Better Size Estimation for Sparse Matrix Products

Rasmus R. Amossen, Andrea Campagna, Rasmus Pagh

IT University of Copenhagen

September 1, 2010

The problem

Boolean matrix multiplication

The problem

Boolean matrix multiplication

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

The problem

Boolean matrix multiplication

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

$$a^3_{i,j} = \bigvee_{h \in \{1,\ldots,n\}} a^1_{i,h} \wedge a^2_{h,j}.$$

The problem

Boolean matrix multiplication

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

$$a^3_{i,j} = \bigvee_{h \in \{1,\ldots,n\}} a^1_{i,h} \wedge a^2_{h,j}.$$

How many true values are there in the resulting matrix?

The problem

Join size estimation

R_1	
a	b
1	1
1	2
3	2
12	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	5
3	12
4	12

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	5
3	2	2
3	2	5
12	3	12
2	3	12
2	4	12

The problem

Join size estimation

$\pi_{a,c}$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	5
3	2	2
3	2	5
12	3	12
2	3	12
2	4	12

$=$

R	
a	c
1	2
1	2
1	5
3	2
3	5
12	12
2	12
2	12

How many distinct rows?

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

$$\begin{pmatrix} \quad \end{pmatrix} \times \begin{pmatrix} \quad \end{pmatrix} = \begin{pmatrix} \quad \end{pmatrix}$$

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

$$\begin{pmatrix} 1 & & \\ & & \\ & & \end{pmatrix} \times \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix} = \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}$$

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

$$\begin{pmatrix} 1 & 1 & & \\ & & & \\ & & & \end{pmatrix} \times \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix} = \begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}$$

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

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

Equivalence of problems

R_1	
a	b
1	1
1	2
3	2
4	3
2	3
2	4

$\bowtie$

R_2	
b	c
1	2
2	2
2	3
3	4
4	4

$=$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	3
3	2	2
3	2	3
4	3	4
2	3	4
2	4	4

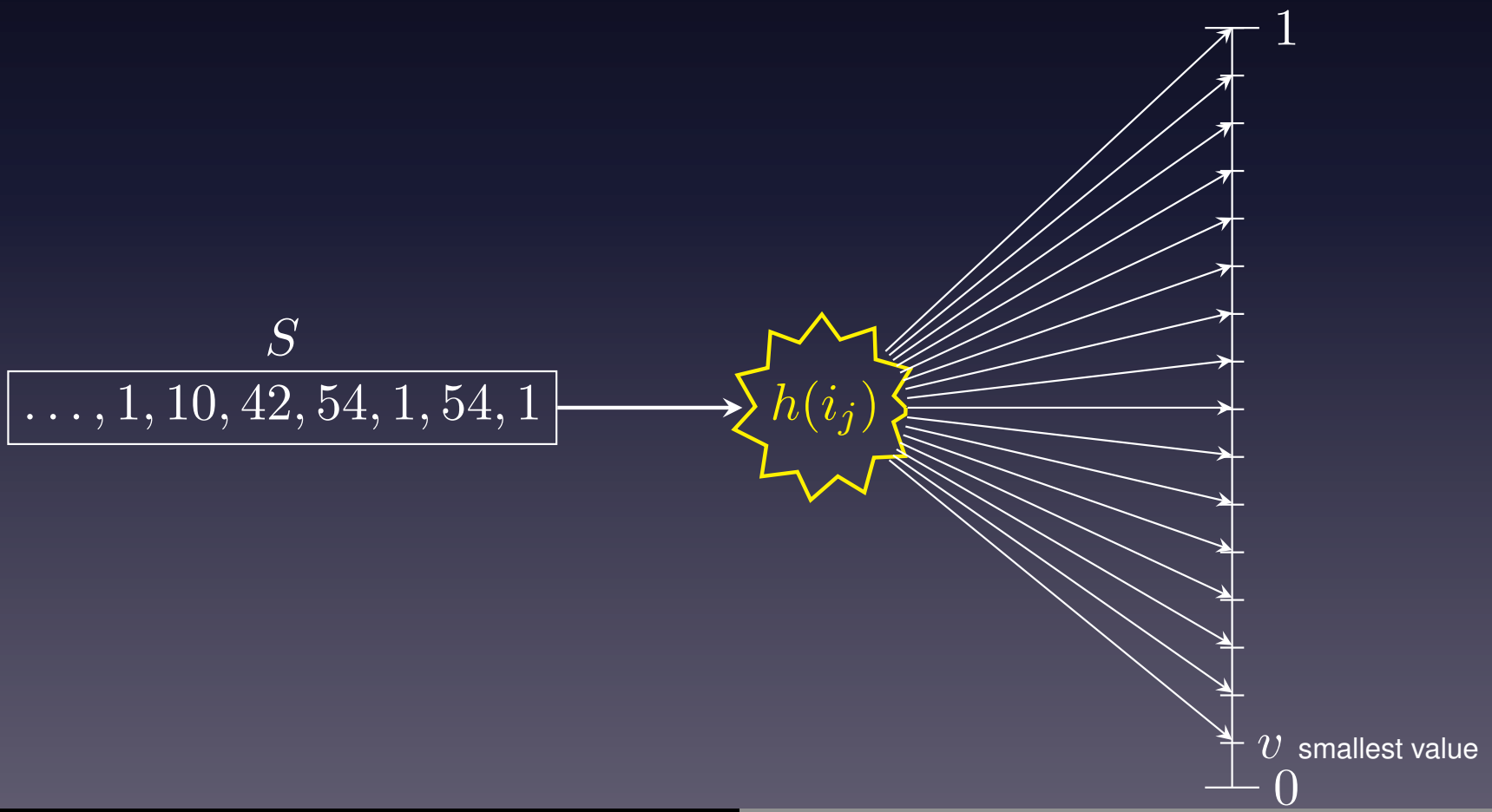
$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Previous and our results

- **Previous results:** $O(n/\varepsilon^2)$ for any fixed constant error probability (Cohen);
- **Our contribution:** $O(n)$ for any $\varepsilon > 4/\sqrt[4]{n}$ and polynomially small error probability for any fixed ε.

The tools

- S is a stream of elements $i \in [n]$;
- $h : [n] \to (0, 1]$ is a pairwise independent hash function;
- If h distributes the values evenly, then $1/v$ distinct items.



The tools

- S is a stream of elements $i \in [n]$;
- $h : [n] \to (0, 1]$ is a pairwise independent hash function;
- If h ditributes evenly the values, then $1/v$ distinct items.

The tools

- S is a stream of elements $i \in [n]$;
- $h : [n] \to (0, 1]$ is a pairwise independent hash function;
- If h ditributes evenly the values, then $1/v$ distinct items.

- **Problem:** large variance;

The tools

- S is a stream of elements $i \in [n]$;
- $h : [n] \to (0, 1]$ is a pairwise independent hash function;
- If h ditributes evenly the values, then $1/v$ distinct items.

- **Problem:** large variance;
- **Solution:**

The tools

- S is a stream of elements $i \in [n]$;
- $h : [n] \to (0, 1]$ is a pairwise independent hash function;
- If h ditributes evenly the values, then $1/v$ distinct items.

