SKETCHING DATA STRUCTURES FOR MASSIVE GRAPH PROBLEMS

Juan P. A. Lopes[1], Fabiano S. Oliveira[2],
Paulo E. D. Pinto[2], Valmir C. Barbosa[1]

August 31st, 2018

[1] Federal University of Rio de Janeiro (UFRJ)
[2] State University of Rio de Janeiro (UERJ)

VLDB2018
VLDB Workshop
Poly'18

Agenda

- Motivation
- Probabilistic Implicit Representations
- Graph streams
- Conclusion

Motivation

Why are **sketching data structures** relevant to **graph** problems?

Some real-life graphs are massive

Observing global structures is **hard**

Facebook

2.2 billion

Number of active users, 2018.

Twitter

233 billion

Estimated number of directed edges, 2018.

Routers

128 MB

Typical amount of RAM in a typical router.

Internet

23 billion

Number of connected devices, 2018.

Metagenomic assemblies

100’s of billions

Number of basepairs in a typical metagenomic sample.

SOME REAL-LIFE GRAPHS ARE MASSIVE AND **DYNAMIC**

How to deal with them?

Probabilistic Implicit **Representations**

Use less **memory** by allowing **errors**

Space Optimal Representations

- A representation is said to be **space optimal** if it requires O(f(n)) bits to represent a class containing $2^{\Theta(f(n))}$ graphs on n vertices;
- Optimality depends on the represented class.

	General Graphs	Trees	Complete Graphs
Adjacency Matrix: $O(n^2)$	✓	✗	✗
Adjacency List: O(m log n)	✗	✓	✗

Spinrad, J. P. (2003). **Efficient graph representations.** American Mathematical Society.

Implicit Representations

A representation is said to be **implicit** if it has the following properties:

Space optimal

$O(f(n))$ bits to represent a class containing $2^{\Theta(f(n))}$ graphs on n vertices;

Distributes information

Each vertex stores $O(f(n)/n)$ bits;

Local adjacency test

Only local vertex information is required to test adjacency;

Spinrad, J. P. (2003). **Efficient graph representations.** American Mathematical Society.

Probabilistic Implicit Representations

For **probabilistic implicit representations**, we introduce a **fourth property**:

Space optimal

O(f(n)) bits to represent a class containing $2^{\Theta(f(n))}$ graphs on n vertices;

Distributes information

Each vertex stores O(f(n)/n) bits;

Local adjacency test

Only local vertex information is required to test adjacency;

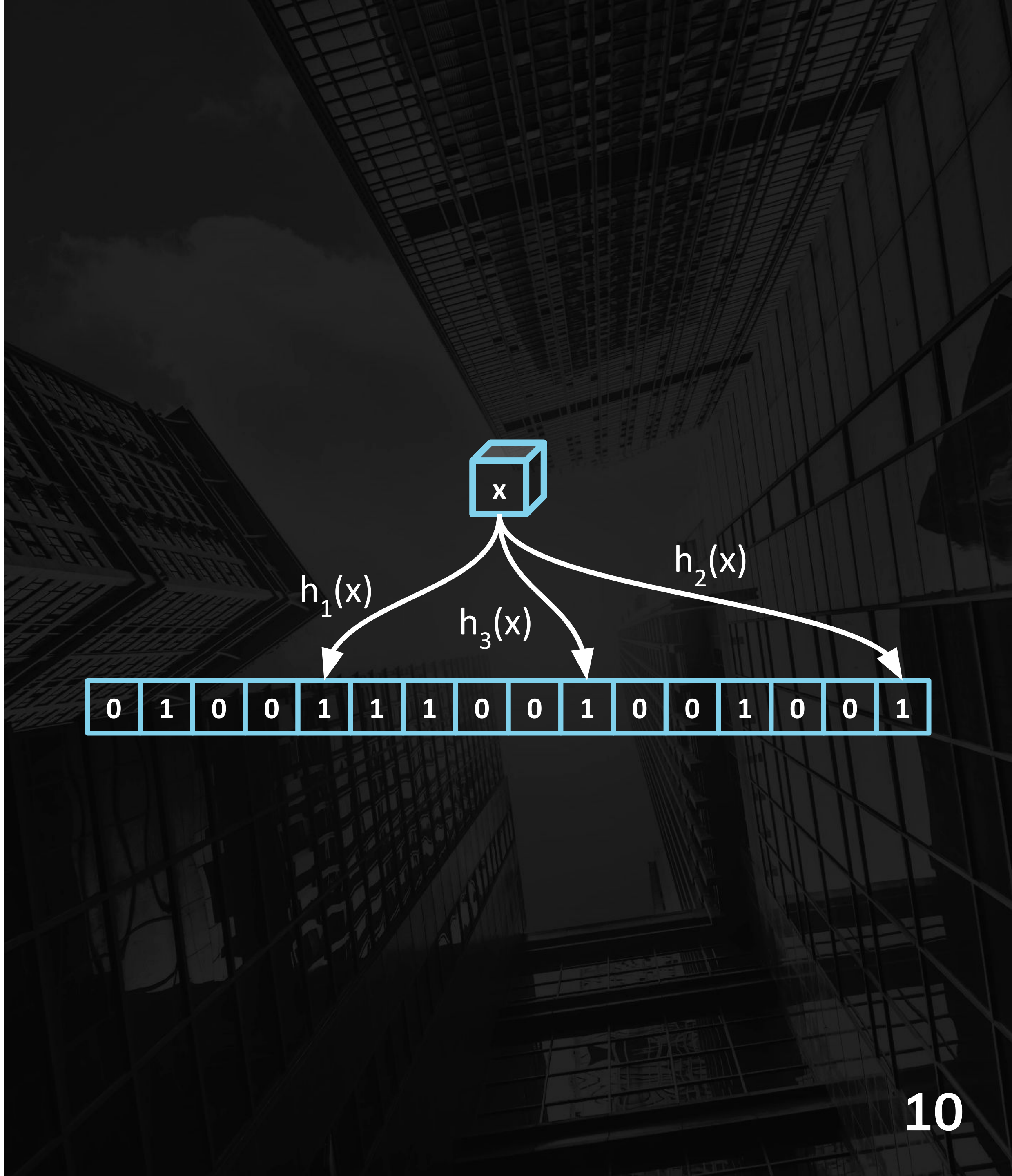
Probabilistic adjacency test

Constant relative probability of false positives or false negatives.

Bloom filter

Represents sets, allowing membership tests with a probability of **false positives**.

- There are **no false negatives**;
- **10 bits** per element are enough to ensure for a false positive probability of **less than 1%**.

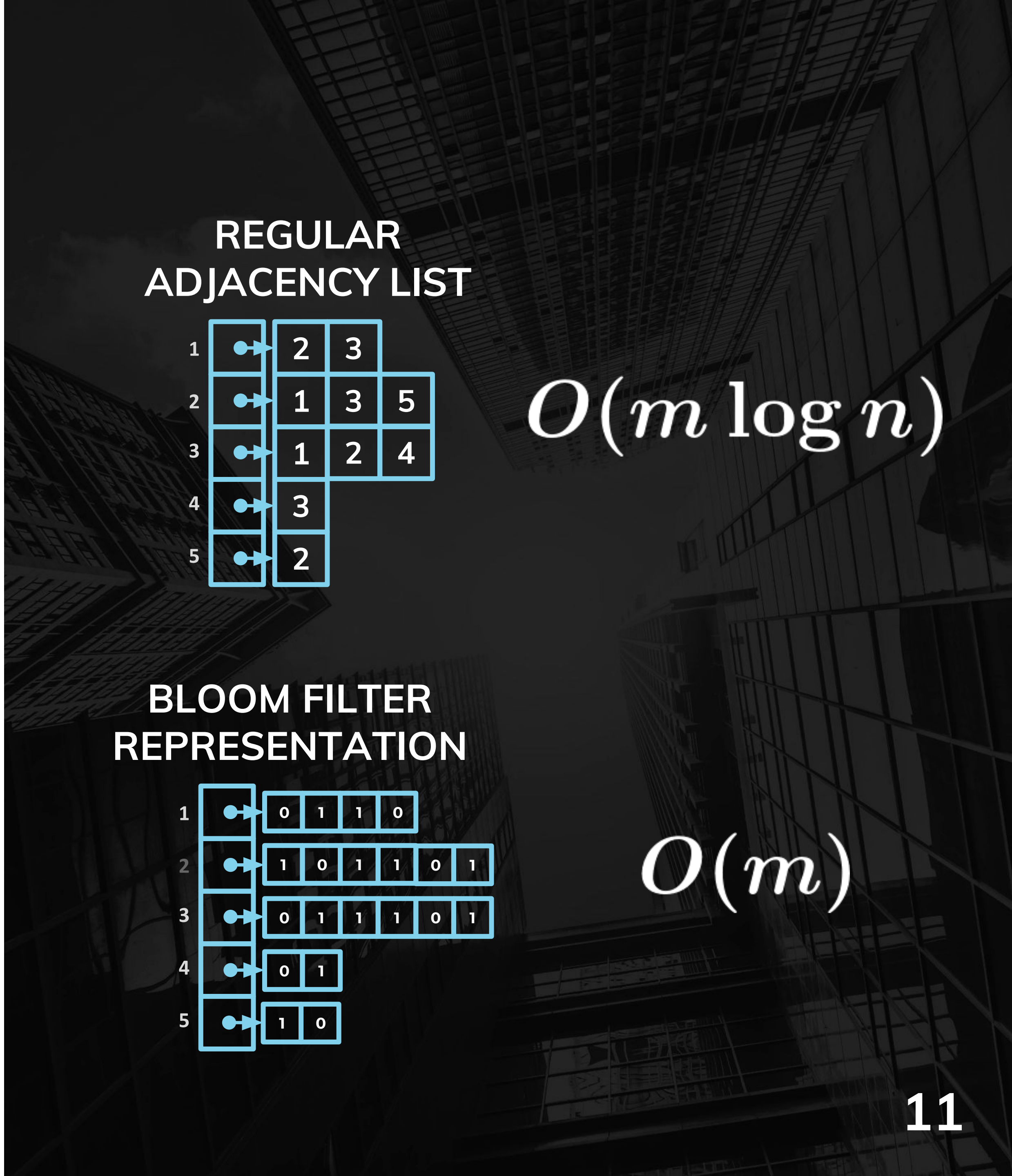


Bloom, B. H. (1970). **Space/time trade-offs in hash coding with allowable errors**. Communications of the ACM.

