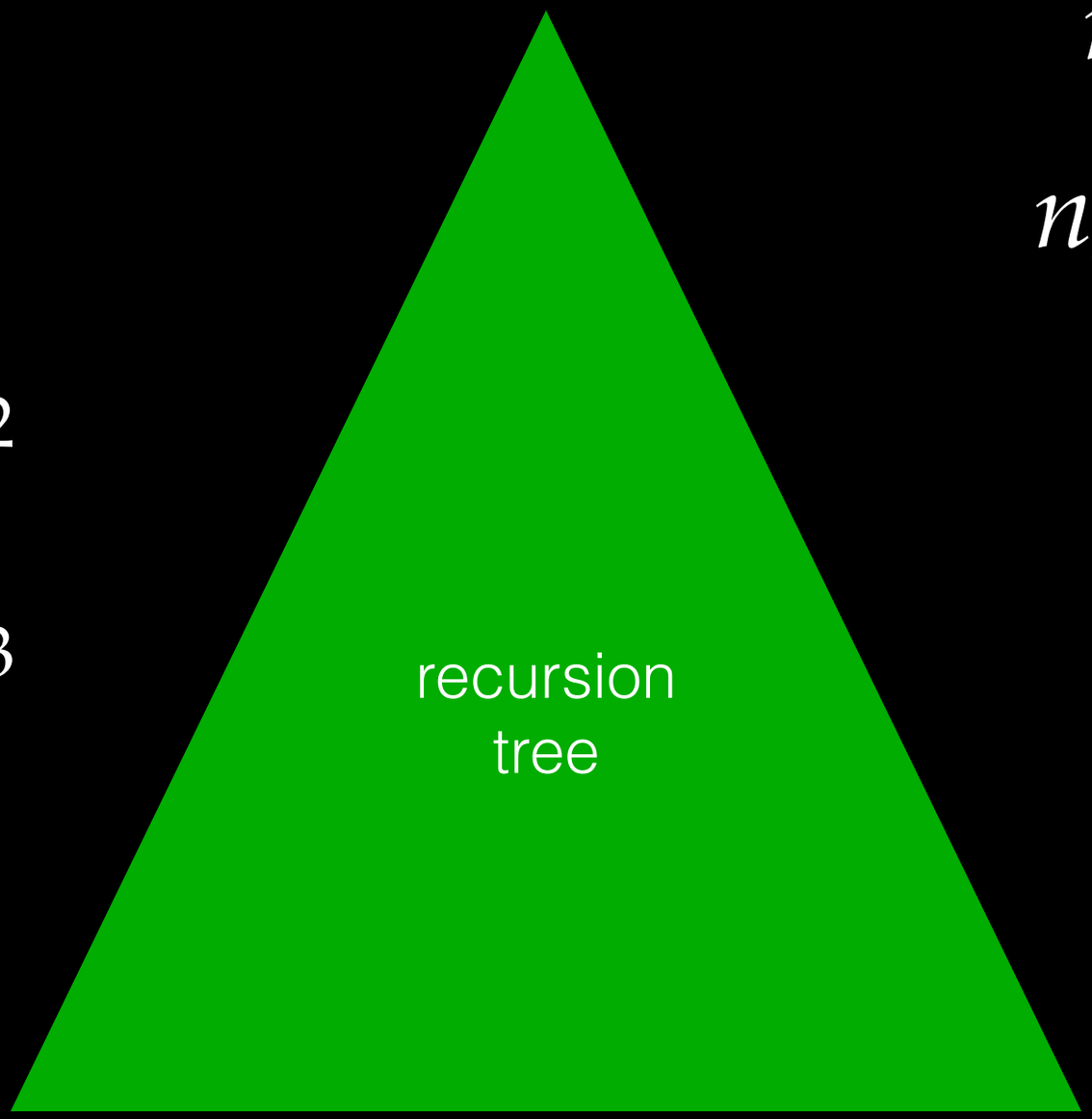
n

nt

nt^2

nt^3

⋮



n

n/t^c

Expected average # points at distance > cr.