- **Problem:** large variance;
- **Solution:**
  - take the k smallest values;

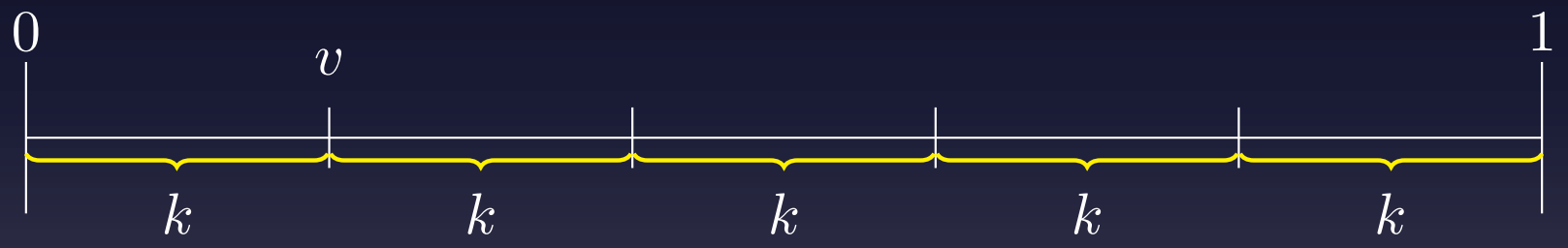
The tools

- S is a stream of elements $i \in [n]$;
- $h : [n] \to (0, 1]$ is a pairwise independent hash function;
- If h ditributes evenly the values, then $1/v$ distinct items.

- **Problem:** large variance;
- **Solution:**
  - take the k smallest values;
  - If v is the value of the kth smallest, estimate k/v distinct items.

The tools

- **Problem:** large variance;
- **Solution:** take the k smallest values (Bar-Yossef et al.);



- Density is $v/k \to k/v$ estimated distinct items.

The tools

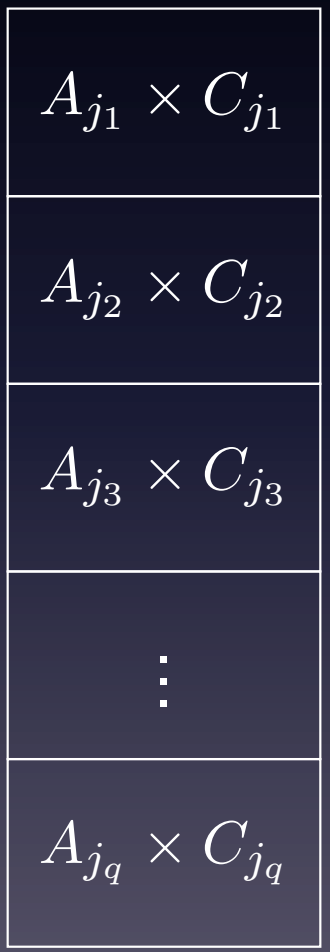
R	
a	c
1	2
1	2
1	5
3	2
3	5
12	12
2	12
2	12

The tools

- **Problem:** find a suitable hash function;
- **Solution:**
  - draw independently at random $h_1 : U \to (0, 1]$ and $h_2 : U \to (0, 1]$ from a family of pairwise independent hash functions;
  - build $h(x, y) = (h_1(x) - h_2(y)) \mod 1$;
  - $h(x, y)$ built in this way is *pairwise independent*.

Notation

$\pi_{a,c}(R_1 \bowtie R_2)$



- $|R_1| + |R_2| = n$;
- $A_j = \{i | (i, j) \in R_1\}$;
- $C_j = \{i | (j, i) \in R_2\}$;
- Note: $\pi_{a,c}(R_1 \bowtie R_2) = \bigcup_j A_j \times C_j$.

Question: how to generate the pairs in $A_j \times C_j$ for any given j?

The algorithm

$R_1 \bowtie R_2 =$

$R_{1,2}$		
a	b	c
2	3	12
1	1	2
1	2	2
12	3	12
1	2	5
3	2	2
2	4	12
3	2	5

The algorithm

$R_1 \bowtie R_2 =$

$R_{1,2}$		
a	b	c
2	3	12
1	1	2
1	2	2
12	3	12
1	2	5
3	2	2
2	4	12
3	2	5

$\longrightarrow$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	5
3	2	2
3	2	5
12	3	12
2	3	12
2	4	12

The algorithm

$R_1 \bowtie R_2 =$

$R_{1,2}$		
a	b	c
2	3	12
1	1	2
1	2	2
12	3	12
1	2	5
3	2	2
2	4	12
3	2	5

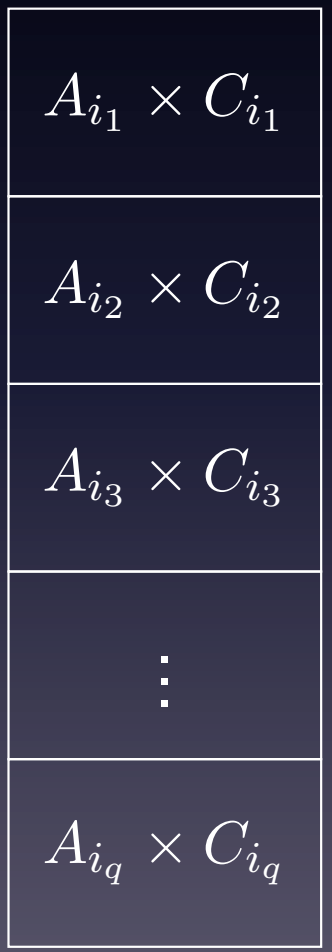
$\longrightarrow$

$R_{1,2}$		
a	b	c
1	1	2
1	2	2
1	2	5
3	2	2
3	2	5
12	3	12
2	3	12
2	4	12

- First step: sort according to b;
- $O(n)$ in the RAM model (e.g. using hashing);
- $sort(n)$ I/Os in the I/O model.

