European Research Council

Established by the European Commission

Streaming lower bounds through boolean Fourier analysis

Michael Kapralov

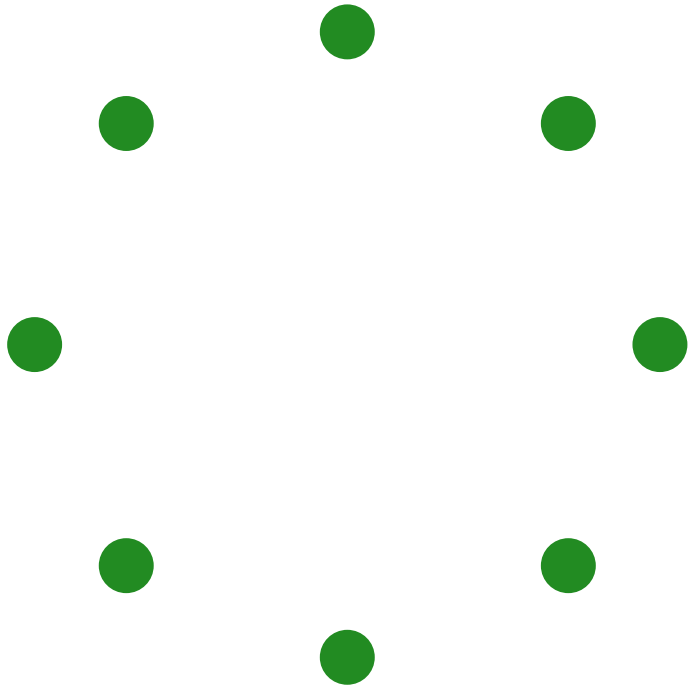
EPFL

Based on joint works with Ashish Chiplunkar, John Kallaugher, Dmitry Krachun and Eric Price

August 4, 2022

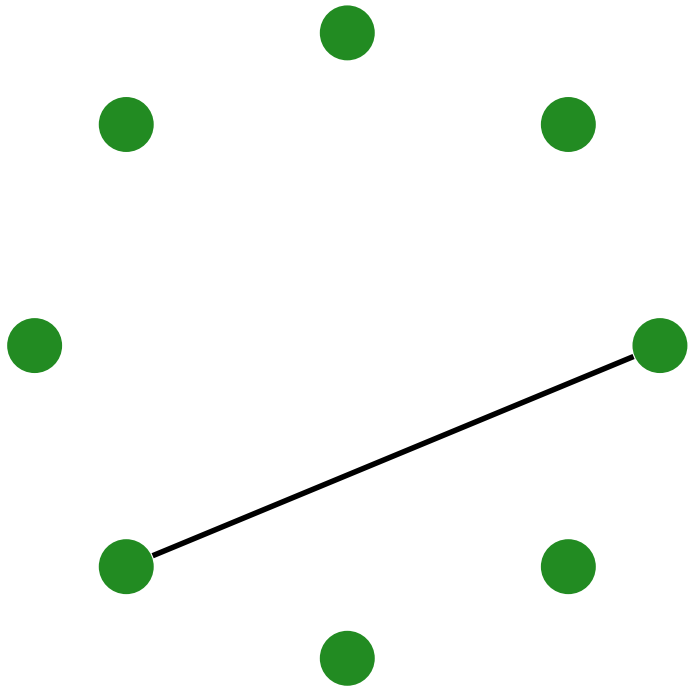
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



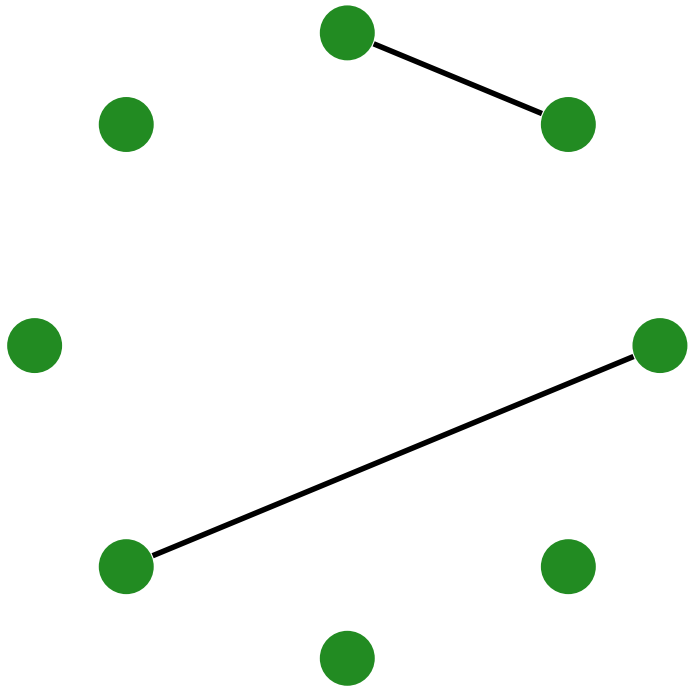
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



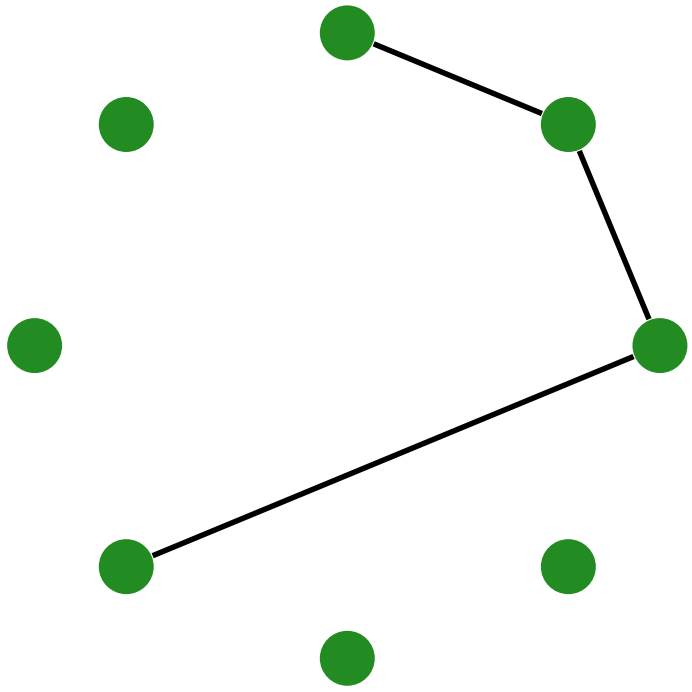
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



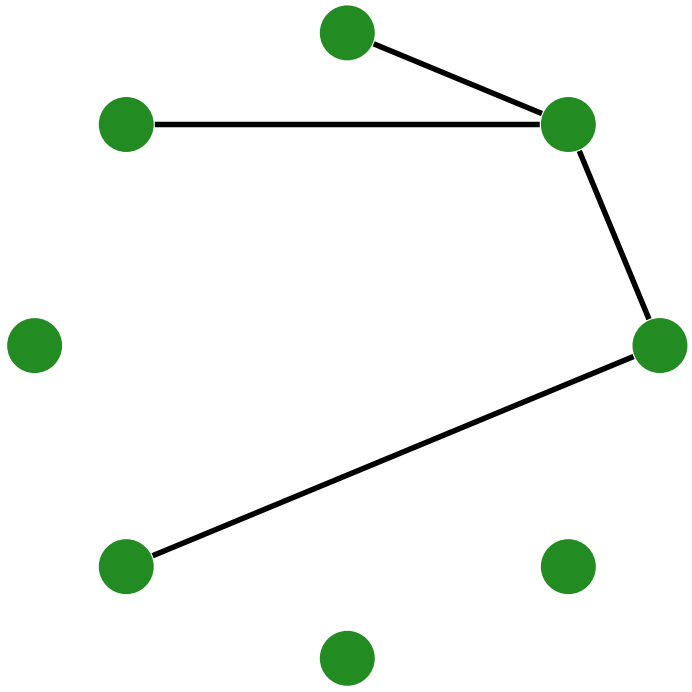
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



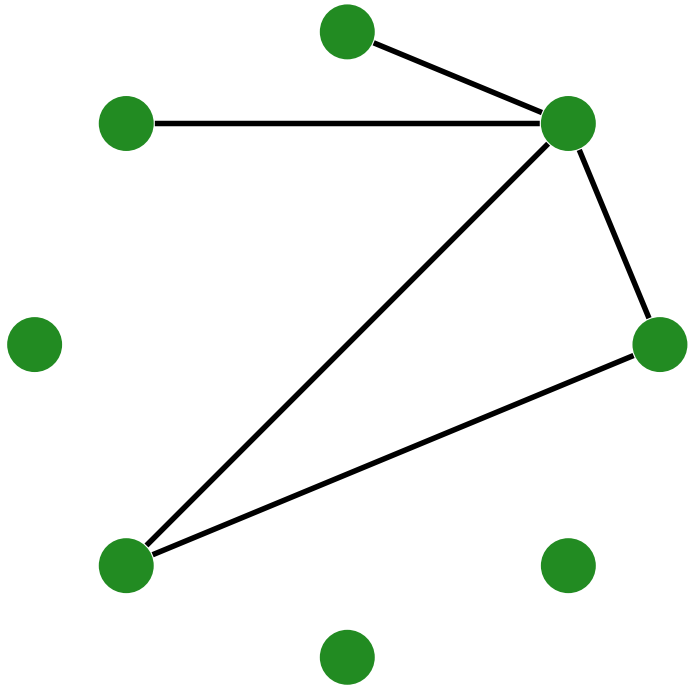
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



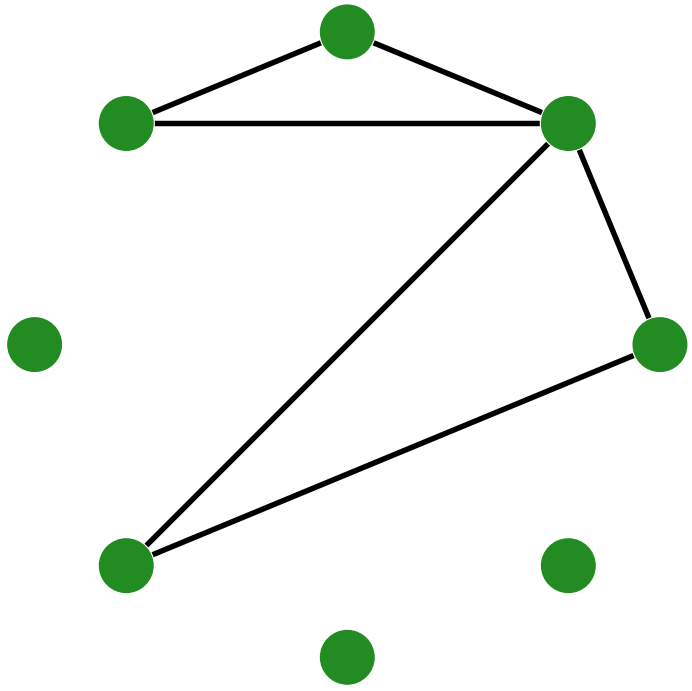
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



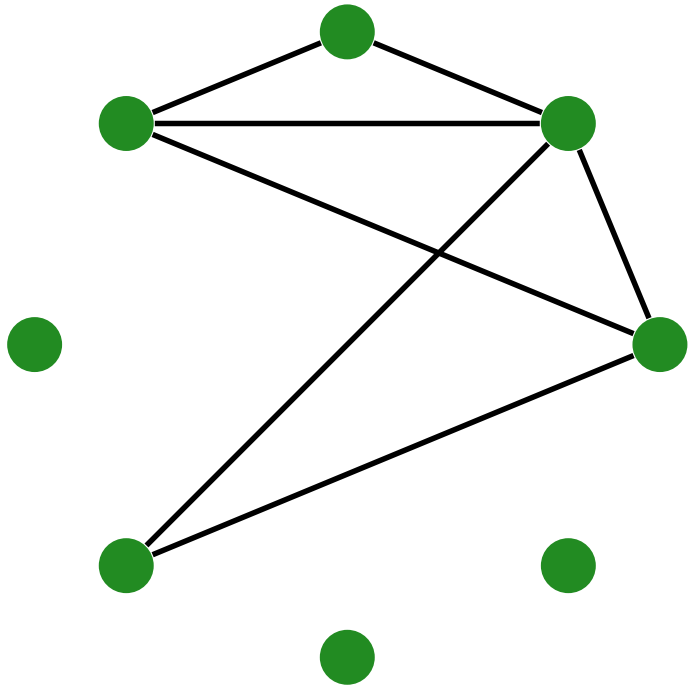
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



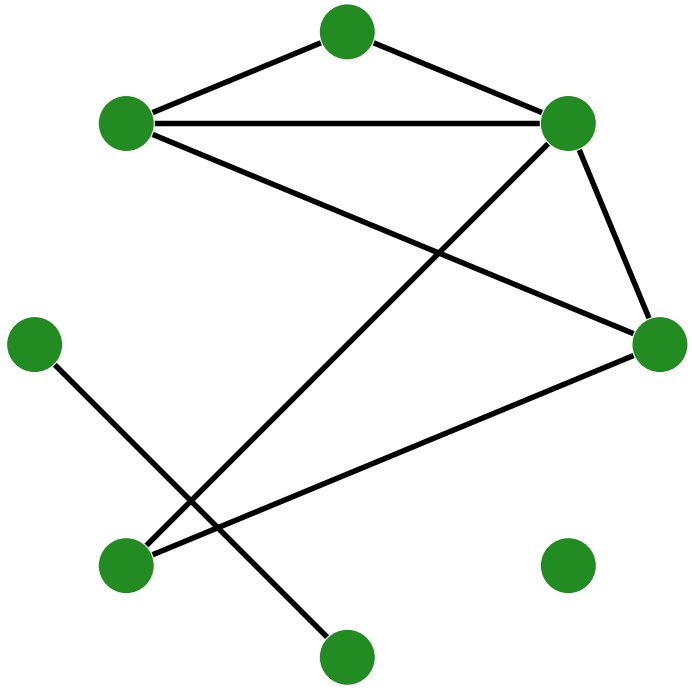
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



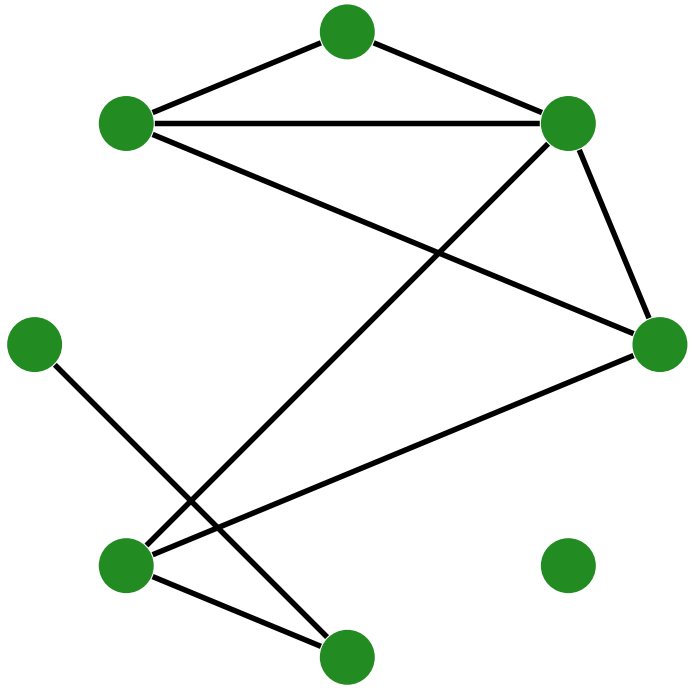
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



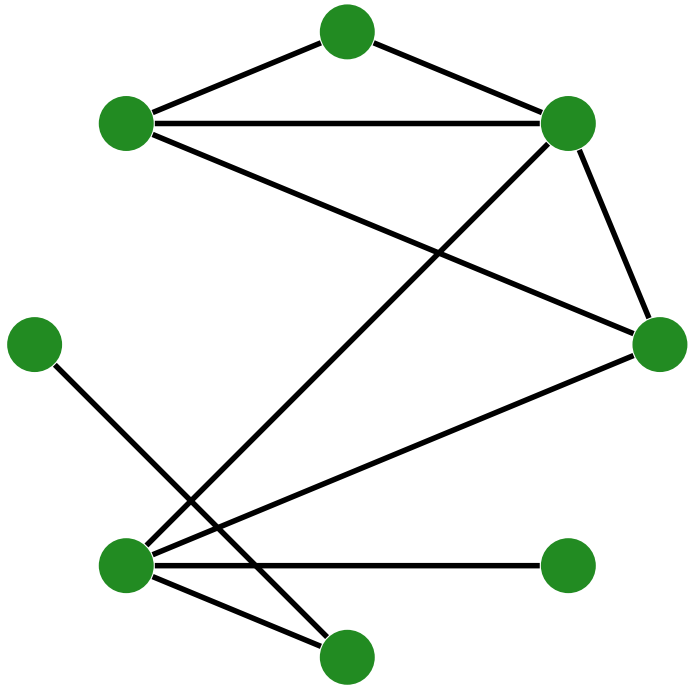
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



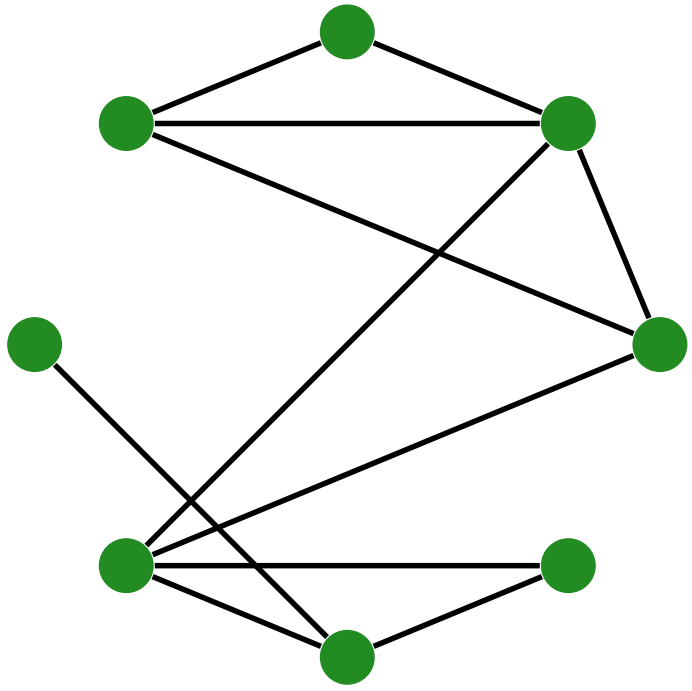
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



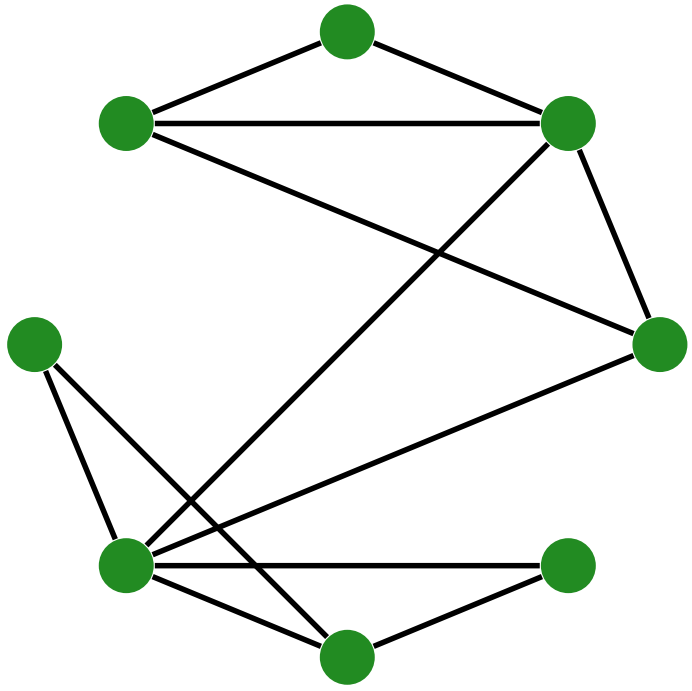
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



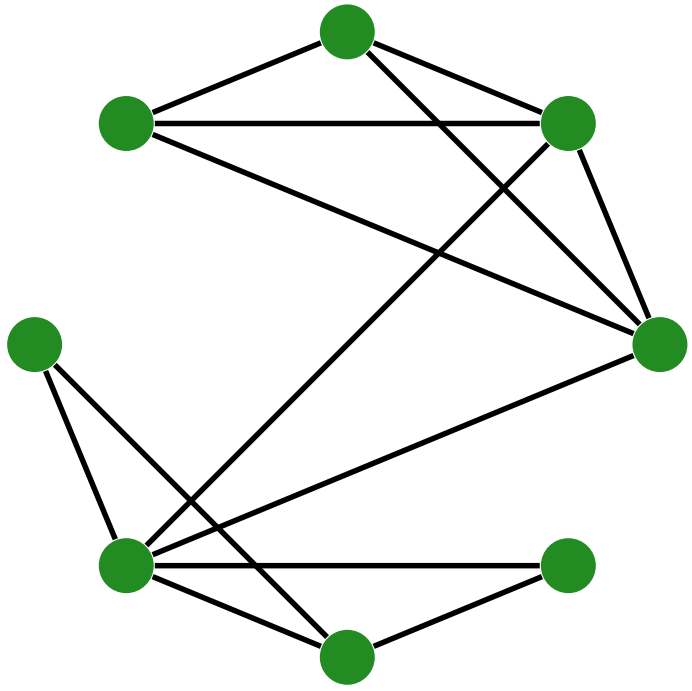
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



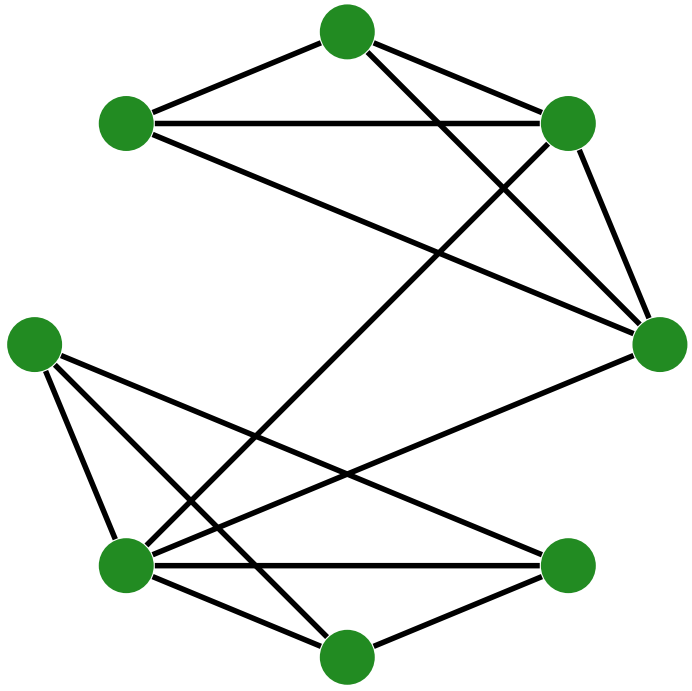
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



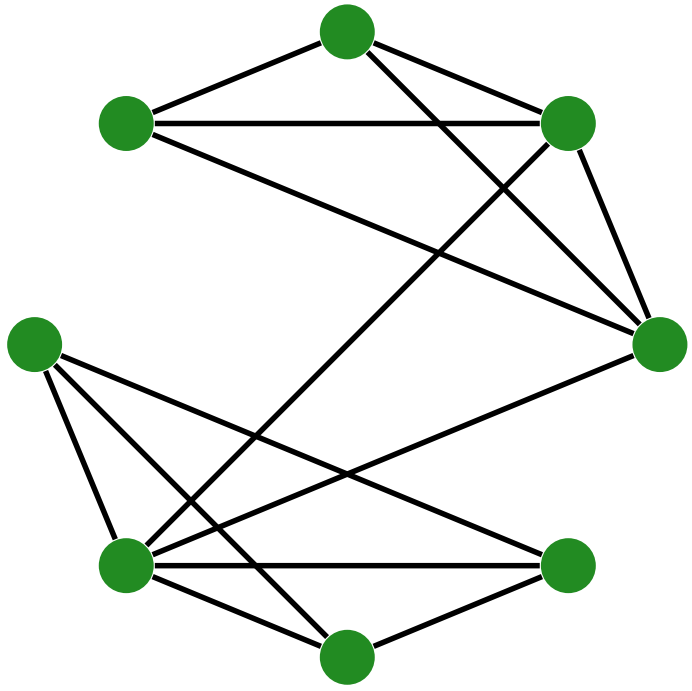
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



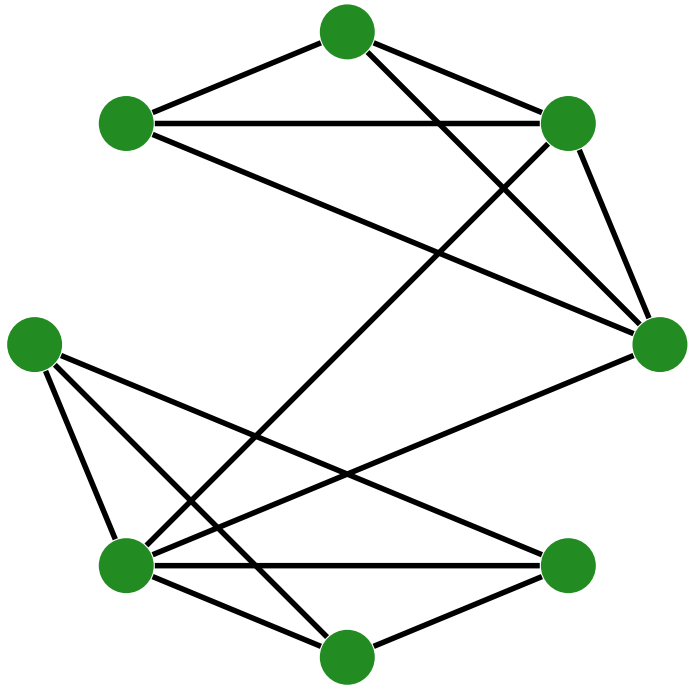
Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream



Streaming model

- edges of $G = (V, E)$ arrive in an arbitrary order in a stream; denote $|V| = n, |E| = m$
- several passes over the stream **(ideally one pass)**



$\Omega(n)$ space is often needed:

- output size often $\Omega(n)$ (e.g., matching, sparsifier, spanner)
- even if output is a number (e.g. testing connectivity)

$\Omega(n)$ space is often needed:

- output size often $\Omega(n)$ (e.g., matching, sparsifier, spanner)
- even if output is a number (e.g. testing connectivity)

But not always:

K.-Khanna-Sudan'14 – can approximate matching size to $\text{poly}(\log n)$ factor using $\text{poly}(\log n)$ space in random streams.

$\Omega(n)$ space is often needed:

- output size often $\Omega(n)$ (e.g., matching, sparsifier, spanner)
- even if output is a number (e.g. testing connectivity)

But not always:

K.-Khanna-Sudan'14 – can approximate matching size to $\text{poly}(\log n)$ factor using $\text{poly}(\log n)$ space in random streams.

Matching, connected components, random walks:
Efsaniari-Hajiaghayi-Liaghat-Monemizadeh-Onak'15, Bury-Schwiegelsohn'15, McGregor-Vorotnikova'16, Cormode-Jowhari-Monemizadeh-Muthukrishnan'16,Peng-Sohler'18, K.-Mitrovic-Norouzi-Fard-Tardos'20, Kallaugher-K.-Price'22,...

$\Omega(n)$ space is often needed:

- output size often $\Omega(n)$ (e.g., matching, sparsifier, spanner)
- even if output is a number (e.g. testing connectivity)

But not always:

K.-Khanna-Sudan'14 – can approximate matching size to $\text{poly}(\log n)$ factor using $\text{poly}(\log n)$ space in random streams.

Matching, connected components, random walks:
Efsaniari-Hajiaghayi-Liaghat-Monemizadeh-Onak'15, Bury-Schwiegelsohn'15, McGregor-Vorotnikova'16, Cormode-Jowhari-Monemizadeh-Muthukrishnan'16,Peng-Sohler'18, K.-Mitrovic-Norouzi-Fard-Tardos'20, Kallaugher-K.-Price'22,...

Streaming complexity of constraint satisfaction problems:
Kogan-Krauthgamer'14, K-Khanna-Sudan'14, K-Khanna-Sudan-Velingker'17, Guruswami-Velingker-Velusamy'17, K.-Krachun'19, Guruswami-Tao, '19, Chou-Golovnev-Velusamy'20, Singer-Sudan-Velusamy'21, Chou-Golovnev-Sudan-Velingker-Velusamy'22, Arunachalam-Doriguello'21,...

Algorithmic techniques

Often a subgraph exploration processes: maintain a (carefully and adaptively) chosen subgraph

Algorithmic techniques

Often a subgraph exploration processes: maintain a (carefully and adaptively) chosen subgraph

In random order streams: use randomness to perform statistical estimation of various quantities

Algorithmic techniques

Often a subgraph exploration processes: maintain a (carefully and adaptively) chosen subgraph

In random order streams: use randomness to perform statistical estimation of various quantities

Rule of thumb: if 'storing a subgraph' is 'optimal', then can get tight lower bound using boolean Fourier analysis

Testing bipartiteness: Goldreich-Ron'00 can tell if graph is bipartite or ε-far from bipartite in $\approx \text{poly}(1/\varepsilon, \log n) n^{1/2}$ queries.
Streaming?

(check if $\approx n^{1/2}$ random walks of even and odd $\approx \text{poly}(\log n)$ length collide)

Testing bipartiteness: Goldreich-Ron'00 can tell if graph is bipartite or ε-far from bipartite in $\approx \text{poly}(1/\varepsilon, \log n) n^{1/2}$ queries. Streaming?

(check if $\approx n^{1/2}$ random walks of even and odd $\approx \text{poly}(\log n)$ length collide)

In a (bounded degree, say) graph, approximate # of triangles

(Kallaugher-K.-Price'18: color coding+careful sampling of a vertex-induced subgraph)

Testing bipartiteness: Goldreich-Ron'00 can tell if graph is bipartite or ε-far from bipartite in $\approx \text{poly}(1/\varepsilon, \log n) n^{1/2}$ queries. Streaming?

(check if $\approx n^{1/2}$ random walks of even and odd $\approx \text{poly}(\log n)$ length collide)

In a (bounded degree, say) graph, approximate # of triangles

(Kallaugher-K.-Price'18: color coding+careful sampling of a vertex-induced subgraph)

Count # of connected components in a graph

(Peng-Sohler'18: approx # of connected components to εn error in $\approx (1/\varepsilon)^{1/\varepsilon^3}$ space)

Testing bipartiteness: Goldreich-Ron'00 can tell if graph is bipartite or ε-far from bipartite in $\approx \text{poly}(1/\varepsilon, \log n) n^{1/2}$ queries. Streaming?

(check if $\approx n^{1/2}$ random walks of even and odd $\approx \text{poly}(\log n)$ length collide)

In a (bounded degree, say) graph, approximate # of triangles

(Kallaugher-K.-Price'18: color coding+careful sampling of a vertex-induced subgraph)

Count # of connected components in a graph

(Peng-Sohler'18: approx # of connected components to εn error in $\approx (1/\varepsilon)^{1/\varepsilon^3}$ space)

Random walk generation, PAGERANK estimation

(Kallaugher-K.-Price'22: (careful) rejection sampling; space exponential in walk length)

Testing bipartiteness: Goldreich-Ron'00 can tell if graph is bipartite or ε-far from bipartite in $\approx \text{poly}(1/\varepsilon, \log n) n^{1/2}$ queries. Streaming?