I/O complexity, sketch

Size of all subproblems

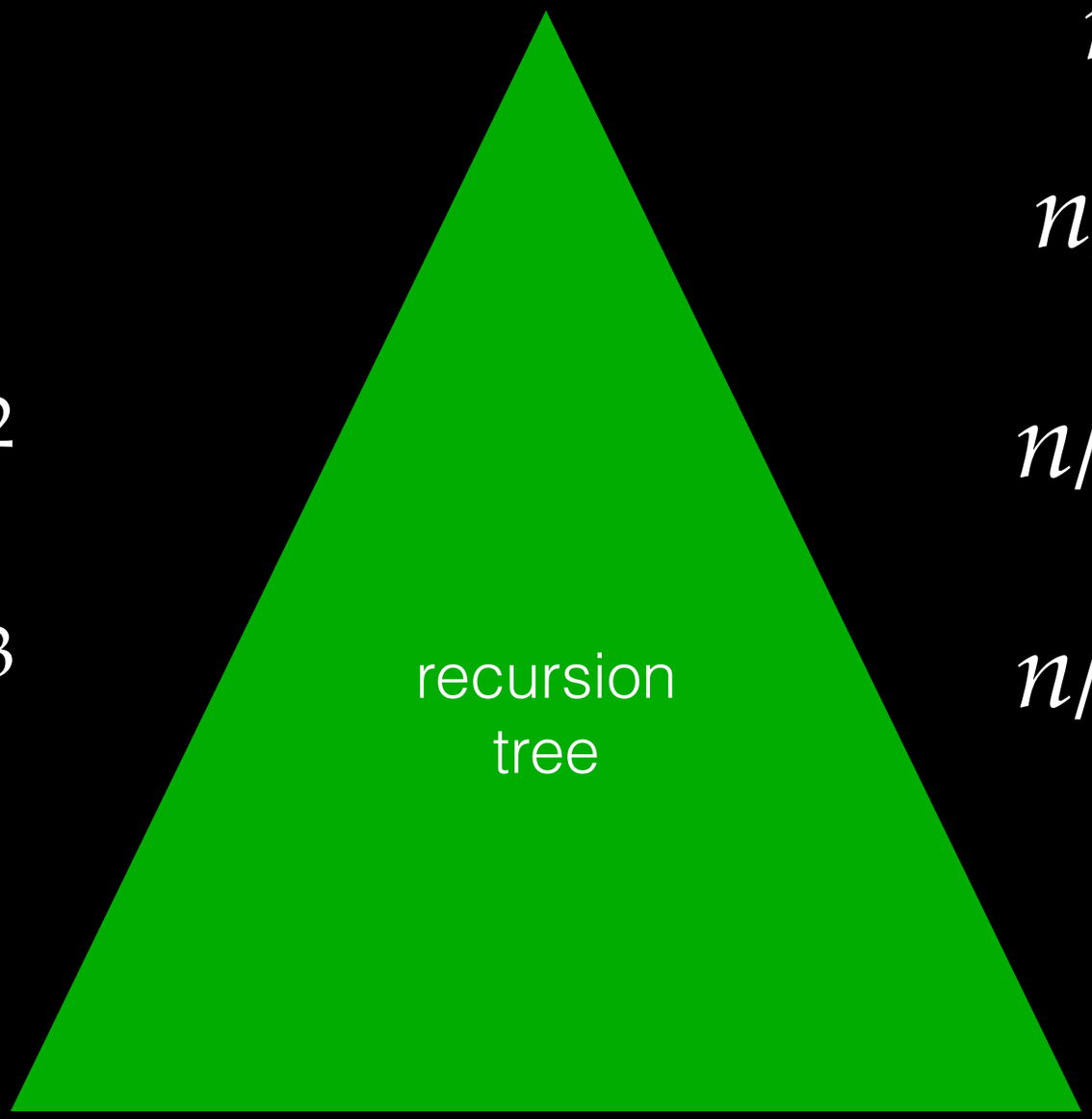
n

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

Size of all subproblems

n

nt

nt^2

nt^3

⋮

recursion tree

In-memory computation

n

n/t^c

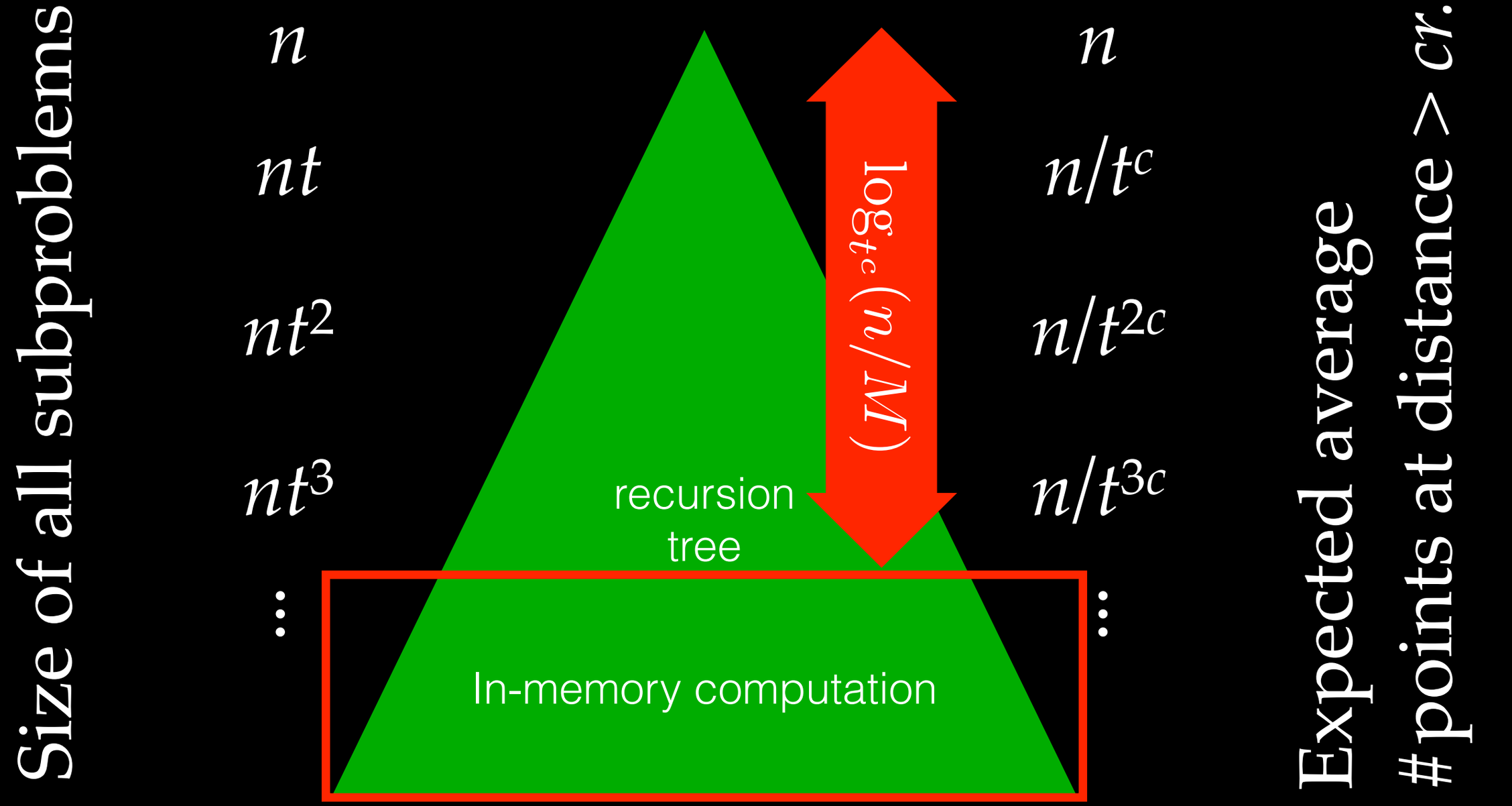
n/t^{2c}

n/t^{3c}

⋮

Expected average # points at distance $< cr$.

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:

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

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:

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

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:

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

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:

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

Part C:

Exact similarity join

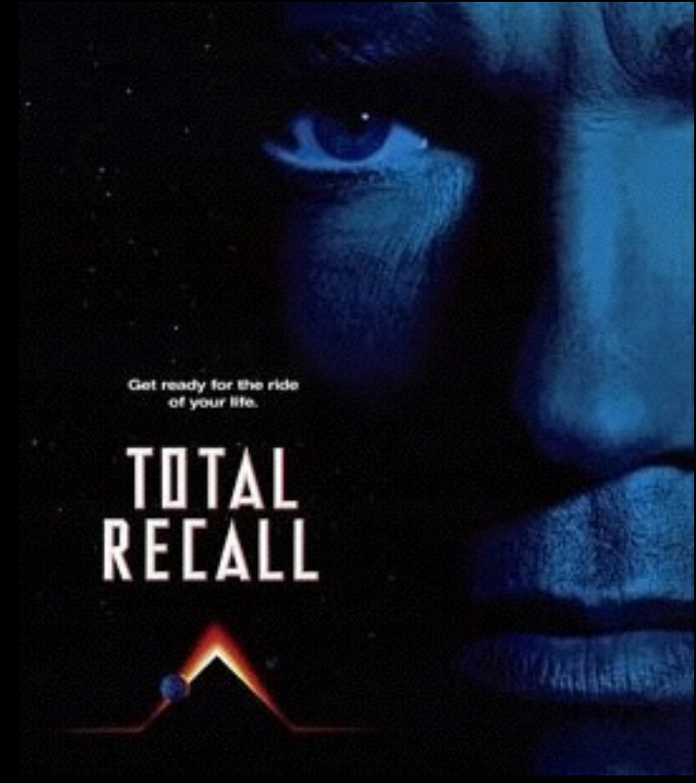
Total recall?

- False negative problem of classical LSH:
  No collision for a close pair with prob. $P > 0$.

Total recall?

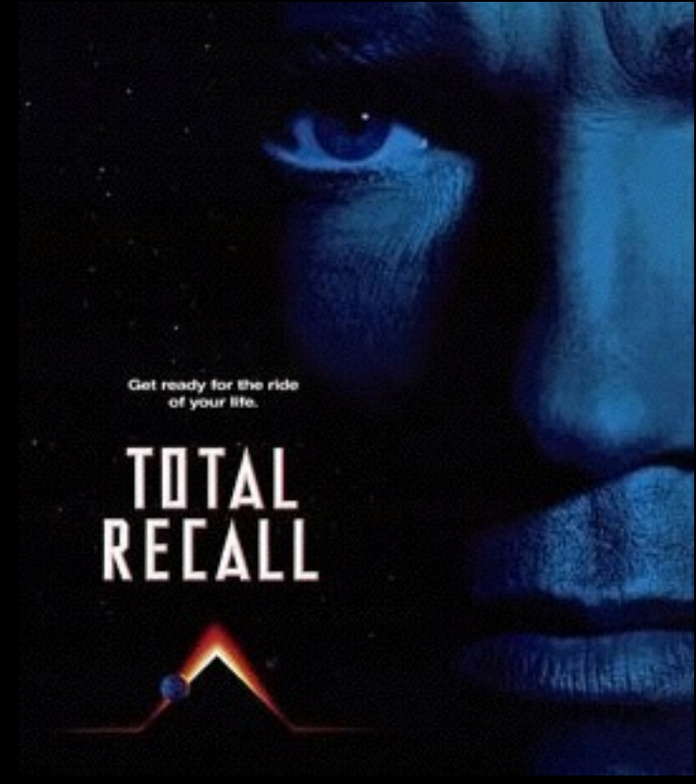
- False negative problem of classical LSH:
  No collision for a close pair with prob. $P > 0$.
- Recall can be made 1-P^k by repeating k times.

Total recall?



- False negative problem of classical LSH:
  No collision for a close pair with prob. $P > 0$.
- Recall can be made 1-P^k by repeating k times.
- Many recent papers deal with deterministic, *exact* similarity joins that achieve 100% recall.

Total recall?



- False negative problem of classical LSH:
  No collision for a close pair with prob. $P > 0$.
- Recall can be made 1-P^k by repeating k times.
- Many recent papers deal with deterministic, *exact* similarity joins that achieve 100% recall.

A Primitive Operator for Similarity Joins in Data Cleaning

Efficient Similarity Joins for Near Duplicate Detection

WWW 2007 / Track: Data Mining

Session: Similarity Search

Scaling Up All Pairs Similarity Search

Correlated LSH

- Idea: Choose hash functions h_1, h_2, … to ensure at least one collision if distance is at most r.

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 propose new algorithms for SSJoin. Our algorithms have two important features: They are exact, i.e., they always produce the correct answer, and they carry

City	*State*
Los Angeles	CA
Palo Alto	CA
San Diego	CA
Santa Barbara	CA
San Francisco	CA
Seattle	WA
...	...

City	*State*
Los Angeles	California
San Diego	California
Santa Barbara	California
San Francisco	California
Sacramento	California
Seattle	Washington
...	...