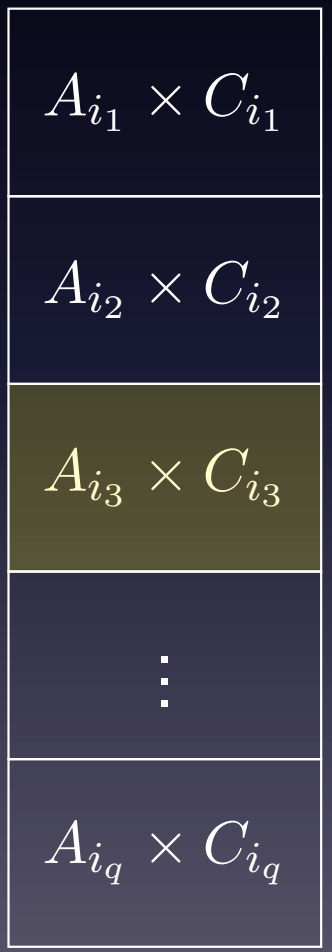
The algorithm

$\pi_{a,b}(R_1 \bowtie R_2)$

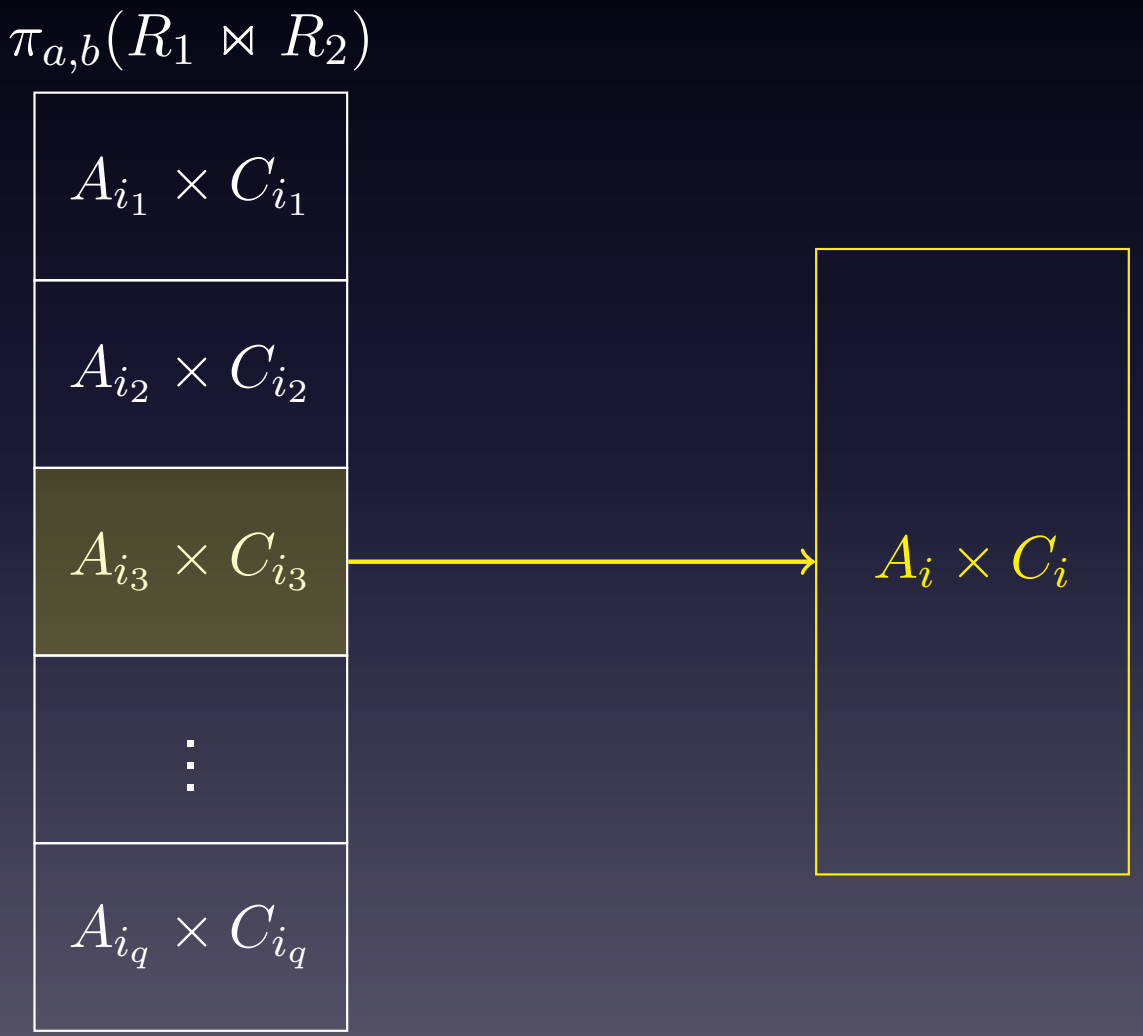


The algorithm

$\pi_{a,b}(R_1 \bowtie R_2)$



The algorithm



The algorithm

- $h(x, y) := h_1(x) - h_2(y) \mod 1$;
- $Z = \pi_{a,b}(R_1 \bowtie R_2) = \bigcup_i A_i \times C_i$, so we can focus on a single generic $A_i \times C_i$;
- find the k smallest values in $h(A_i \times C_i) = (h_1(A_i) - h_2(C_i)) \mod 1$.

The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

$h_1(a)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

$h_1(a)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

threshold p

The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

$h_1(a)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$h(x_{i-1}, y)$
$\vee$
$h(x_i, y_j)$

The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$h_1(a)$

$0.9 > 0.85$

The algorithm

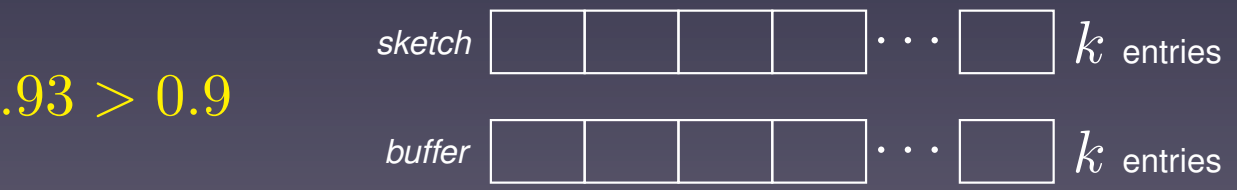
$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$h_1(a)$

$0.93 > 0.9$



The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$h_1(a)$

$0.02 < 0.93$

The algorithm

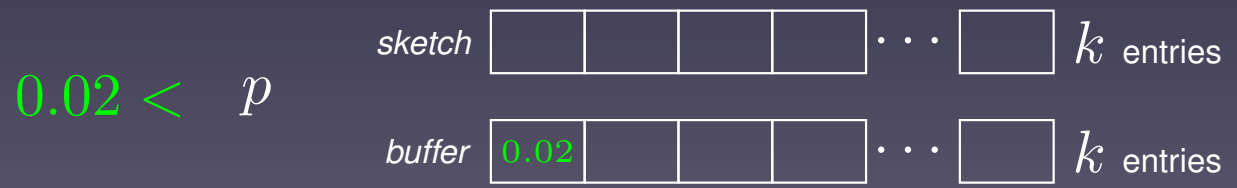
$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

$h_1(a)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$0.02 < p$



The algorithm

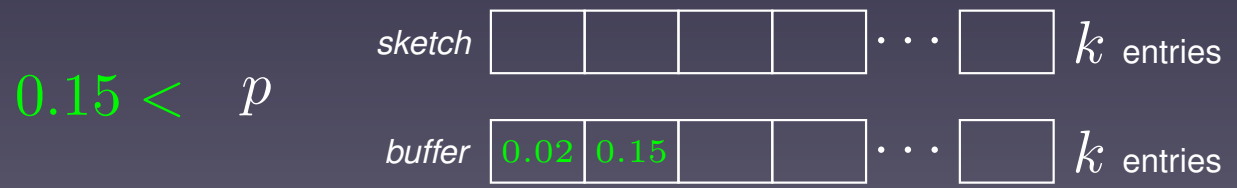
$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