Bloom filter

Idea: to **replace** each vertex set in an adjacency list with a **Bloom filter.**

- Each edge would require only **O(1) bits**, instead of O(log n);
- By using Bloom filters, there would be **no false negatives**, only false positives.
- Similarly, a **single Bloom filter** could be used to store the **entire edge set**, but technically this would not be an implicit representation.



MinHash

Represents sets through a constant-sized signature and allow computing the Jaccard coefficient between two or more sets.

	h_1	h_2	h_3	h_4	h_5	h_6	h_7	h_8
MinHash(A)	11	6	1	6	71	34	57	106
MinHash(B)	11	6	1	81	80	34	73	88

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

$$h_{\text{min}}(A) = min\{h(x), x \in A\}$$

$$\Pr[h_{\text{min}}(A) = h_{\text{min}}(B)] = J(A,B)$$

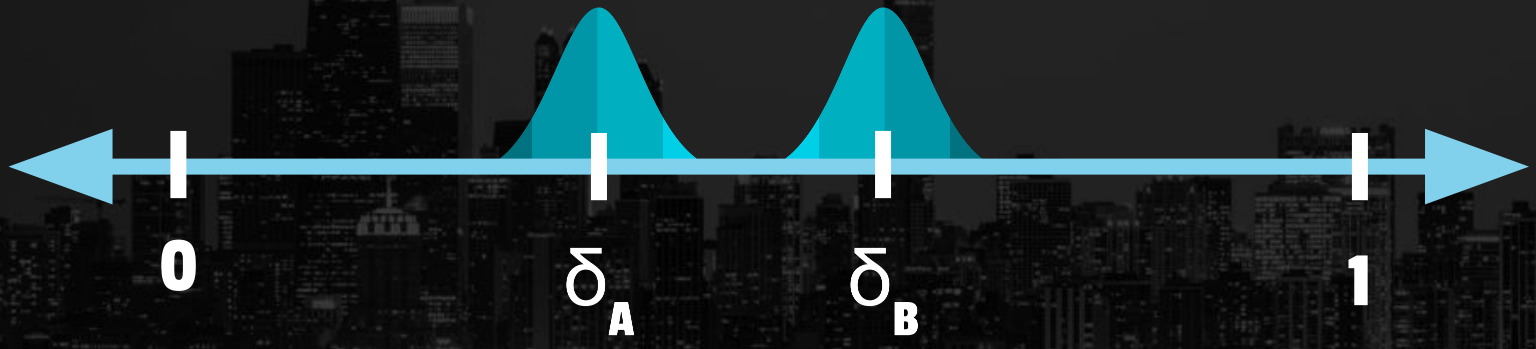
Broder, A. Z. (1997). **On the resemblance and containment of documents**. In Compression and complexity of sequences.

MinHash

Idea: construct a set for each vertex, such that the Jaccard index between any pair of vertices encodes their adjacency.

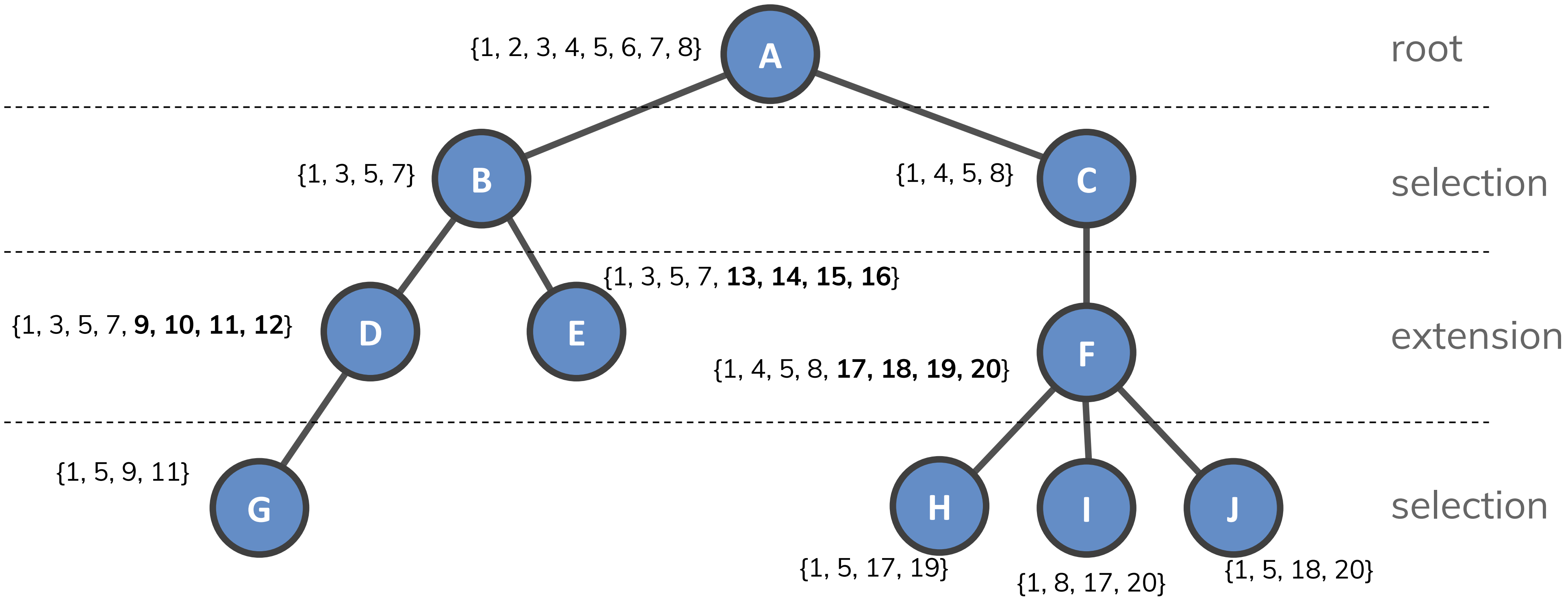
$$(v_i, v_j) \notin E \leftrightarrow J(S_i, S_j) \leq \delta_A$$

$$(v_i, v_j) \in E \leftrightarrow J(S_i, S_j) \geq \delta_B$$



MinHash

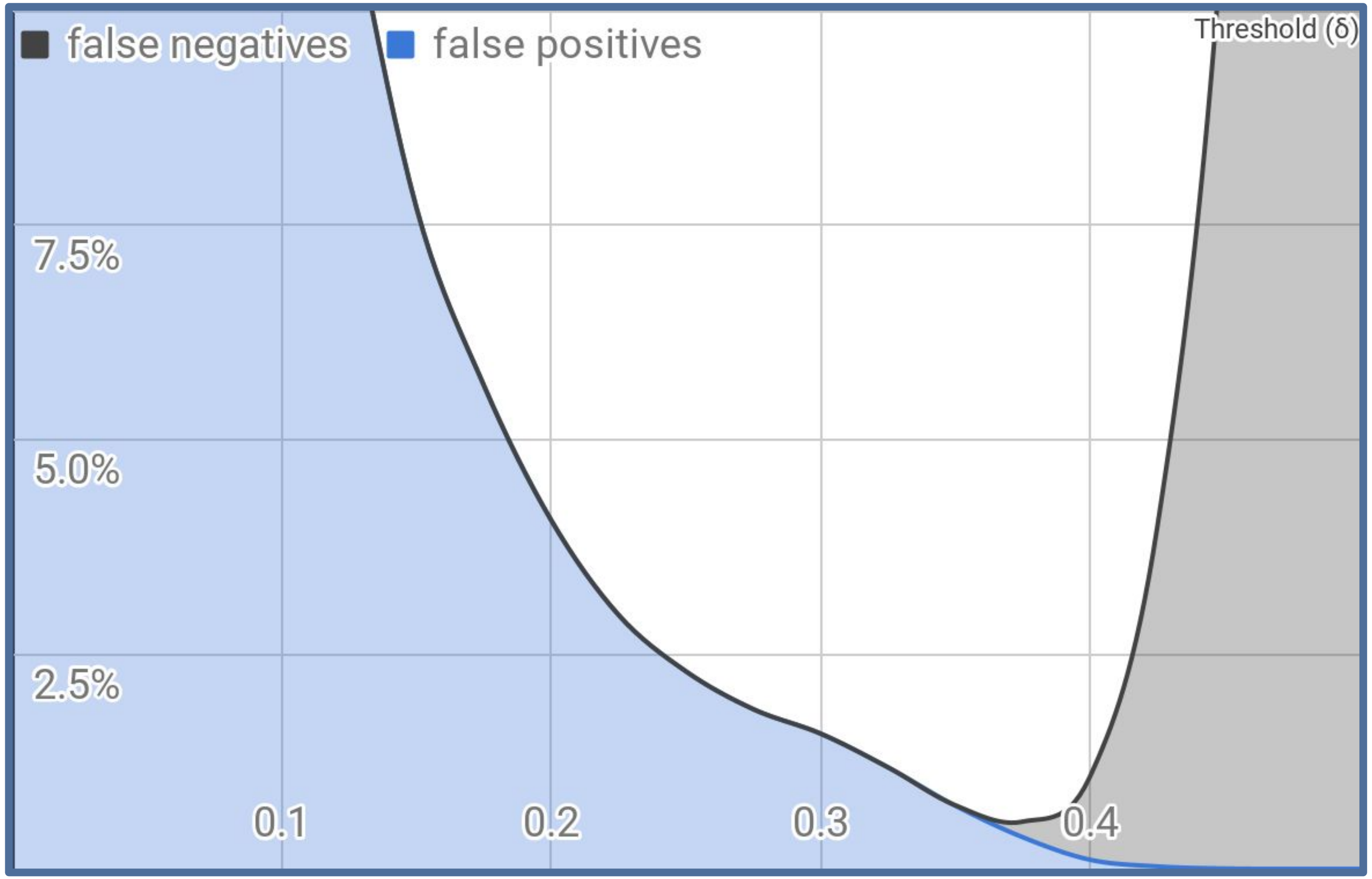
Example of sets construction for $\boldsymbol{\delta}_A = \frac{1}{3}$ and $\boldsymbol{\delta}_B = \frac{1}{2}$.