(check if $\approx n^{1/2}$ random walks of even and odd $\approx \text{poly}(\log n)$ length collide)

In a (bounded degree, say) graph, approximate # of triangles

(Kallaugher-K.-Price'18: color coding+careful sampling of a vertex-induced subgraph)

Count # of connected components in a graph

(Peng-Sohler'18: approx # of connected components to εn error in $\approx (1/\varepsilon)^{1/\varepsilon^3}$ space)

Random walk generation, PAGERANK estimation

(Kallaugher-K.-Price'22: (careful) rejection sampling; space exponential in walk length)

(Nearly) tight lower bounds using Fourier analytic techniques

1. The Boolean Hidden Matching problem

2. Multiplayer games via the convolution theorem

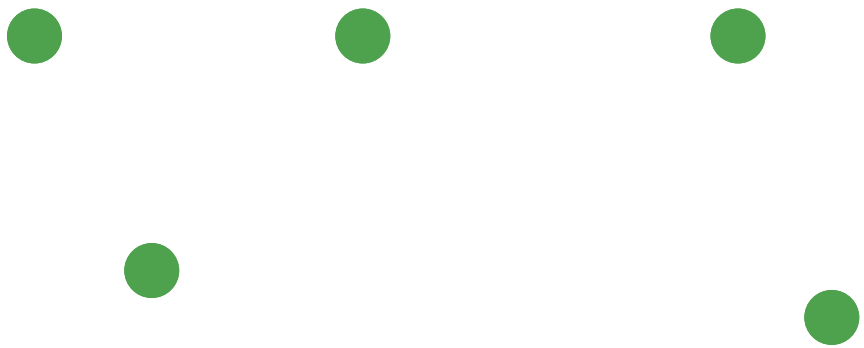
3. A game with a ‘infinite’ number of players

1. The Boolean Hidden Matching problem
2. Multiplayer games via the convolution theorem
3. A game with a ‘infinite’ number of players

Boolean hidden matching problem (BHM)

Alice

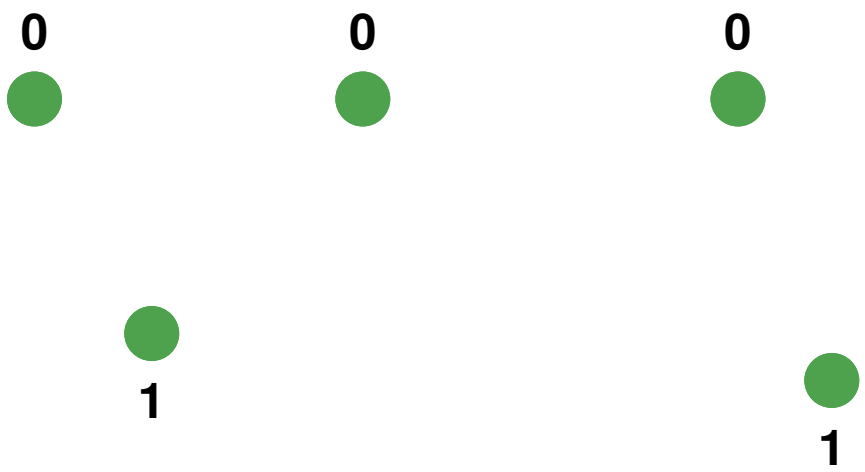
binary string $x \in \{0,1\}^n$



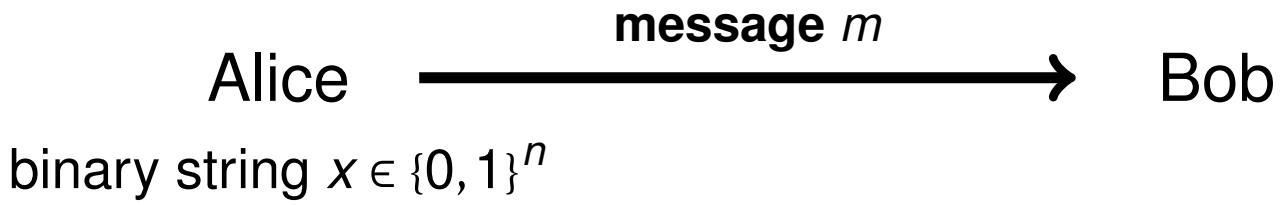
Boolean hidden matching problem (BHM)

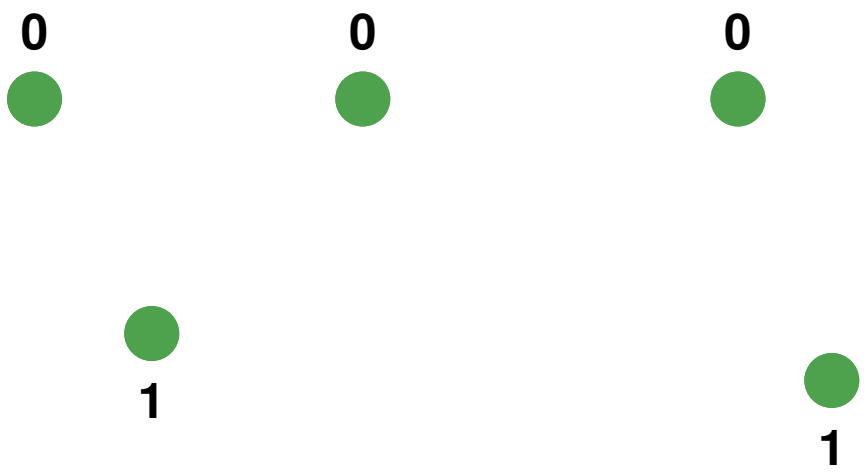
Alice

binary string $x \in \{0,1\}^n$

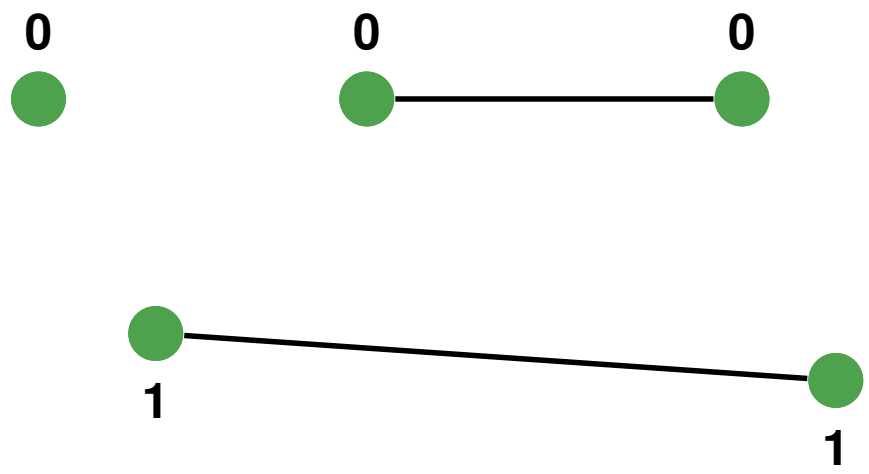


Boolean hidden matching problem (BHM)

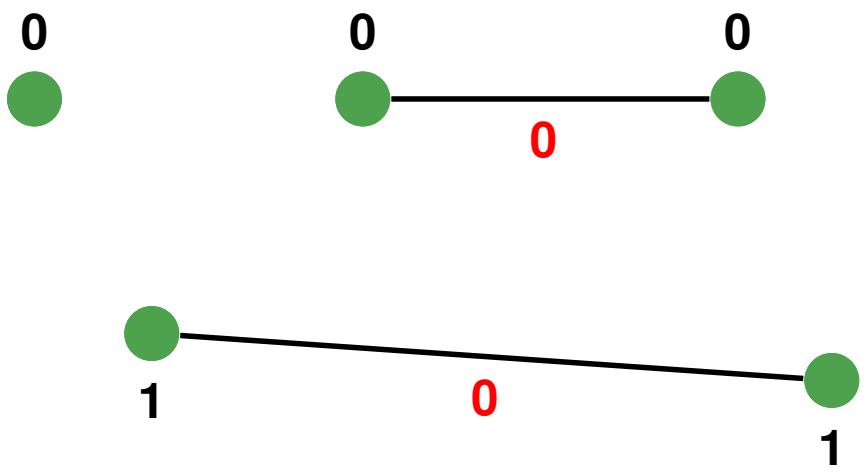




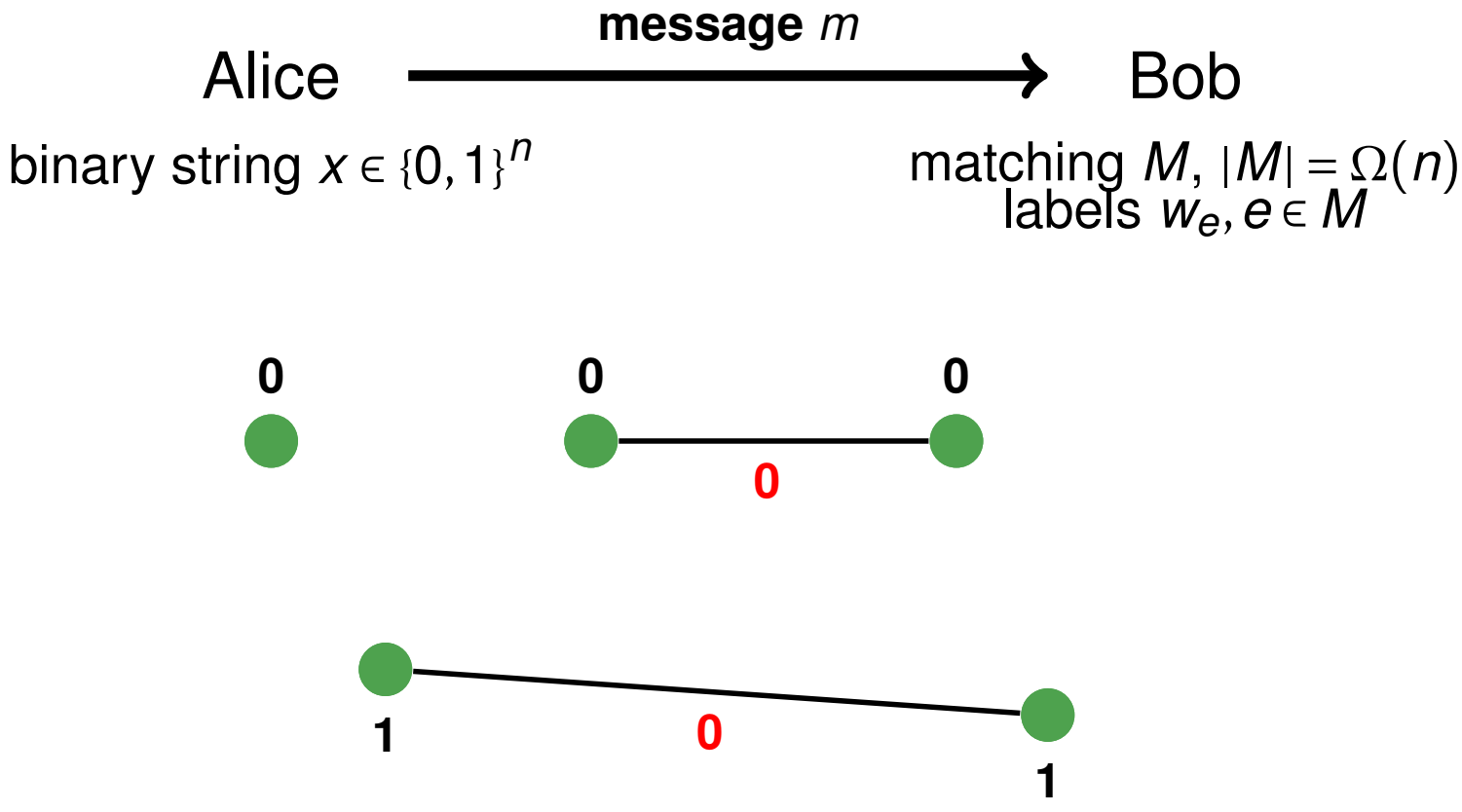
Boolean hidden matching problem (BHM)



Boolean hidden matching problem (BHM)



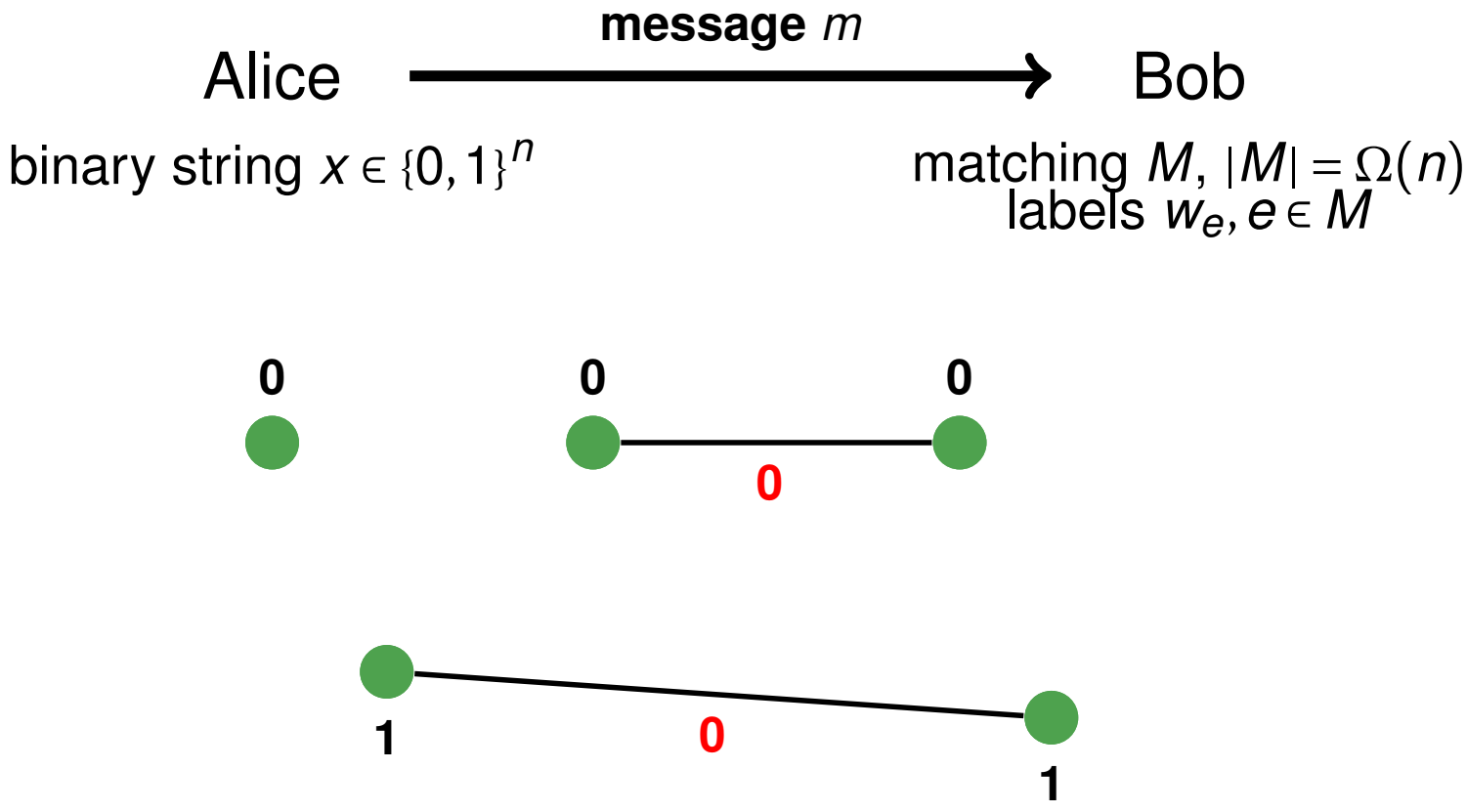
Boolean hidden matching problem (BHM)



In the YES case $w_e = x_u + x_v$ for every $e = (u, v) \in M$

In the NO case $w_e \sim UNIF(\{0,1\})$ for every $e = (u, v) \in M$

Boolean hidden matching problem (BHM)

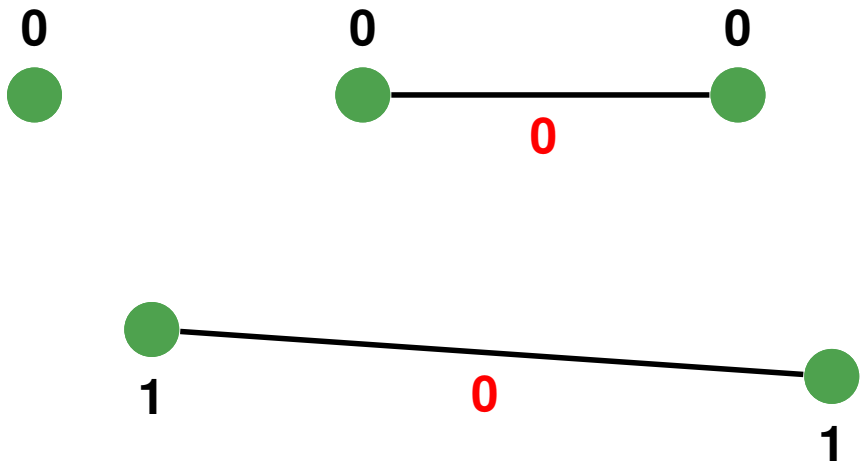


In the YES case $w_e = x_u + x_v$ for every $e = (u, v) \in M$

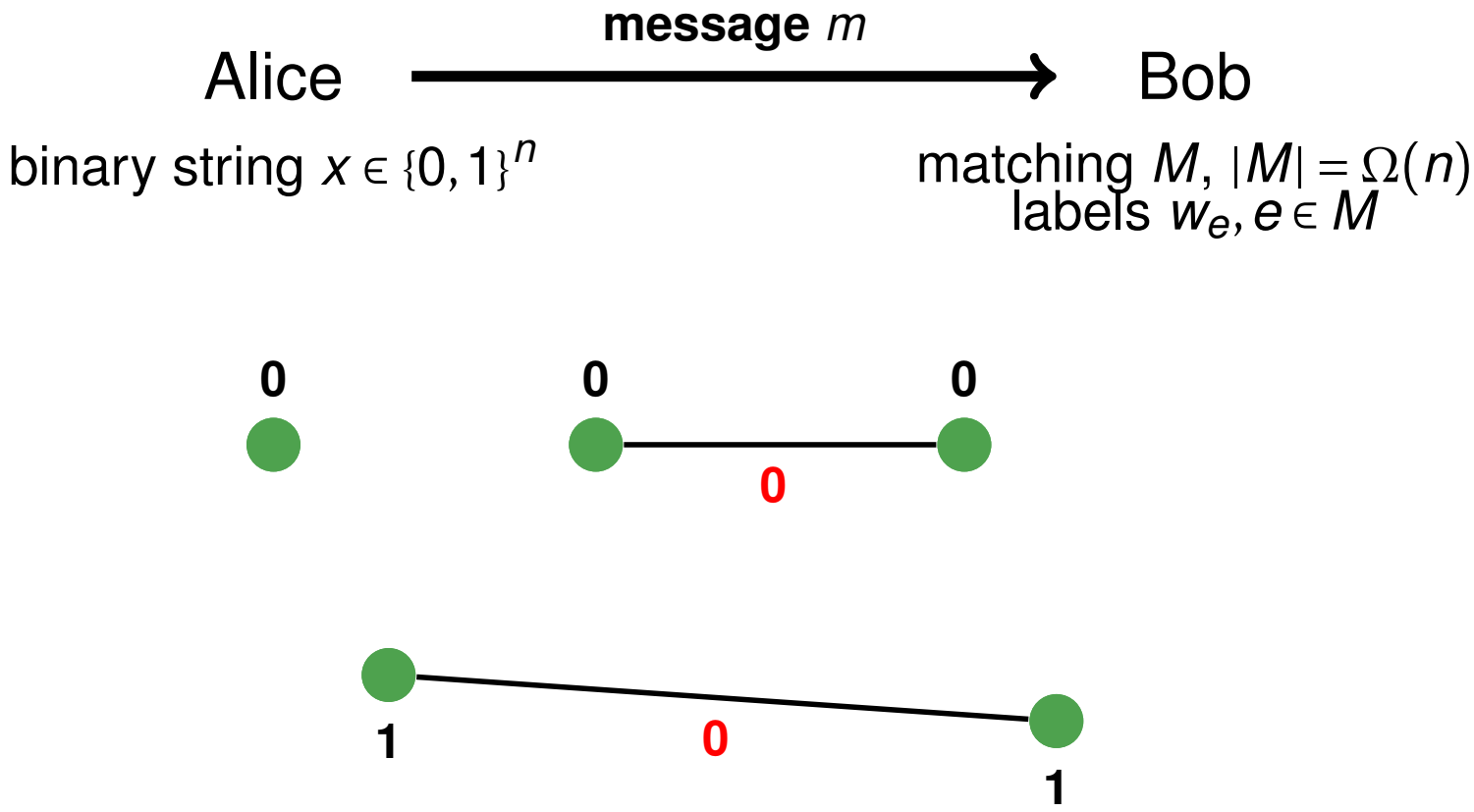
In the NO case $w_e \sim UNIF(\{0,1\})$ for every $e = (u, v) \in M$

Bob's task: distinguish between YES and NO cases

Boolean hidden matching problem (BHM)



Boolean hidden matching problem (BHM)



Bob's task: distinguish between YES and NO cases

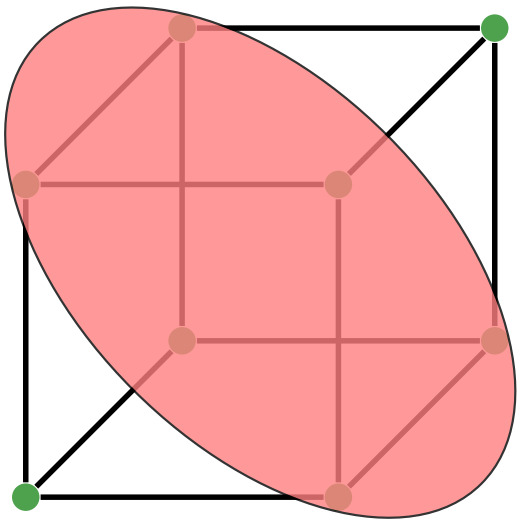
Sample $\approx \sqrt{n}$ coordinates of x, send values to Bob. Bob will know x_u and x_v for some $e = (u, v) \in M$, can check if $w_e = x_u + x_v$

Gavinsky et al.'07 showed $\Omega(\sqrt{n})$ space is needed

Conditioned on Alice's message, what is the distribution of MX?

Conditioned on Alice's message, what is the distribution of MX?

$X \sim UNIF(A)$
conditioned on m



$|A| \approx 2^{n-s}$

$f(x) :=$indicator of A

Chance of guessing $x_i + x_j$ is

$$\approx \frac{1}{2} + \left|\widetilde{f}(\{i,j\})\right|,$$

where for $z \in \{0,1\}^n$

$$\widetilde{f}(z) = \mathbb{E}_{x \sim UNIF(A)}[(-1)^{x \cdot z}]$$

is a normalized Fourier transform of f.

Chance of guessing $x_i + x_j$ is

$$\approx \frac{1}{2} + \left|\widetilde{f}(\{i,j\})\right|,$$

where for $z \in \{0,1\}^n$

$$\widetilde{f}(z) = \mathbb{E}_{x \sim UNIF(A)}[(-1)^{x \cdot z}]$$

is a normalized Fourier transform of f.

Show that

$$\mathbb{E}\left[\sum_{e=\{i,j\} \in M} \widetilde{f}(\{i,j\})^2\right] = o(1)?$$

Lemma (Gavinsky et al'07; from hypercontractivity)

If $f : \{0,1\}^n \to \{0,1\}$ *is the indicator function of a set* $A \subset \{0,1\}^n$, $|A| \geq 2^{n-s}$, *then*

$$\sum_{i,j} \widetilde{f}(\{i,j\})^2 \leq O(s^2).$$

Lemma (Gavinsky et al'07; from hypercontractivity)

If $f : \{0,1\}^n \to \{0,1\}$ *is the indicator function of a set* $A \subset \{0,1\}^n$, $|A| \geq 2^{n-s}$, *then*

$$\sum_{i,j} \widetilde{f}(\{i,j\})^2 \leq O(s^2).$$

For the trivial protocol A is a coordinate subspace, so

$$\left|\widetilde{f}(\{i,j\})\right| = \begin{cases} 1 & \text{if both } i \text{ and } j \text{ are known} \\ 0 & \text{o.w.} \end{cases}$$

and $\sum_{i<j} |\widetilde{f}(\{i,j\})| = \binom{s}{2}$

For every $i, j \in [n], i \neq j$, one has

$$\Pr[\{i, j\} \in M] \approx 1/n$$

For every $i,j \in [n], i \neq j$, one has

$$\Pr[\{i,j\} \in M] \approx 1/n$$

So

$$\mathbb{E}\left[\sum_{e=\{i,j\} \in M} \widetilde{f}(\{i,j\})^2\right] = O(c^2/n),$$

since $s = O(c)$ with constant probability

For every $i,j \in [n], i \neq j$, one has

$$\Pr[\{i,j\} \in M] \approx 1/n$$

So

$$\mathbb{E}\left[\sum_{e=\{i,j\} \in M} \widetilde{f}(\{i,j\})^2\right] = O(c^2/n),$$

since $s = O(c)$ with constant probability

Bob cannot guess parity of any edge in M unless $c = \Omega(\sqrt{n})$.

1. The Boolean Hidden Matching problem

2. Multiplayer games via the convolution theorem

3. A game with a ‘infinite’ number of players

1. The Boolean Hidden Matching problem

2. Multiplayer games via the convolution theorem

3. A game with a ‘infinite’ number of players

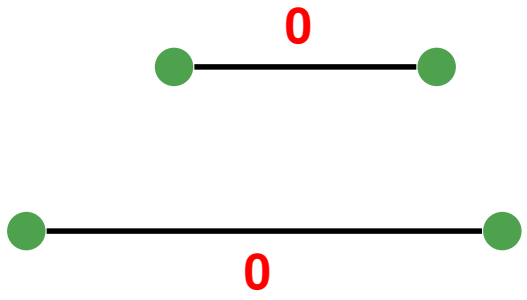
Testing bipartiteness: Goldreich-Ron'00: can tell if graph is bipartite or ε-far from bipartite in $\approx \mathrm{poly}(1/\varepsilon, \log n) n^{1/2}$ queries. Streaming?

(check if $\approx n^{1/2}$ random walks of even and odd $\approx \mathrm{poly}(\log n)$ length collide)

An Optimal Space Lower Bound for Approximating MAX-CUT

(STOC 2019; joint work with Dmitry Krachun)

Implicit Hidden Partition Problem

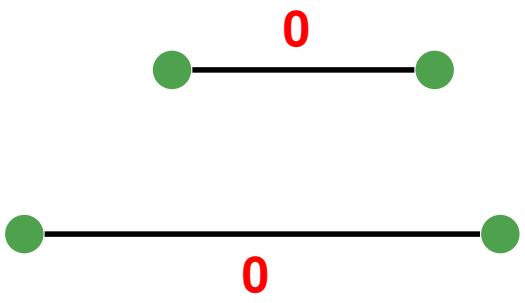


Player 1

matching M_1, labels

w^1 on edges

Implicit Hidden Partition Problem

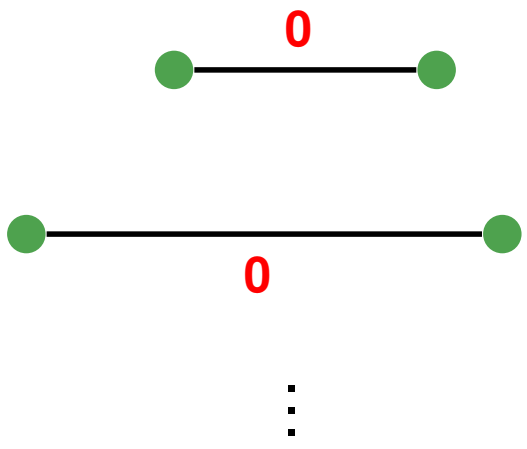


Player 1 $\longrightarrow$ m_1

matching M_1, labels

w^1 on edges

Implicit Hidden Partition Problem



Player 1 $\longrightarrow$ m_1

matching M_1, labels

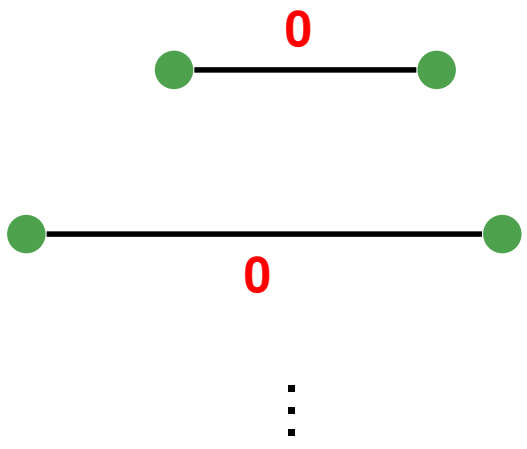
w^1 on edges

Player T

matching M_T, labels

w^T on edges

Implicit Hidden Partition Problem



Player 1 $\longrightarrow$ m_1

matching M_1, labels

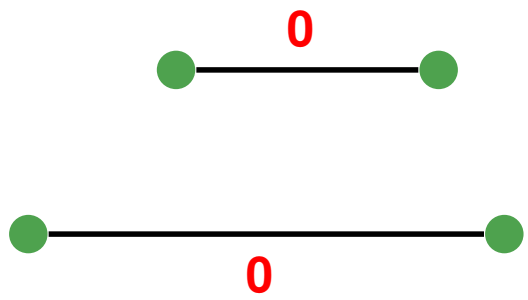
w^1 on edges

Player T $\longrightarrow$ m_T

matching M_T, labels

w^T on edges

Implicit Hidden Partition Problem



Player 1 $\longrightarrow$ m_1

matching M_1, labels

w^1 on edges

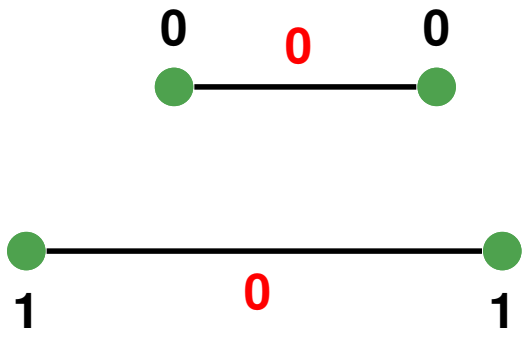
Player T $\longrightarrow$ m_T

matching M_T, labels

w^T on edges

YES case: $\exists$ partition $x \in \{0,1\}^n$ such that $w^t = M^t x$ for $1 \leq t \leq T$

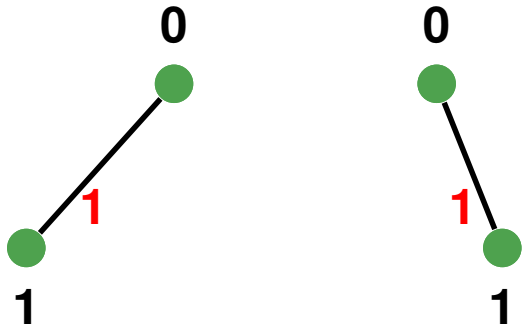
Implicit Hidden Partition Problem



Player 1 $\longrightarrow$ m_1

matching M_1, labels

w^1 on edges



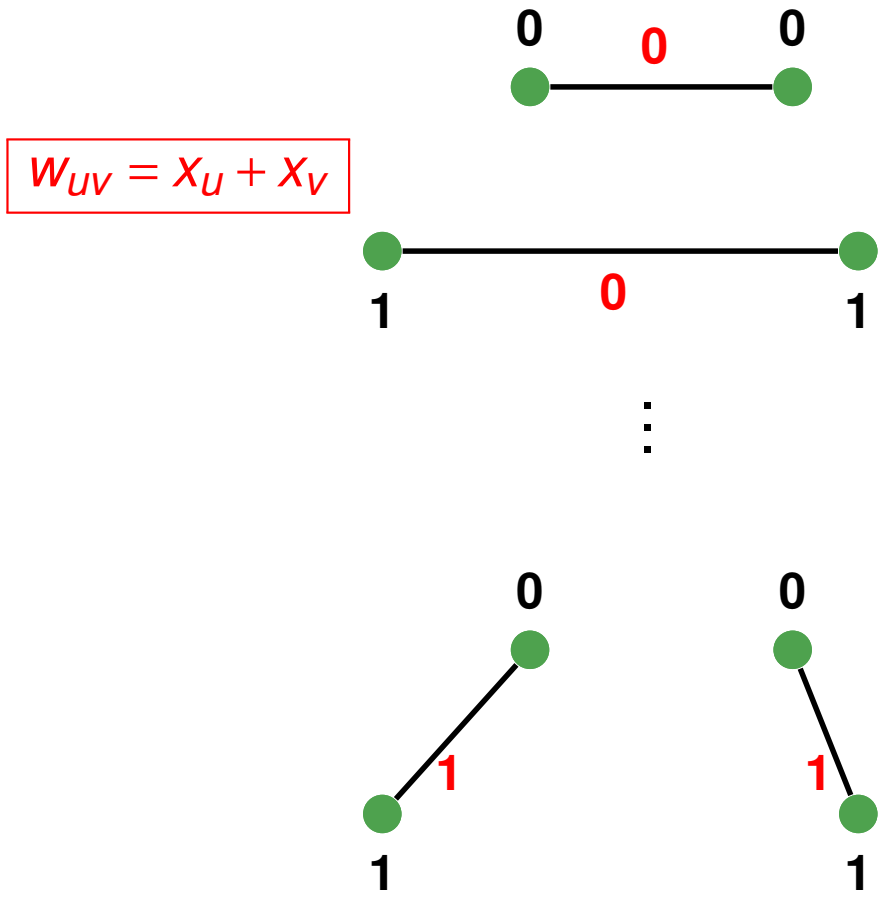
Player T $\longrightarrow$ m_T

matching M_T, labels

w^T on edges

YES case: $\exists$ partition $x \in \{0,1\}^n$ such that $w^t = M^t x$ for $1 \leq t \leq T$

Implicit Hidden Partition Problem



Player 1 $\longrightarrow$ m_1

matching M_1, labels

w^1 on edges

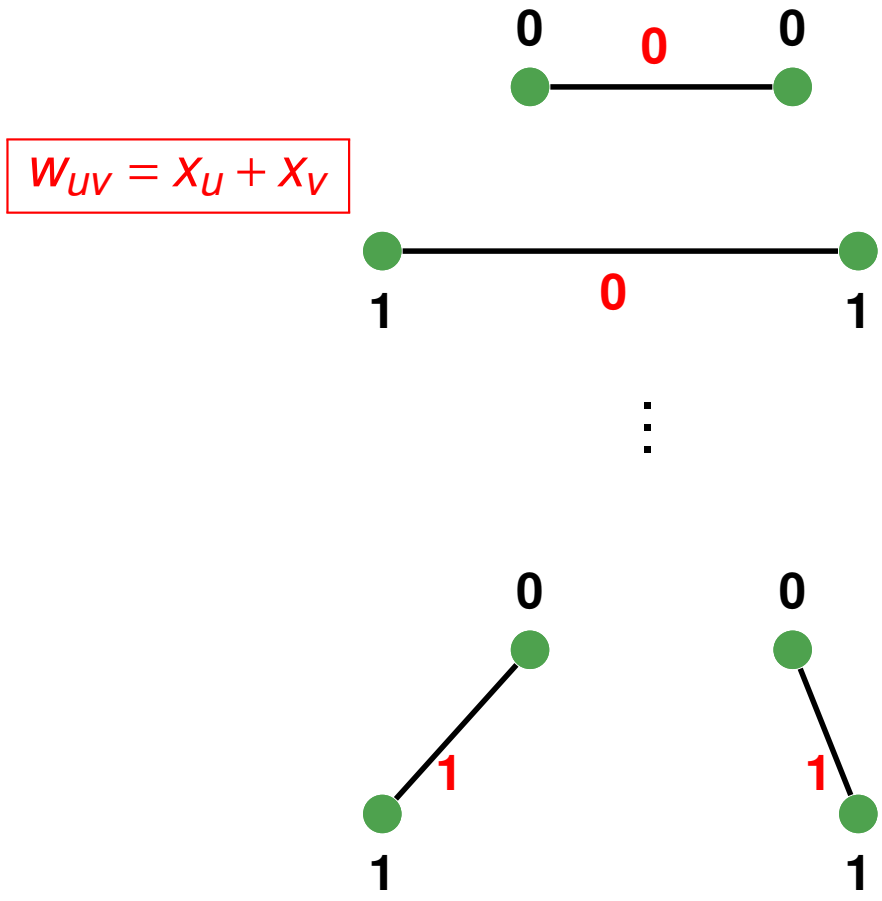
Player T $\longrightarrow$ m_T

matching M_T, labels

w^T on edges

YES case: $\exists$ partition $x \in \{0,1\}^n$ such that $w^t = M^t x$ for $1 \le t \le T$

Implicit Hidden Partition Problem



Player 1 $\longrightarrow$ m_1

matching M_1, labels

w^1 on edges

Player T $\longrightarrow$ m_T

matching M_T, labels

w^T on edges

YES case: $\exists$ partition $x \in \{0,1\}^n$ such that $w^t = M^t x$ for $1 \le t \le T$