$h_1(a)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$0.15 < p$



The algorithm

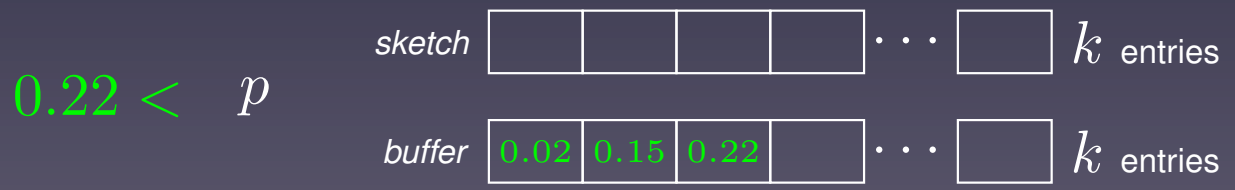
$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

$h_1(a)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$0.22 < \quad p$



The algorithm

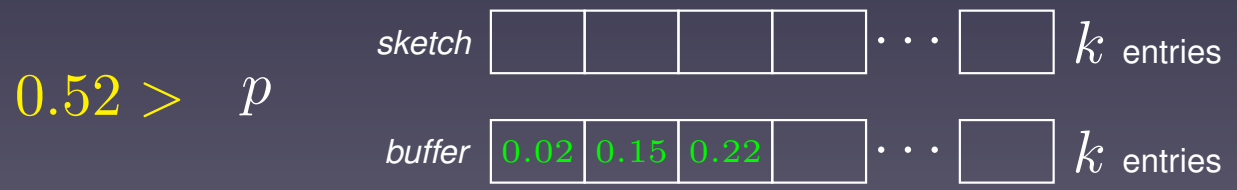
$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

$h_1(a)$

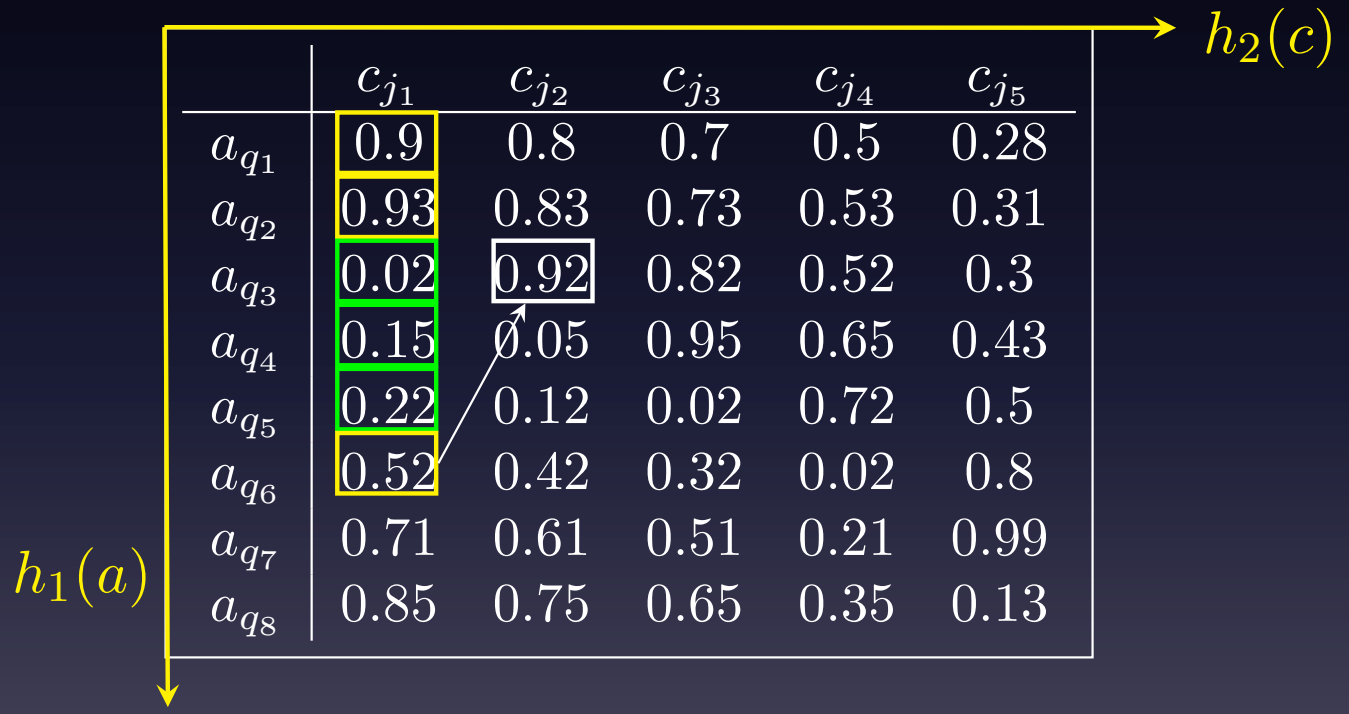
	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$0.52 > \quad p$



The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$



	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

The algorithm

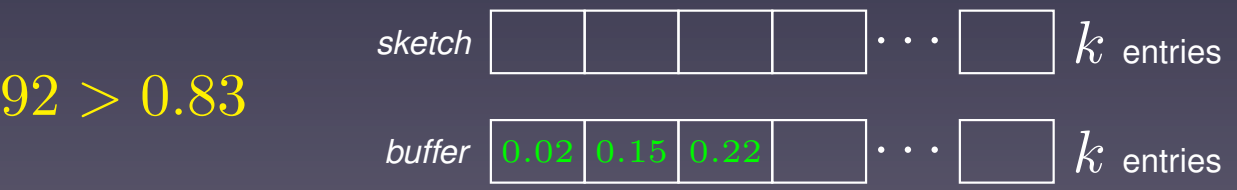
$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

$h_1(a)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$0.92 > 0.83$



The algorithm

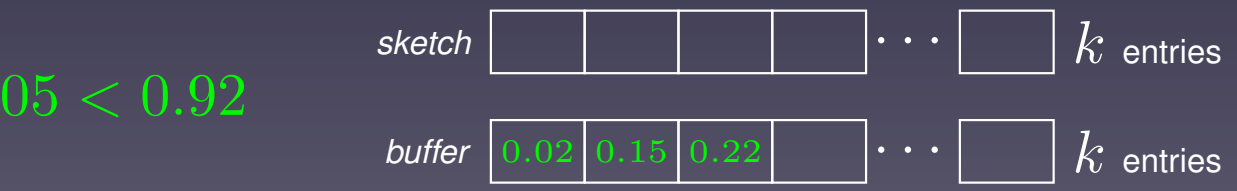
$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

$h_1(a)$

	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$0.05 < 0.92$



The algorithm

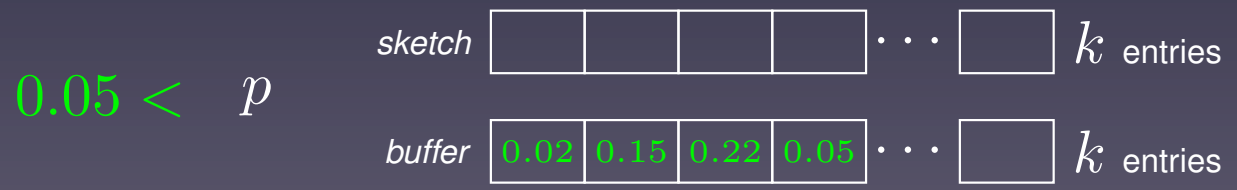
$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