O(n) bits

Experimental Results

For MinHash-based representation



Observations

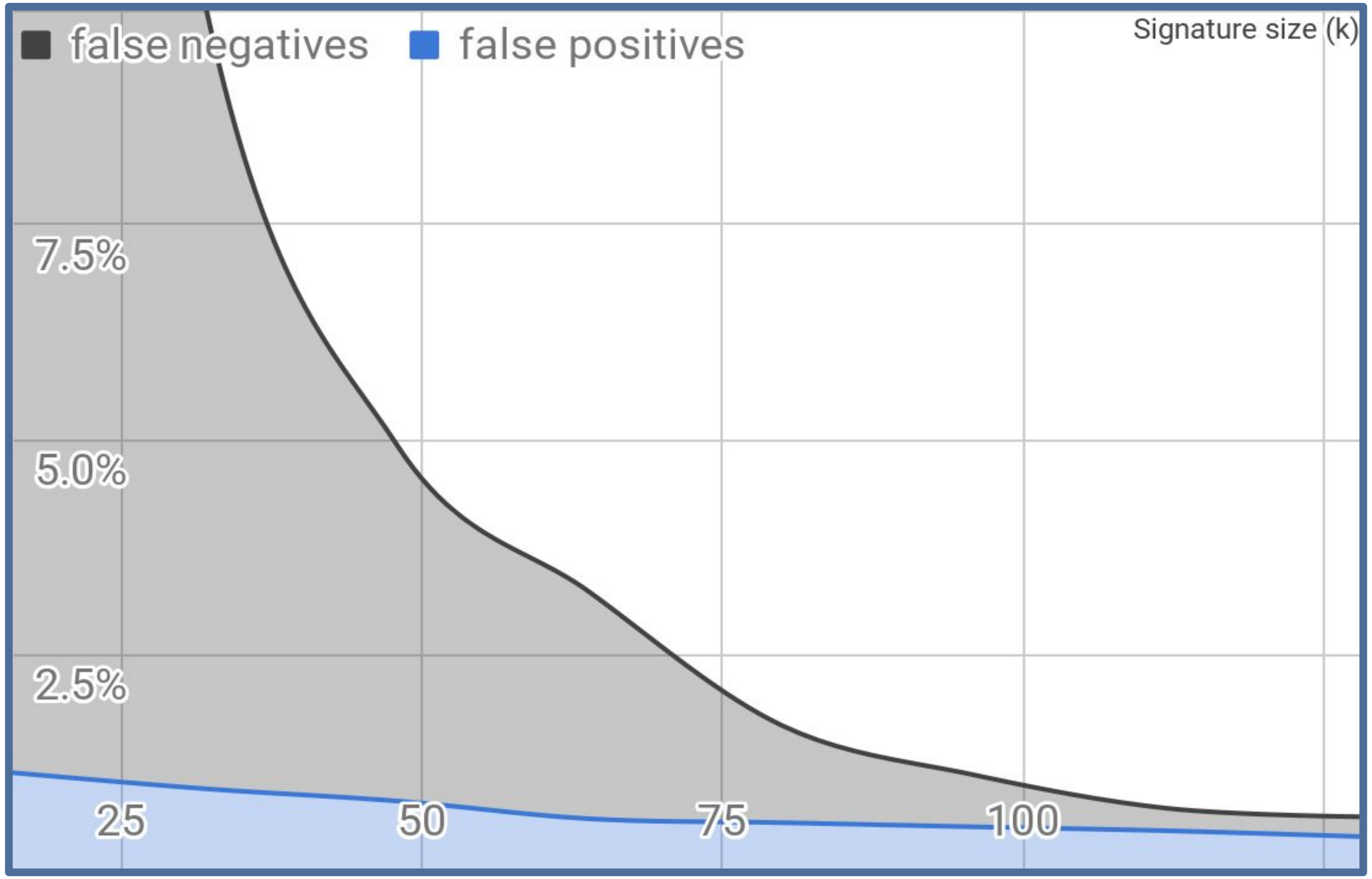
1. The experiment was run with k=128 hash functions and a graph with n=200 vertices.

2. Increasing the threshold seems to increase the rate of false negatives and decrease false positives.

3. The perfect threshold depends on the application tolerance for false positives and false negatives.

Experimental Results

For MinHash-based representation

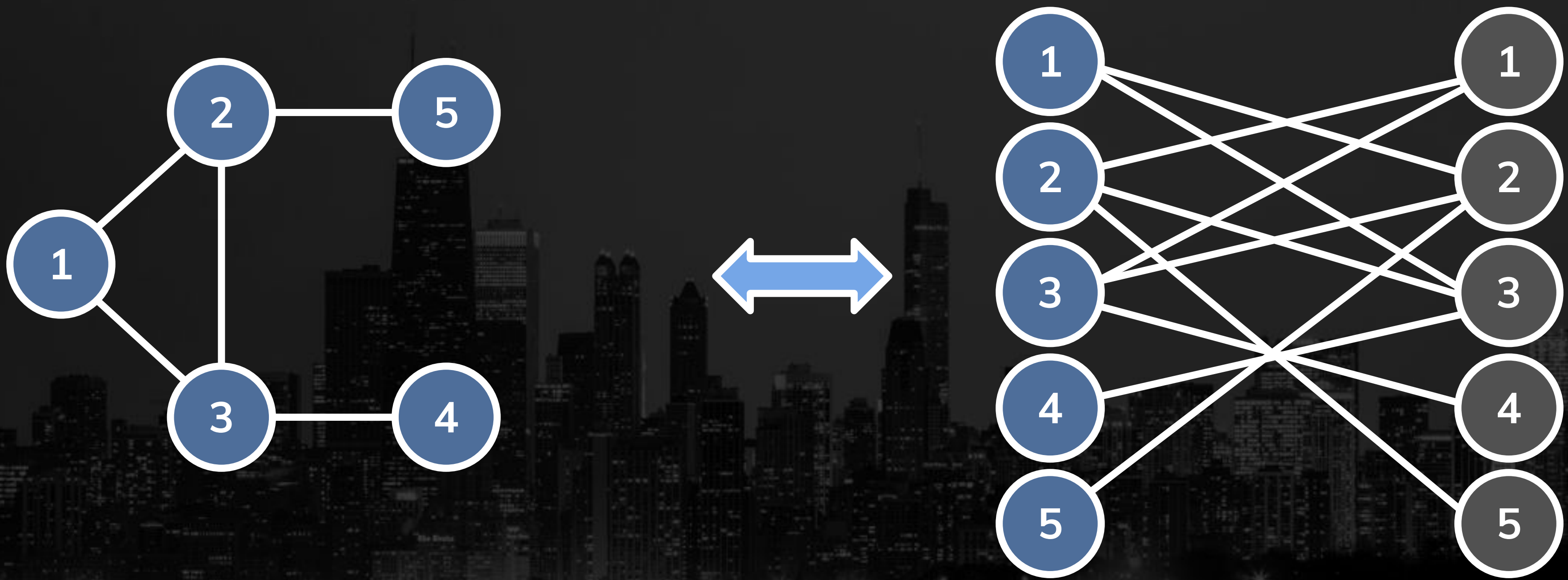


Observations

1. The experiment was run with δ = 0.375 and a graph with n=200 vertices.
2. Increasing the signature size seems to have more effect on the rate of false negatives than positives.
3. This effect appears the same for whatever choice of threshold.

Other results

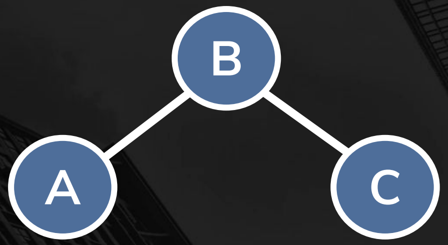
Any efficient representation for bipartite, co-bipartite or split graphs can be used to represent general graphs efficiently.

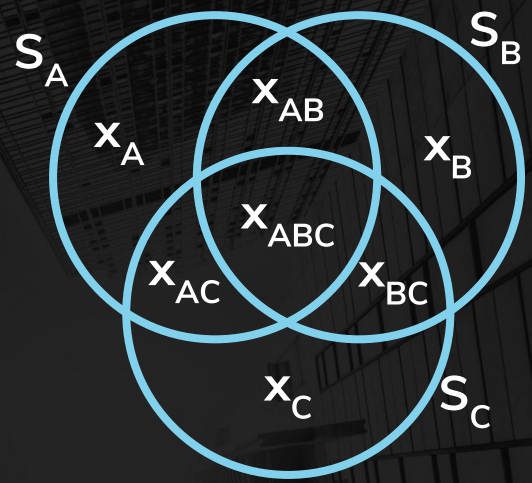


Other results

Modeling this problem through integer programming allows proving the infeasibility of specific configurations.

- Each possible subset of vertices is modelled as a variable.
- Each variable describes the size of the set intersection between those vertices.





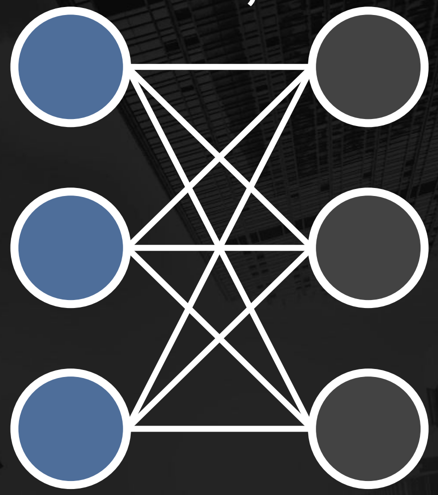
$$
\begin{aligned}
\min \quad & x_A + x_B + x_{AB} + x_C + x_{AC} + x_{BC} + x_{ABC} \\
\text{s.t.} \quad & 6x_A + 6x_B - 4x_{AB} + 6x_{AC} + 6x_{BC} - 4x_{ABC} \leq 0 \\
& -4x_A - 4x_{AB} - 4x_C + 6x_{AC} - 4x_{BC} + 6x_{ABC} \leq 0 \\
& 6x_B + 6x_{AB} + 6x_C + 6x_{AC} - 4x_{BC} - 4x_{ABC} \leq 0 \\
& x_A + x_{AB} + x_{AC} + x_{ABC} \geq 1 \\
& x_B + x_{AB} + x_{BC} + x_{ABC} \geq 1 \\
& x_C + x_{AC} + x_{BC} + x_{ABC} \geq 1
\end{aligned}
$$

Other results

Modeling this problem through integer programming allows proving the infeasibility of specific configurations.

- Each possible subset of vertices is modelled as a variable.
- Each variable describes the size of the set intersection between those vertices.
- Do all threshold values have an infeasible bipartite graph? Still an open problem.

$K_{3,3}$



- **Impossible** for $\delta_A = 0.4$ e $\delta_B = 0.6$.
- **Possible** for $\delta_A = 1/3$ e $\delta_B = 1/2$.