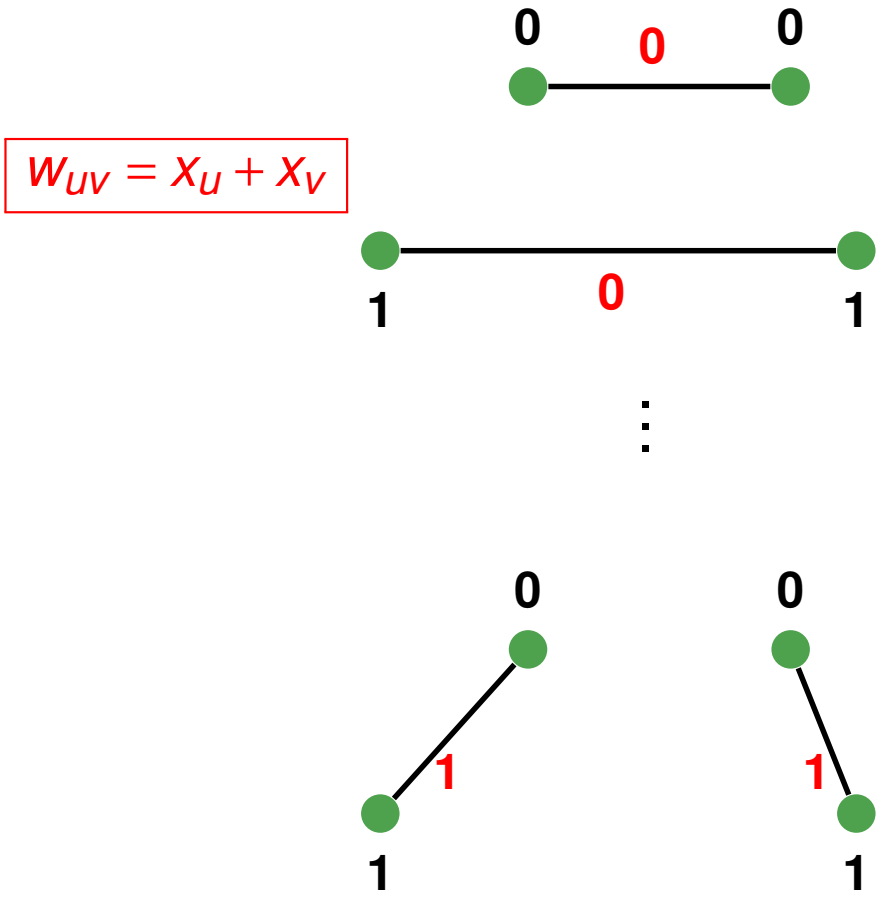
NO case: no such partition exists

Distributional communication problem

Choose a hidden partition $X \in UNIF(\{0,1\}^n)$

Distributional communication problem

Choose a hidden partition $X \in UNIF(\{0,1\}^n)$



Player 1 $\longrightarrow$ m_1

matching M_1, labels

w^1 on edges

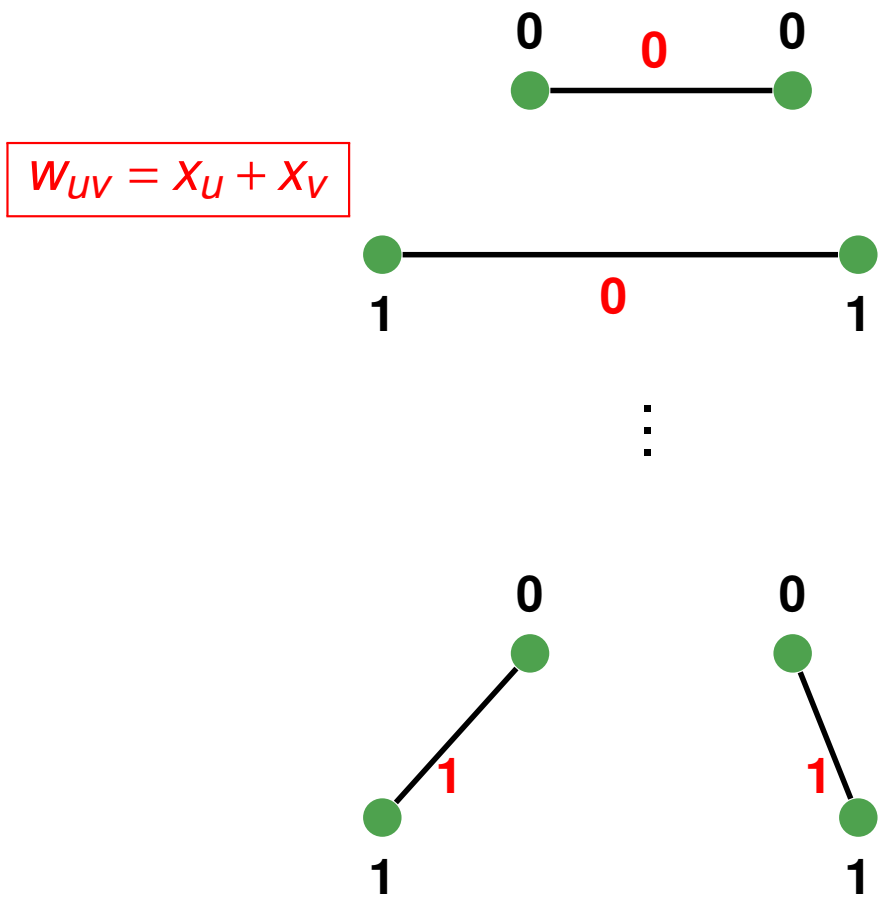
Player T $\longrightarrow$ m_T

matching M_T, labels

w^T on edges

Distributional communication problem

Choose a hidden partition $X \in UNIF(\{0,1\}^n)$



Player 1 $\longrightarrow$ m_1

matching M_1, labels

w^1 on edges

Player T $\longrightarrow$ m_T

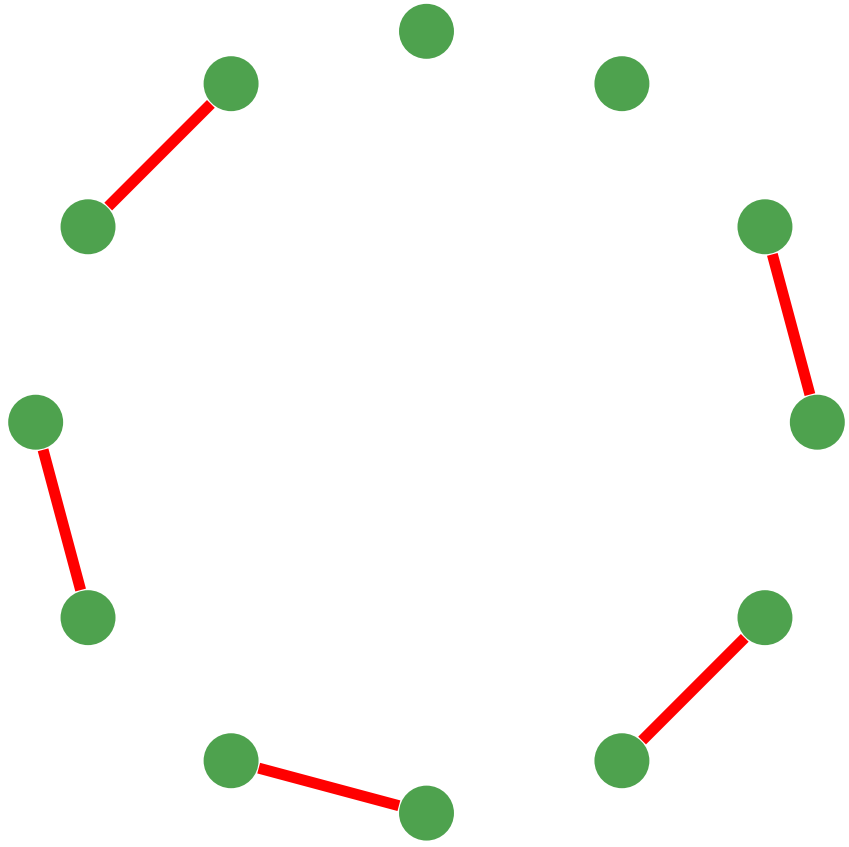
matching M_T, labels

w^T on edges

YES case: labels satisfy $w^t = M^t X$ for $1 \le t \le T$

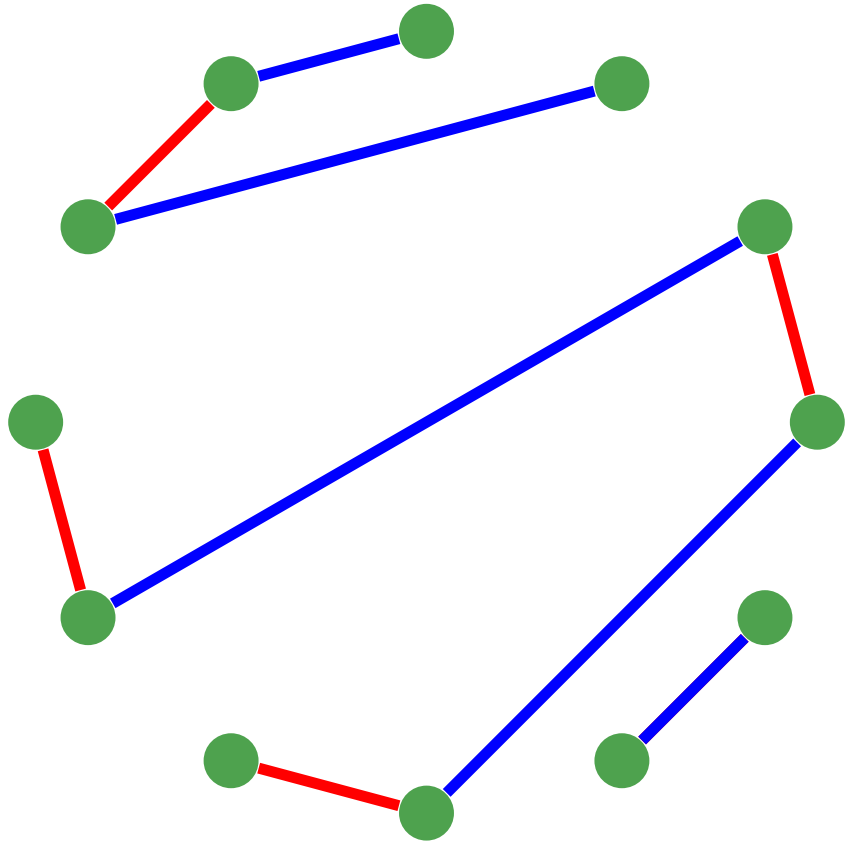
NO case: labels are random: $w^t \sim UNIF$

Random matchings



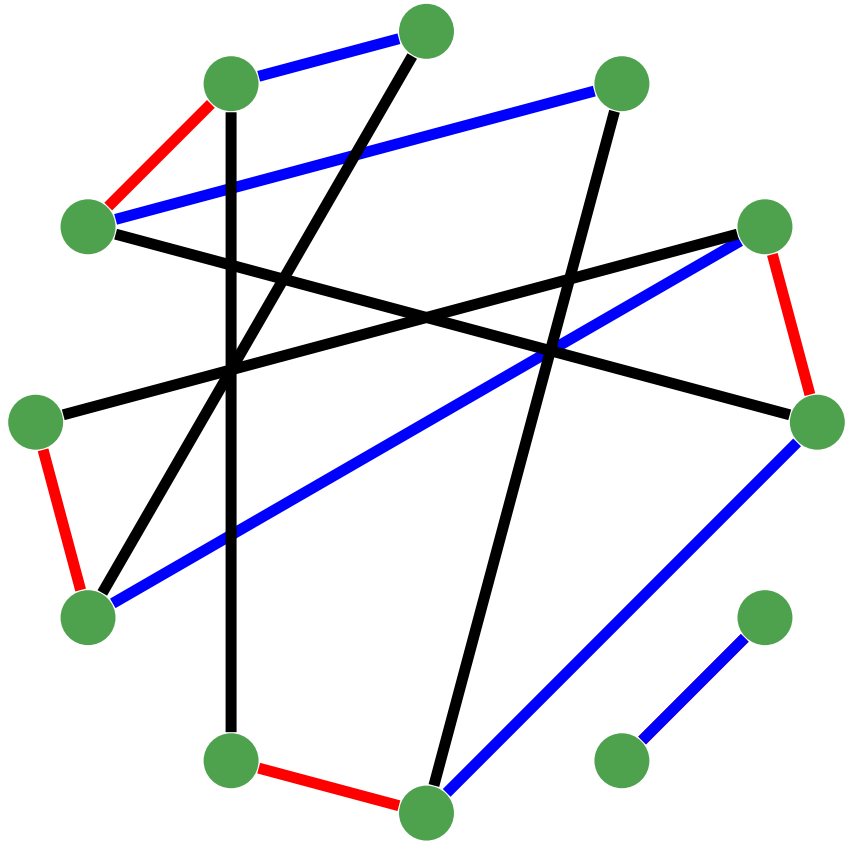
Random matchings of size $\approx n/100$

Random matchings



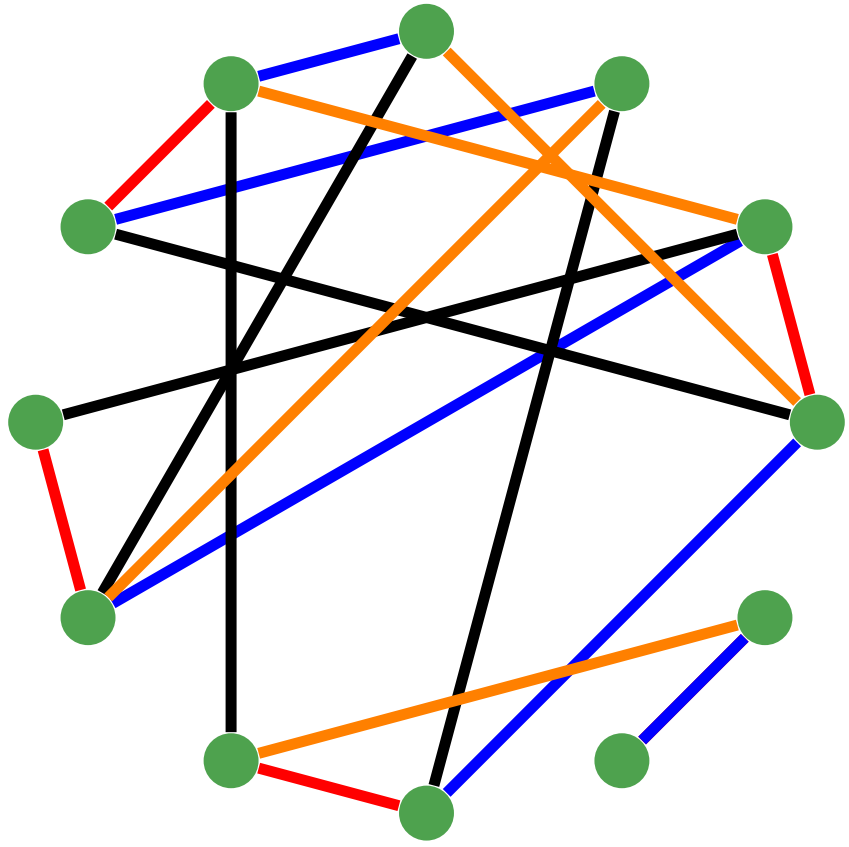
Random matchings of size $\approx n/100$

Random matchings



Random matchings of size $\approx n/100$

Random matchings



Random matchings of size $\approx n/100$

Reduction from MAX-CUT ($T \approx 1/\varepsilon^2$ for $2-\varepsilon$ approx)

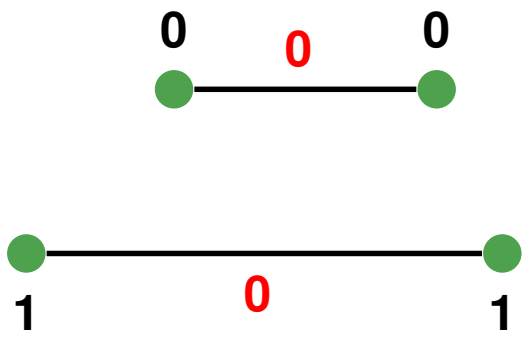
YES: random bipartite (multi)graph with expected degree $\approx \frac{1}{\varepsilon^2}$

NO: non-bipartite (multi)graph with expected degree $\approx \frac{1}{\varepsilon^2}$

Reduction from MAX-CUT ($T \approx 1/\varepsilon^2$ for $2-\varepsilon$ approx)

YES: random bipartite (multi)graph with expected degree $\approx \frac{1}{\varepsilon^2}$

NO: non-bipartite (multi)graph with expected degree $\approx \frac{1}{\varepsilon^2}$



Player t $\longrightarrow$ m_t

matching M_t, labels

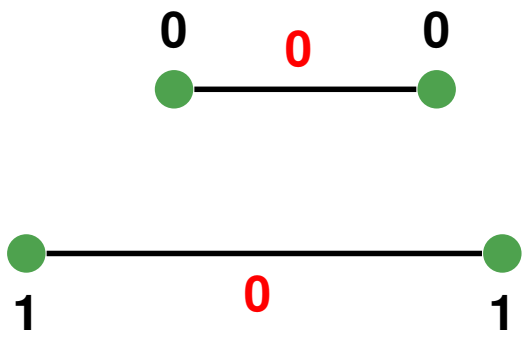
w^t on edges

t-th player generates graph G'_t by including edges $e \in G_t$ with $w^t_e = 1$

Reduction from MAX-CUT ($T \approx 1/\varepsilon^2$ for $2-\varepsilon$ approx)

YES: random bipartite (multi)graph with expected degree $\approx \frac{1}{\varepsilon^2}$

NO: non-bipartite (multi)graph with expected degree $\approx \frac{1}{\varepsilon^2}$



Player t $\longrightarrow$ m_t

matching M_t, labels

w^t on edges

t-th player generates graph G'_t by including edges $e \in G_t$ with $w^t_e = 1$

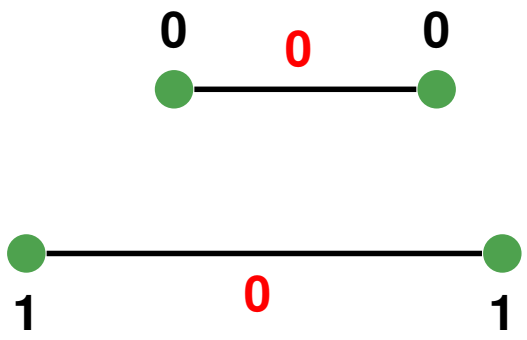
YES case: labels satisfy $w^t = M^t X$ for $1 \le t \le T$

$\bigcup_t G'_t$ is bipartite

Reduction from MAX-CUT ($T \approx 1/\varepsilon^2$ for $2-\varepsilon$ approx)

YES: random bipartite (multi)graph with expected degree $\approx \frac{1}{\varepsilon^2}$

NO: non-bipartite (multi)graph with expected degree $\approx \frac{1}{\varepsilon^2}$



Player t $\longrightarrow$ m_t

matching M_t, labels

w^t on edges

t-th player generates graph G'_t by including edges $e \in G_t$ with $w^t_e = 1$

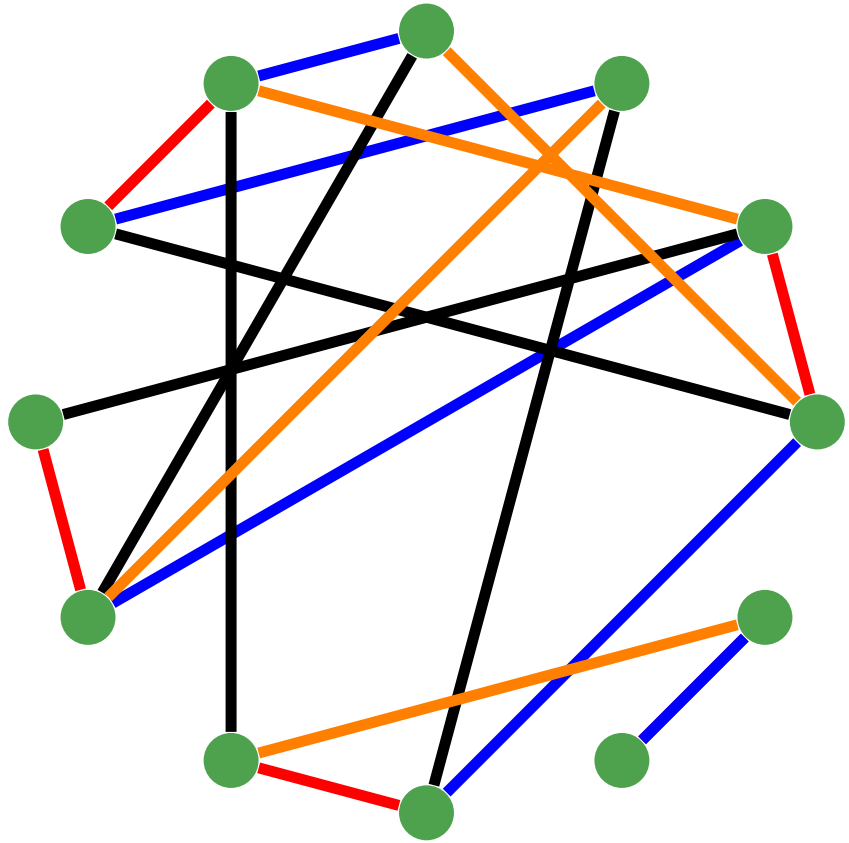
YES case: labels satisfy $w^t = M^t X$ for $1 \le t \le T$

$\bigcup_t G'_t$ is bipartite

NO case: labels are random: $w^t \sim UNIF$

$\bigcup_t G'_t$ is a sample of $\bigcup_t G_t$ at rate $1/2$

Complexity of Implicit Hidden Partition



Theorem (K.-Khanna-Sudan'14; Informal)

Any T-player protocol that obtains constant advantage over random guessing for the Implicit Hidden Partition problem requires $\sqrt{n}/poly(T)$ communication.

Add a zero-th player Alice, who holds the bipartition X

Add a zero-th player Alice, who holds the bipartition X

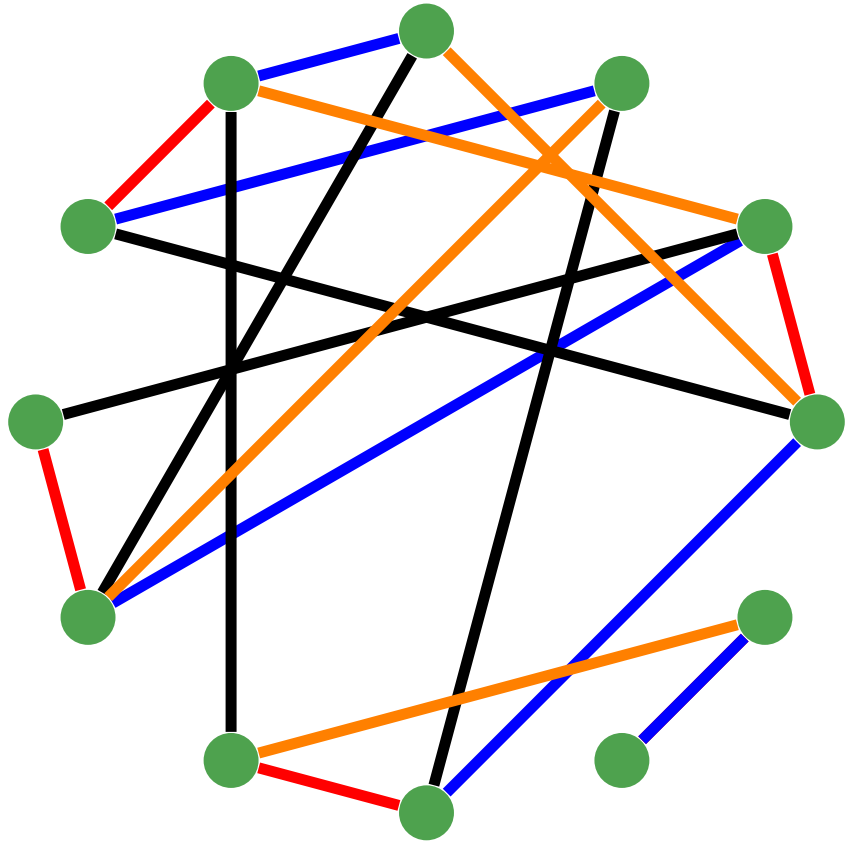
Roughly speaking, Alice should get advantage at least $1/T$ with at least one of Bobs.

Add a zero-th player Alice, who holds the bipartition X

Roughly speaking, Alice should get advantage at least $1/T$ with at least one of Bobs.

So $\sqrt{n}/\text{poly}(T)$ communication is needed

Complexity of Implicit Hidden Partition



Theorem (K.-Krachun'19; Informal)

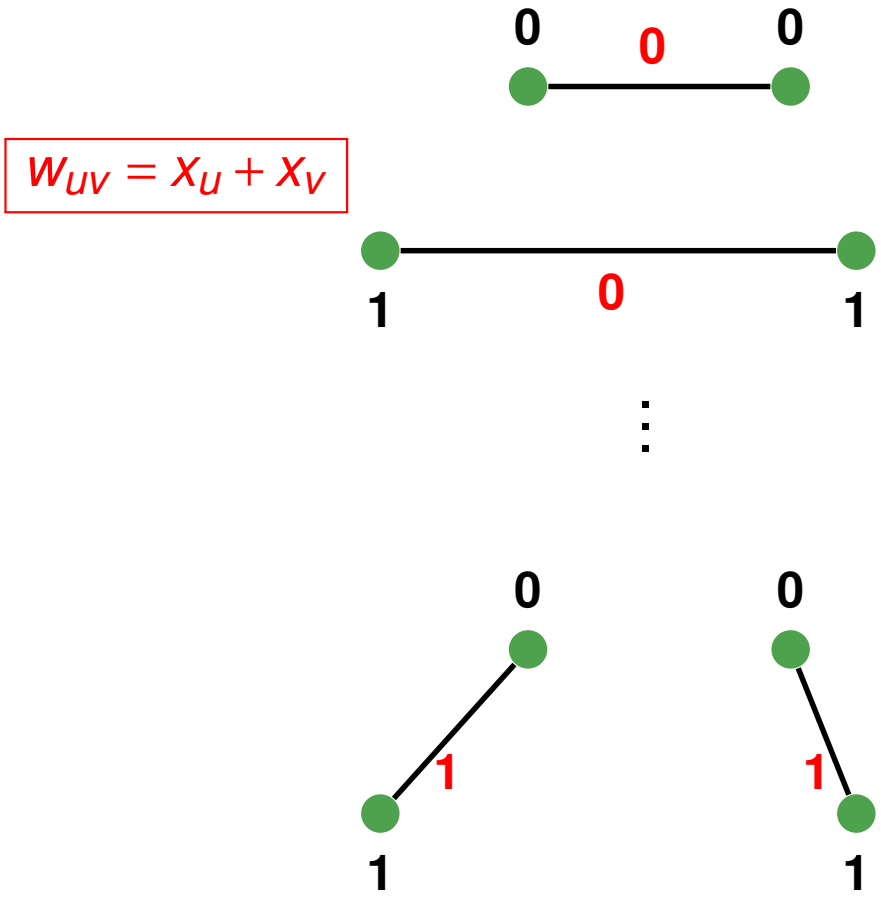
Any T-player protocol that obtains constant advantage over random guessing for the Implicit Hidden Partition problem requires $n/T^{O(T)}$ communication.

Distributional communication problem

Choose a hidden partition $X \in UNIF(\{0,1\}^n)$

Distributional communication problem

Choose a hidden partition $X \in UNIF(\{0,1\}^n)$



Player 1 $\longrightarrow$ m_1

matching M_1, labels

w^1 on edges

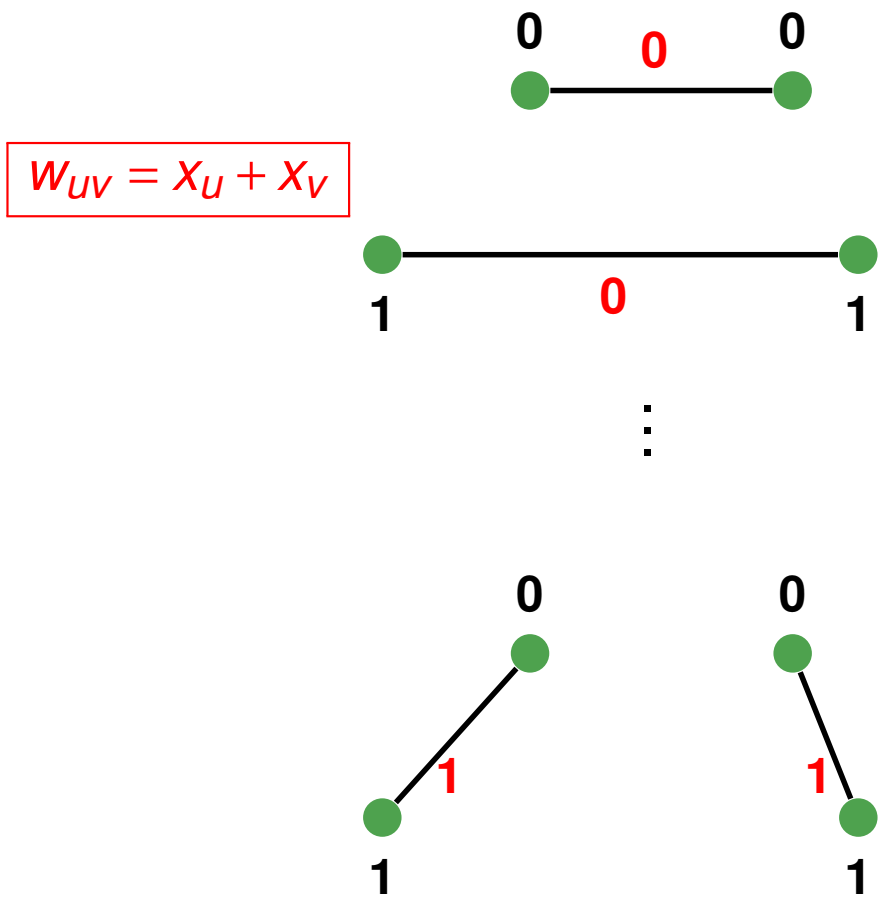
Player T $\longrightarrow$ m_T

matching M_T, labels

w^T on edges

Distributional communication problem

Choose a hidden partition $X \in UNIF(\{0,1\}^n)$



Player 1 $\longrightarrow$ m_1

matching M_1, labels

w^1 on edges

Player T $\longrightarrow$ m_T

matching M_T, labels

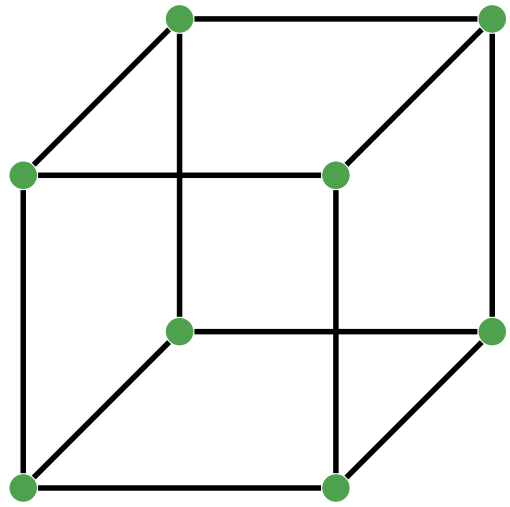
w^T on edges

YES case: labels satisfy $w^t = M^t X$ for $1 \le t \le T$

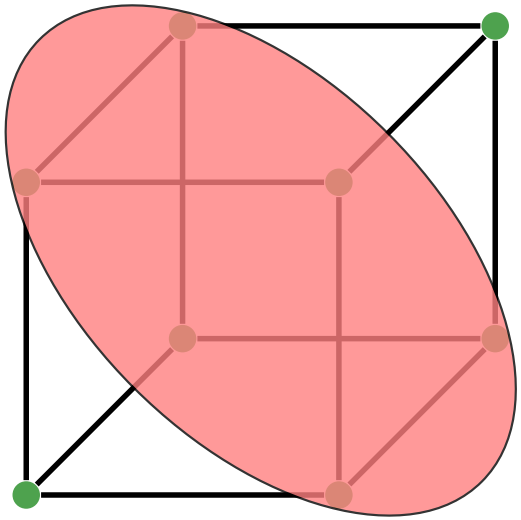
NO case: labels are random: $w^t \sim UNIF$

Conditioned on messages of player 1, player 2 and player 3, is distribution of $M_4 X$ close to uniform?

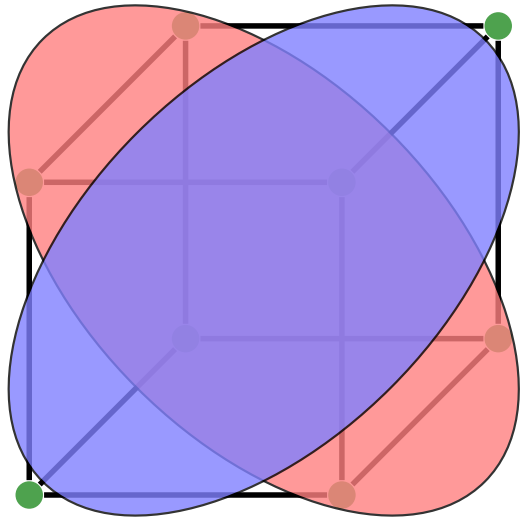
Conditioned on messages of player 1, player 2 and player 3, is distribution of $M_4 X$ close to uniform?



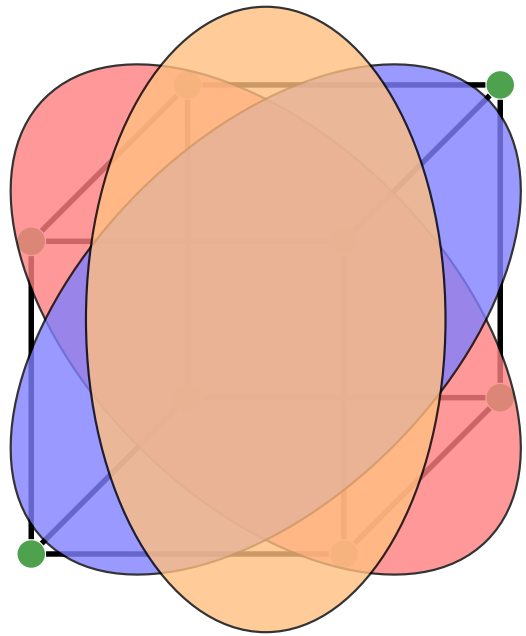
Conditioned on messages of player 1, player 2 and player 3, is distribution of $M_4 X$ close to uniform?



Conditioned on messages of player 1, player 2 and player 3, is distribution of $M_4 X$ close to uniform?

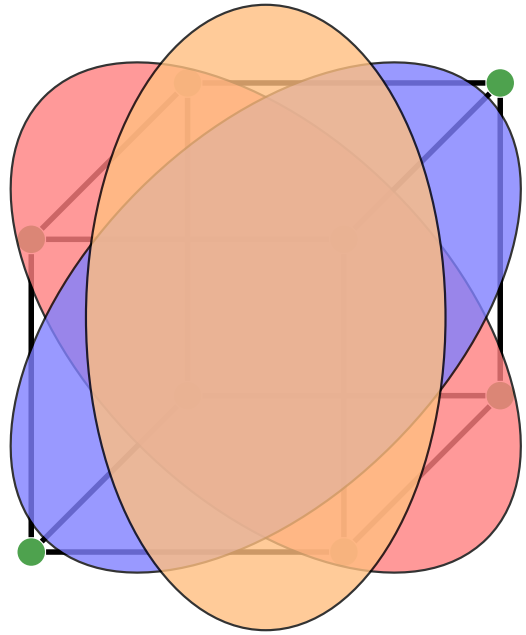


Conditioned on messages of player 1, player 2 and player 3, is distribution of M_4X close to uniform?



Conditioned on messages of player 1, player 2 and player 3, is distribution of M_4X close to uniform?

$X \sim UNIF(A_1 \cap A_2 \cap A_3)$
conditioned on (m_1, m_2, m_3)



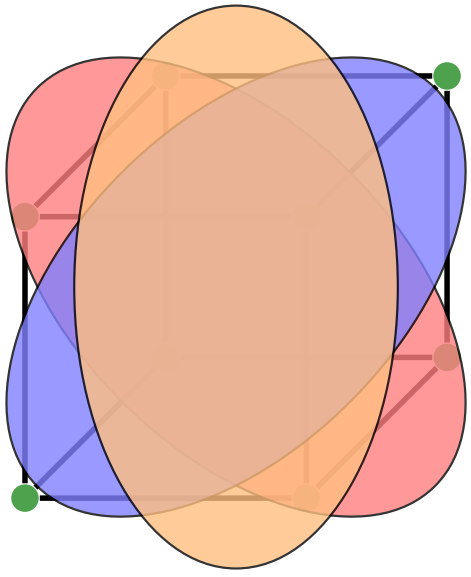
$|A_i|/2^n \geq 2^{-s}$

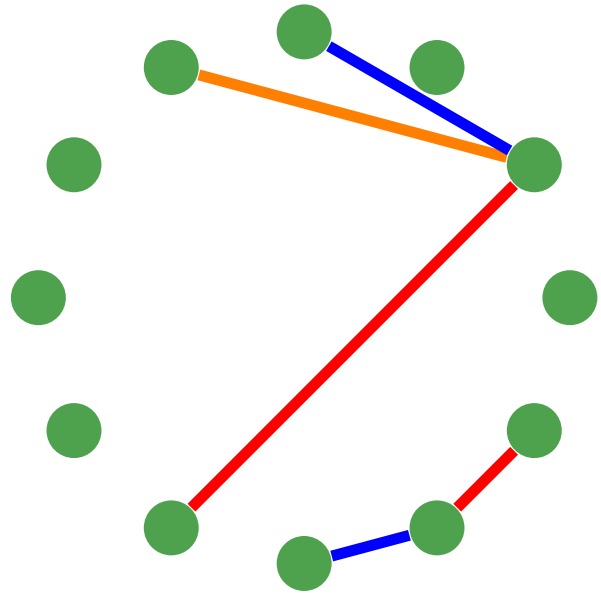
$f_1(x) :=$ indicator of A_1
$f_2(x) :=$ indicator of A_2
$f_3(x) :=$ indicator of A_3

The indicator of $A_1 \cap A_2 \cap A_3$ is $h := f_1 \cdot f_2 \cdot f_3$.

Bound distance of M_4X to uniform?

Bound $\sum_{\substack{v \in \{0,1\}^n \\ |v|=2k}} \widehat{h}(v)^2$ for $k = 1, \ldots, n/2$?

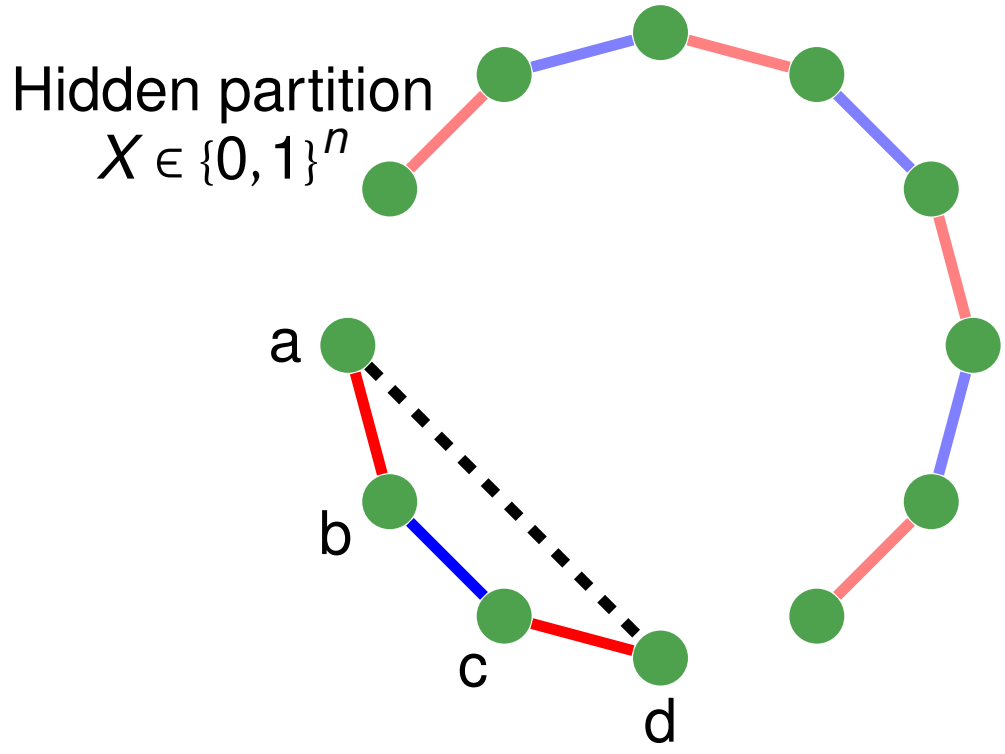




Proving stronger bounds on $\widetilde{f}_i$ is easy (since $\widetilde{f}_i$ are supported on matchings), use convolution theorem?