$h_2(c)$

$h_1(a)$

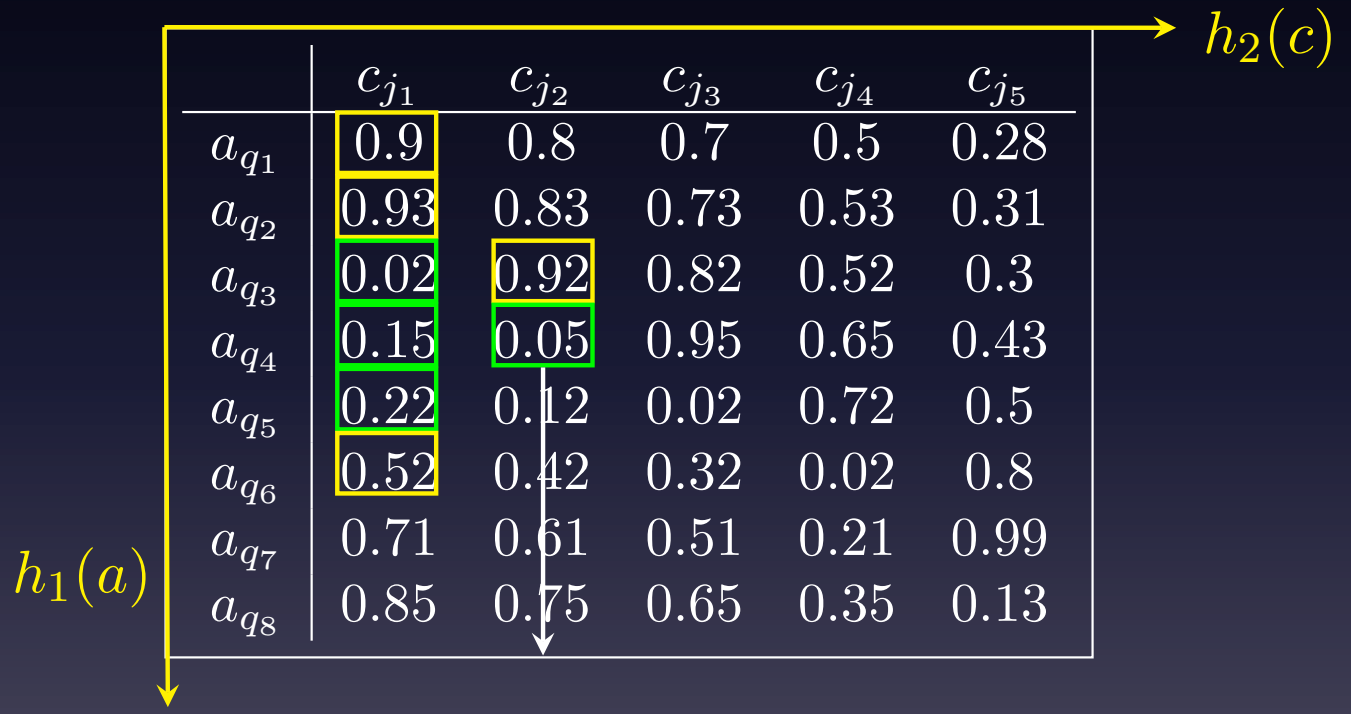
	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13