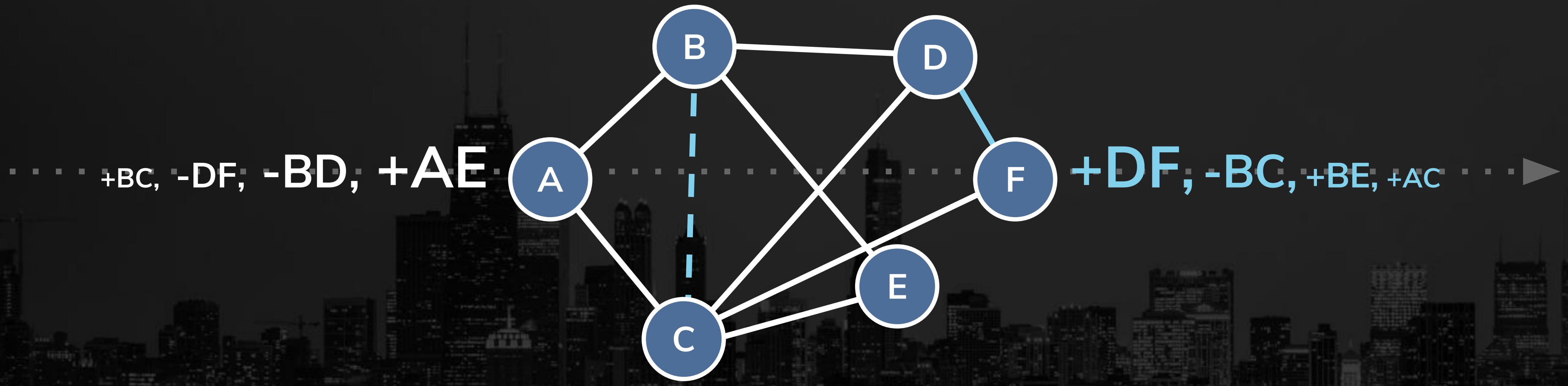
Graph **Streams**

How to represent **dynamic graphs** in sublinear space?

Graph Streams

Graph Streams are graphs represented in the data stream model, i.e. single-pass through a stream of edge insertions and deletions.

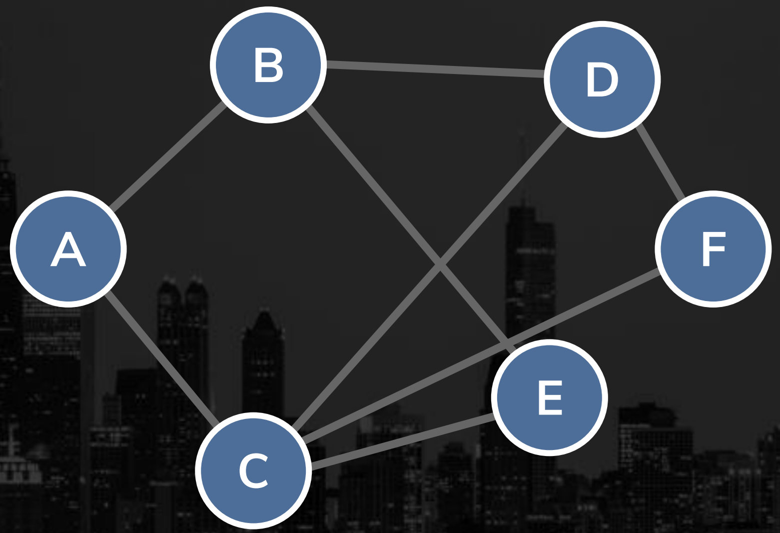
Can we compute global parameters in **sublinear space**?



Ahn, K. J., Guha, S., and McGregor, A. (2012). **Analyzing graph structure via linear measurements**. In Proceedings of SODA'12.
McGregor, A. (2014). **Graph stream algorithms: a survey**. ACM SIGMOD.

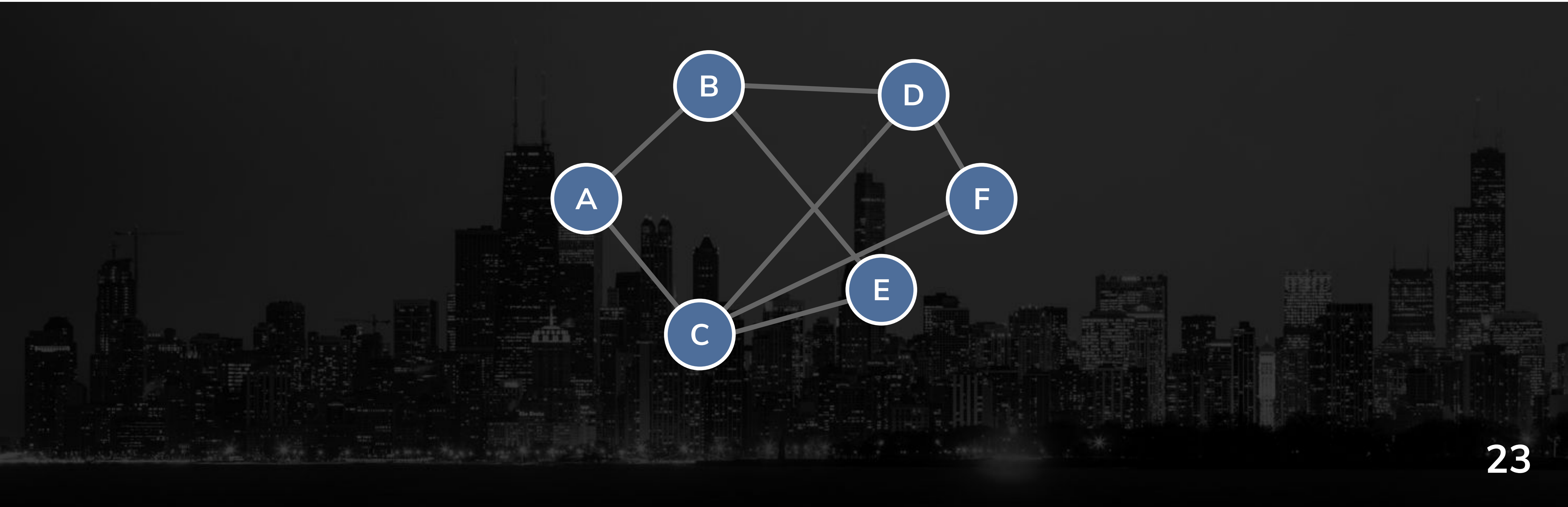
Graph Streams

Can we construct a full spanning forest of the graph in **sublinear space**?



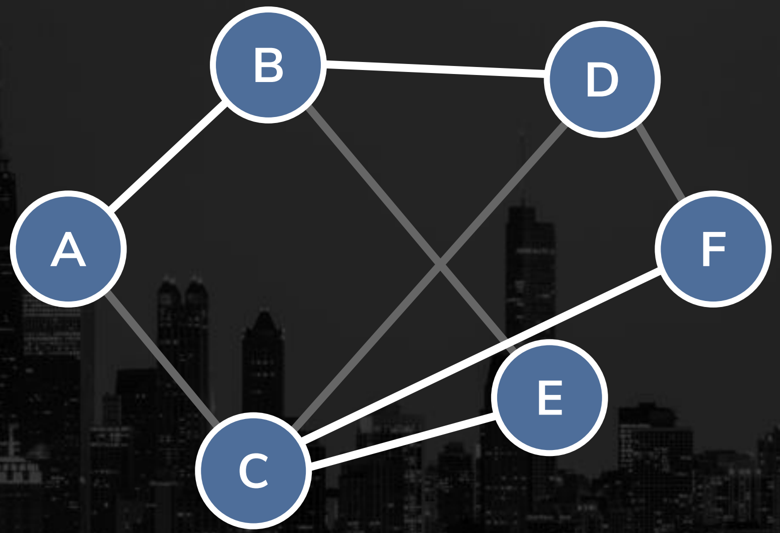
Graph Streams

Idea: we can sample an edge from each vertex and merge its endpoints in a single “super-vertex”. Repeat. This procedures finishes in O(log n) steps.



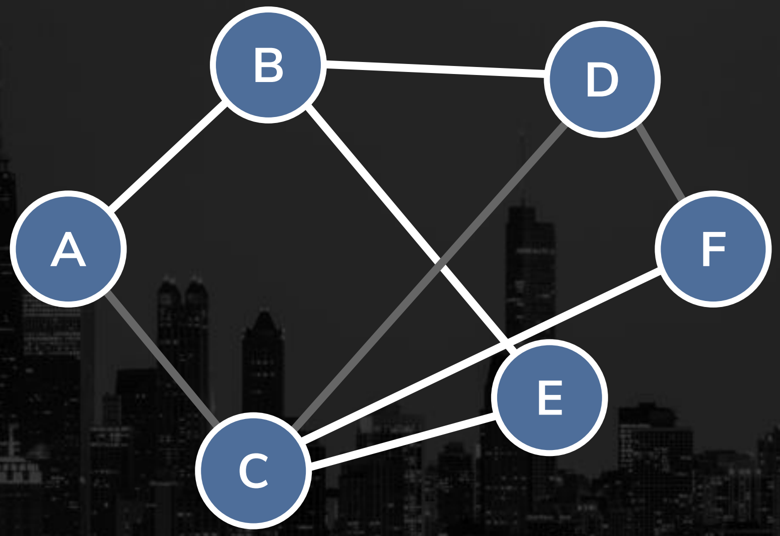
Graph Streams

Idea: we can sample an edge from each vertex and merge its endpoints in a single “super-vertex”. Repeat. This procedures finishes in O(log n) steps.



Graph Streams

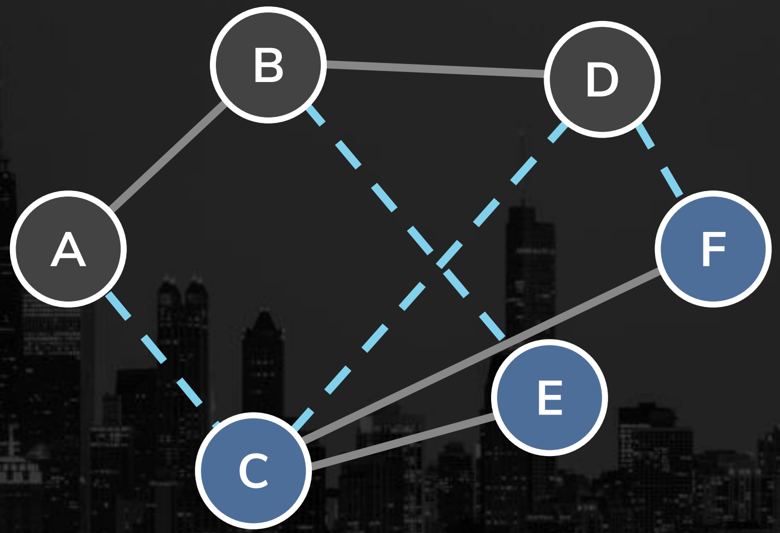
Idea: we can sample an edge from each vertex and merge its endpoints in a single “super-vertex”. Repeat. This procedures finishes in O(log n) steps.