Convolution theorem:

$$\widetilde{h}_t = \widetilde{f_1 \cdot \ldots \cdot f_t} \approx \widetilde{f}_1 * \ldots * \widetilde{f}_t$$



Intuition: $\widetilde{f}_1(a,b,c,d)^2 \approx$ how much information player 1 transmits about parity $X_a + X_b + X_c + X_d$

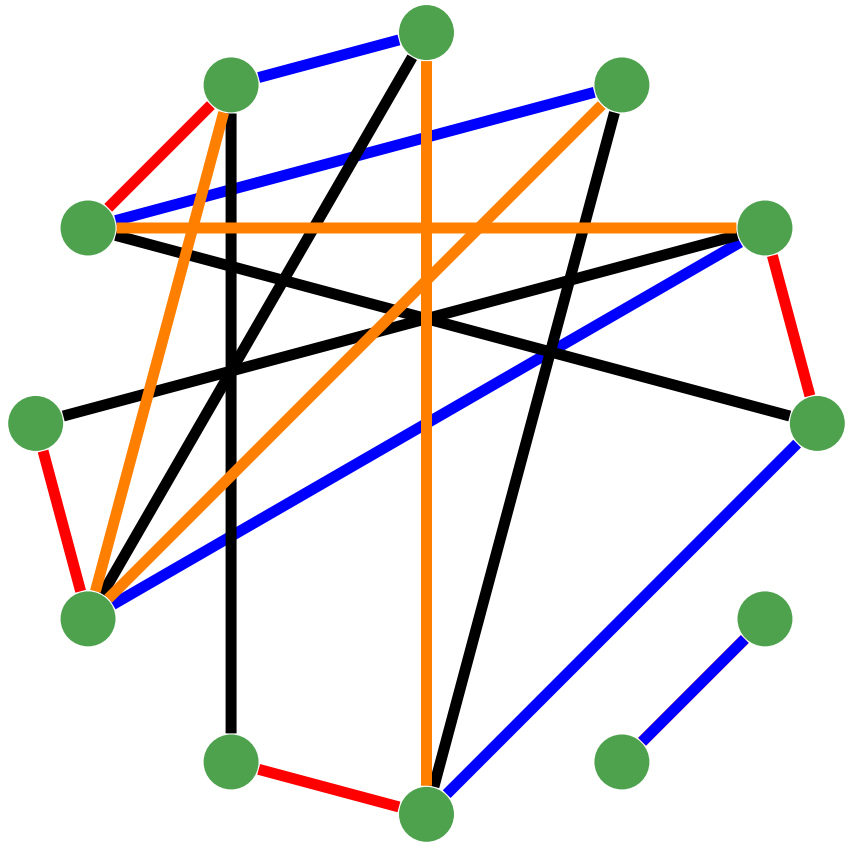
$\widetilde{f}_2(b,c)^2 \approx$ how much information player 2 transmits about parity $X_b + X_c$

$\widetilde{f_1 \cdot f_2}(a,d)^2 = \widetilde{f}_1(a,b,c,d)^2 \cdot \widetilde{f}_2(b,c)^2 \approx$ how much information players 1 and 2 transmit about parity $X_a + X_d$

Main technical contribution:

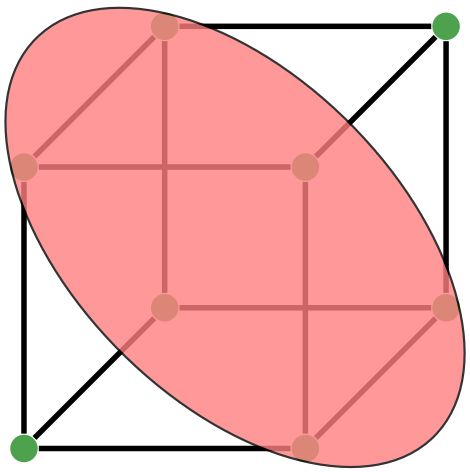
Analysis of $\widetilde{f}_1 * \widetilde{f}_2 * \cdots * \widetilde{f}_T$ for arbitrarily large T

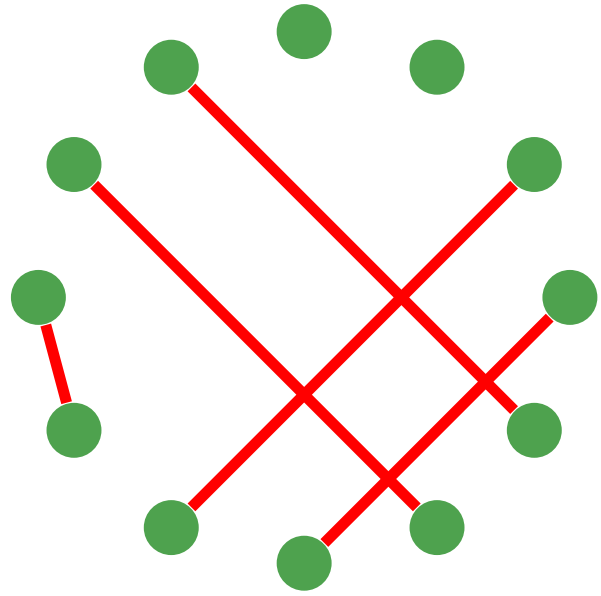
Main challenge: giant component in players' input:



ℓ_2 bounds are not sufficient for inductive hypothesis

Our approach: bound ℓ_1 norm of the Fourier transform!





ℓ_1 upper bounds on spectrum seamlessly translate to quantum setting: Kallaugher-Parekh'22

Subgraph counting through sketching

Theorem

In a bounded degree graph, can approximate triangle count in $O^*(m/T^{2/3})$ *space in the sketching model*

Subgraph counting through sketching

Theorem

In a bounded degree graph, can approximate triangle count in $O^*(m/T^{2/3})$ *space in the sketching model*

Theorem

The sketching complexity of triangle counting is $\Omega(m/T^{2/3})$

Subgraph counting through sketching

Theorem

In a bounded degree graph, can approximate triangle count in $O^*(n^{1/3})$ space in the sketching model

Set $T = \Theta(n)$, i.e. roughly maximal number of triangles.

Theorem

The sketching complexity of triangle counting is $\Omega(n^{1/3})$

Set $T = \Theta(n)$, i.e. roughly maximal number of triangles.

Subgraph counting through sketching

Theorem

In a bounded degree graph, can approximate triangle count in $O^*(n^{1/3})$ space in the sketching model

Set $T = \Theta(n)$, i.e. roughly maximal number of triangles.

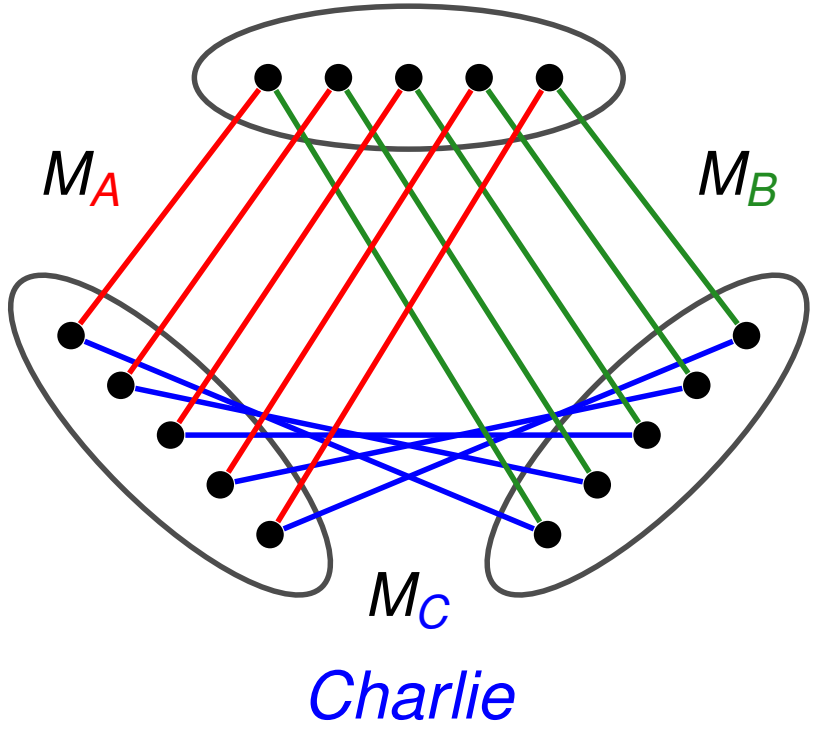
Theorem

The sketching complexity of triangle counting is $\Omega(n^{1/3})$

Set $T = \Theta(n)$, i.e. roughly maximal number of triangles.

Generalizes beyond triangles, and to hypergraphs

Communication game



M_A, M_B, M_C random

subject to $M_A M_B M_C = I$

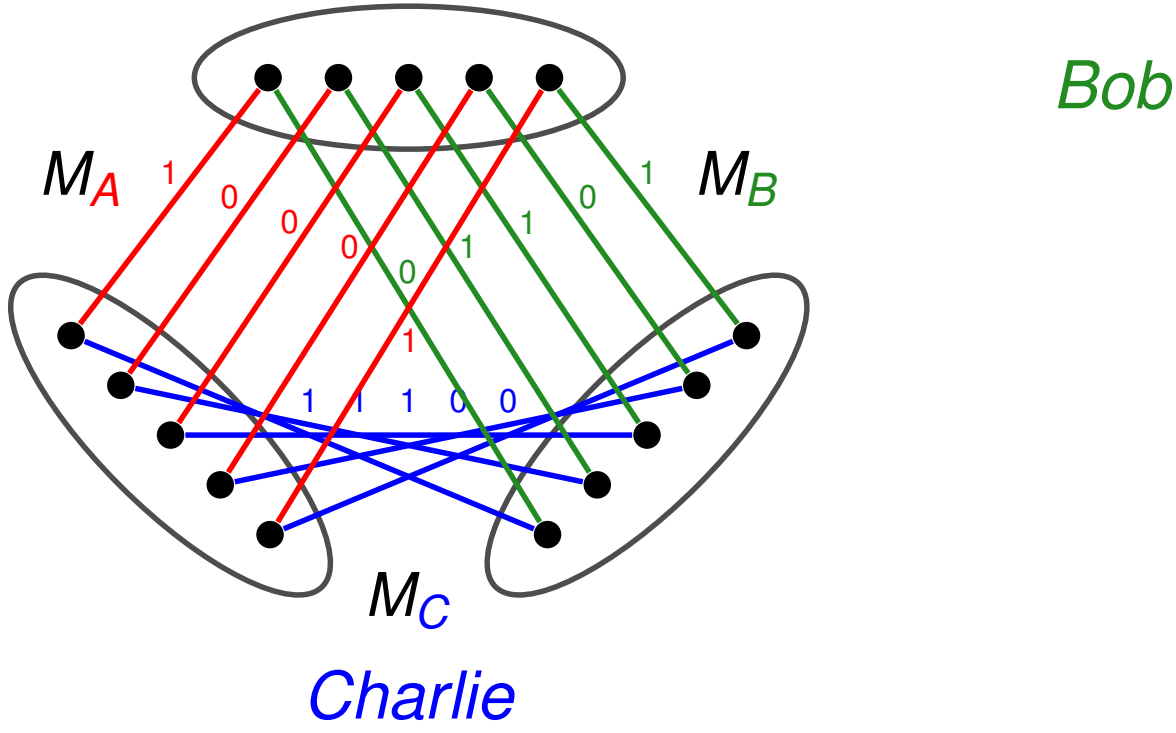
Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one

Communication game



M_A, M_B, M_C random

subject to $M_A M_B M_C = I$

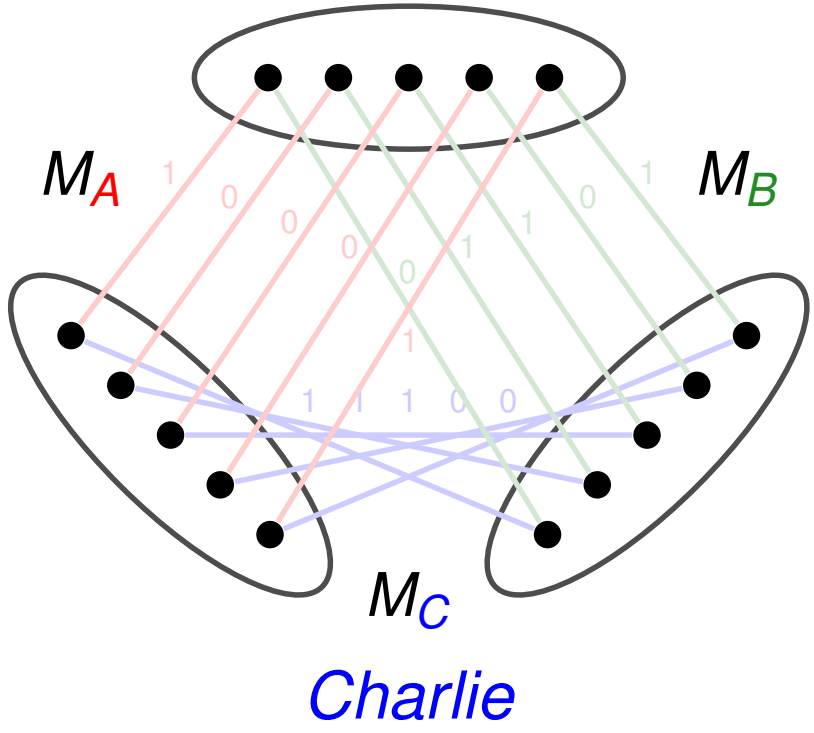
Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one

Communication game



M_A, M_B, M_C random

subject to $M_A M_B M_C = I$

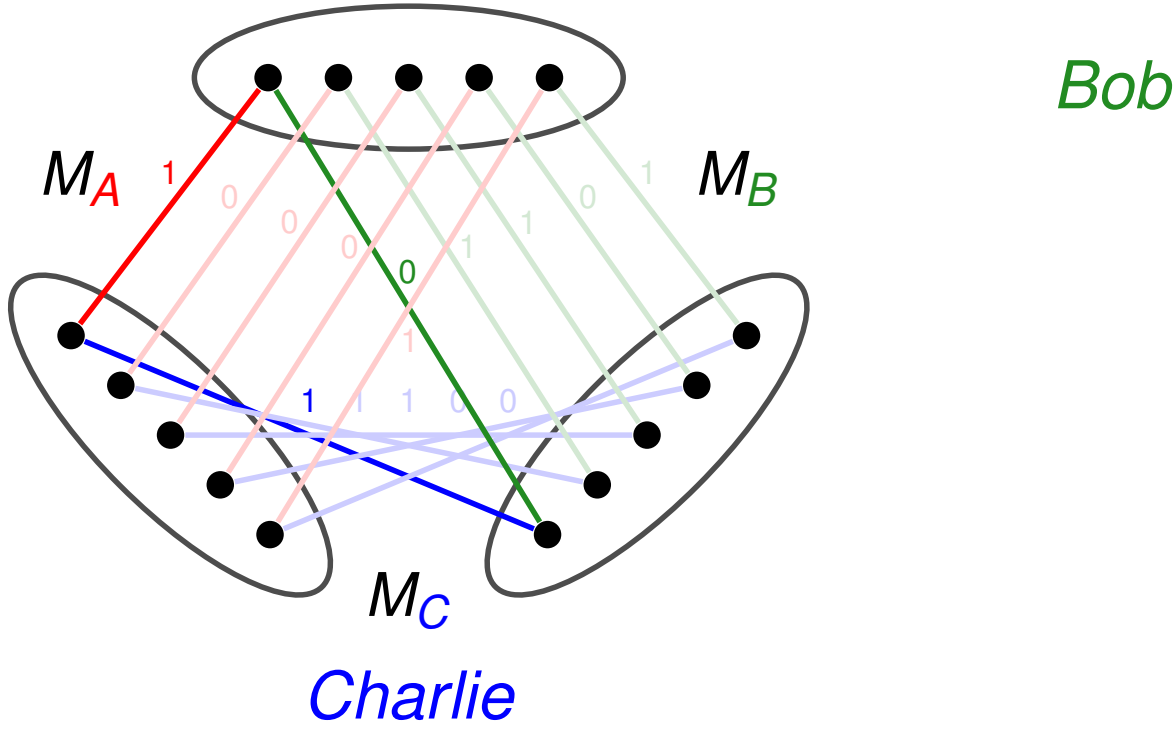
Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one

Communication game



M_A, M_B, M_C random

subject to $M_A M_B M_C = I$

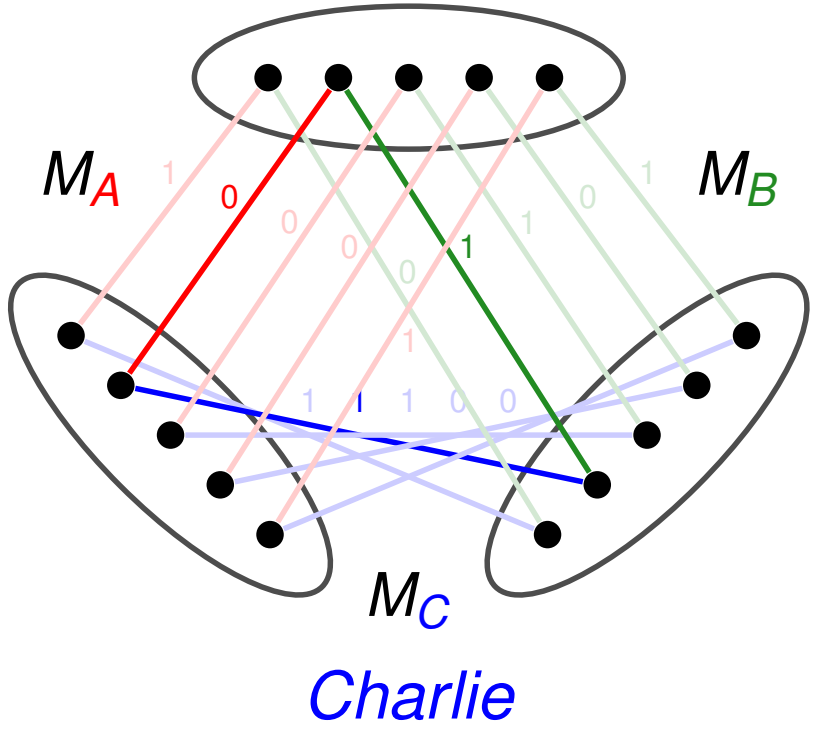
Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one

Communication game



M_A, M_B, M_C random

subject to $M_A M_B M_C = I$

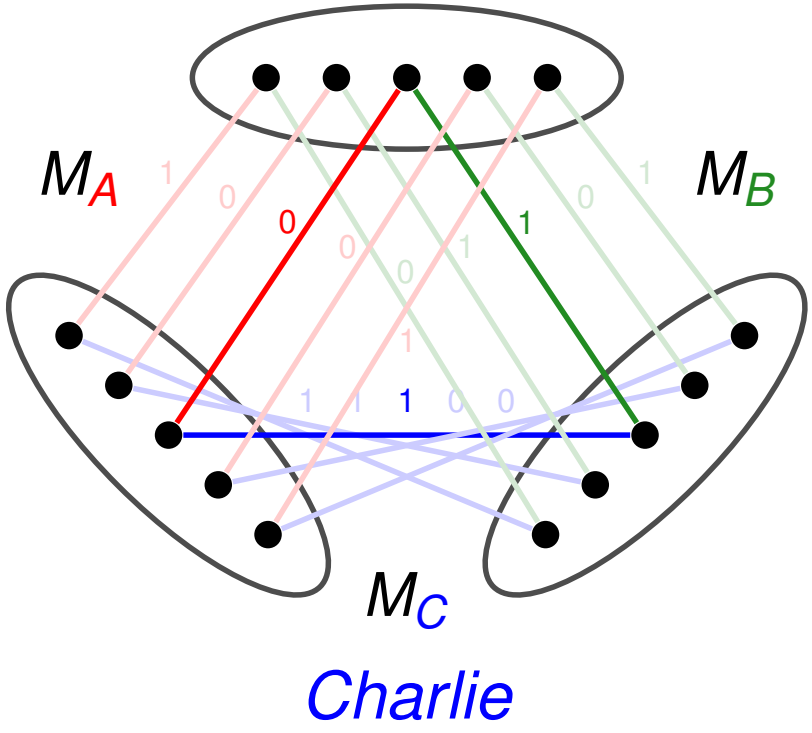
Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one

Communication game



M_A, M_B, M_C random

subject to $M_A M_B M_C = I$

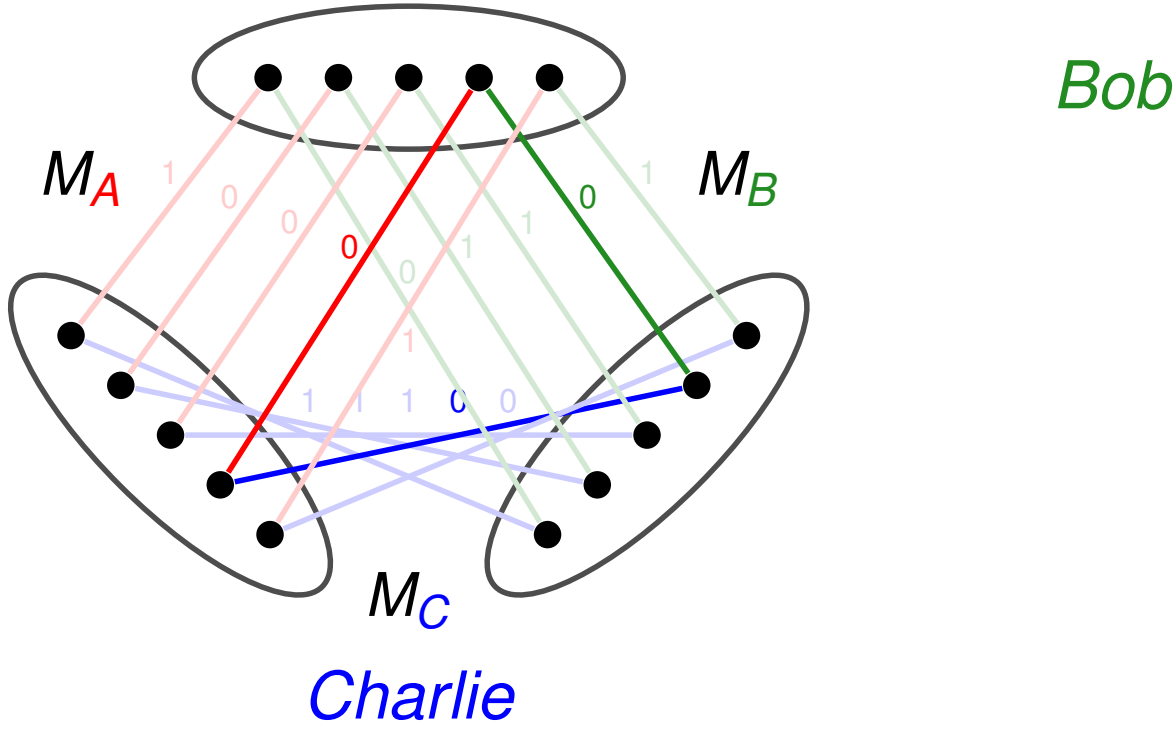
Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one

Communication game



M_A, M_B, M_C random

subject to $M_A M_B M_C = I$

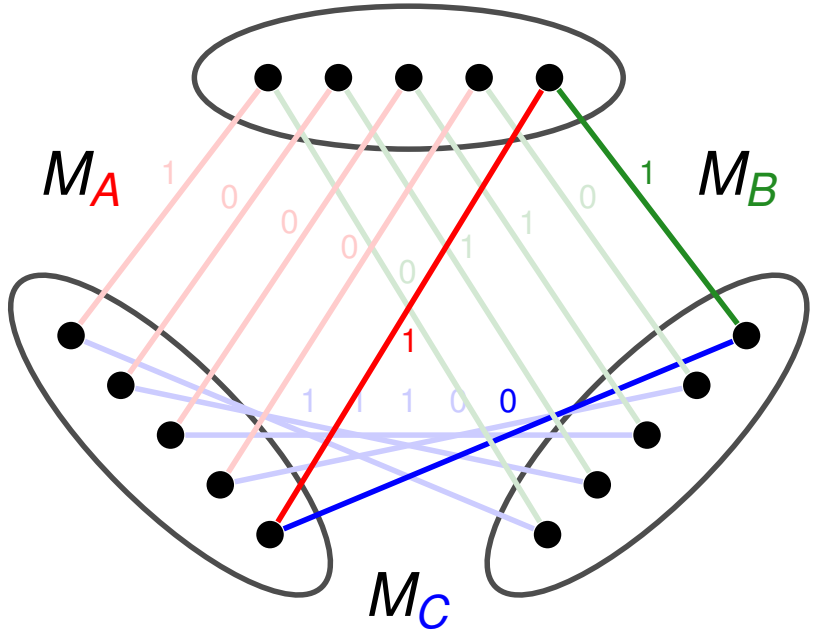
Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one

Communication game



M_A, M_B, M_C random

subject to $M_A M_B M_C = I$

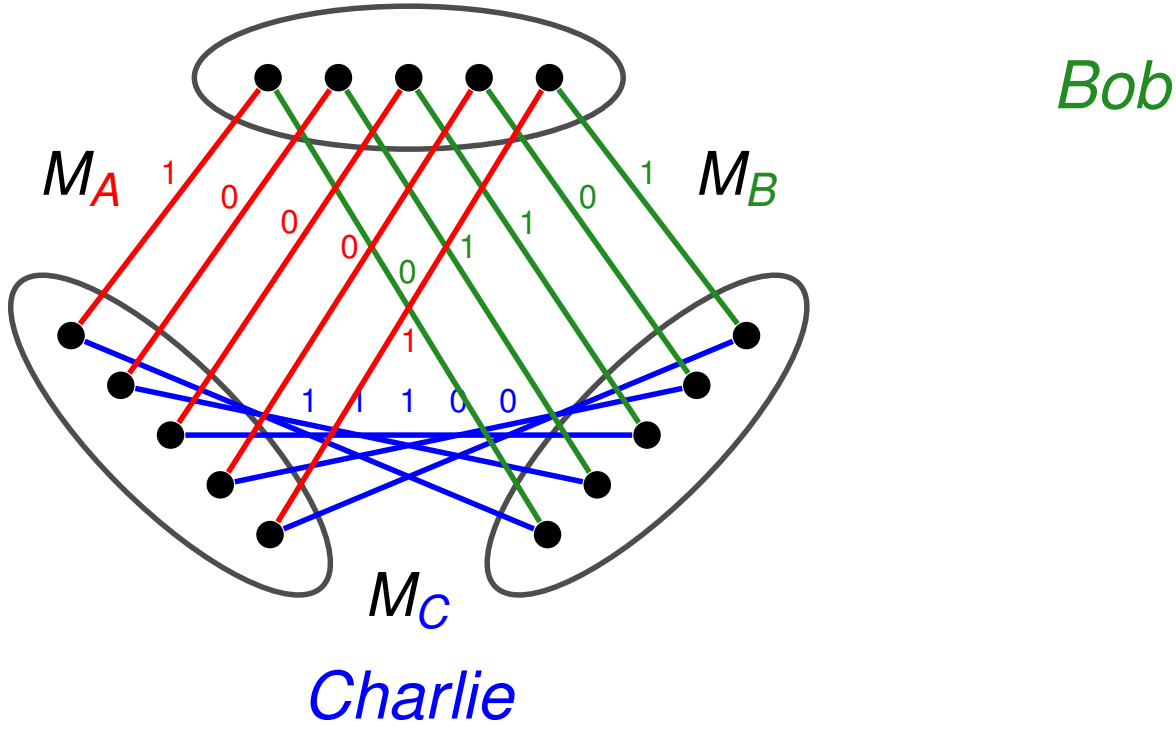
Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one

Communication game



M_A, M_B, M_C random

subject to $M_A M_B M_C = I$ *Referee*

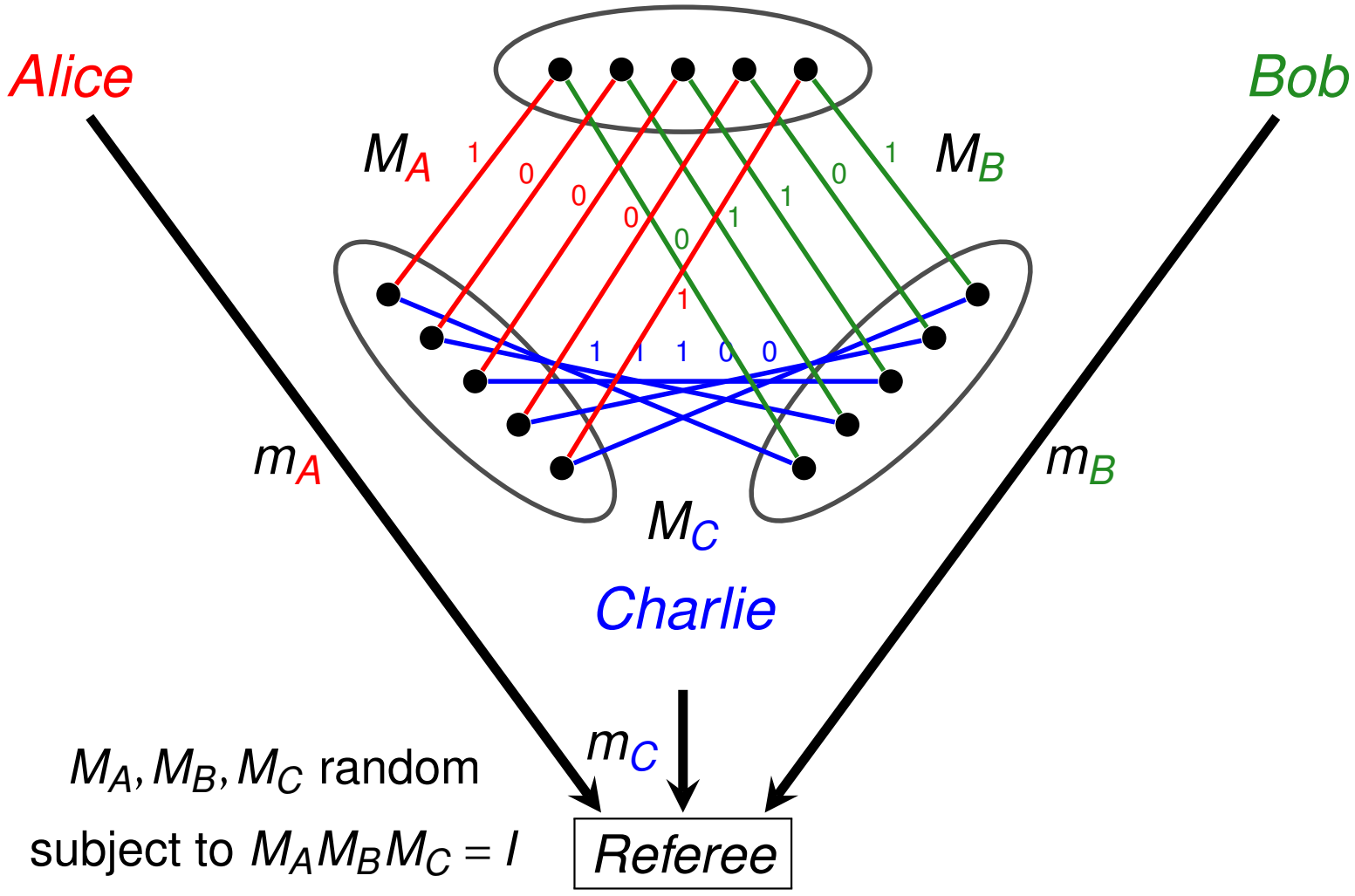
Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one

Communication game



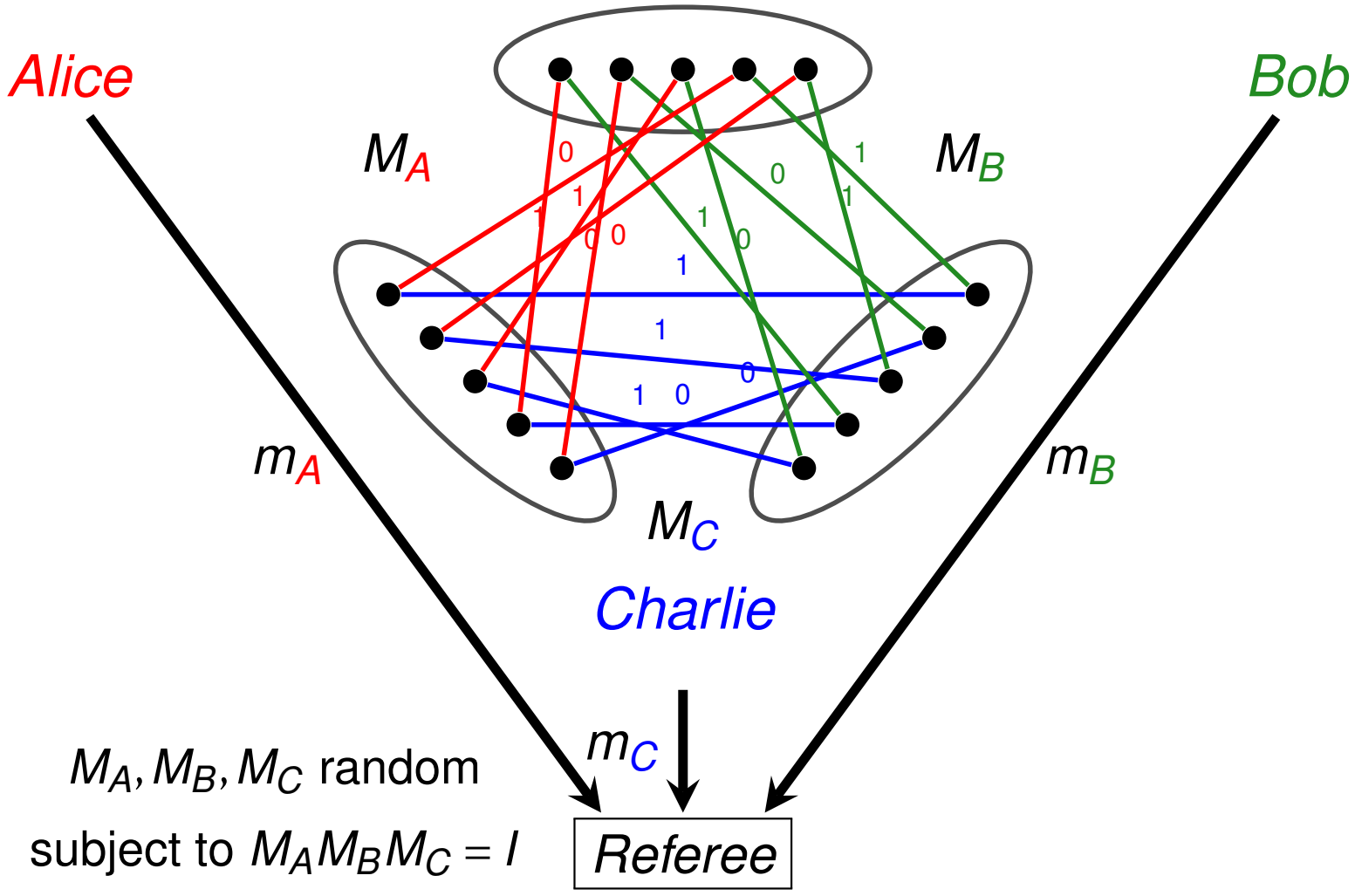
Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one

Communication game

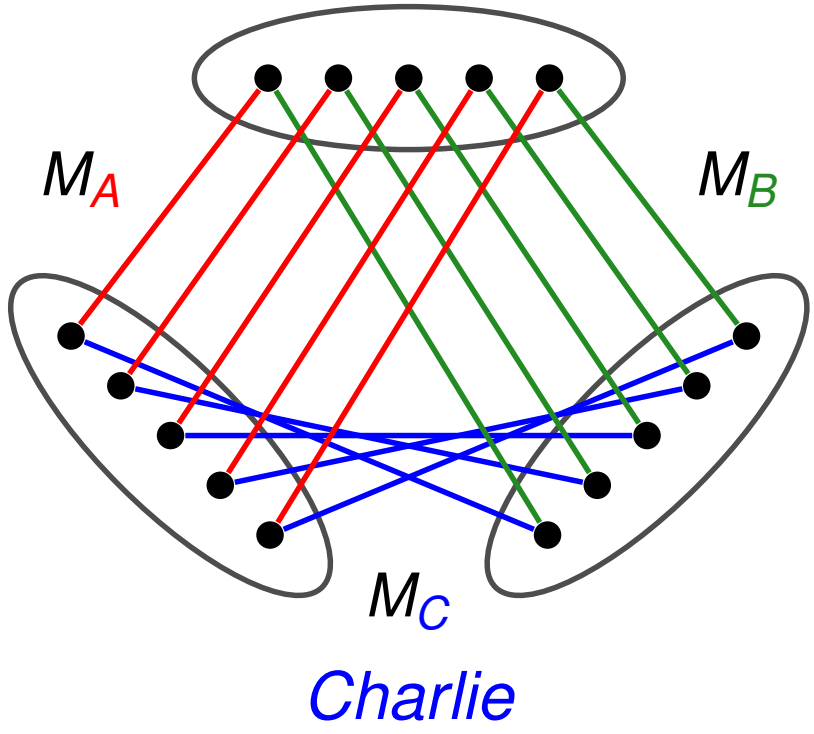


Bit strings $x^A, x^B, x^C \in \{0,1\}^n$
uniformly random subject to:

YES case: $x^A + x^B + x^C = 0^n$

NO case: $x^A + x^B + x^C = 1^n$

Sum to zero across all triangles or sum to one



M_A, M_B, M_C random

subject to $M_A M_B M_C = I$

Theorem

$\Omega(n^{1/3})$ communication is required to get $\Omega(1)$ advantage over random guessing

1. The Boolean Hidden Matching problem

2. Multiplayer games via the convolution theorem

3. A game with a ‘infinite’ number of players

1. The Boolean Hidden Matching problem

2. Multiplayer games via the convolution theorem

3. A game with an ‘infinite’ number of players

Count # of connected components in a graph

(Peng-Sohler'18: approx # of connected components to εn error in $\approx (1/\varepsilon)^{1/\varepsilon^3}$ space)

Random walk generation, PAGERANK estimation

(Kallaugher-K.-Price'22: generate s walks of length ℓ using space $\approx 2^{O(\ell^2)} s$)

Factorial Lower Bounds for (Almost) Random Order Streams

(FOCS 2022; joint work with Ashish Chiplunkar, John Kallaugher and Eric Price)

~~Count # of connected components in a graph~~

(Peng-Sohler'18: approx # of connected components to εn error in $\approx (1/\varepsilon)^{1/\varepsilon^3}$ space)

Random walk generation, PAGERANK estimation

(Kallaugher-K.-Price'22: generate s walks of length k using space $\approx 2^{O(k^2)}s$)

Factorial Lower Bounds for (Almost) Random Order Streams

(FOCS 2022; joint work with Ashish Chiplunkar, John Kallaugher and Eric Price)

Find a component of size $\leq \ell$ (assuming there are many)

(Peng-Sohler'18: component finding in $\approx \ell^{\ell^3}$ space)

Random walk generation, PAGERANK estimation

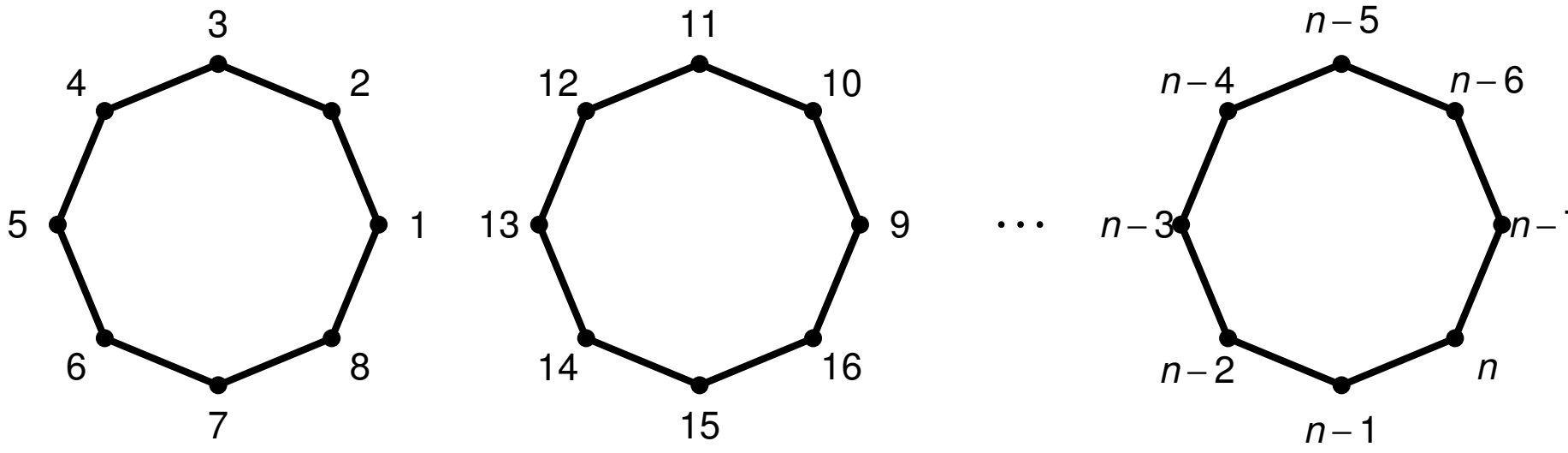
(Kallaugher-K.-Price'22: generate s walks of length ℓ using space $\approx 2^{O(\ell^2)}s$)

Factorial Lower Bounds for (Almost) Random Order Streams

(FOCS 2022; joint work with Ashish Chiplunkar, John Kallaugher and Eric Price)

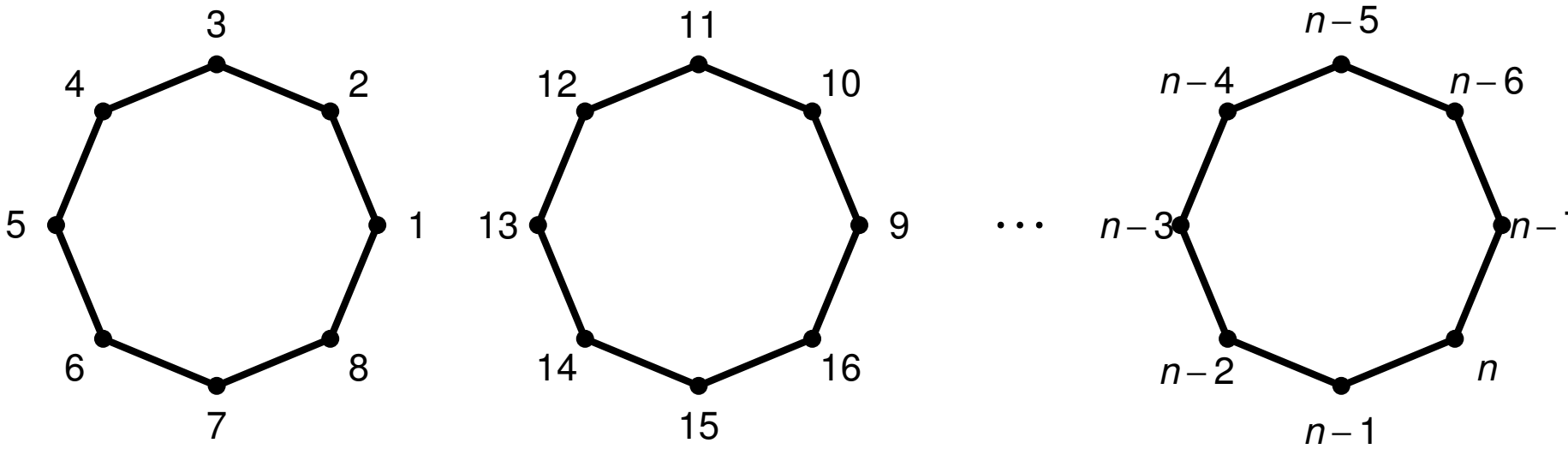
The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order



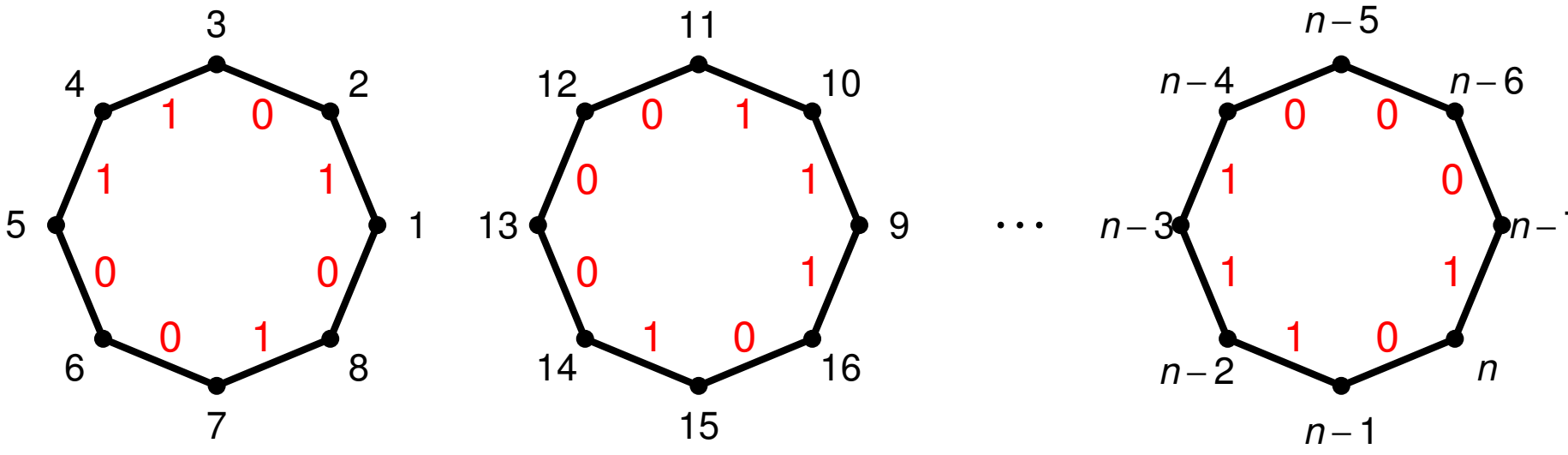
The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits



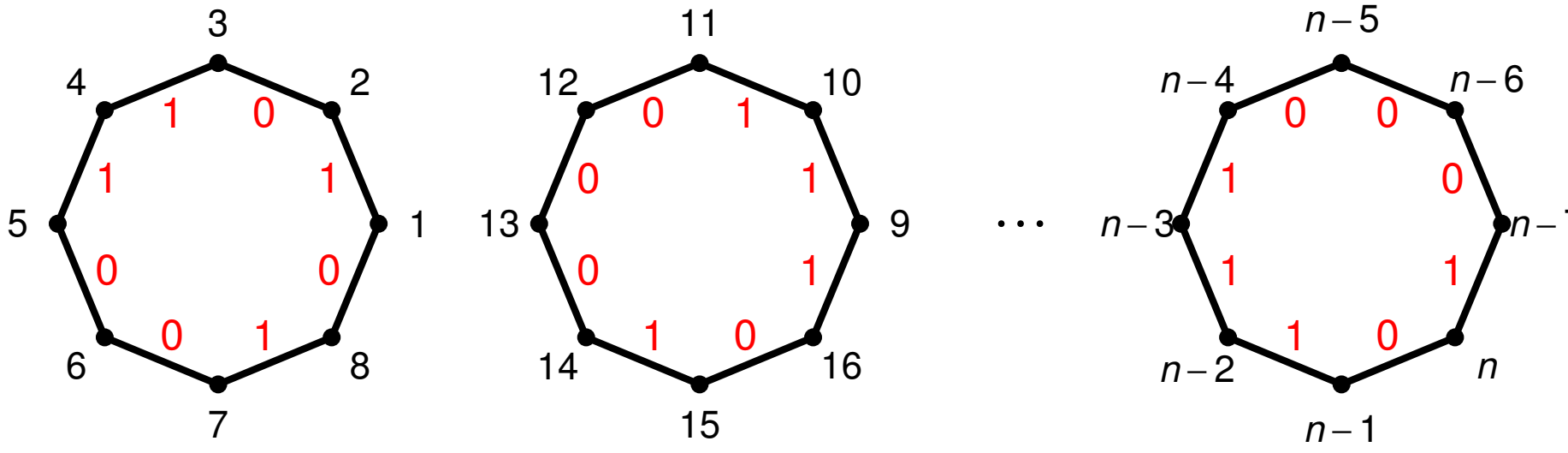
The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits



The StreamingCycles problem

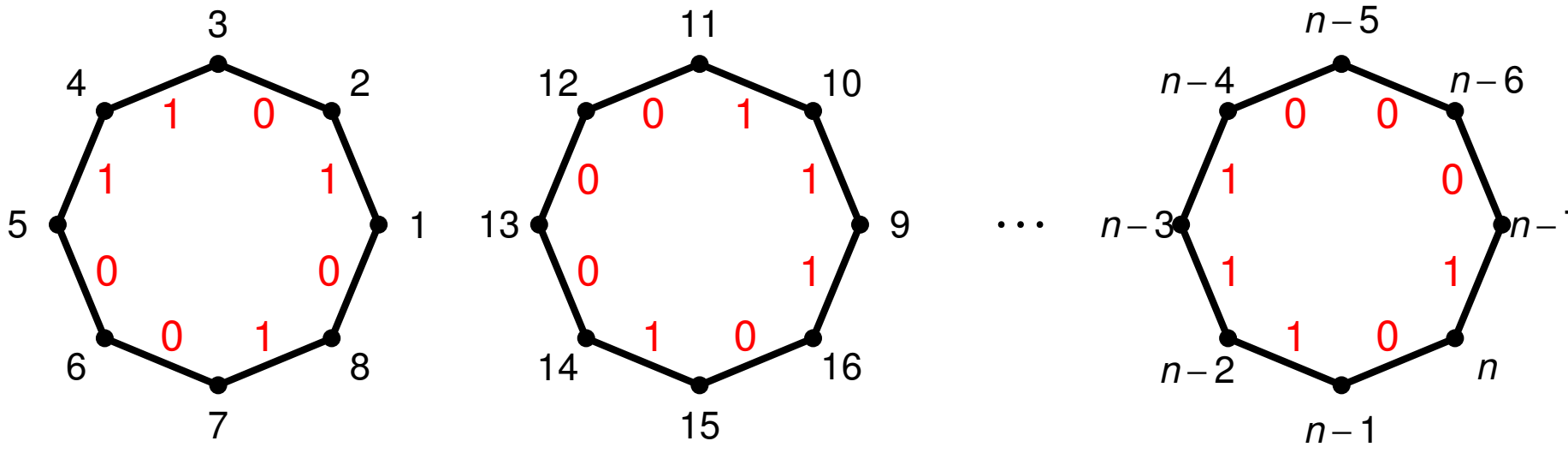
Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits



The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

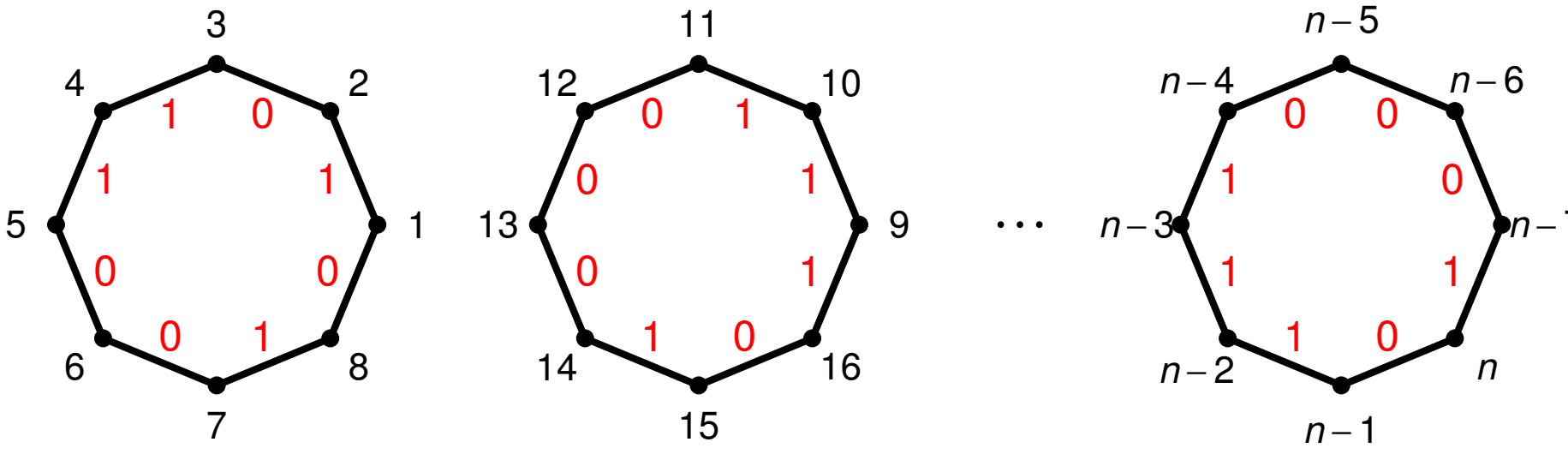
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

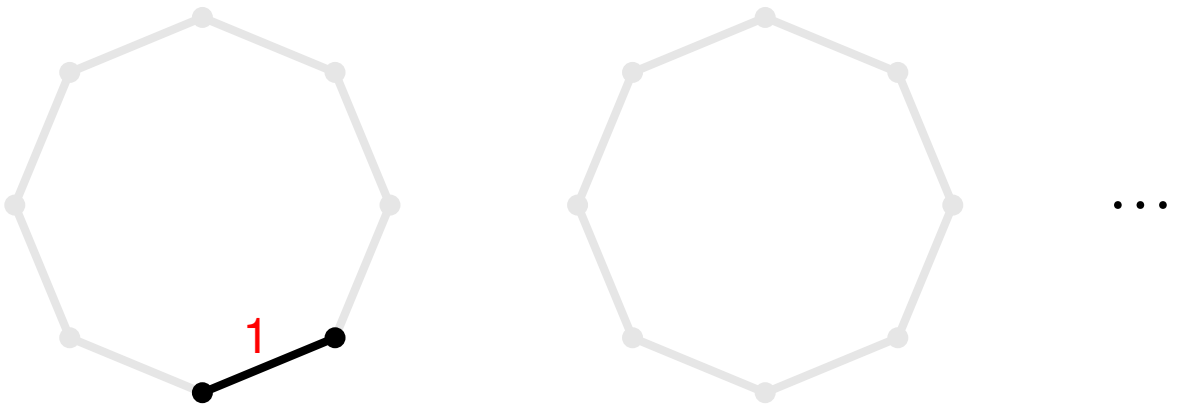
...

Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

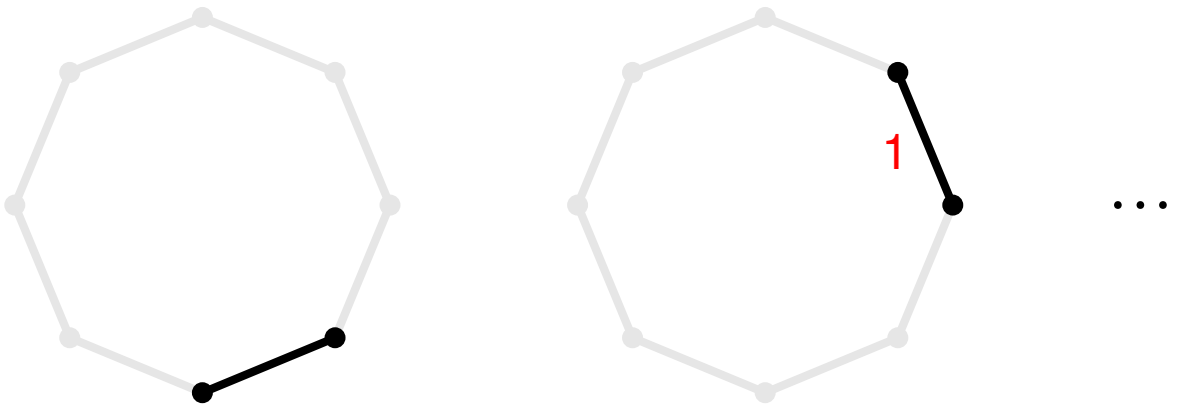


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

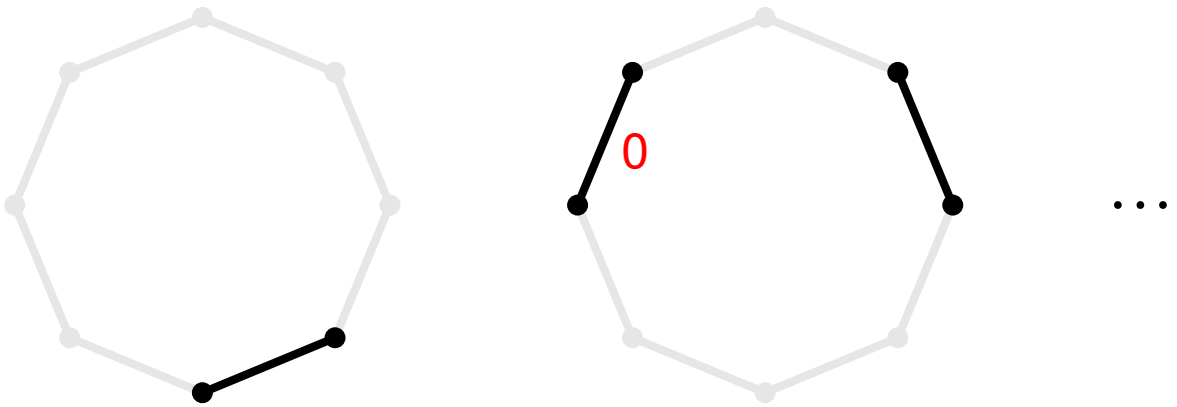


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

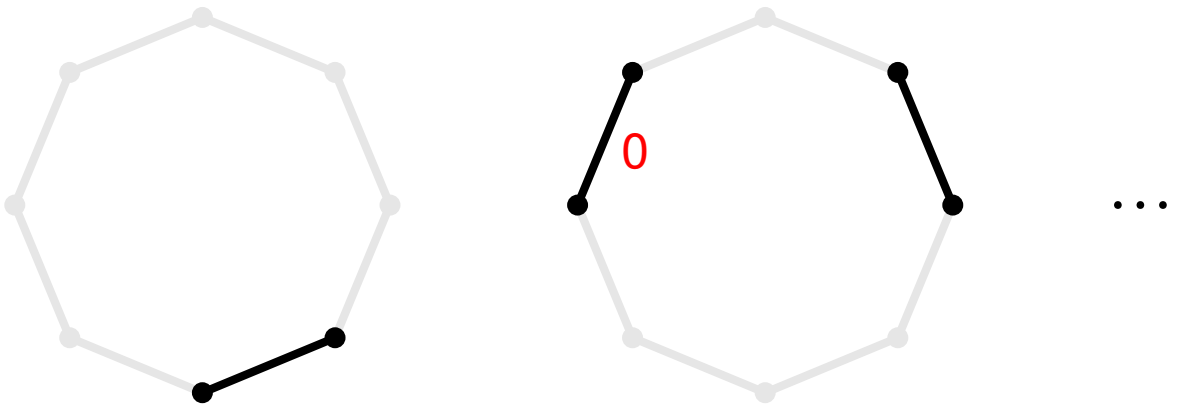


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

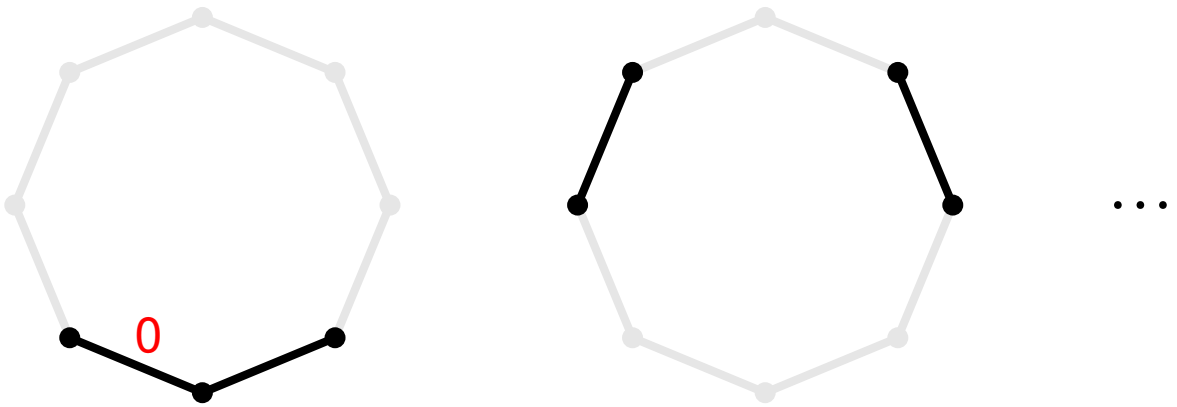


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

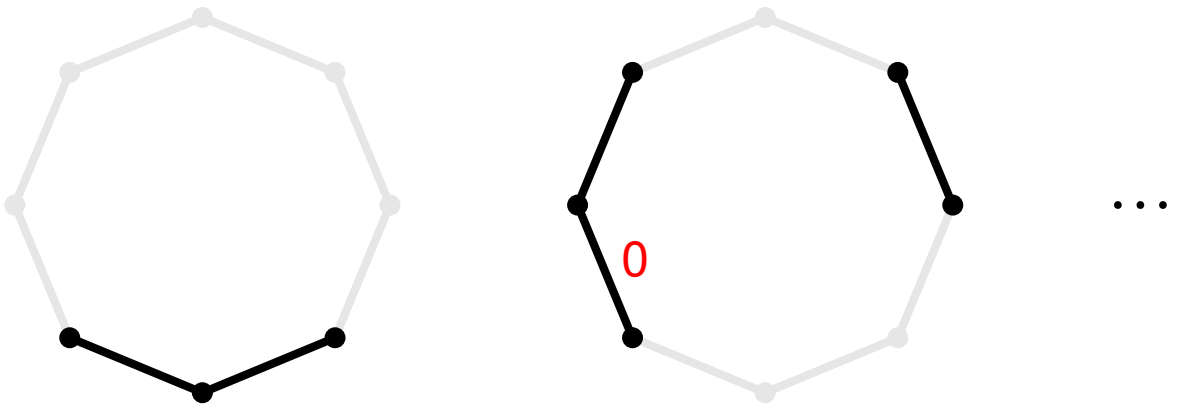


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

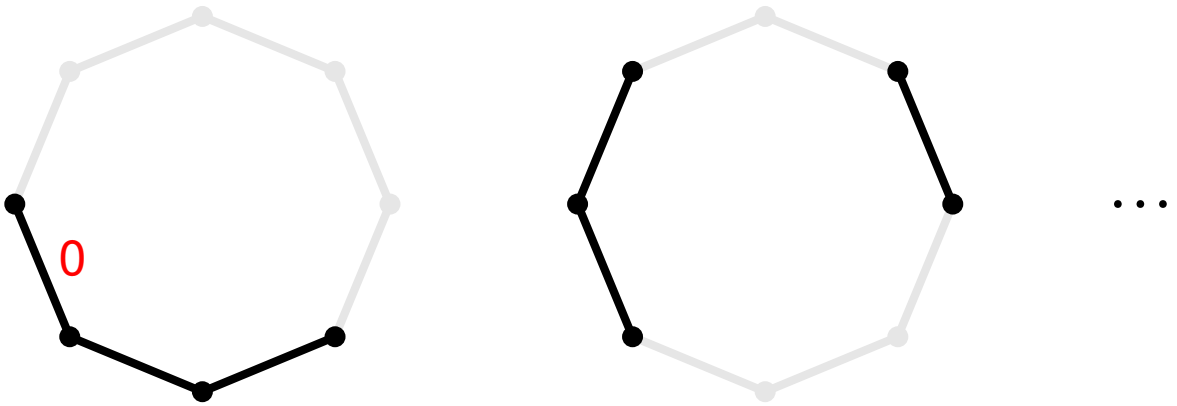


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

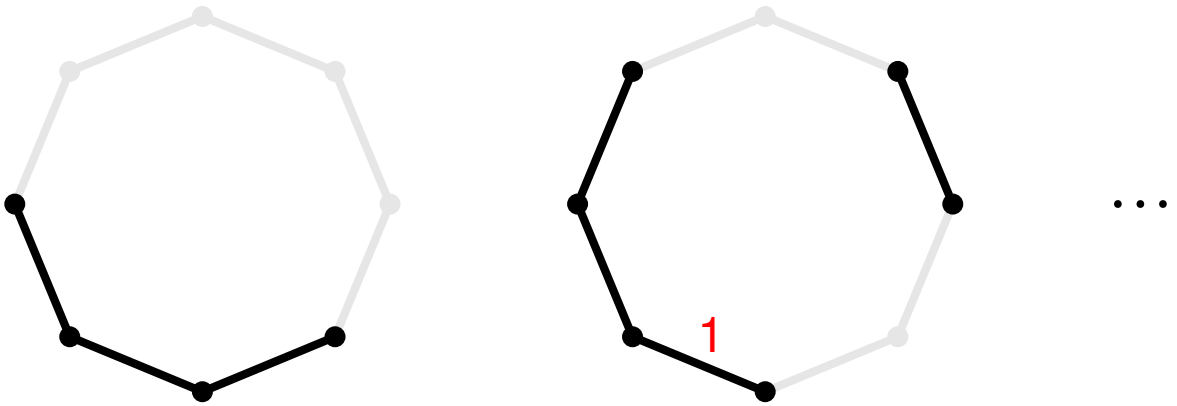


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

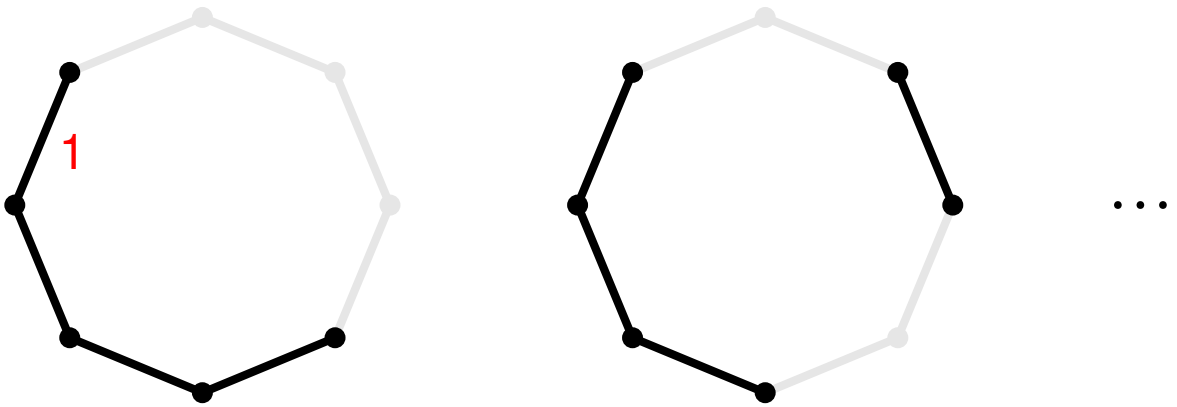


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

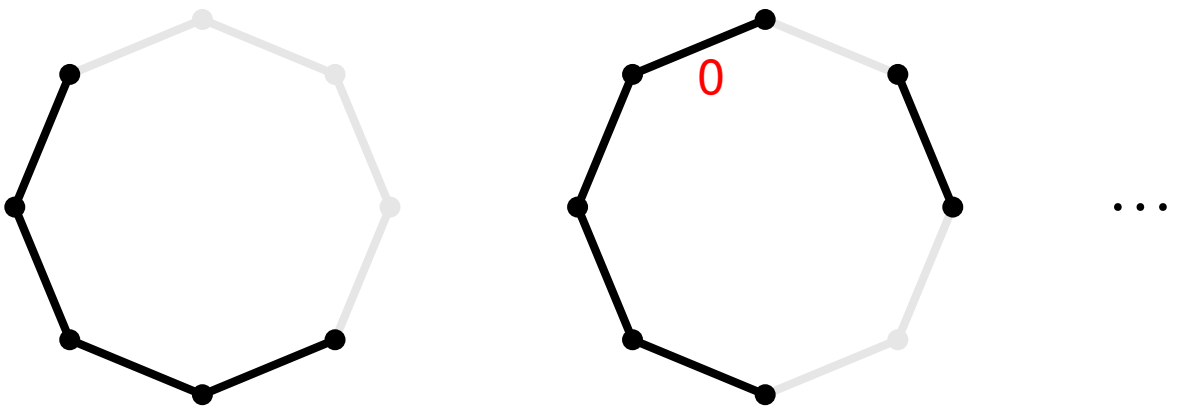


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

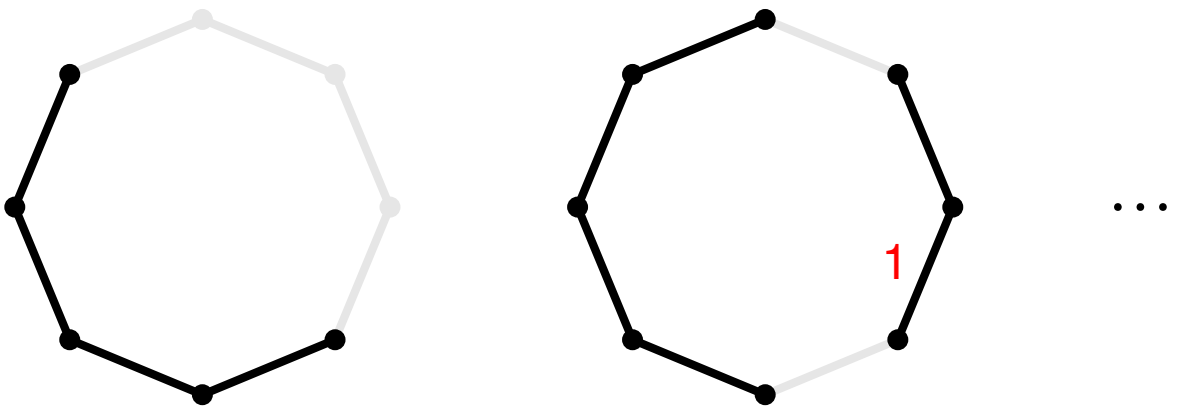


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

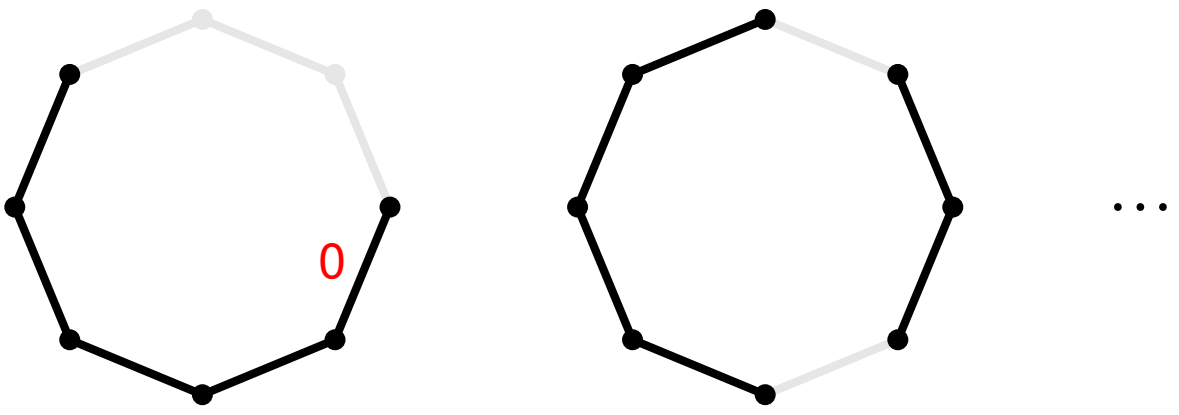


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

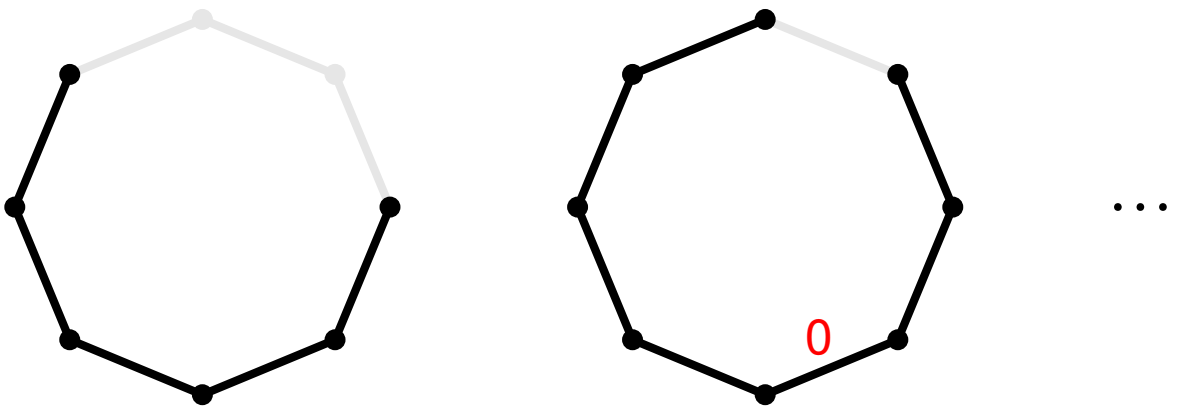


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

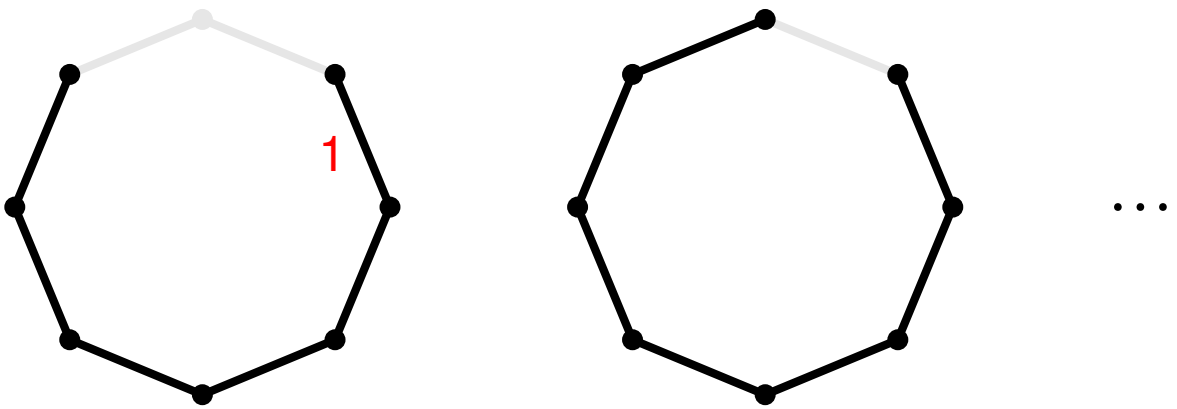


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

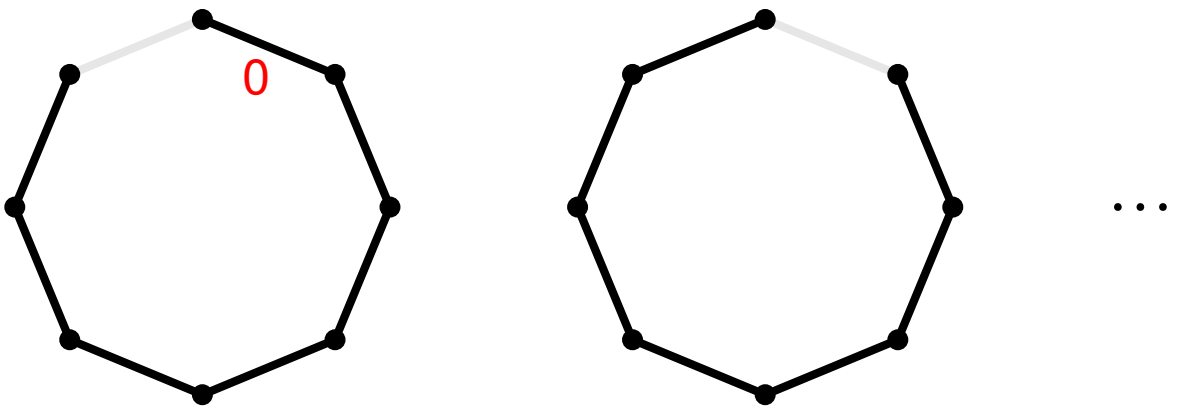


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

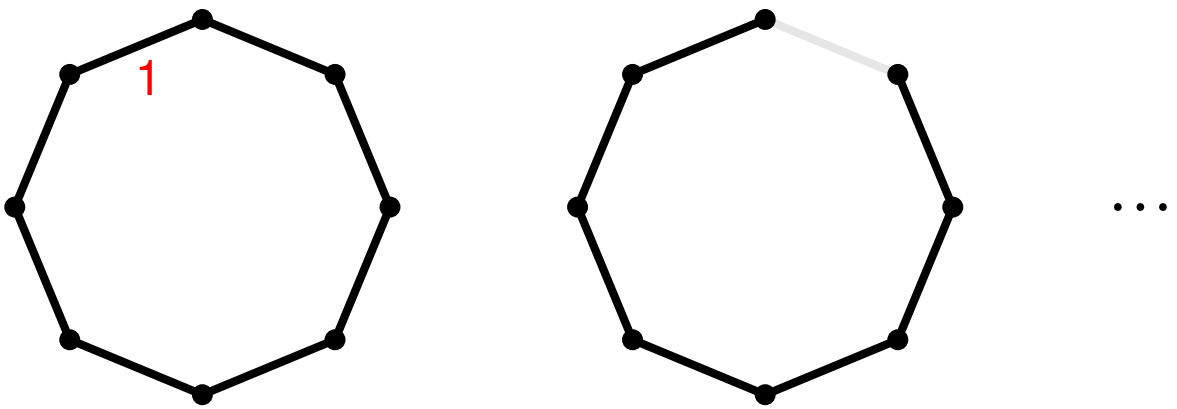


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

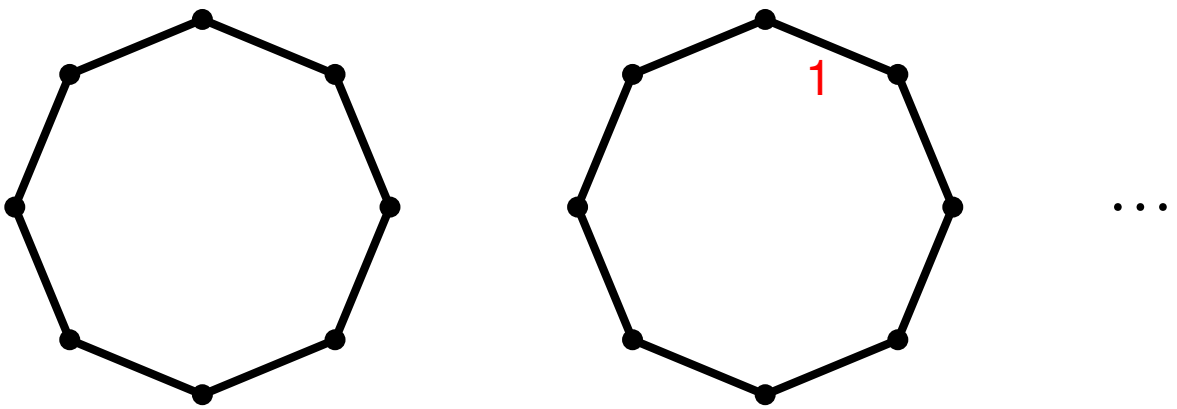


Simplification: edges arrived so far posted on a board

The StreamingCycles problem

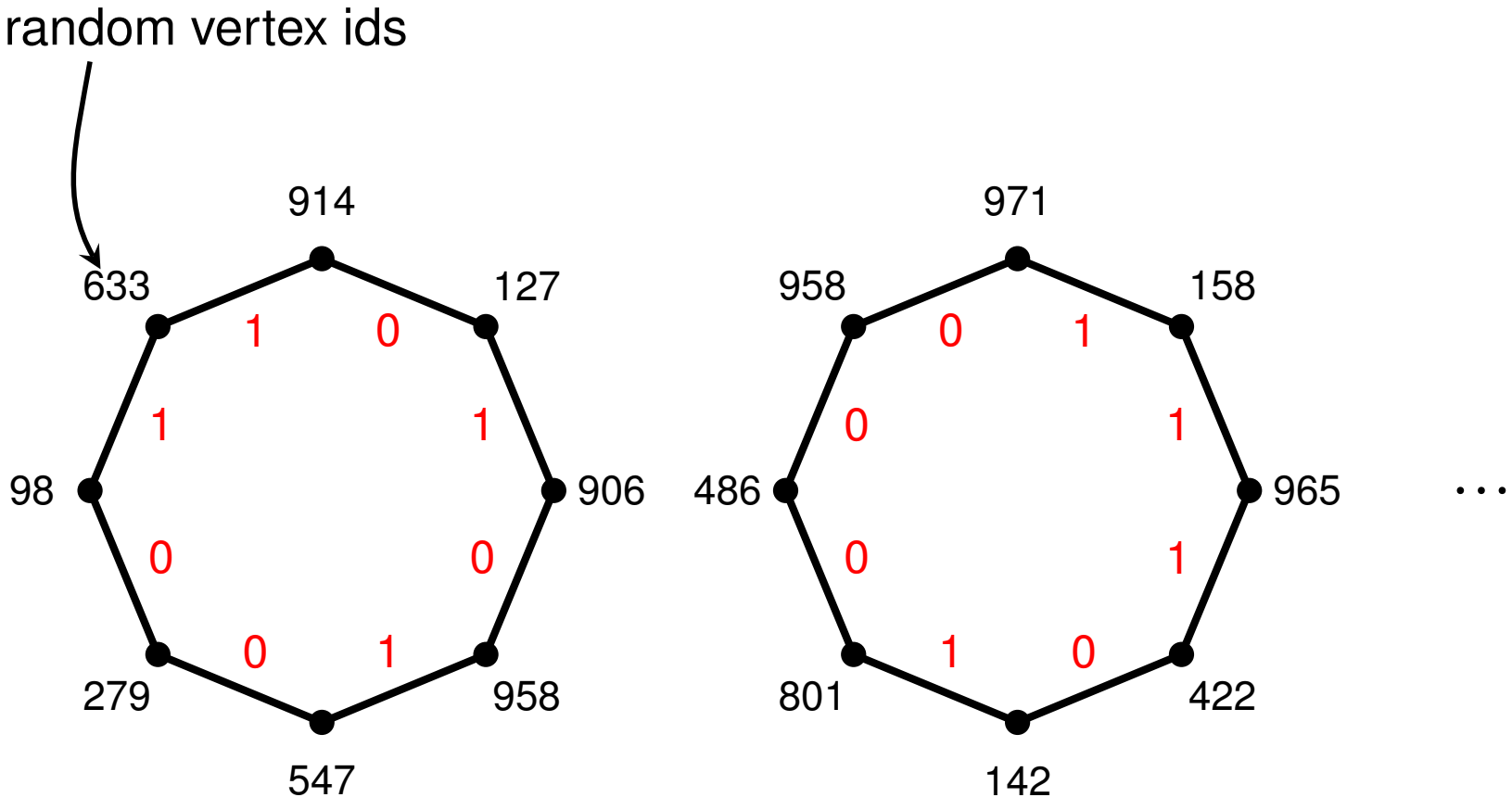
Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

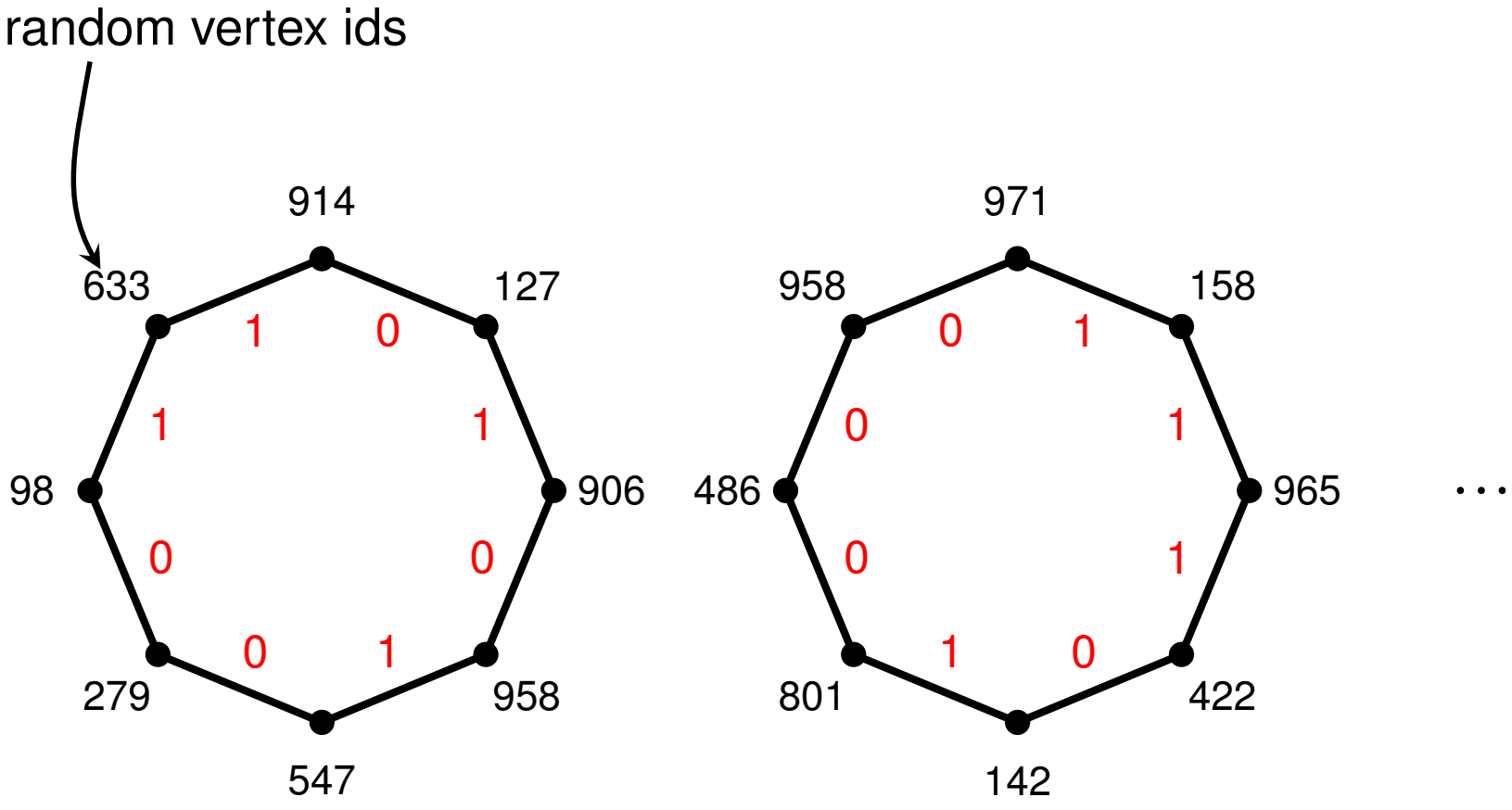


Simplification: edges arrived so far posted on a board

The StreamingCycles problem



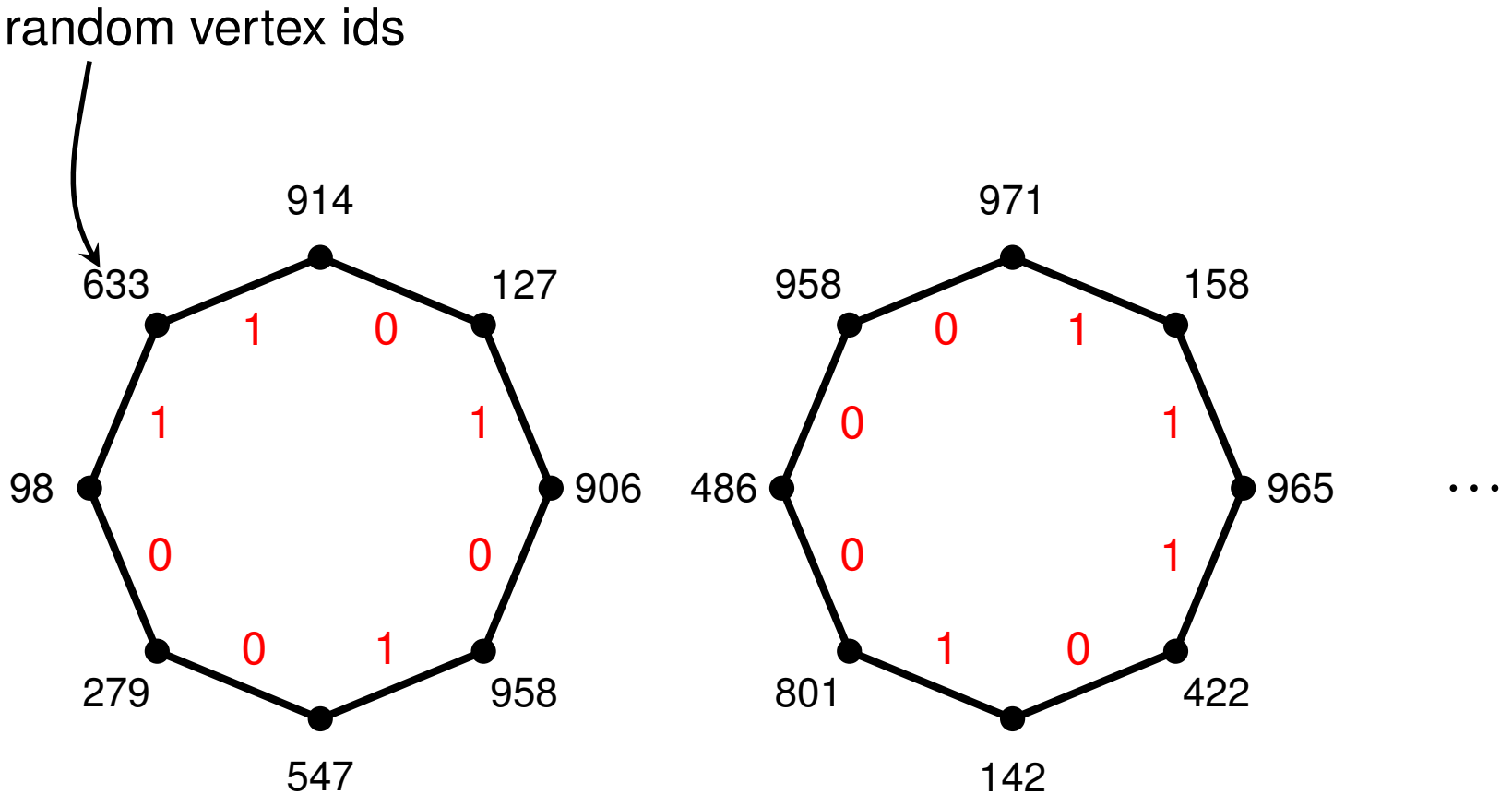
The StreamingCycles problem



Example stream:

$e_1 = (234, 345), \quad x_{234} = 0$

The StreamingCycles problem

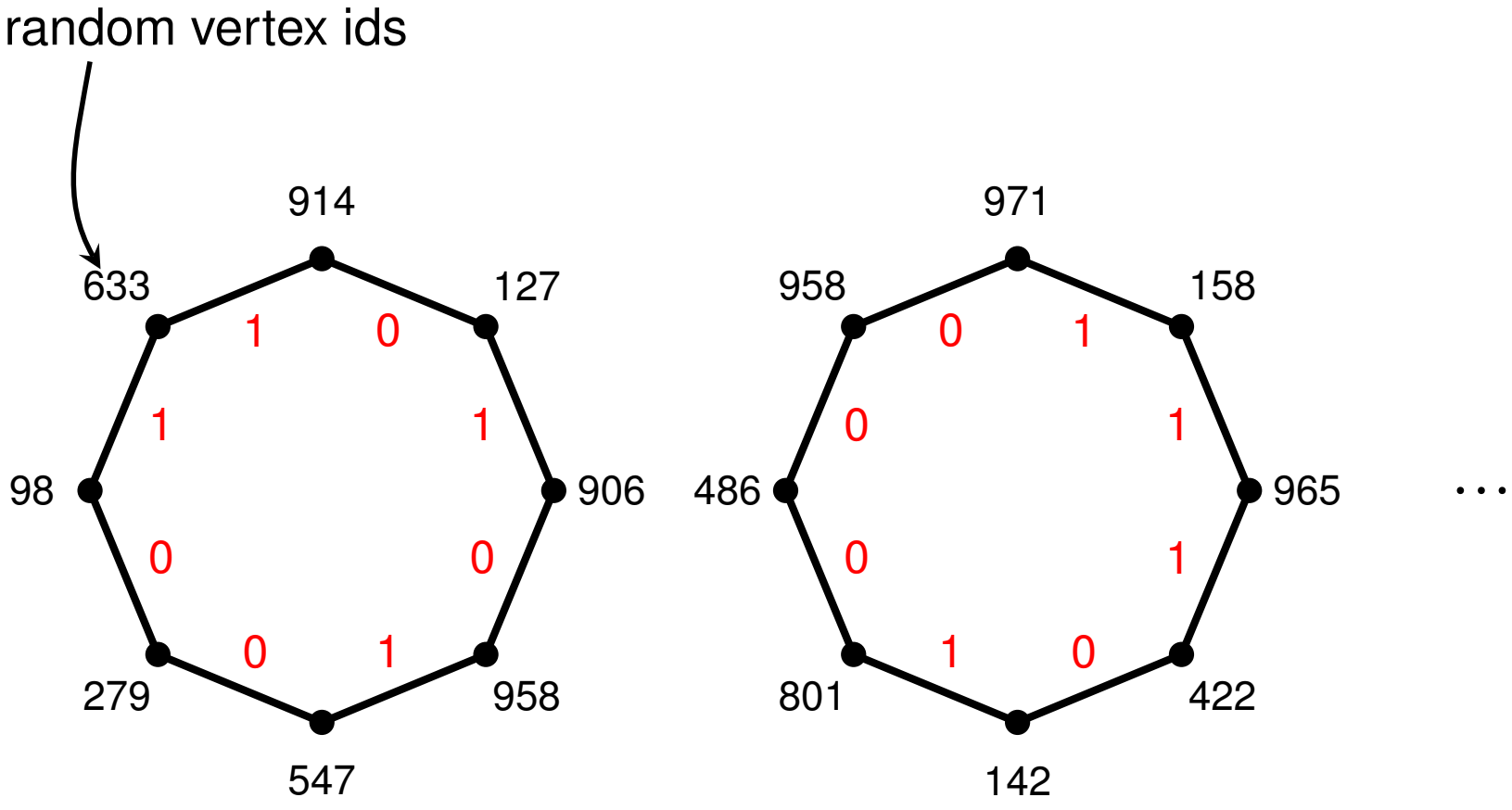


Example stream:

$e_1 = (234, 345), \quad x_{234} = 0$

$e_2 = (584, 745), \quad x_{584} = 1$

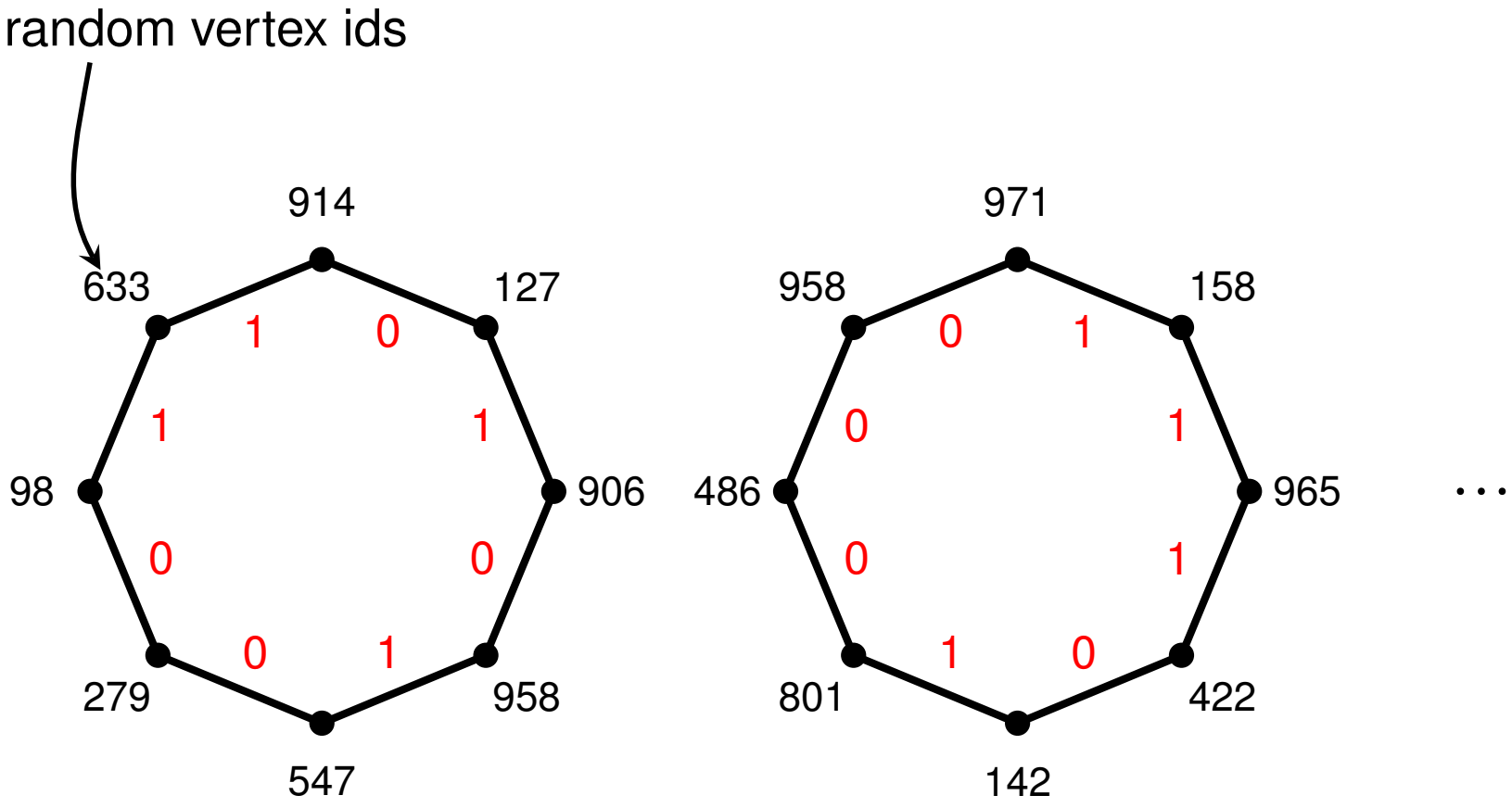
The StreamingCycles problem



Example stream:

$$e_1 = (234, 345), \quad x_{234} = 0$$
$$e_2 = (584, 745), \quad x_{584} = 1$$
$$e_2 = (123, 574), \quad x_{123} = 1$$

The StreamingCycles problem



Example stream:

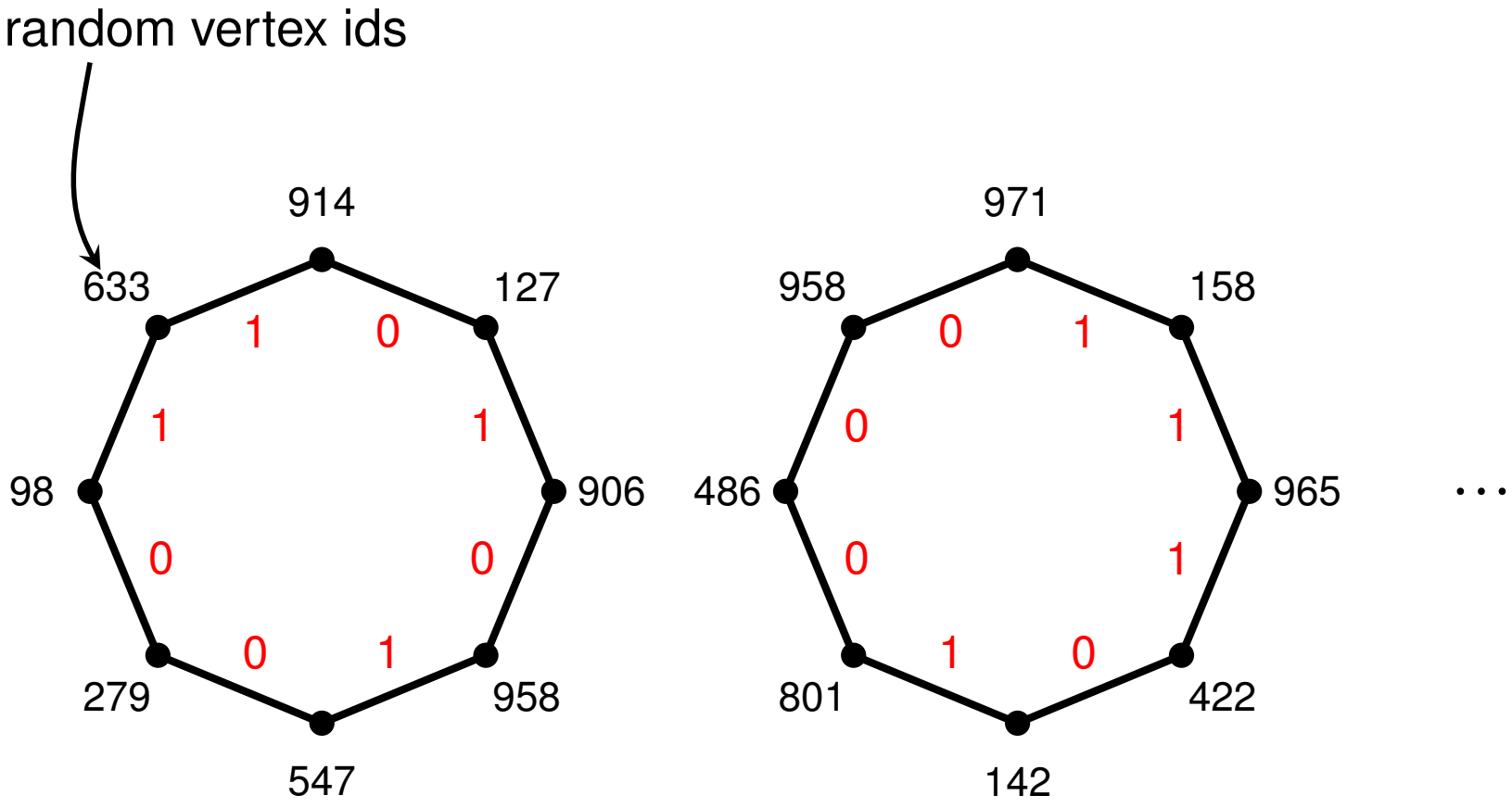
$e_1 = (234, 345), \quad x_{234} = 0$

$e_2 = (584, 745), \quad x_{584} = 1$

$e_2 = (123, 574), \quad x_{123} = 1$

...

The StreamingCycles problem



Example stream:

$e_1 = (234, 345), \quad x_{234} = 0$

$e_2 = (584, 745), \quad x_{584} = 1$

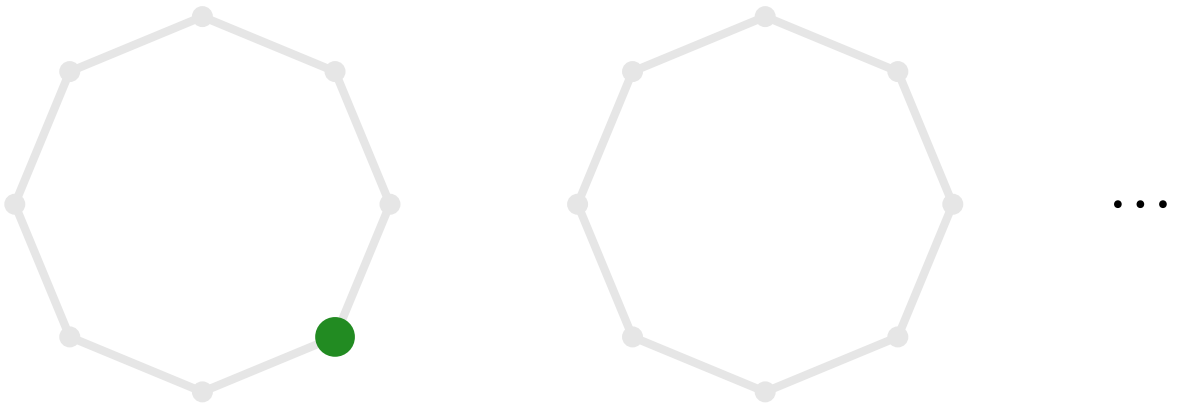
$e_2 = (123, 574), \quad x_{123} = 1$

...

Space complexity of StreamingCycles?