$0.05 < p$

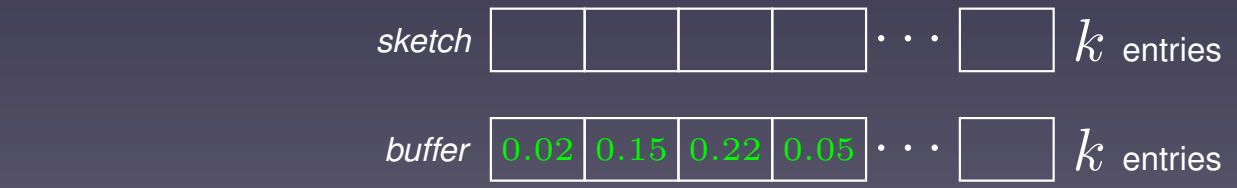


The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$

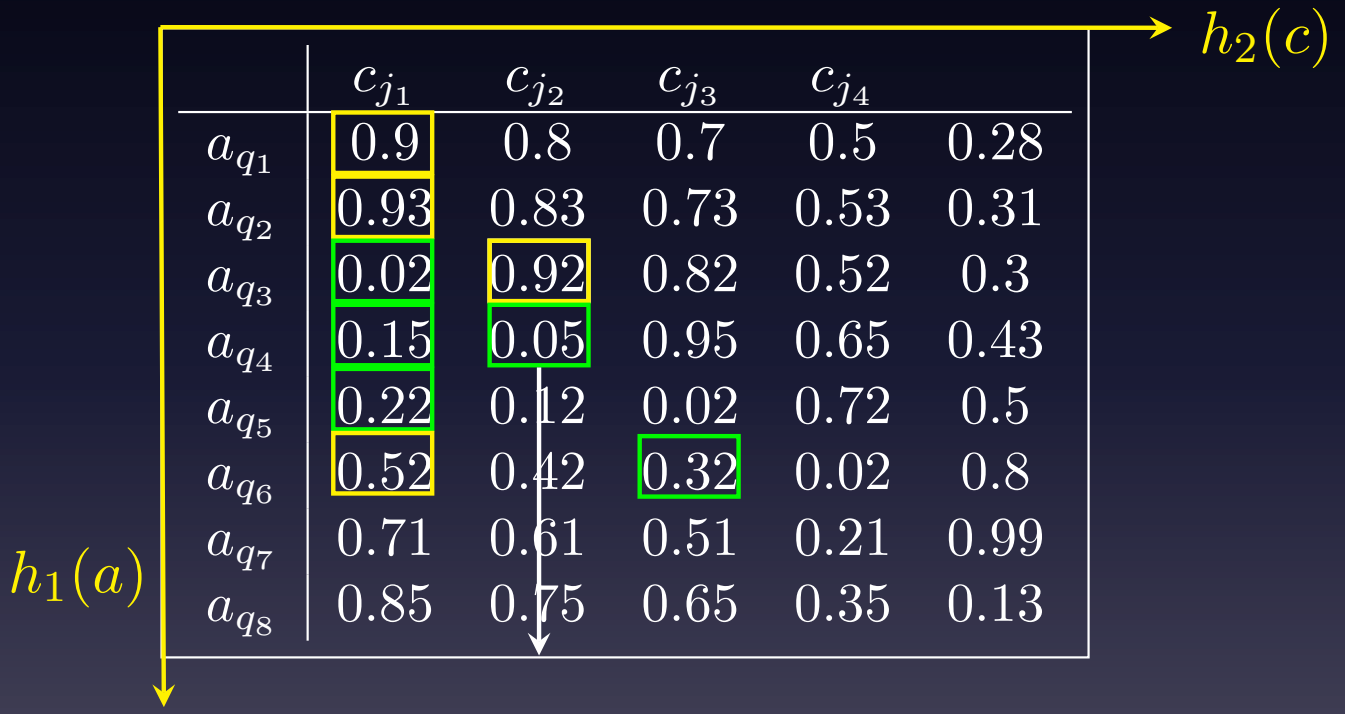


	c_{j_1}	c_{j_2}	c_{j_3}	c_{j_4}	c_{j_5}
a_{q_1}	0.9	0.8	0.7	0.5	0.28
a_{q_2}	0.93	0.83	0.73	0.53	0.31
a_{q_3}	0.02	0.92	0.82	0.52	0.3
a_{q_4}	0.15	0.05	0.95	0.65	0.43
a_{q_5}	0.22	0.12	0.02	0.72	0.5
a_{q_6}	0.52	0.42	0.32	0.02	0.8
a_{q_7}	0.71	0.61	0.51	0.21	0.99
a_{q_8}	0.85	0.75	0.65	0.35	0.13



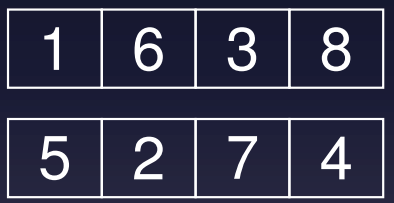
The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$



Not sorted

The algorithm (Buffers merging)



The algorithm (Buffers merging)

- Find the median m;

The algorithm (Buffers merging)

- Find the median m;
- $\forall i \in \text{sketch}$ if $i \leq m$ keep i in the sketch;

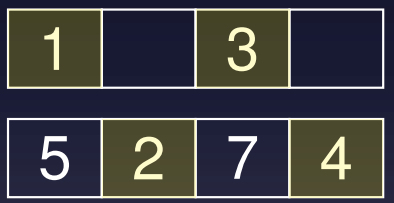
The algorithm (Buffers merging)

- Find the median m;
- $\forall i \in \text{sketch}$ if $i \leq m$ keep i in the sketch;

The algorithm (Buffers merging)

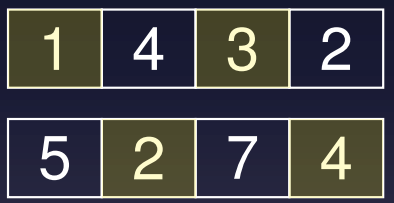
- Find the median m;
- $\forall i \in \text{sketch}$ if $i \leq m$ keep i in the sketch;

The algorithm (Buffers merging)



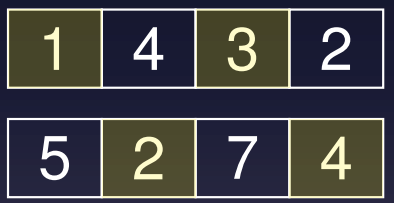
- Find the median m;
- $\forall i \in \text{sketch}$ if $i \leq m$ keep i in the sketch;
- $\forall i \in \text{buffer}$ if $i \leq m$ put i in the sketch.

The algorithm (Buffers merging)



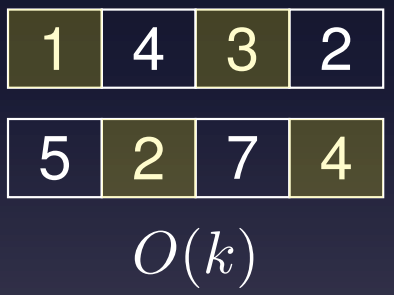
- Find the median m;
- $\forall i \in \text{sketch}$ if $i \leq m$ keep i in the sketch;
- $\forall i \in \text{buffer}$ if $i \leq m$ put i in the sketch.

The algorithm (Buffers merging)



- Find the median m;
- $\forall i \in \text{sketch}$ if $i \leq m$ keep i in the sketch;
- $\forall i \in \text{buffer}$ if $i \leq m$ put i in the sketch.
- Note: *no need to keep the two structures sorted!*

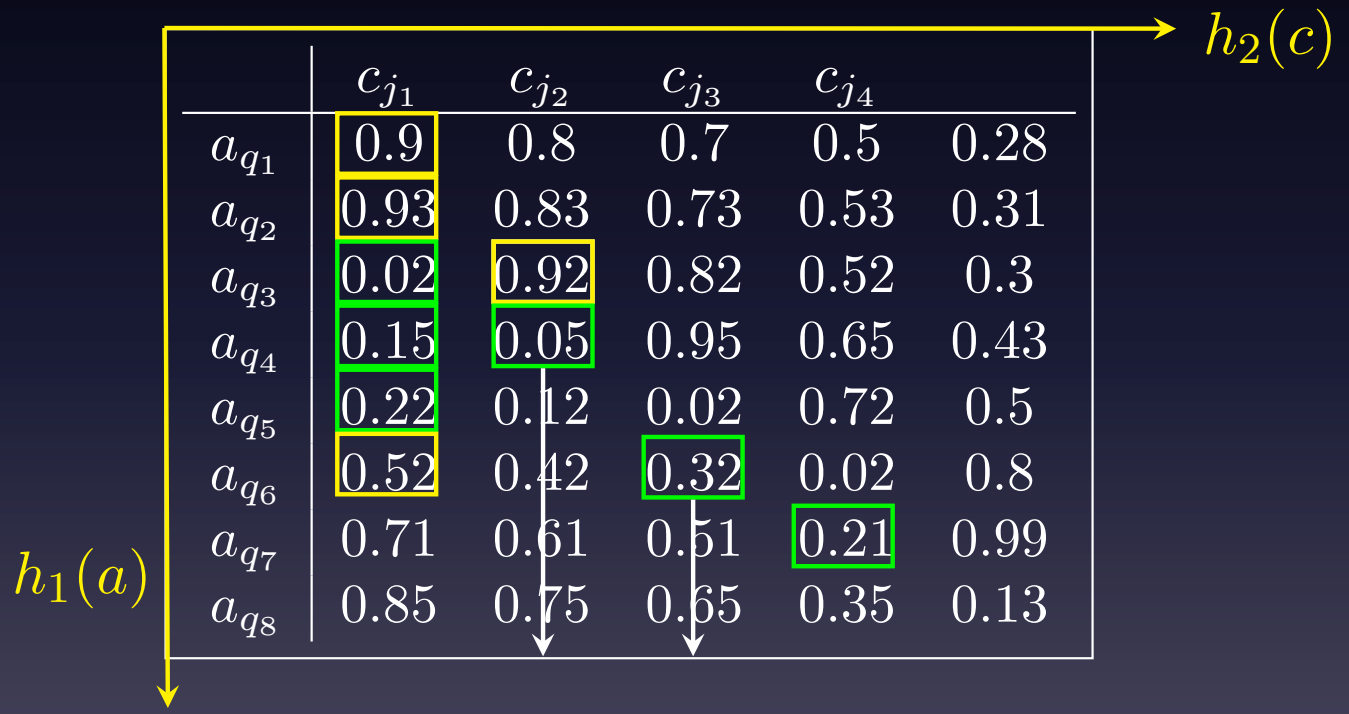
The algorithm (Buffers merging)