Arasu et al.
VLDB 2006

Basic correlated LSH
partitioning

Arasu et al.
VLDB 2006

Basic correlated LSH

partitioning

h_1

Arasu et al.
VLDB 2006

Basic correlated LSH

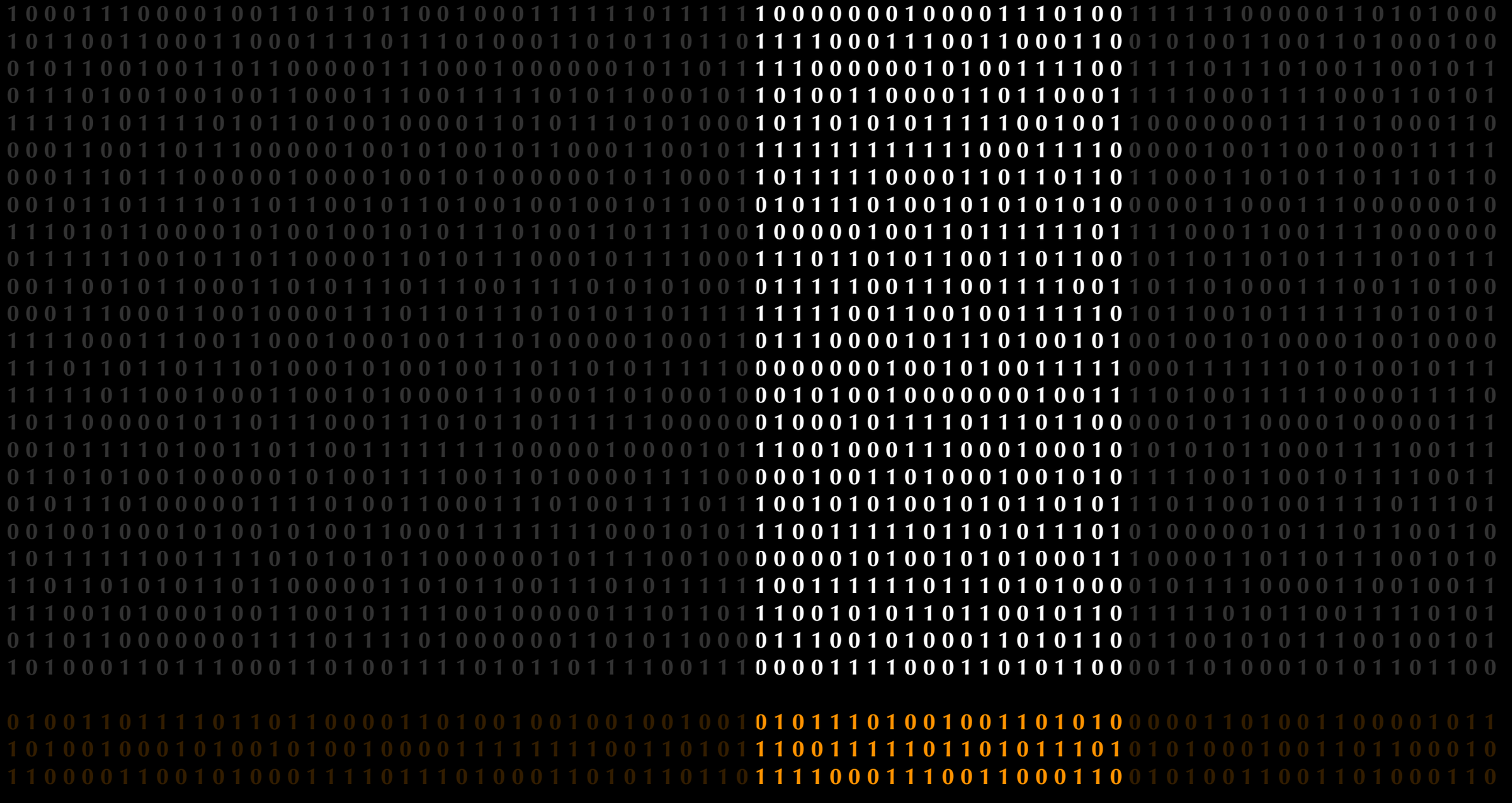
partitioning

h_2

Arasu et al.
VLDB 2006

Basic correlated LSH

partitioning

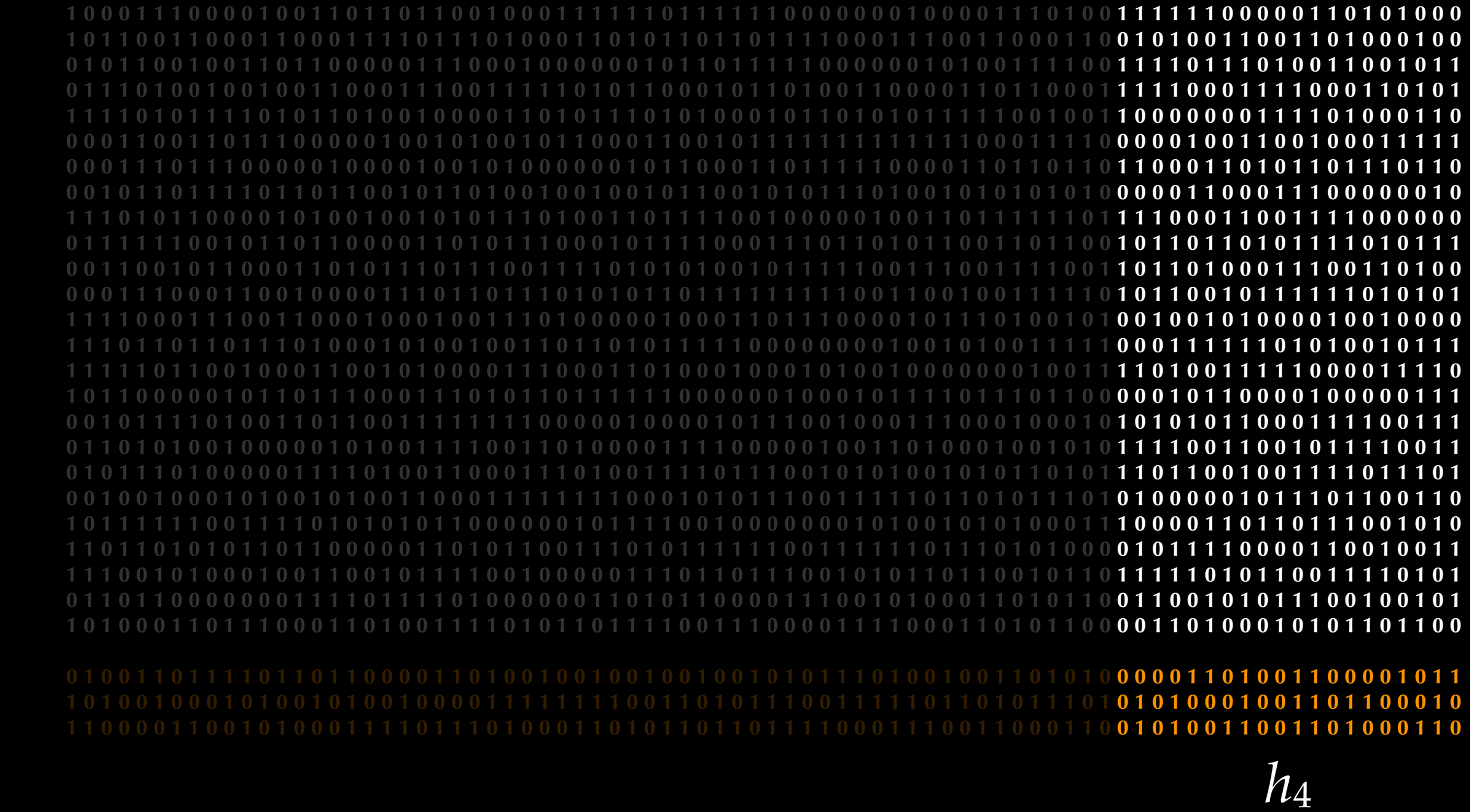


h_3

Arasu et al.
VLDB 2006

Basic correlated LSH

partitioning



h_4

Arasu et al.
VLDB 2006

Basic correlated LSH
partitioning

For Hamming distance ≤ 3,
a collision is guaranteed!

Arasu et al.
VLDB 2006

Basic correlated LSH
partitioning

For Hamming distance ≤ 3, a collision is guaranteed!