Graph Streams

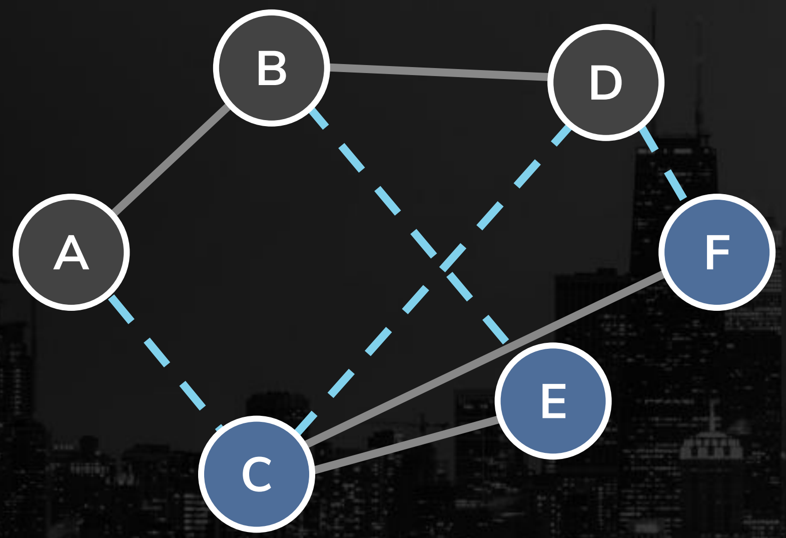
A simpler problem:

Is it possible to sample a **random edge** from any **cut-set** [S, V\S] in a graph stream storing **less than $O(n^2)$ bits**?



Sampling edges from cut-set

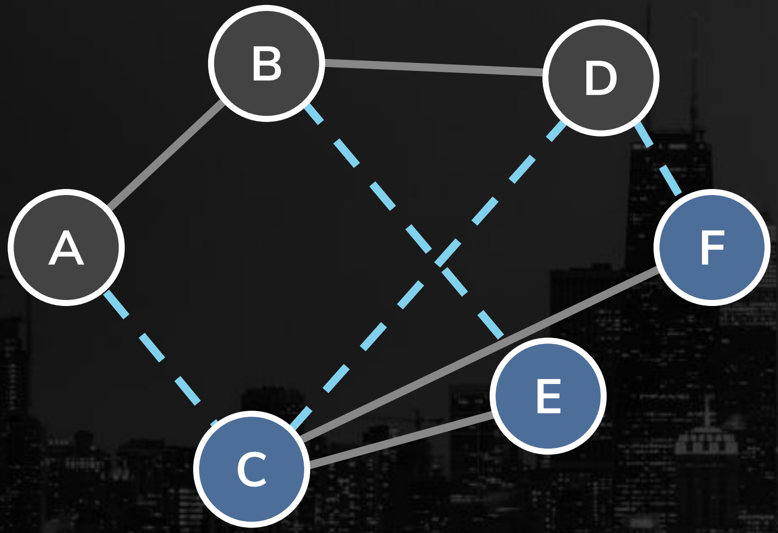
Idea: to represent graph through a modified **incidence matrix**, where each edge is represented **twice** (once in each “direction”).



	AB	BA	AC	CA	BD	DB	BE	EB	CD	DC	CE	EC	CF	FC	DF	FD
A	1	-1	1	-1	0	0	0	0	0	0	0	0	0	0	0	0
B	-1	1	0	0	1	-1	1	-1	0	0	0	0	0	0	0	0
C	0	0	-1	1	0	0	0	0	1	-1	1	-1	1	-1	0	0
D	0	0	0	0	-1	1	0	0	-1	1	0	0	0	0	1	-1
E	0	0	0	0	0	0	-1	1	0	0	-1	1	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	0	-1	1	-1	1

Sampling edges from cut-set

The main benefit from this representation is the ability to **sum incidence vectors** to find the corresponding vector of a cut-set. Being able to **sample nonzero coordinates** from this vector implies sampling edges from such cut-set.

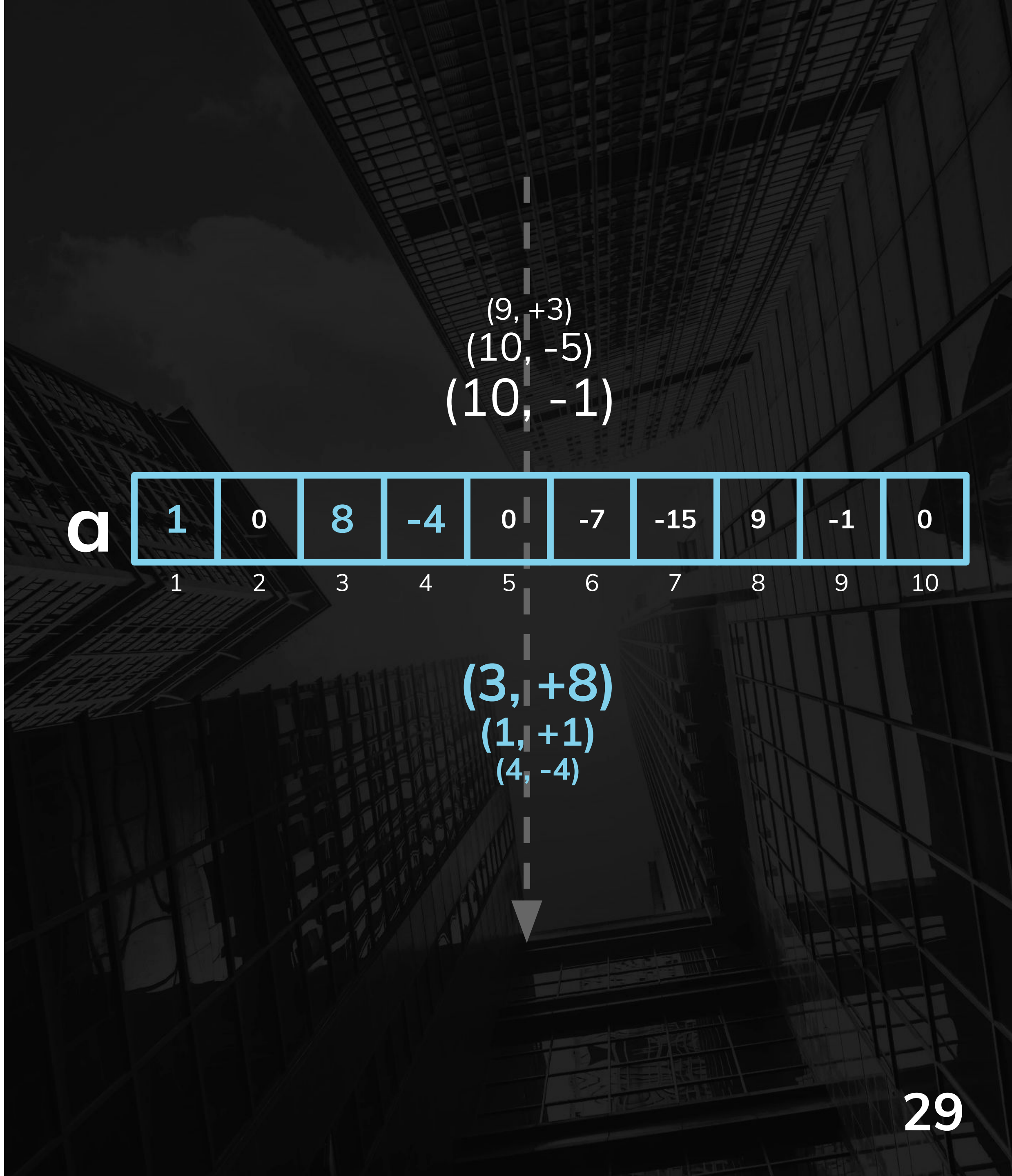


	AB	BA	AC	CA	BD	DB	BE	EB	CD	DC	CE	EC	CF	FC	DF	FD
A	1	-1	1	-1	0	0	0	0	0	0	0	0	0	0	0	0
+B	-1	1	0	0	1	-1	1	-1	0	0	0	0	0	0	0	0
+D	0	0	0	0	-1	1	0	0	-1	1	0	0	0	0	1	-1
{A, B, D}	0	0	1	-1	0	0	-1	1	-1	1	0	0	0	0	1	-1

What is ℓ_0-sampling?

Sampling, with **uniform probability**, of a nonzero coordinate from a vector **a**, represented **incrementally** by a stream of updates.

- Some updates may **cancel** others;
- Must be done in **sublinear** space;
- Known lower-bound: $\mathbf{\Omega(\log^2 n)}$.