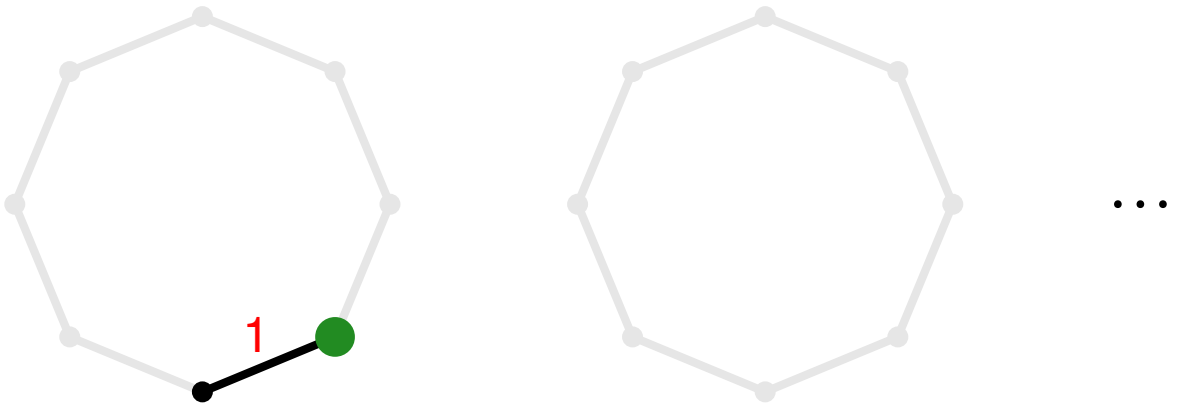
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



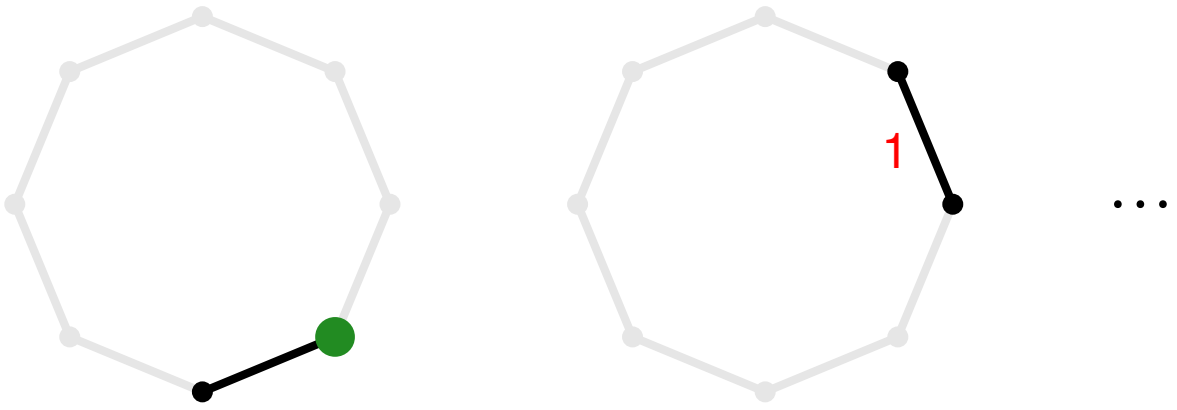
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



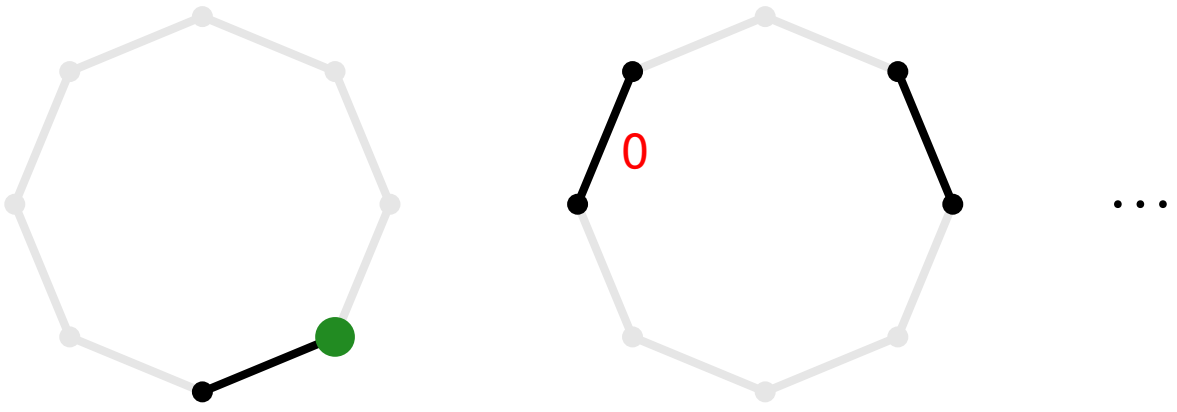
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



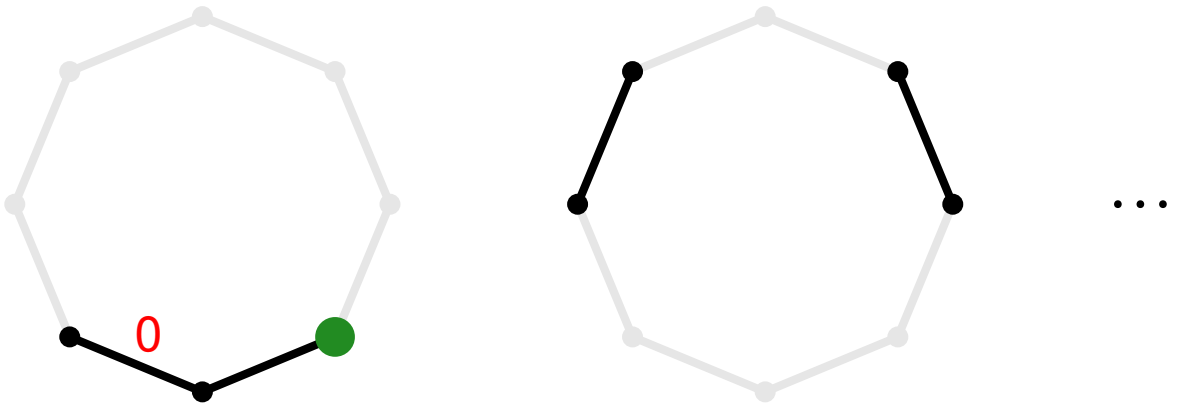
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



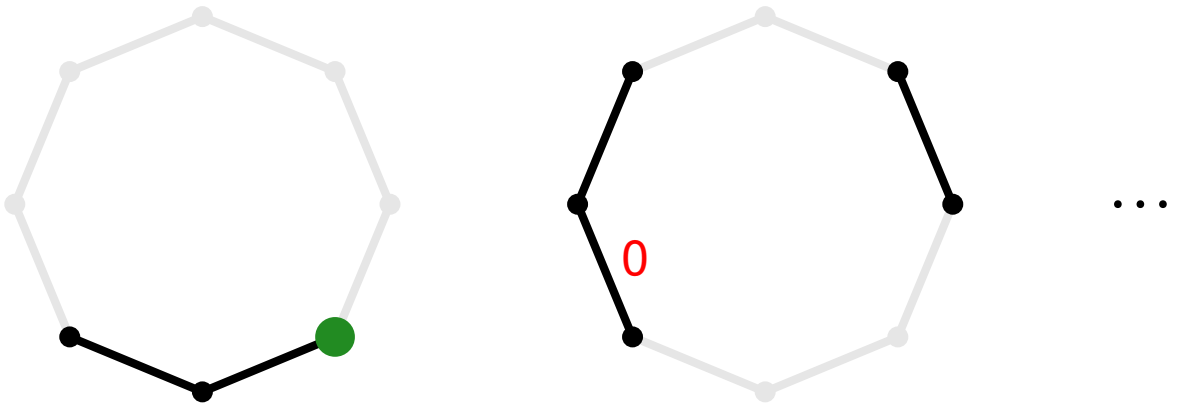
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



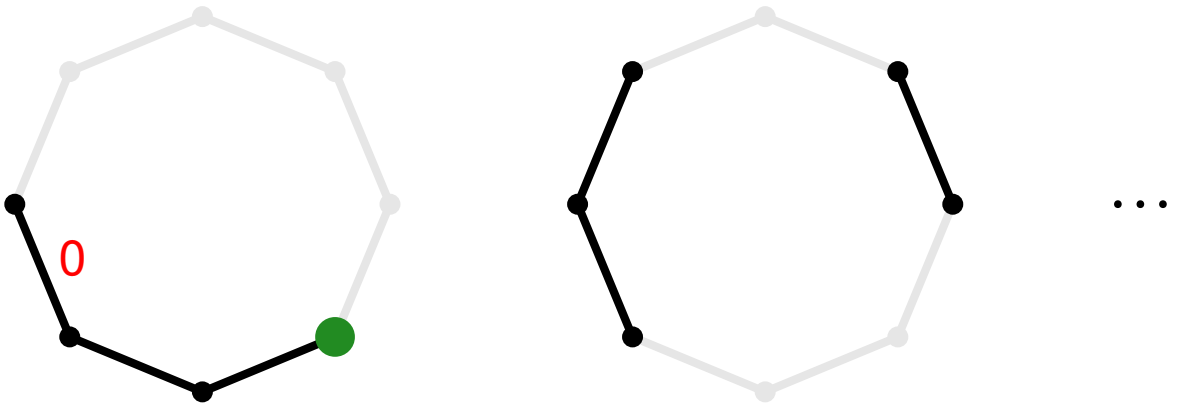
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



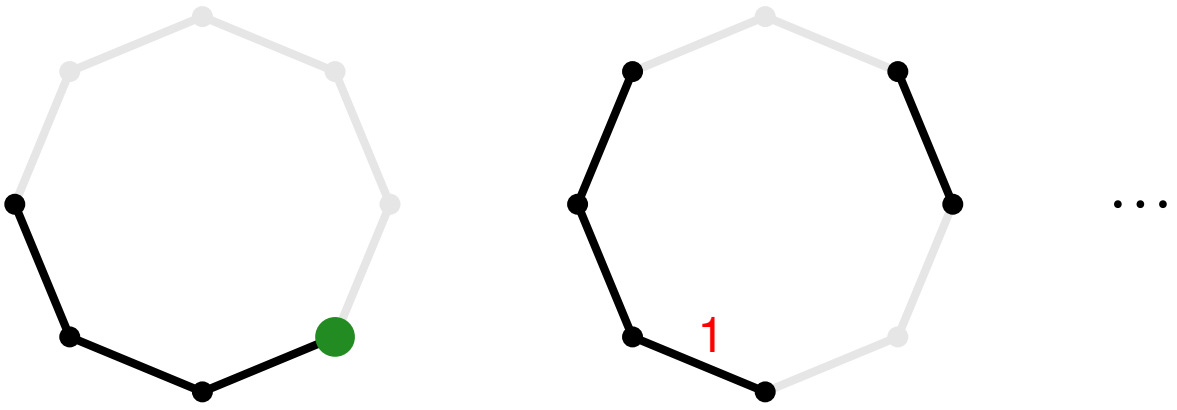
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



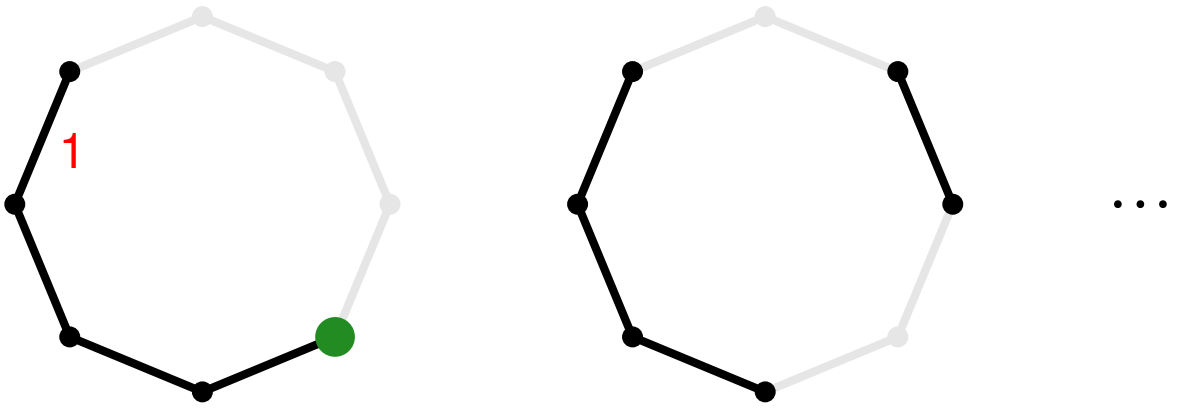
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



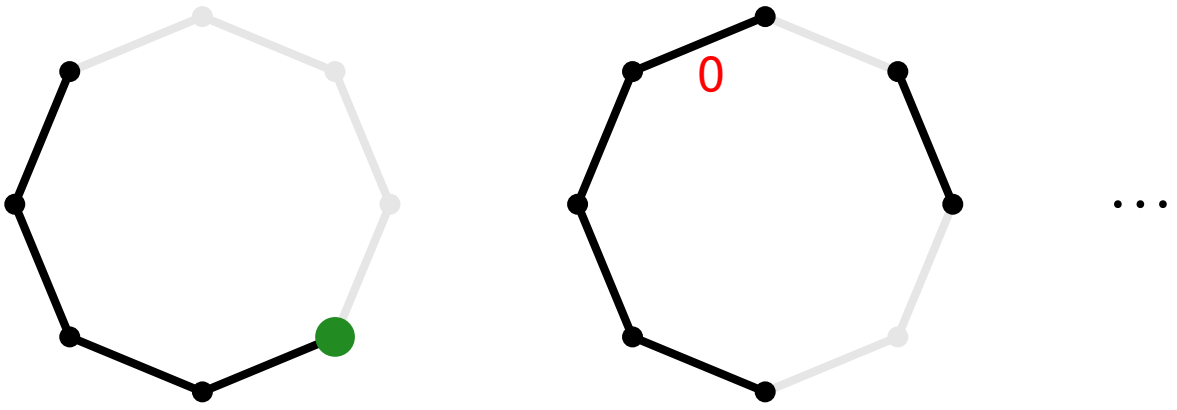
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



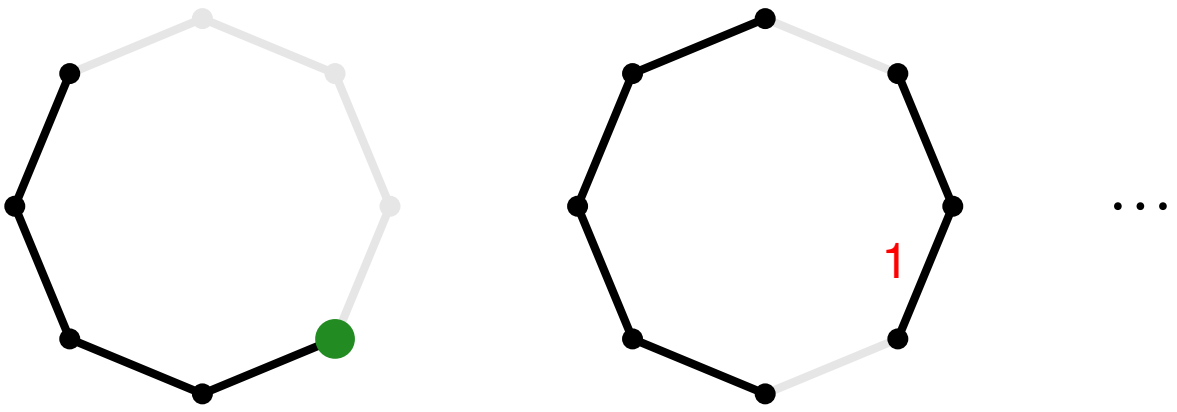
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



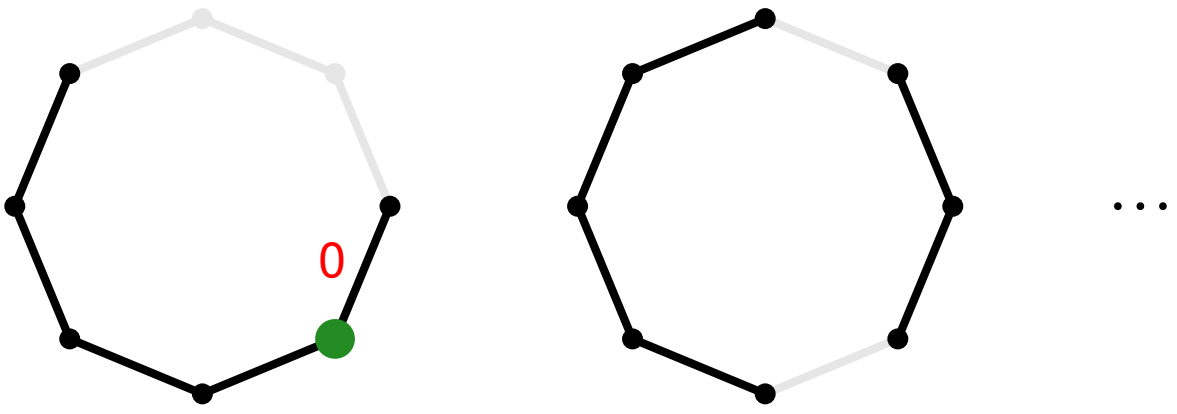
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



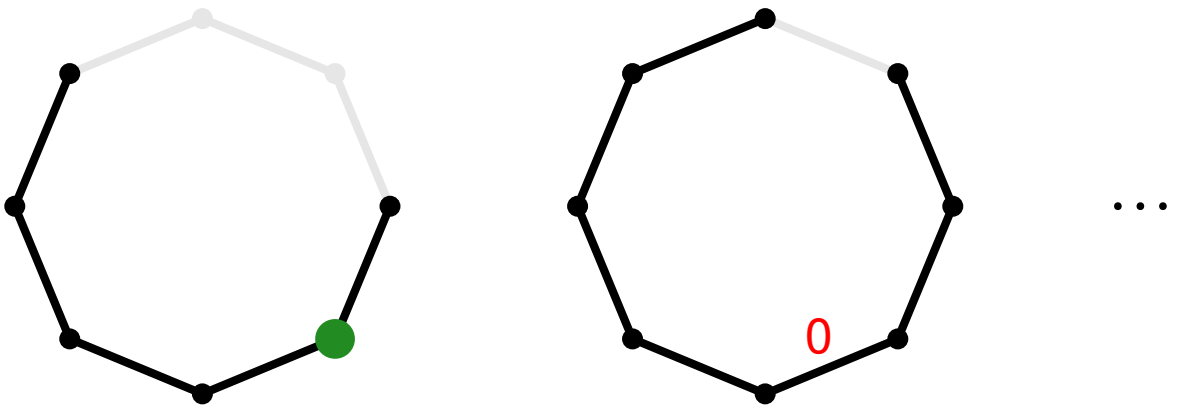
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



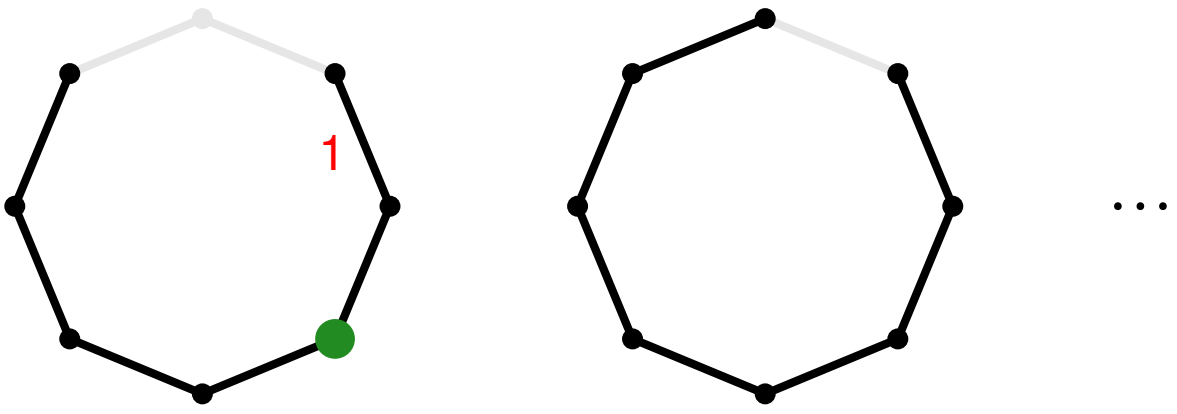
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



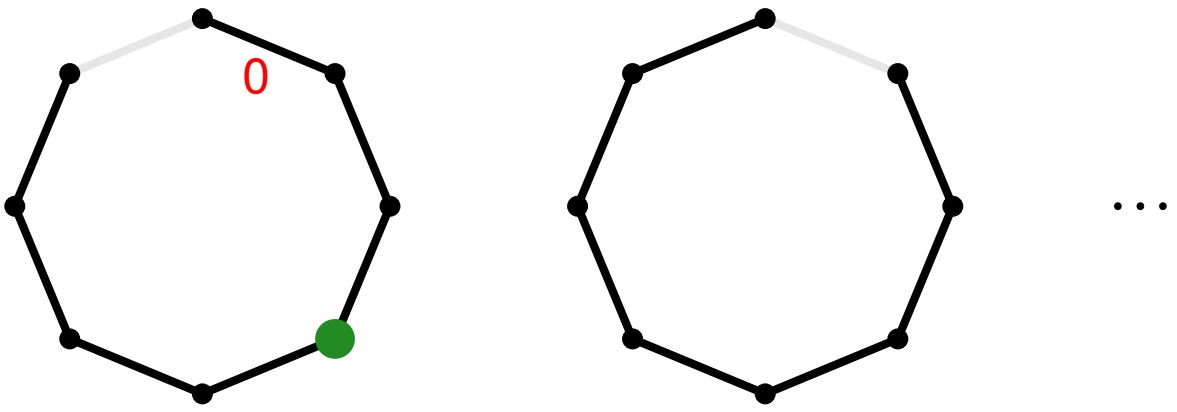
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



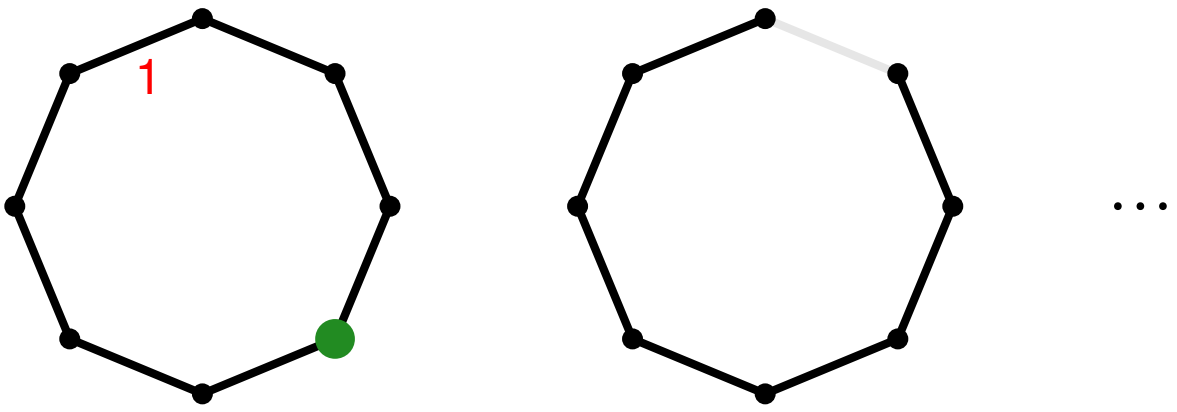
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



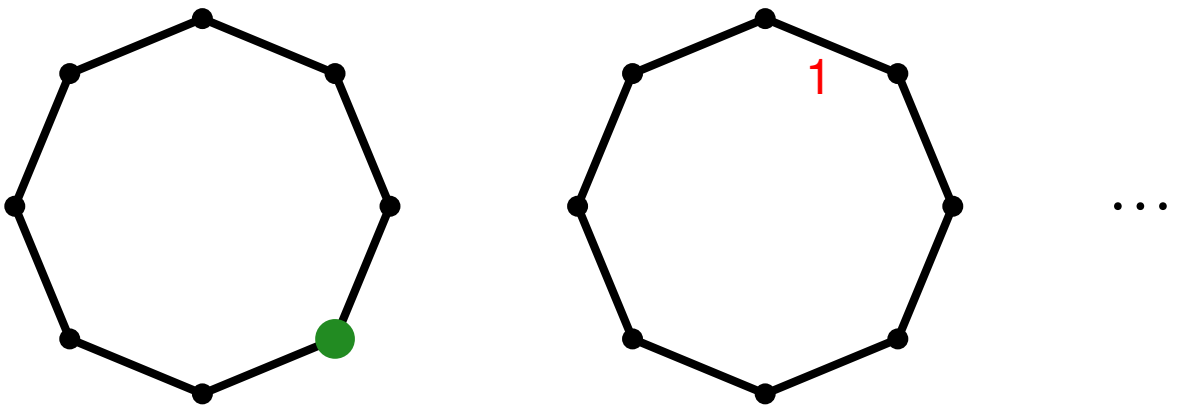
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



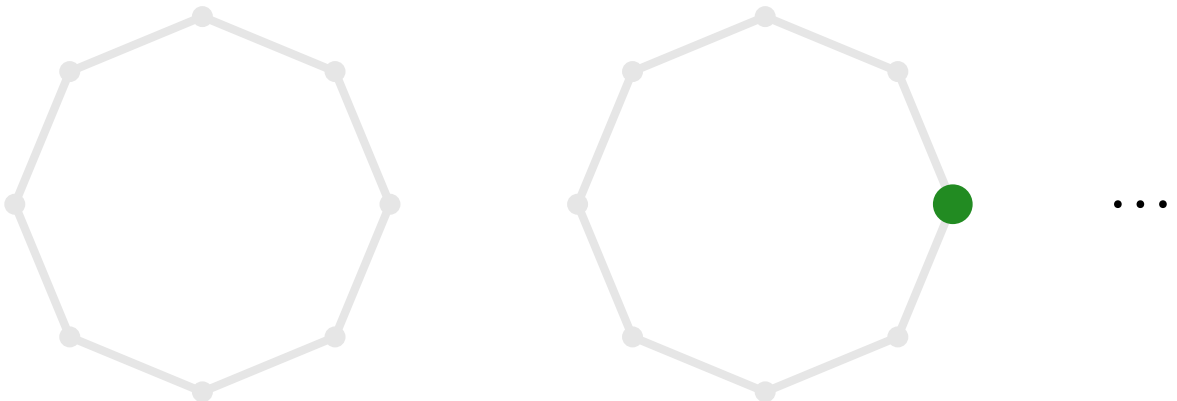
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



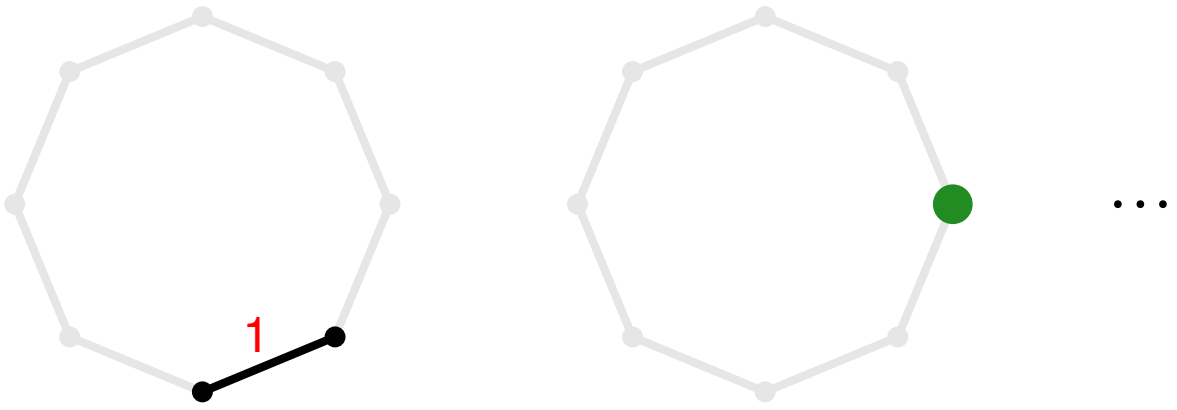
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



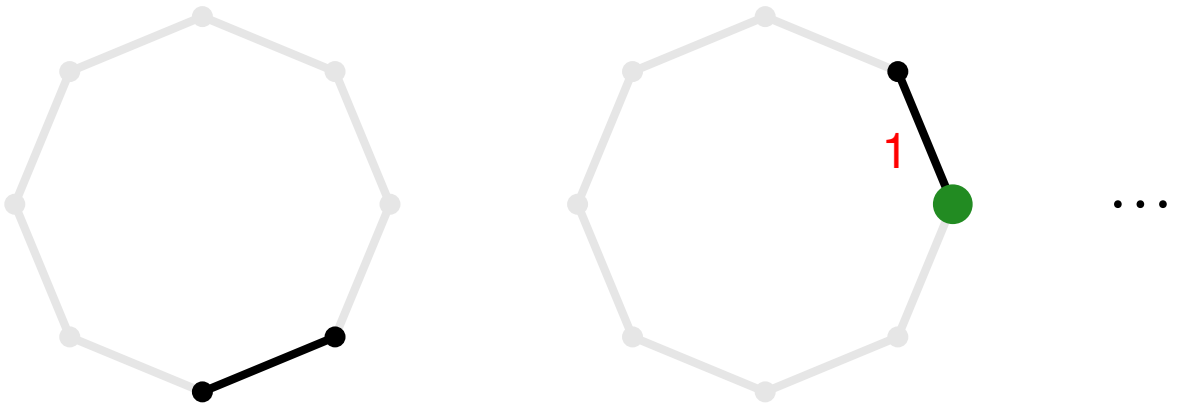
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



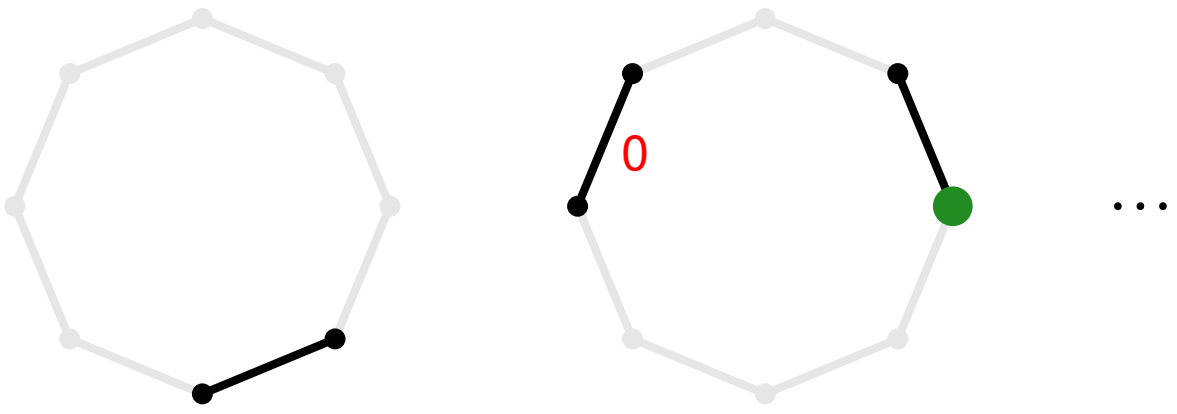
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



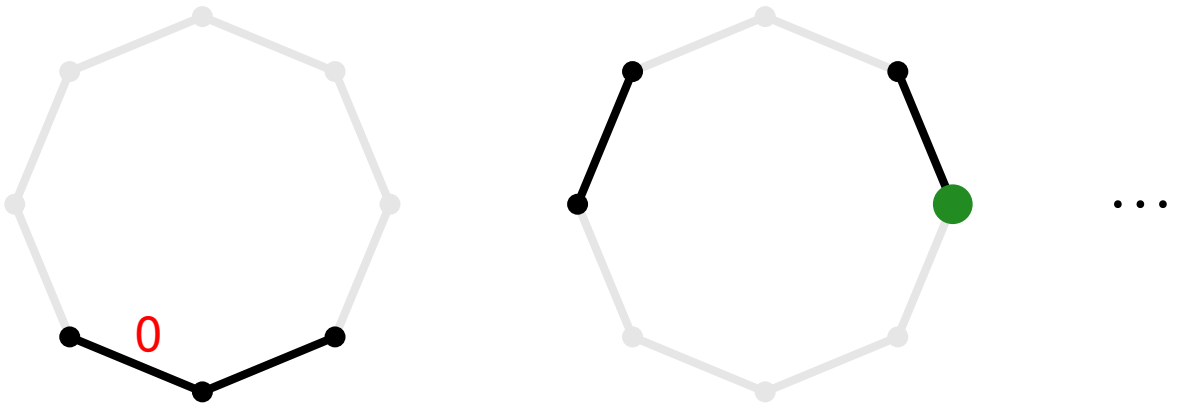
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



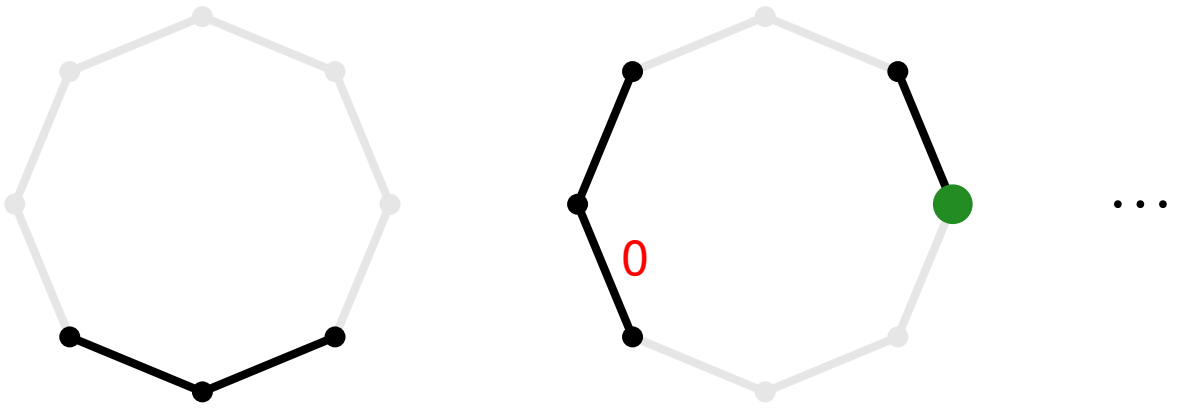
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



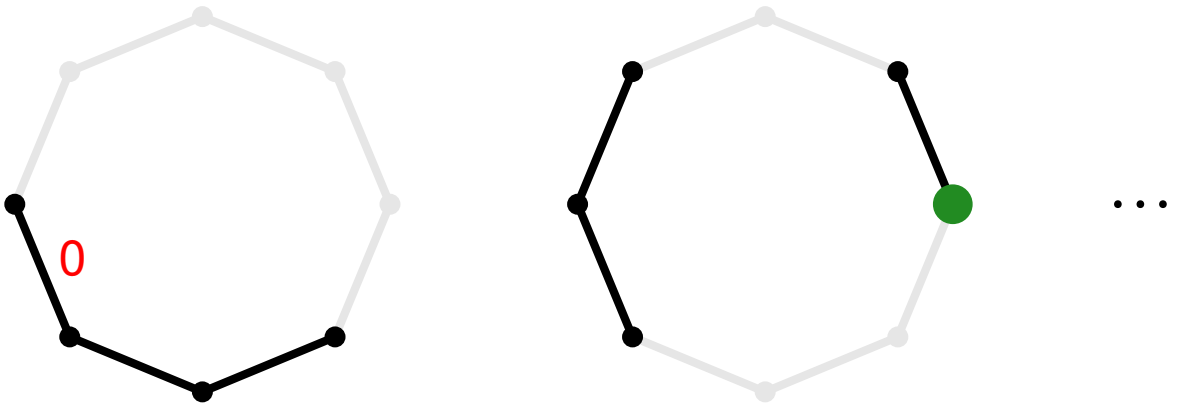
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



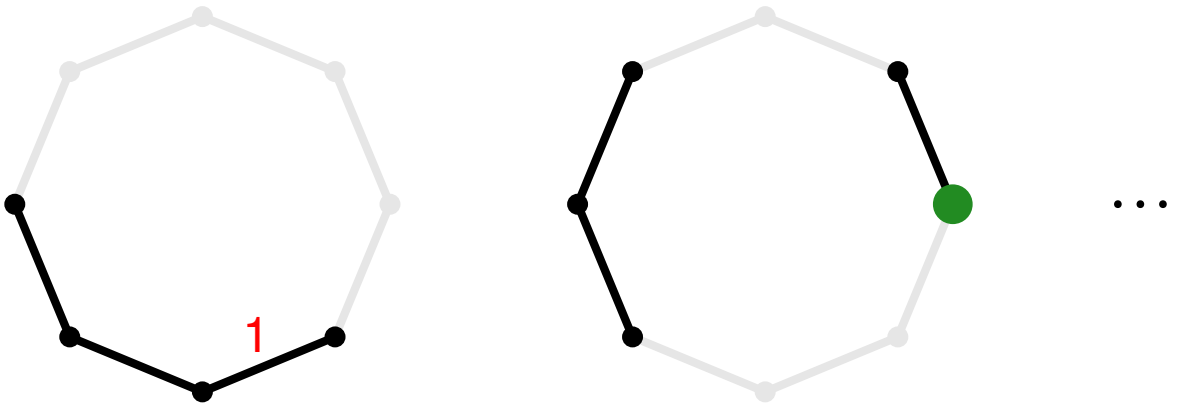
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



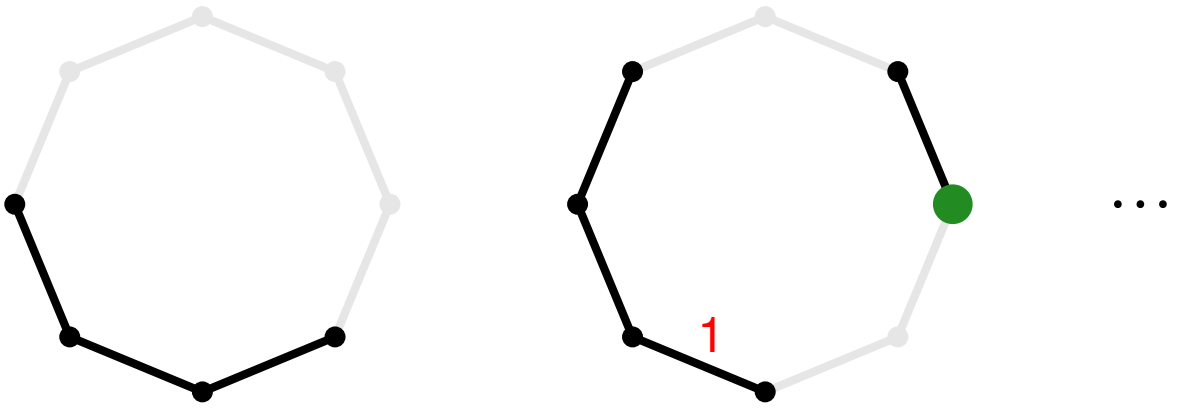
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



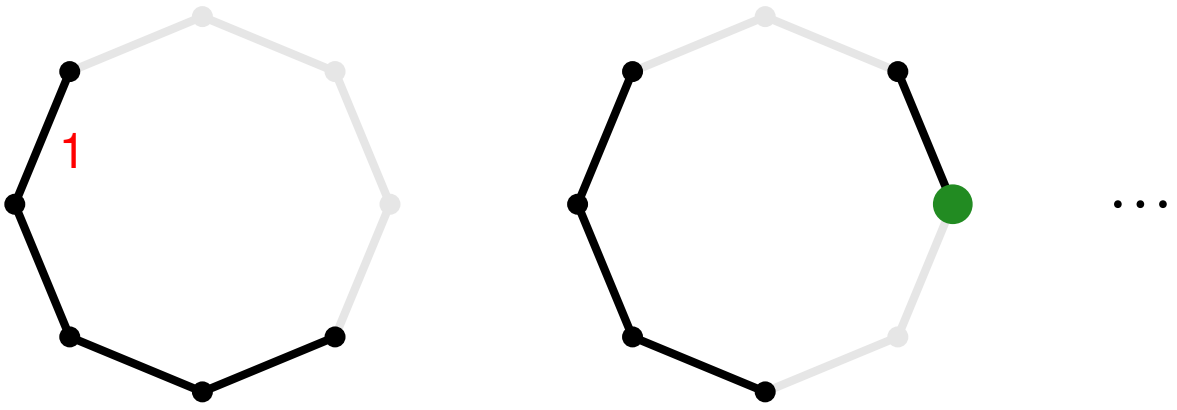
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



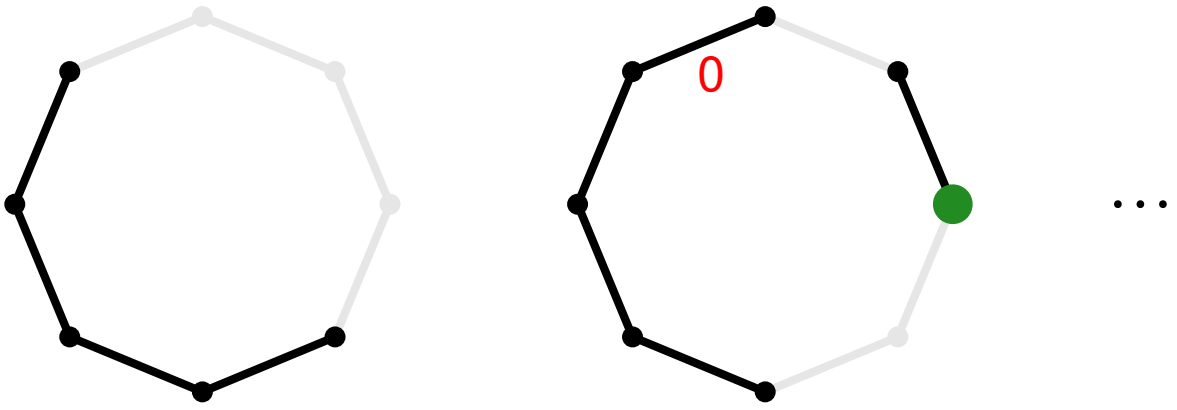
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