To bound probability of collision for distance > 3 randomly permute the dimensions

Arasu et al.
VLDB 2006

Basic correlated LSH
enumeration

h_{12}

Arasu et al.
VLDB 2006

Basic correlated LSH
enumeration

h_{13}

Arasu et al.
VLDB 2006

Basic correlated LSH
enumeration

h_{14}

Arasu et al.
VLDB 2006

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}

Arasu et al.
VLDB 2006

Result in LSH framework

- The Gionis et al. LSH achieves:

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

Arasu et al.
VLDB 2006

Result in LSH framework

- The Gionis et al. LSH achieves:

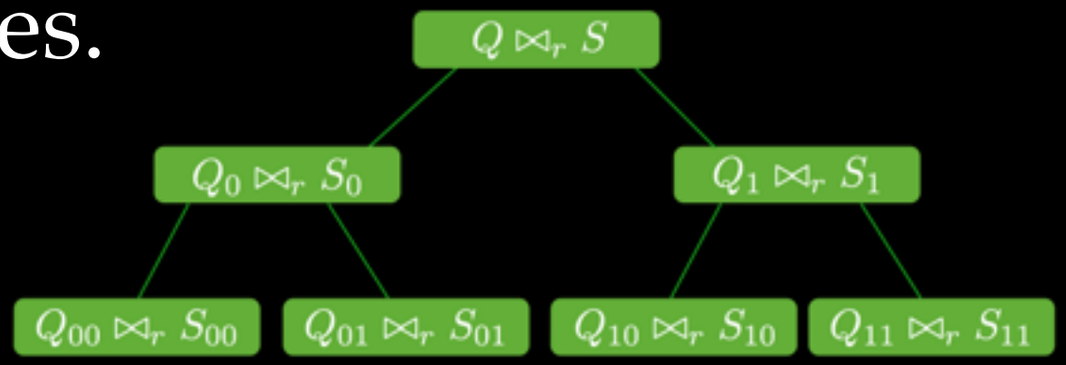
$$\Pr[h(q) = h(x) \mid d(q,x) = cr] \leq \Pr[h(q) = h(x) \mid d(q,x) = r]^{c}$$

- Arasu et al. construction combining partitioning, enumeration, and random permutation:

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

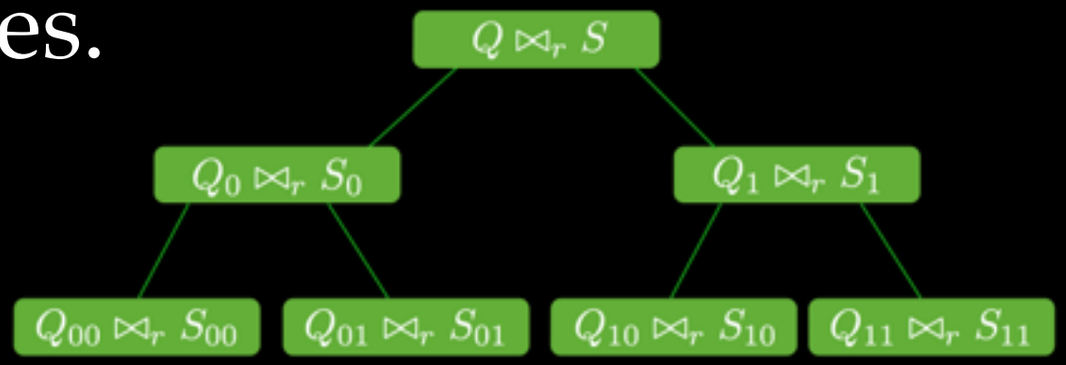
Conclusions

- Recursive application of LSH is a promising approach to large-scale similarity joins with theoretical guarantees.



Conclusions

- Recursive application of LSH is a promising approach to large-scale similarity joins with theoretical guarantees.



- Guarantees can include "total recall", though with present constructions this requires a constant $c > 3$.

Some open questions

- Practicality of recursive approach
  - Many choices made to enable theoretical reasoning should be reconsidered.
  - How about parallelism?

Some open questions

- Practicality of recursive approach
  - Many choices made to enable theoretical reasoning should be reconsidered.
  - How about parallelism?
- Better correlated LSH
  - Ideally matching the best performance possible without 100% recall guarantee.

Thank you!

IT UNIVERSITY OF COPENHAGEN

DET FRIE FORSKNINGSRÅD

European Research Council

Established by the European Commission

Supporting top researchers
from anywhere in the world

“There's nothing an agnostic can't do if he doesn't know whether he believes in anything or not.”

–Monty Python