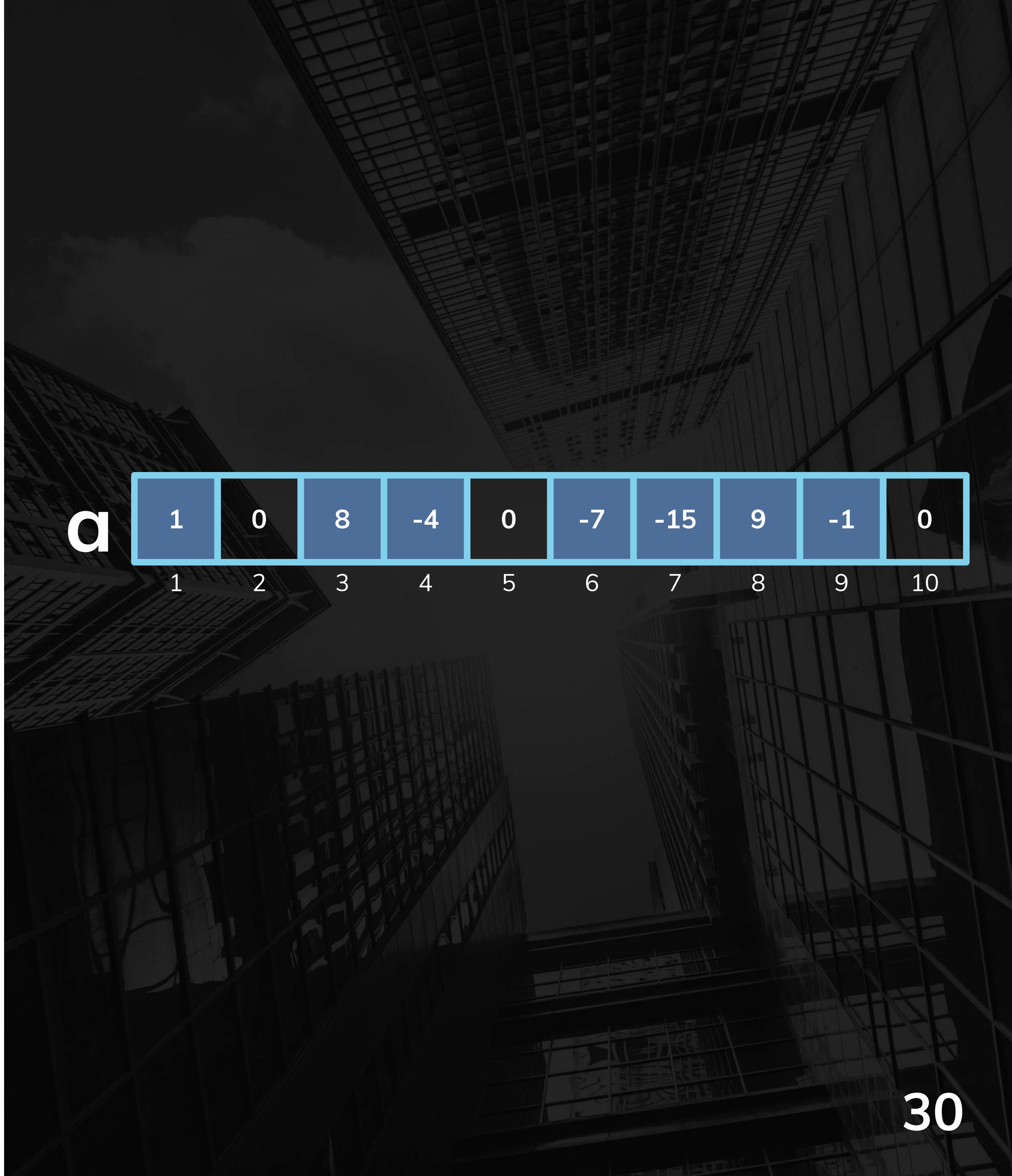
Cormode, G., Muthukrishnan, S., and Rozenbaum, I. (2005). **Summarizing and mining inverse distributions on data streams via dynamic inverse sampling**. In Proceedings of VLDB'05.

Jowhari, H., Saglam, M., and Tardos, G. (2011). **Tight bounds for lp-samplers, finding duplicates in streams, and related problems**. In Proceedings of PODS'11.

What is ℓ_0-sampling?

Sampling, with **uniform probability**, of a nonzero coordinate from a vector **a**, represented **incrementally** by a stream of updates.

- Some updates may **cancel** others;
- Must be done in **sublinear** space;
- Known lower-bound: **$\Omega(\log^2 n)$**.

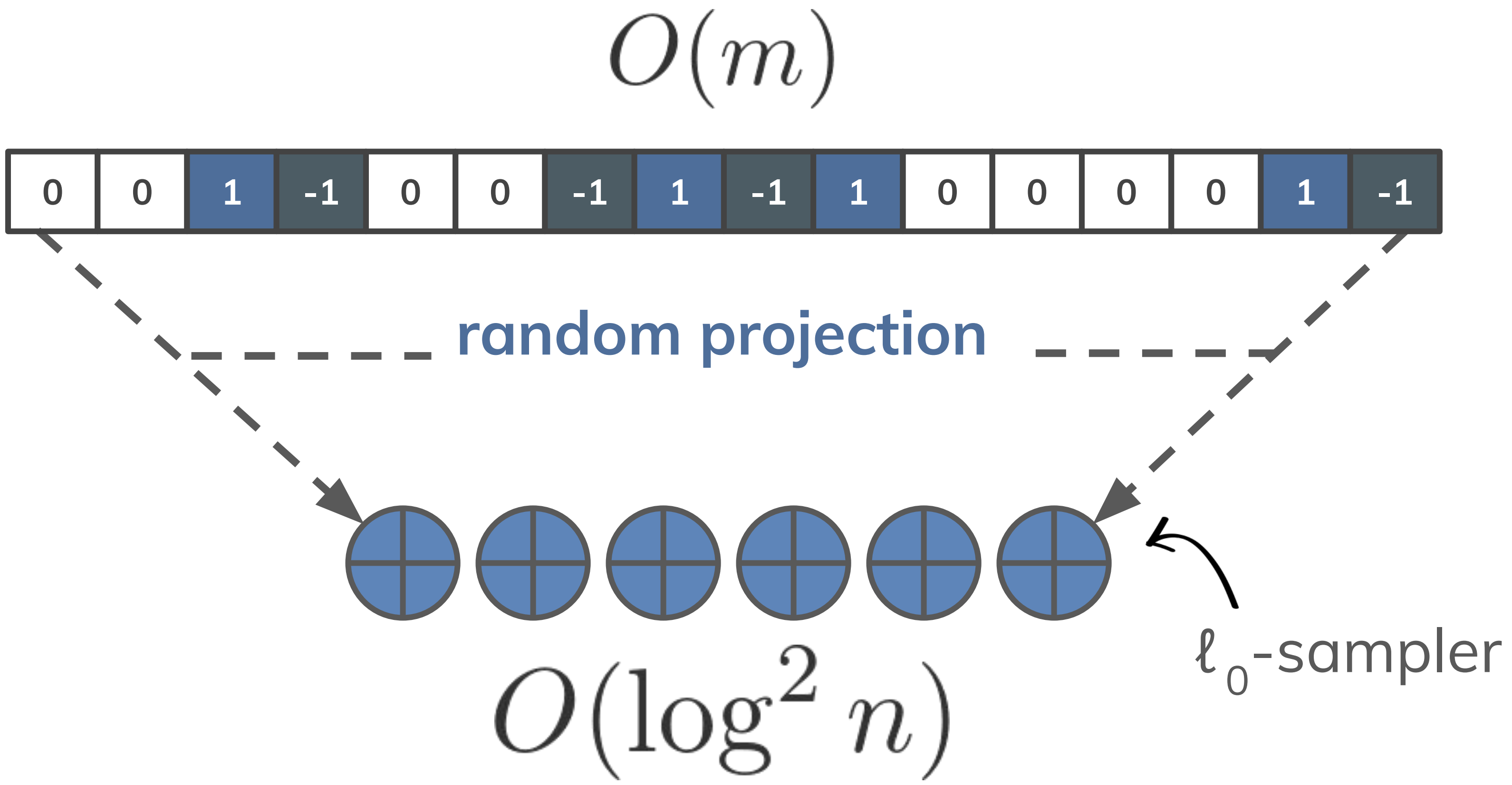


Cormode, G., Muthukrishnan, S., and Rozenbaum, I. (2005). **Summarizing and mining inverse distributions on data streams via dynamic inverse sampling**. In Proceedings of VLDB'05.

Jowhari, H., Saglam, M., and Tardos, G. (2011). **Tight bounds for lp-samplers, finding duplicates in streams, and related problems**. In Proceedings of PODS'11.

Sampling edges from cut-set

Is it possible to encode each incidence vector in a compact representation?



ℓ_0-sampling algorithm

The sampling algorithm is based on the following idea:

Assign each coordinate a random bucket

Use hash functions. Each bucket must have **exponentially decreasing** probabilities of representing each coordinate.

Find 1-sparse vector

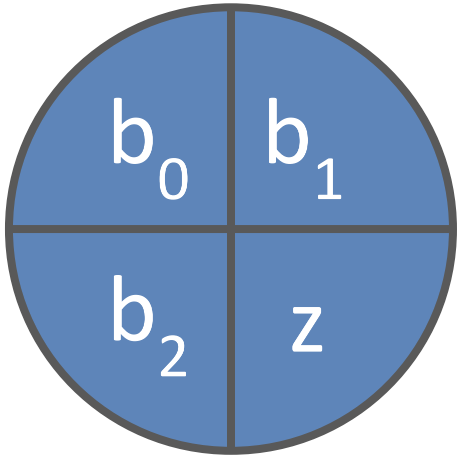
There is a **high probability** that at least one bucket will represent a 1-sparse vector, that is, a vector with a single nonzero coordinate.

Recover its only nonzero coordinate

Through a randomized procedure called **1-sparse recovery**, it is possible to recover the nonzero coordinates from 1-sparse vectors, using O(log n) bits.

1-sparse recovery

Tests if a vector is 1-sparse. If yes, it recovers the single nonzero coordinate.



O(log n) bits

linear transform

$$b_0 = \sum \Delta_i$$
$$b_1 = \sum \Delta_i u_i$$
$$b_2 = \sum \Delta_i z^{u_i} \quad \bmod p$$

$$b_2 \equiv b_0 z^{b_1/b_0} \quad \bmod p?$$

yes:

$$i = b_1/b_0$$
$$a_i = b_0$$

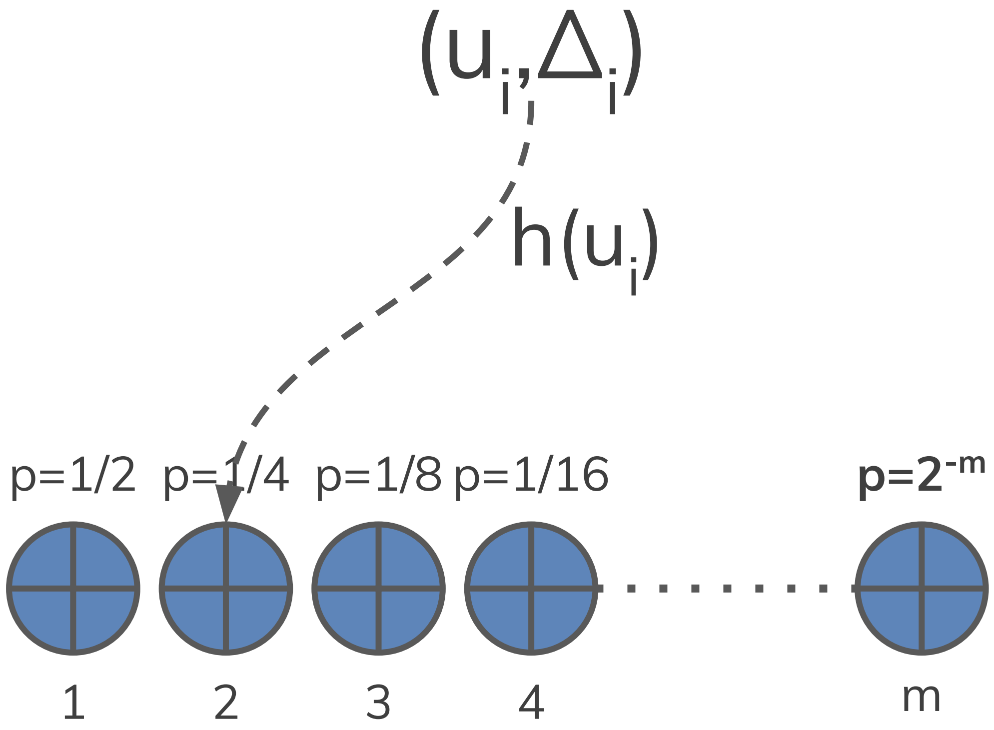
prob. ≥ 1 - n/p

no:

not 1-sparse

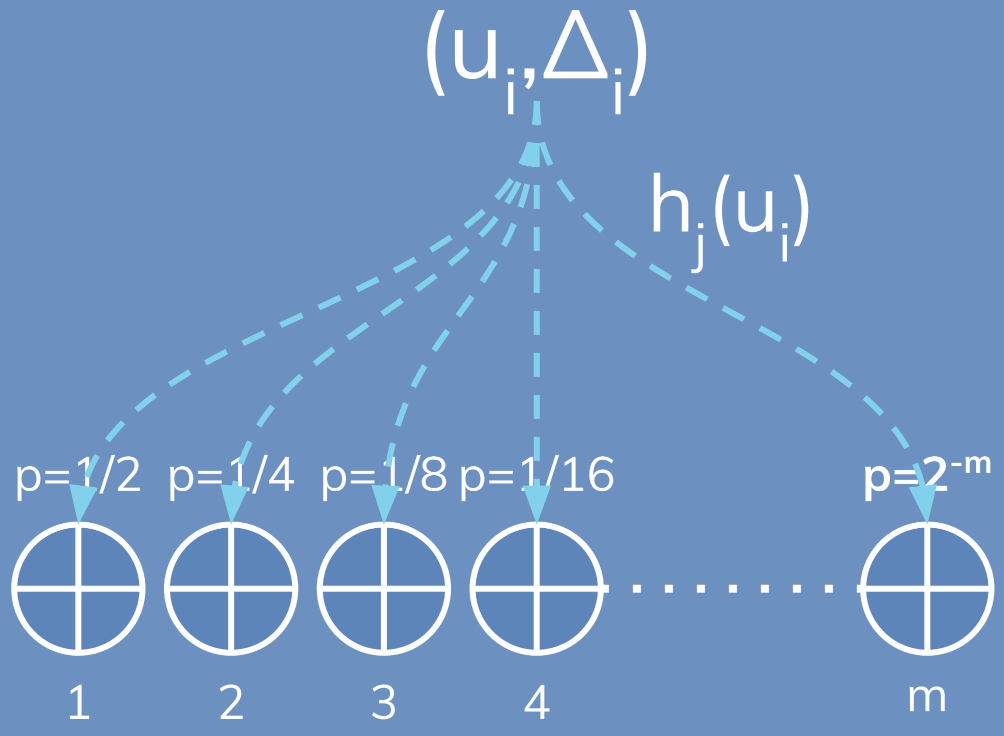
100% sure

Variant (a)



- Single hash function (more efficient);
- Non-independent buckets.

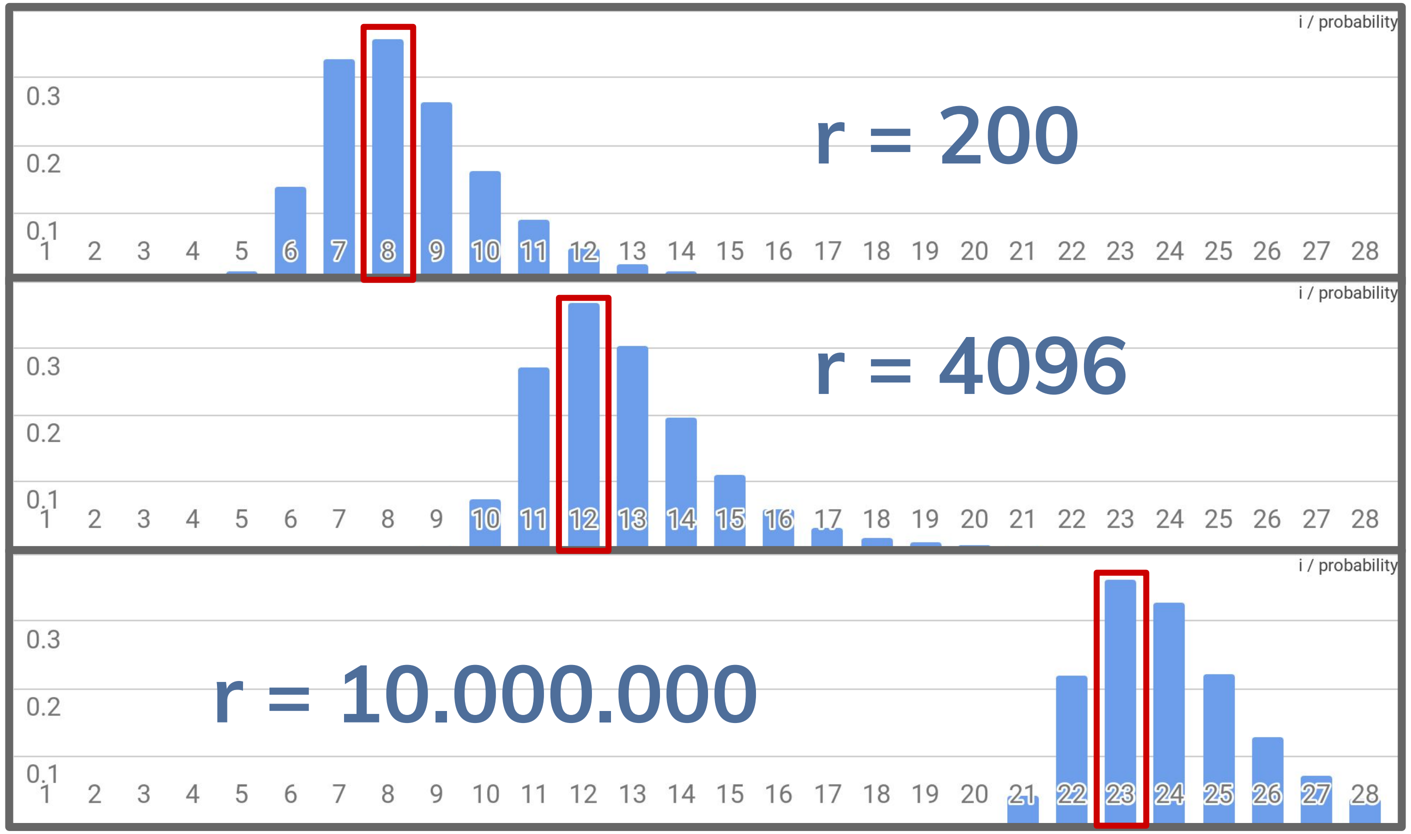
Variant (b)



- Multiple hash function;
- Independent buckets (easier).

ℓ_0-sampling algorithm

$$p_i = r2^{-i} \exp(-r2^{-i})$$



Observations

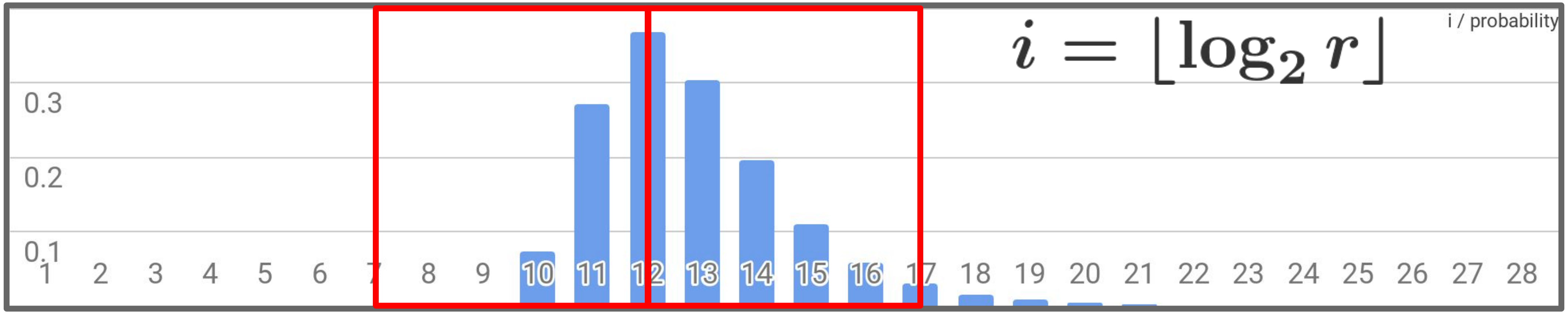
1. We define r, the number of nonzero coordinates in a vector. p_i is the probability of the i^{th} bucket being 1-sparse.