- Find the median m;
- $\forall i \in \text{sketch}$ if $i \leq m$ keep i in the sketch;
- $\forall i \in \text{buffer}$ if $i \leq m$ put i in the sketch.
- Note: *no need to keep the two structures sorted!*

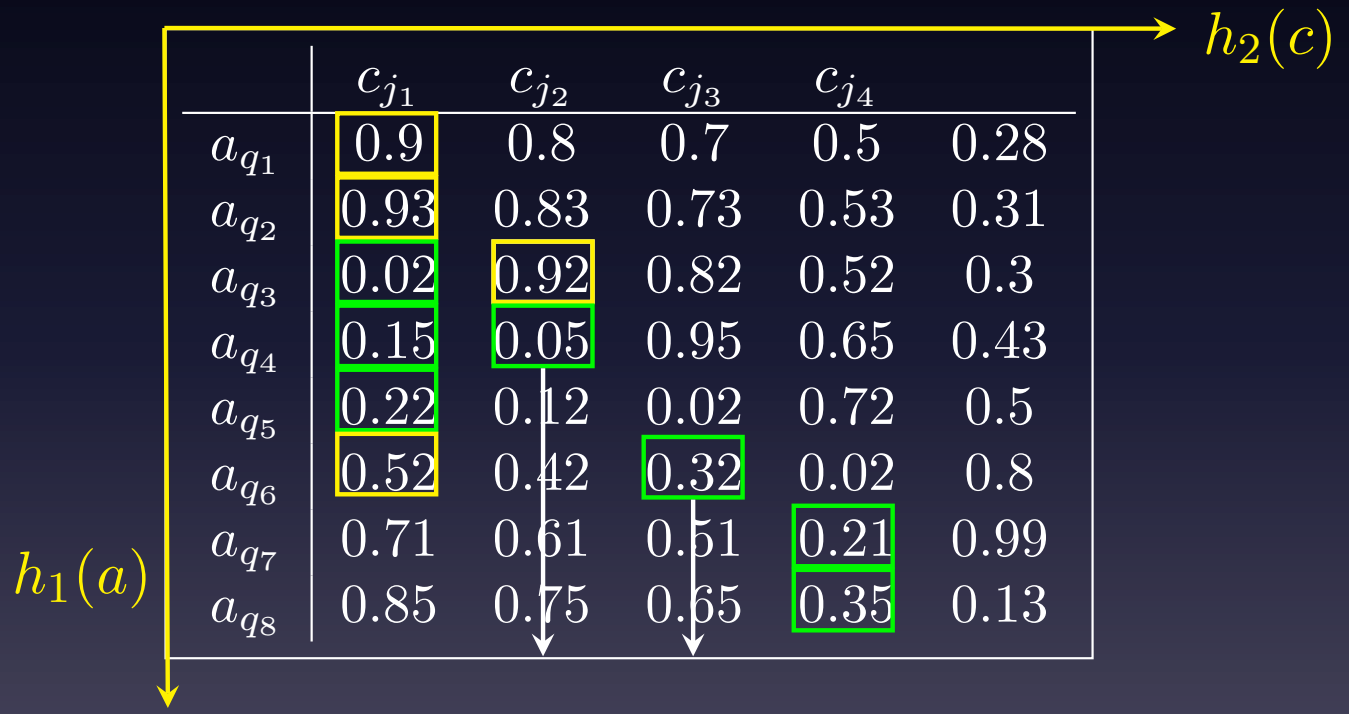
The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$



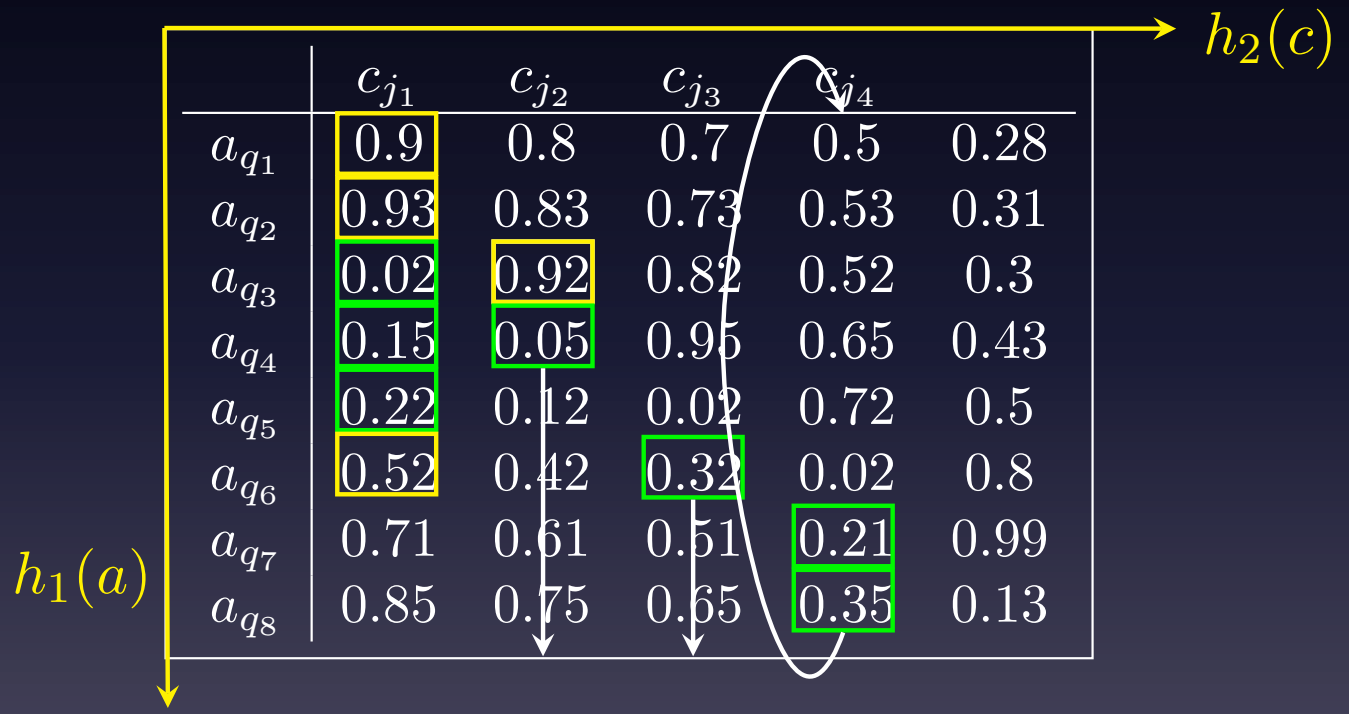
The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$