2. It is easy to see that for every value of r, there will always be a bucket with high probability of recovery (~0.35).

3. There will also be other adjacent buckets with high probability of recovery.

ℓ_0-sampling algorithm

$m = \lceil \log_2 n + 5 \rceil$ is enough to ensure a failure probability of **less than 0.31.**

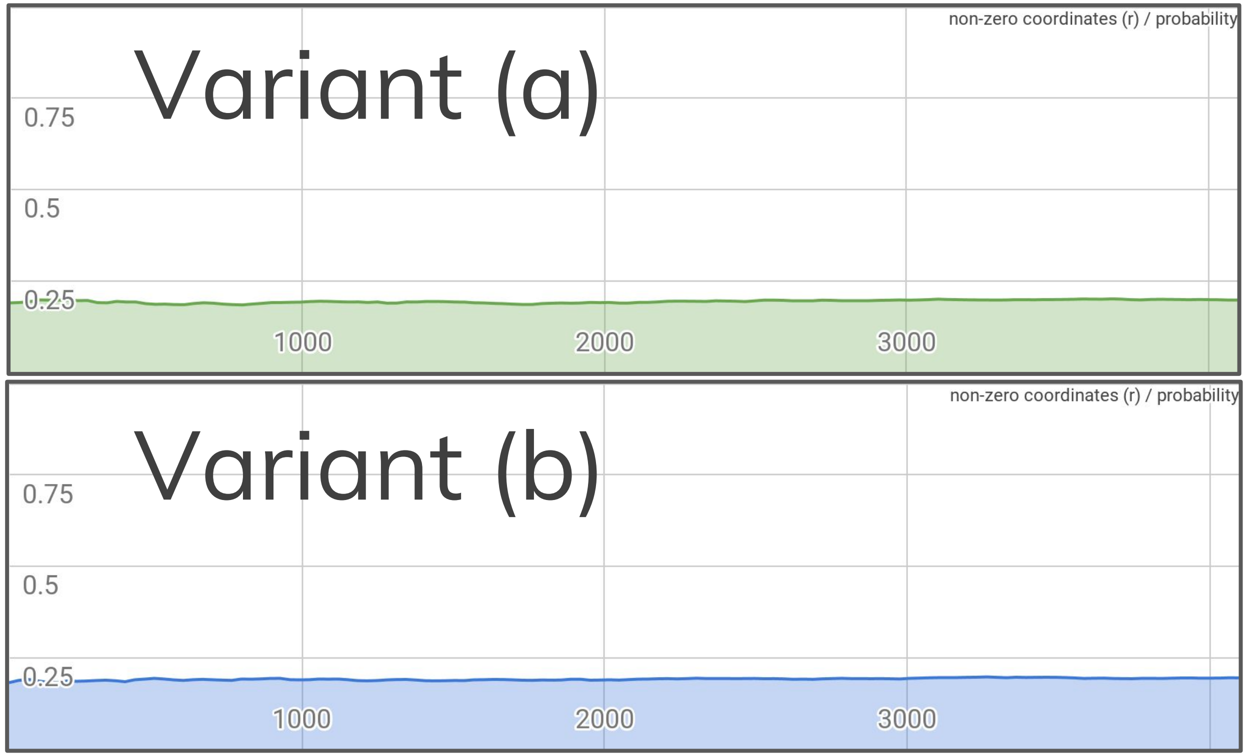


$$\mathbf{Pr}[\text{FAILURE}] \le \prod_{k=i-5}^{i+5} 1 - r2^{-k} \exp(-r2^{-k}) \le 0.31$$

analyzing factors' maxima

Experimental results

Correcly sized setup.



Observations

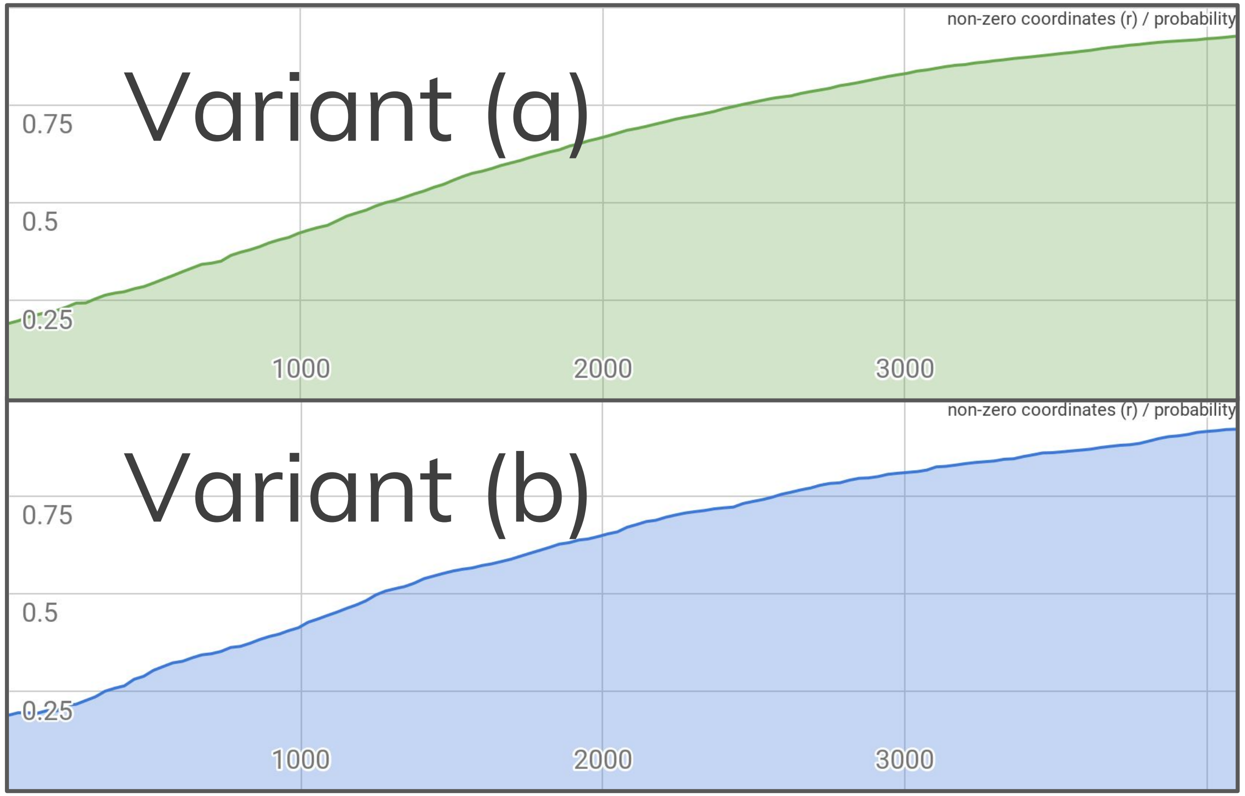
1. We tested both variants in a correctly sized setup, i.e. $r \leq 4096$, $m = 17$.

2. Variants behave similarly, with error apparently constant under 20% in both tests.

3. The distribution of sampled coordinates (not shown) was also similar in both tests.

Experimental results

Undersized setup.



Observations

1. We tested both variants in an **undersized** setup, i.e. $r \leq 4096$, **m = 10**.

2. Variants behave similarly, with error growing from under 20% to almost 100% in both tests.

3. The distribution of sampled coordinates (not shown) was also similar in both tests.

Conclusion

What should we expect from **sketching data structures** in a near future?

In this talk...

... I presented the application of three **sketching data structures** for massive graph problems.

Bloom Filter

Adjacency test on general graphs in O(m) bits. Specially useful for sparse massive graphs. Has constant probability of false positives. No false negatives.

MinHash

Adjacency test on trees in O(n) bits. Better space complexity than the optimal deterministic representation. Useful for giant trees (over a billion nodes).

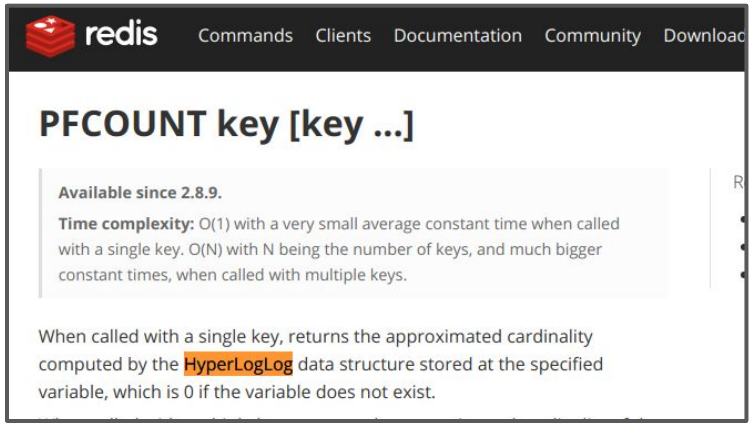
ℓ_0-Sampler

Dynamic spanning forest in $O(n \log^3 n)$ bits. Useful for very dense graphs.

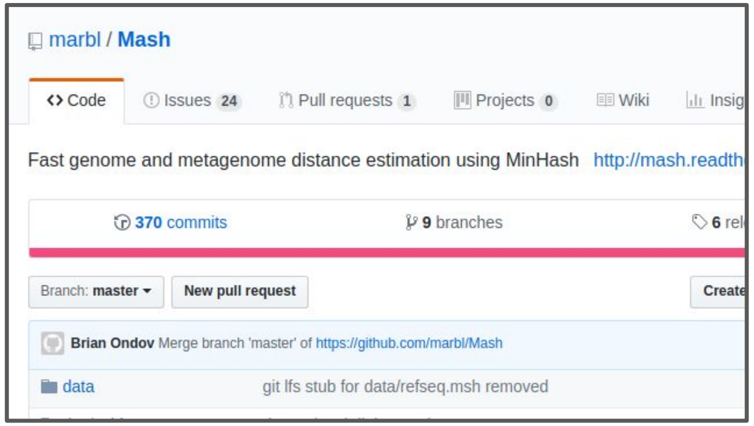
Sketching data structures are growing

Not only a theory. Not only for graphs.

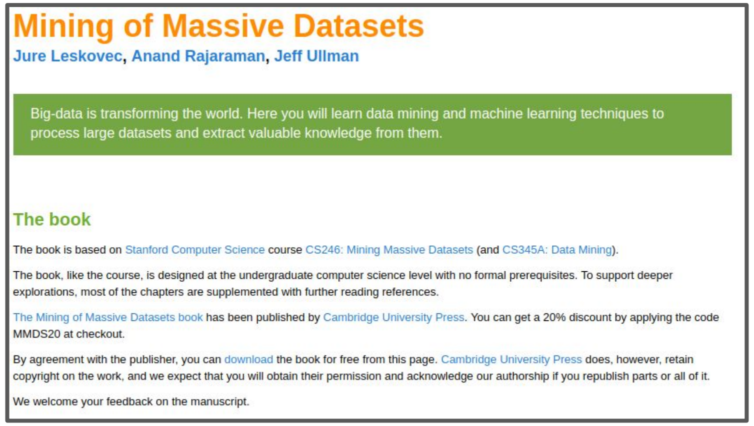
Mash: Fast genome and metagenome distance estimation using **MinHash**.



MMDS book chapter 4: several **sketch-based** stream algorithms.



Redis **PFCOUNT**: set distinct count using **HyperLogLog**.



Our next steps

We are searching for new algorithms that use ℓ_0-sampling as a primitive

ℓ_0-Sampler

The ability to sample edges from cut-sets is very useful and can help to produce many new graph algorithms.

Questions?

Slidedeck available at:
juanlopes.net/poly18