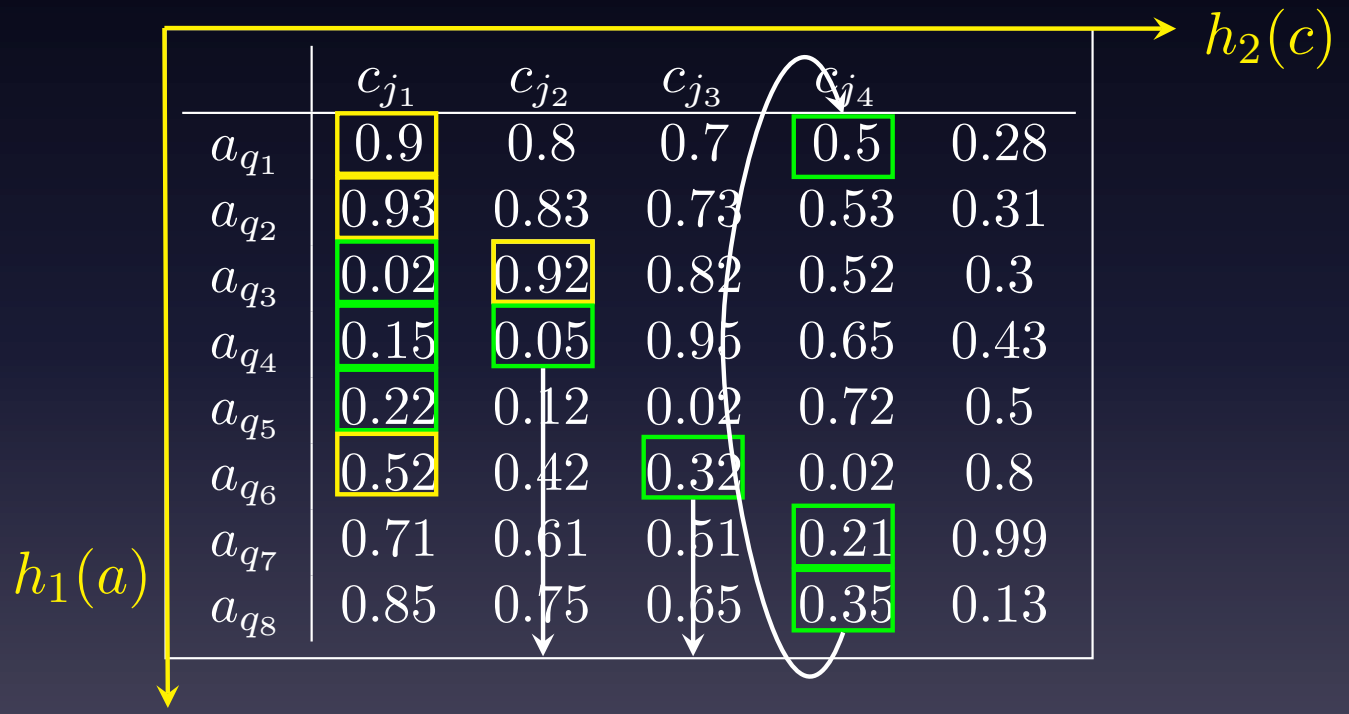
The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$



The algorithm

$$m_{x,y} = h(a_{q_x}, c_{q_y}) = h_1(a_{q_x}) - h_2(c_{q_y}) \mod 1.$$



Time analysis: RAM

- Sorting the values of h_1 and h_2 takes $O(|A_i| + |C_i|)$ (sorting of pairwise independent values);
- given the threshold p, for each cluster we expect $O(p|A_i||C_i|)$ scanned pairs;

Time analysis: RAM

- Sorting the values of h_1 and h_2 takes $O(|A_i| + |C_i|)$ (sorting of pairwise independent values);
- given the threshold p, for each cluster we expect $O(p|A_i||C_i|)$ scanned pairs;

$$A_i \times C_i$$

Time analysis: RAM

- Sorting the values of h_1 and h_2 takes $O(|A_i| + |C_i|)$ (sorting of pairwise independent values);
- given the threshold p, for each cluster we expect $O(p|A_i||C_i|)$ scanned pairs;

Time analysis: RAM

- Sorting the values of h_1 and h_2 takes $O(|A_i| + |C_i|)$ (sorting of pairwise independent values);
- given the threshold p, for each cluster we expect $O(p|A_i||C_i|)$ scanned pairs;

- merging the two buffers costs $O(k)$ but at least k pairs have to be scanned before, so $O(1)$ amortized;

Time analysis: RAM

- Sorting the values of h_1 and h_2 takes $O(|A_i| + |C_i|)$ (sorting of pairwise independent values);
- given the threshold p, for each cluster we expect $O(p|A_i||C_i|)$ scanned pairs;

- merging the two buffers costs $O(k)$ but at least k pairs have to be scanned before, so $O(1)$ amortized;
- We choose p in order to have $O(\sum_i p|A_i||C_i|) = O(n)$ and enough reported pairs ($\geq k$).

Time analysis: RAM

- Sorting the values of h_1 and h_2 takes $O(|A_i| + |C_i|)$ (sorting of pairwise independent values);
- given the threshold p, for each cluster we expect $O(p|A_i||C_i|)$ scanned pairs;

- merging the two buffers costs $O(k)$ but at least k pairs have to be scanned before, so $O(1)$ amortized;
- We choose p in order to have $O(\sum_i p|A_i||C_i|) = O(n)$ and enough reported pairs ($\geq k$).

Total cost: $O(n)$.

Time analysis: I/O

- Single global (cache oblivious) sorting of values $(b, h_1(a))$ and $(b, h_2(c))$;
- the rest of the algorithm works directly in the cache oblivious model;

Time analysis: I/O

- Single global (cache oblivious) sorting of values $(b, h_1(a))$ and $(b, h_2(c))$;
- the rest of the algorithm works directly in the cache oblivious model;
- the first operation dominates, so:

Time analysis: I/O

- Single global (cache oblivious) sorting of values $(b, h_1(a))$ and $(b, h_2(c))$;
- the rest of the algorithm works directly in the cache oblivious model;
- the first operation dominates, so:

Total cost: $O(sort(n))$ I/Os.

Error probability

- Three possible sources of errors:
  - the kth value v found is too small $\rightarrow$ estimate too large;
  - the kth value v found is too large $\rightarrow$ estimate too small;
  - too few values are found smaller than $p \rightarrow$ we allow an additive error.
- analysis essentially equivalent in all the cases;
- follows the one in Bar-Yossef et al.;
- error probability for all of the cases: 1/3.

Conclusions and open questions

- First algorithm allowing $o(1)$ relative error:
  - in expected linear time;
  - with polynomially small error probability;

Conclusions and open questions

- First algorithm allowing $o(1)$ relative error:
  - in expected linear time;
  - with polynomially small error probability;
- can this method be extended to products of multiple boolean matricies?

Conclusions and open questions

- First algorithm allowing $o(1)$ relative error:
  - in expected linear time;
  - with polynomially small error probability;
- can this method be extended to products of multiple boolean matricies?
- can this method be extended for transitive closure of general graphs?

Conclusions and open questions

- First algorithm allowing $o(1)$ relative error:
  - in expected linear time;
  - with polynomially small error probability;
- can this method be extended to products of multiple boolean matricies?
- can this method be extended for transitive closure of general graphs?

If you would be a real seeker after truth,
you must at least once in your life doubt,
as far as possible,
all things.