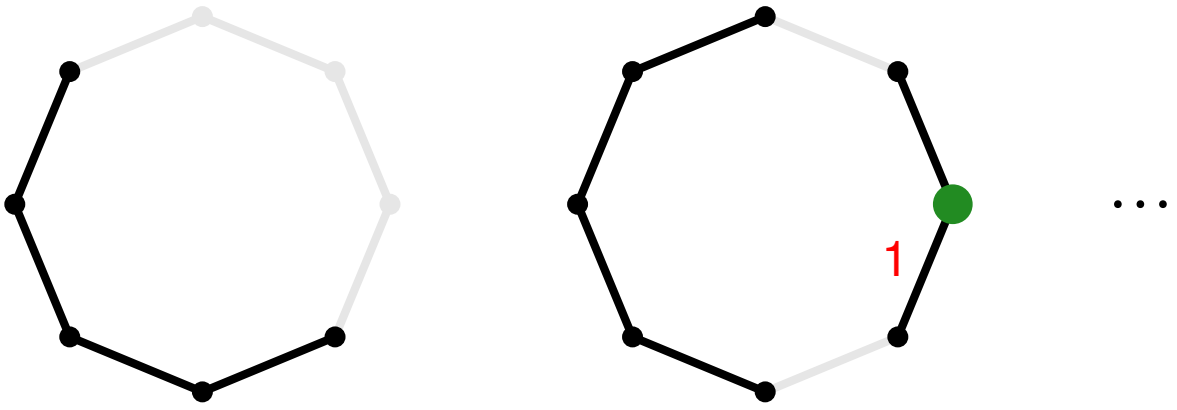
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



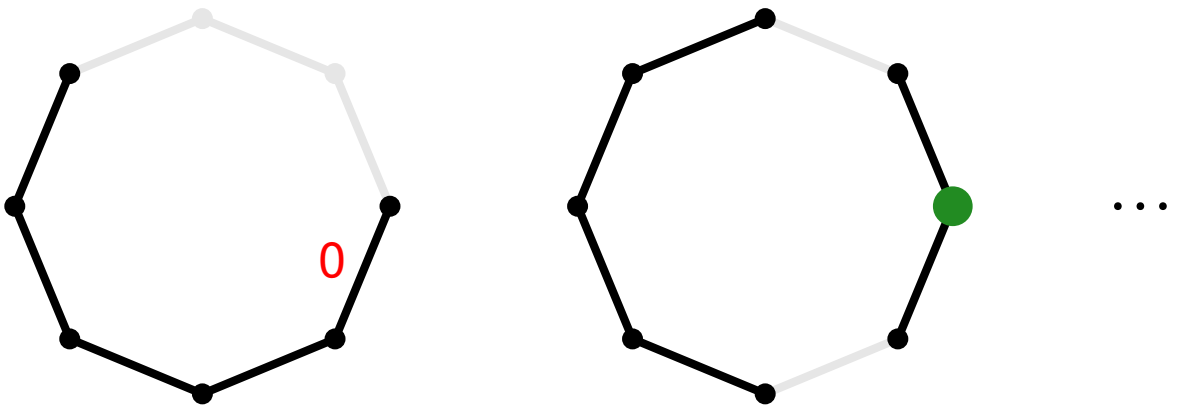
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



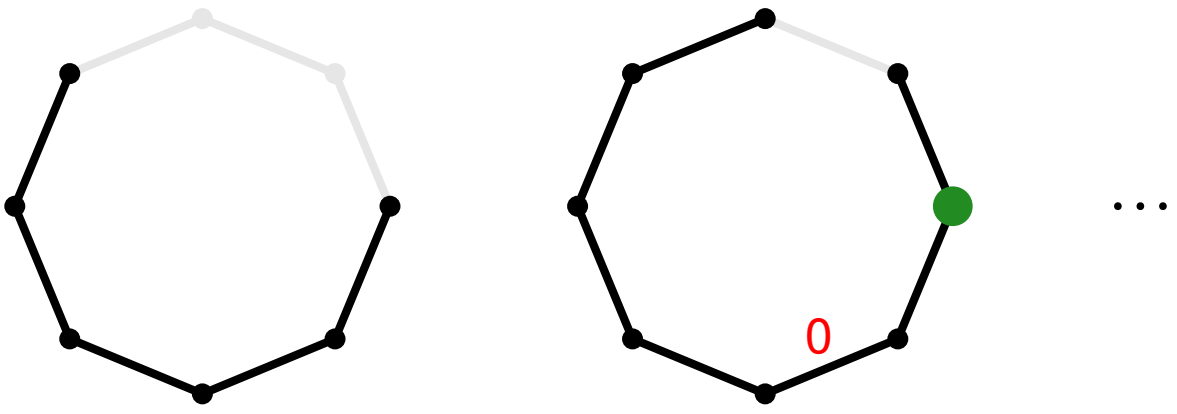
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



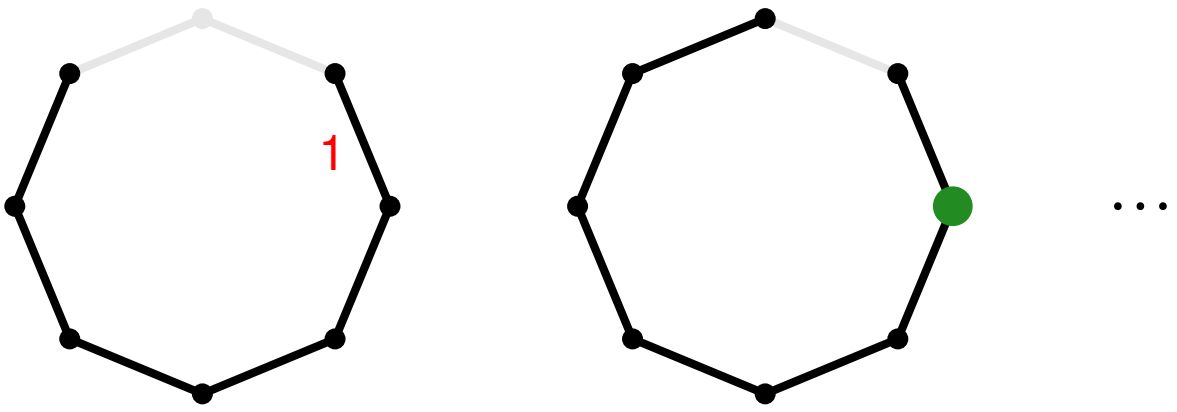
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



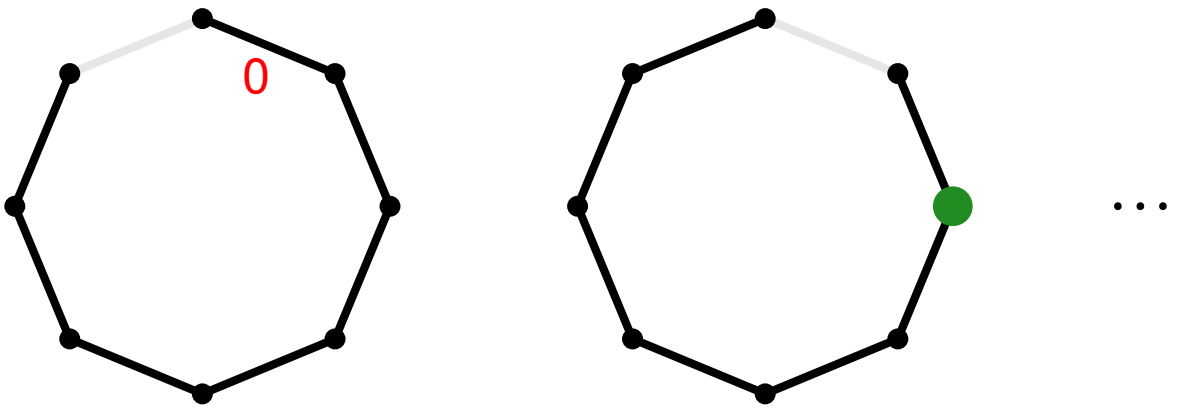
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



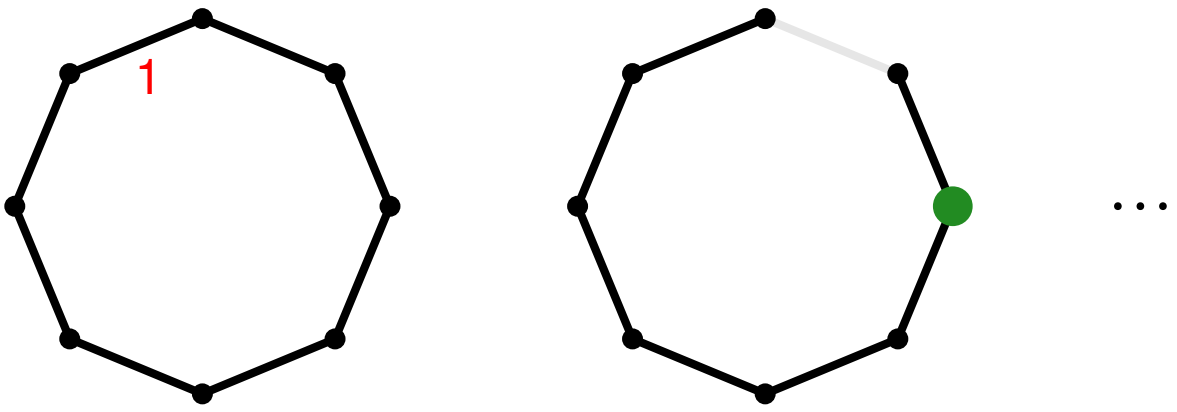
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



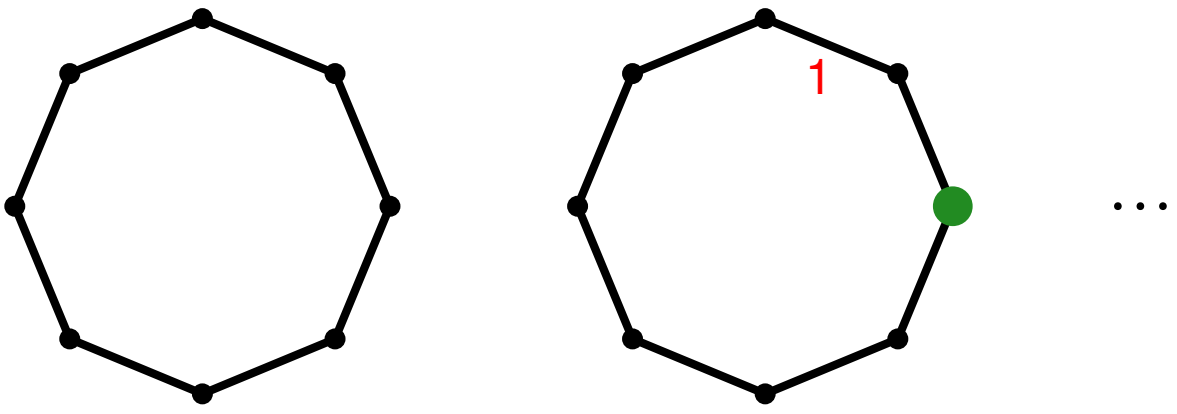
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



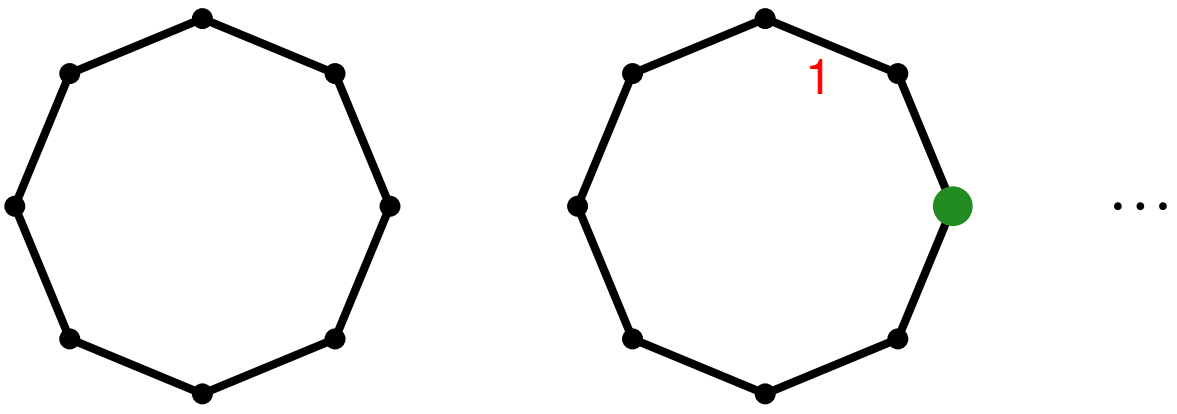
The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



The StreamingCycles problem

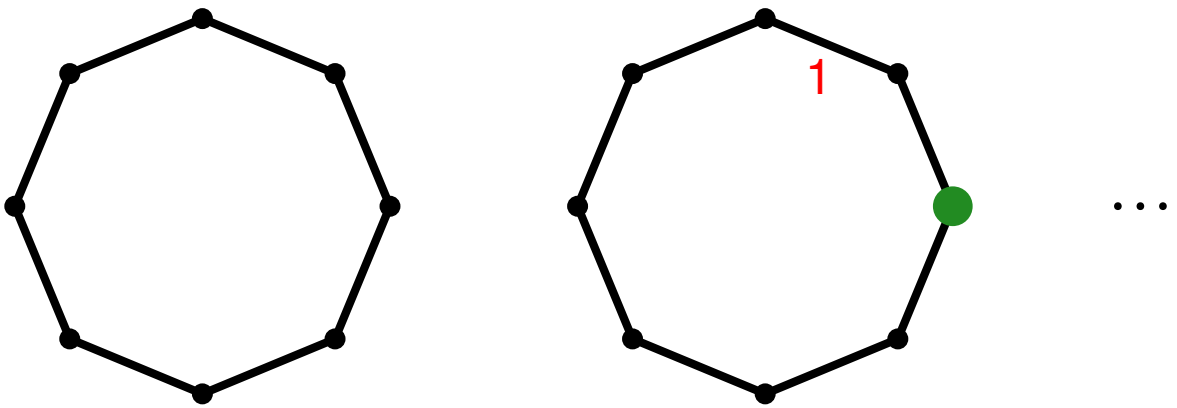
A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



Success probability=

The StreamingCycles problem

A simple protocol: maintain parity of connected component of vertex 1. Space=1 bit.



Success probability= $\ell^{-\Theta(\ell)}$.

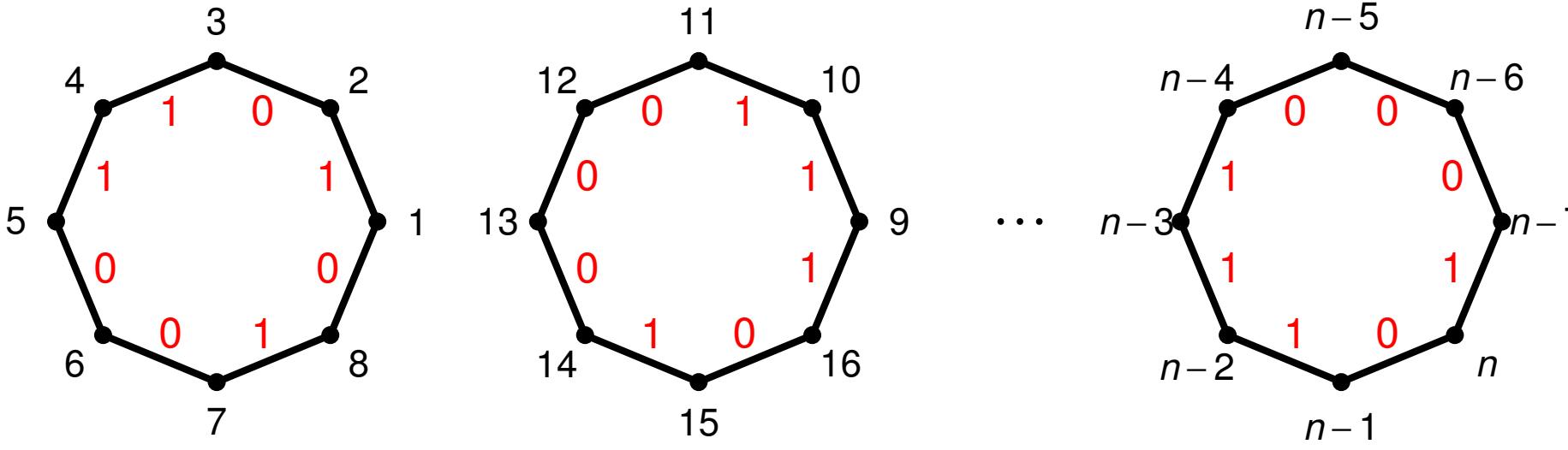
Maintain connected component of $\ell^{O(\ell)}$ random 'seed' vertices.
Space=$\ell^{O(\ell)}$, success probability= $1 - o(1)$.

Maintain connected component of $\ell^{O(\ell)}$ random 'seed' vertices.
Space=$\ell^{O(\ell)}$, success probability= $1 - o(1)$.

Theorem (Main result; informal)

This is tight up to constant factors in the exponent.

The StreamingCycles problem



Maintain connected component of $\ell^{O(\ell)}$ random 'seed' vertices.
Space=$\ell^{O(\ell)}$, success probability= $1 - o(1)$.

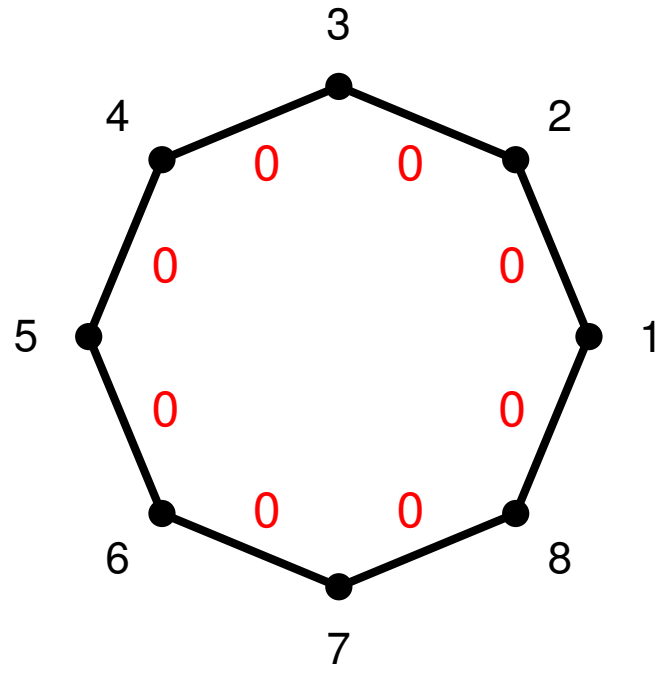
Theorem (Main result (informal))

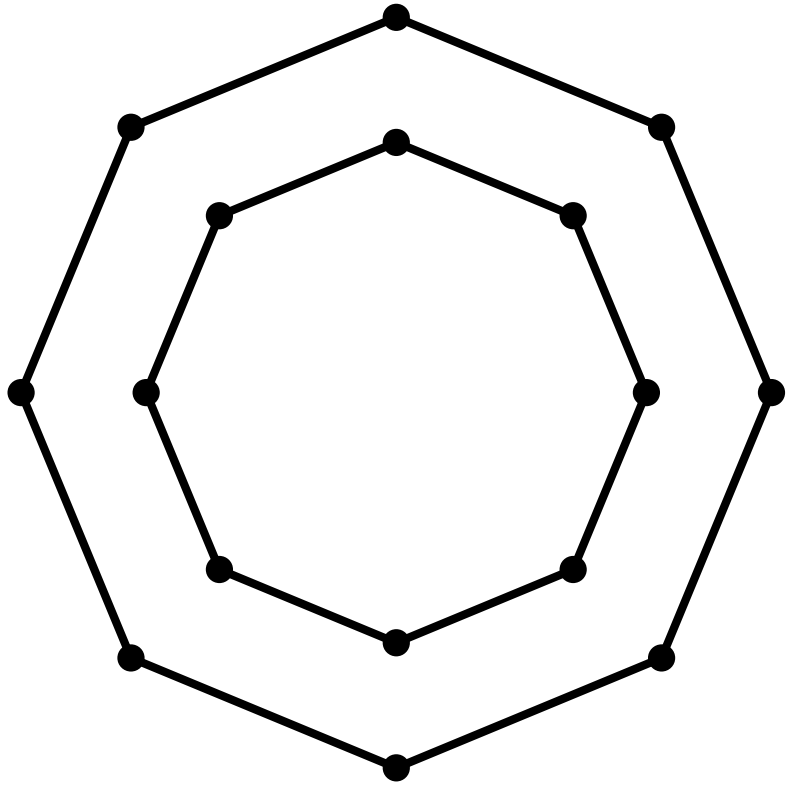
The communication complexity of StreamingCycles *is* $\ell^{\Omega(\ell)}$.

Space complexity of component finding

For every $i \in [n]$ create two vertices i_0 and i_1.

For $\{i,j\}$ insert $\begin{cases} \{i_0,j_0\},\{i_1,j_1\} & \text{if } x_{ij}=0 \\ \{i_0,j_1\},\{i_1,j_0\} & \text{o.w.} \end{cases}$

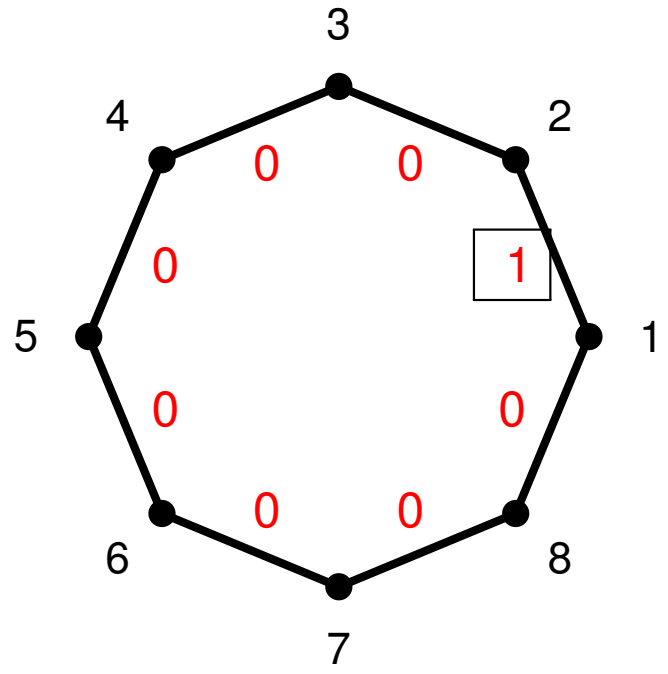


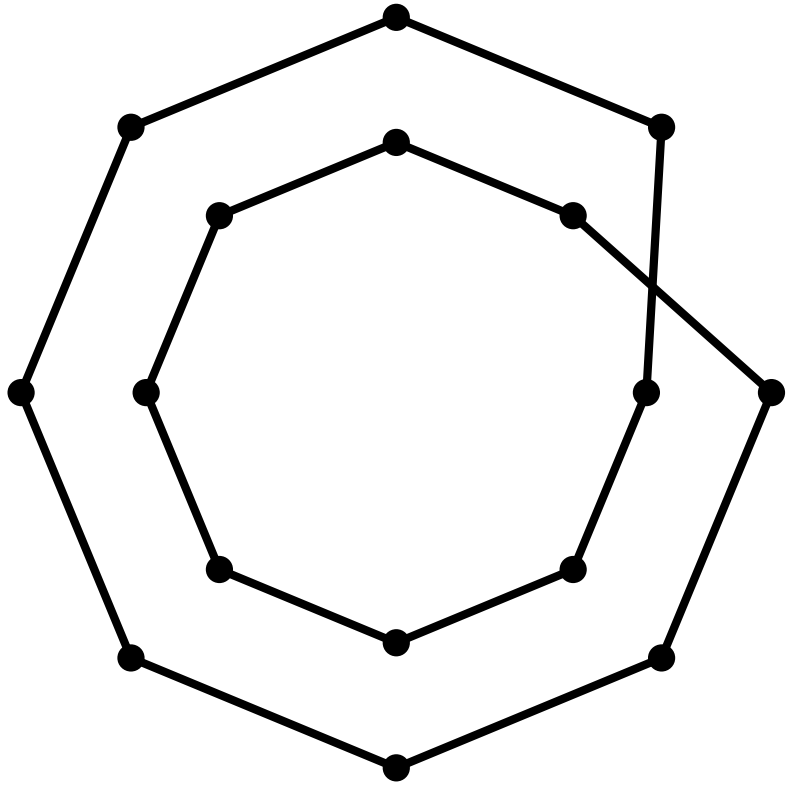


Space complexity of component finding

For every $i \in [n]$ create two vertices i_0 and i_1.

For $\{i,j\}$ insert $\begin{cases} \{i_0,j_0\},\{i_1,j_1\} & \text{if } x_{ij}=0 \\ \{i_0,j_1\},\{i_1,j_0\} & \text{o.w.} \end{cases}$

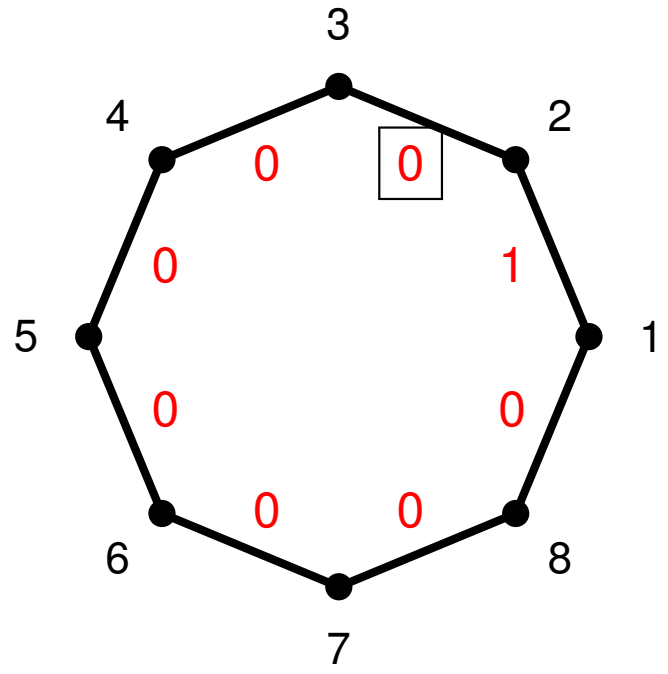


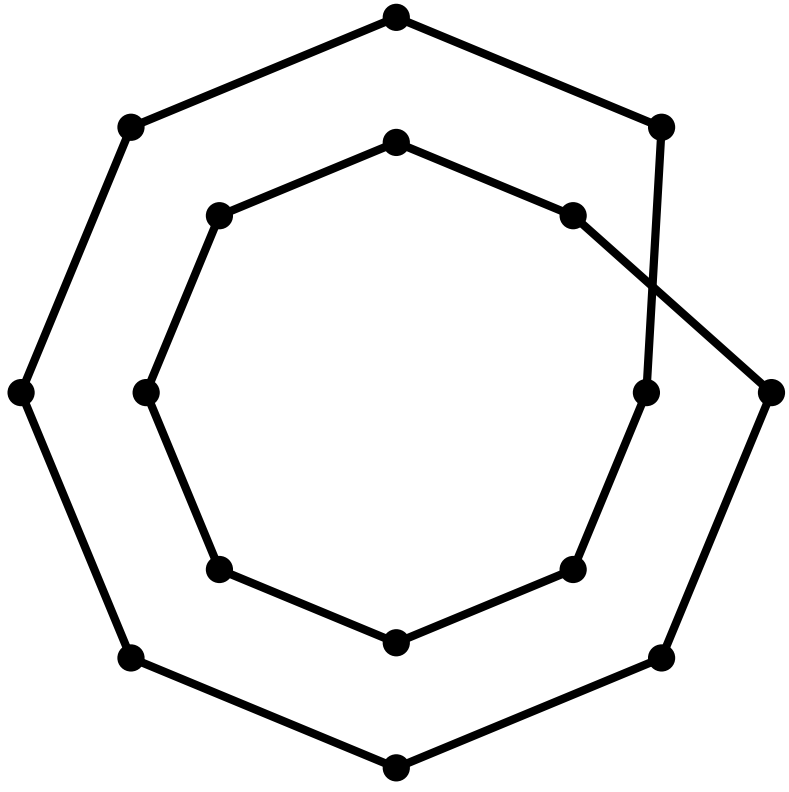


Space complexity of component finding

For every $i \in [n]$ create two vertices i_0 and i_1.

$$\text{For } \{i,j\} \text{ insert } \begin{cases} \{i_0,j_0\},\{i_1,j_1\} & \text{if } x_{ij}=0 \\ \{i_0,j_1\},\{i_1,j_0\} & \text{o.w.} \end{cases}$$

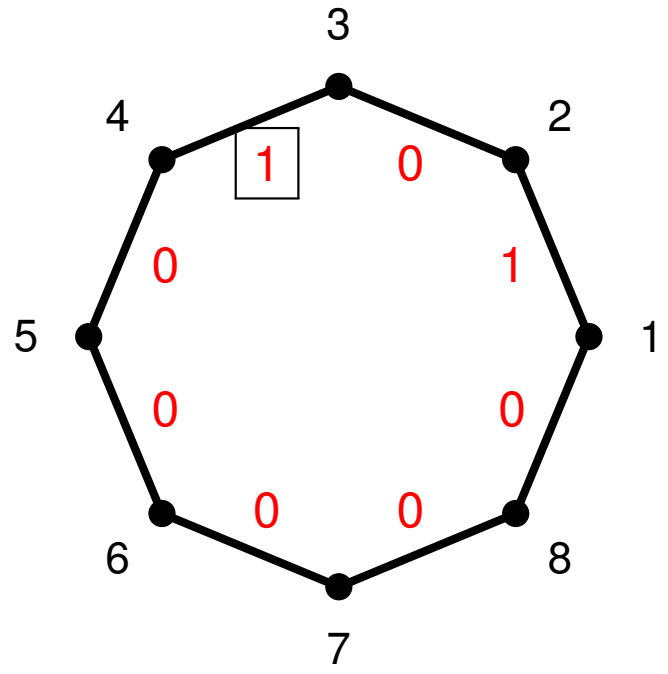


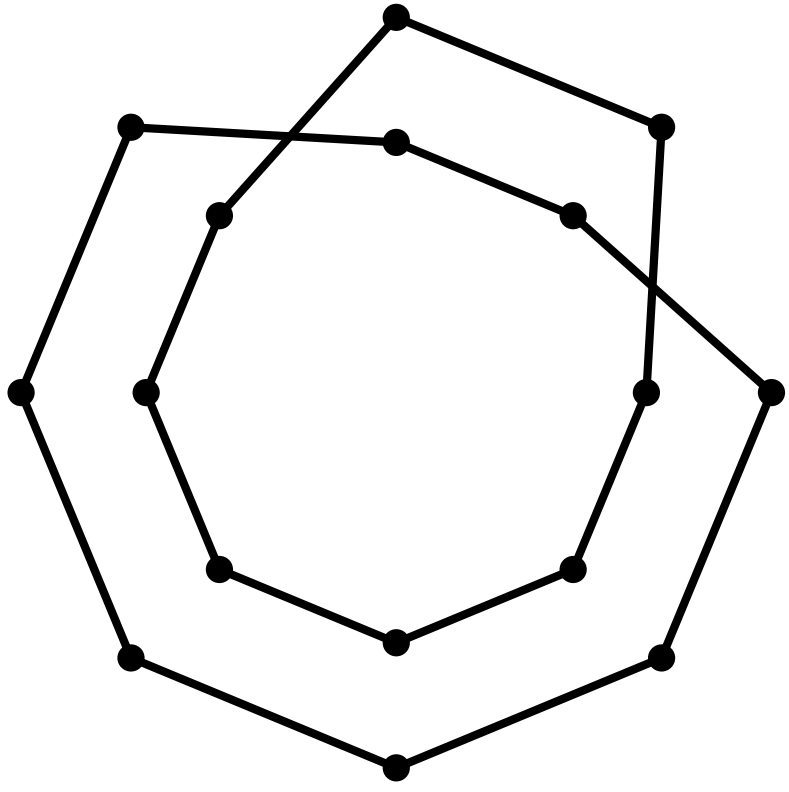


Space complexity of component finding

For every $i \in [n]$ create two vertices i_0 and i_1.

$$\text{For } \{i,j\} \text{ insert } \begin{cases} \{i_0,j_0\},\{i_1,j_1\} & \text{if } x_{ij}=0 \\ \{i_0,j_1\},\{i_1,j_0\} & \text{o.w.} \end{cases}$$

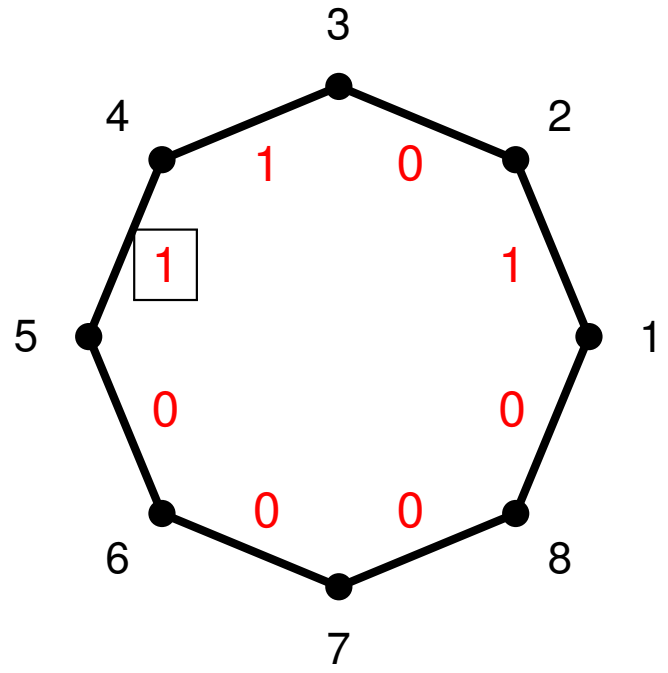


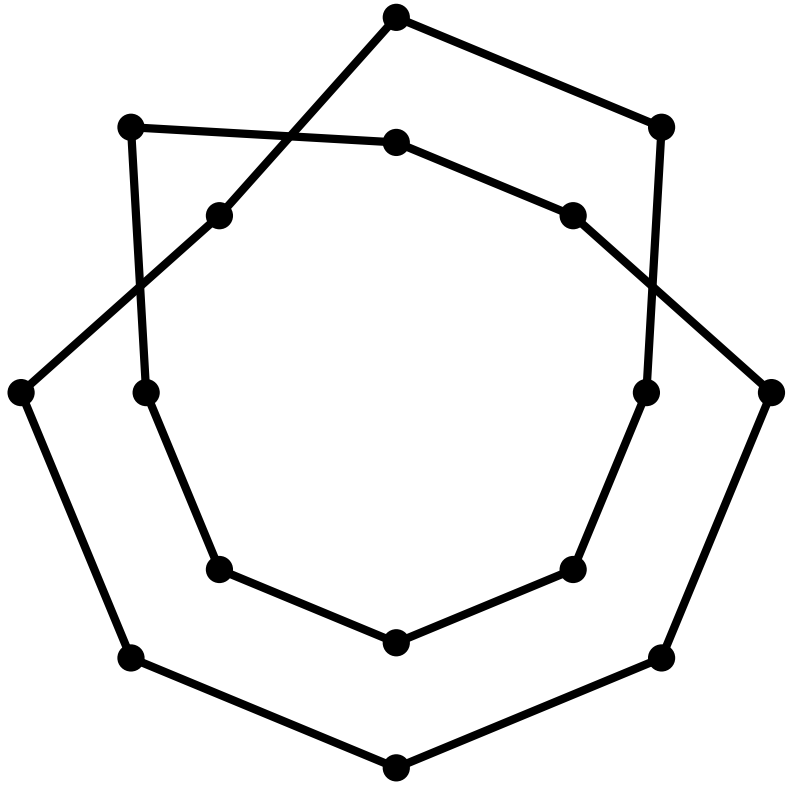


Space complexity of component finding

For every $i \in [n]$ create two vertices i_0 and i_1.

$$\text{For } \{i,j\} \text{ insert } \begin{cases} \{i_0,j_0\},\{i_1,j_1\} & \text{if } x_{ij}=0 \\ \{i_0,j_1\},\{i_1,j_0\} & \text{o.w.} \end{cases}$$

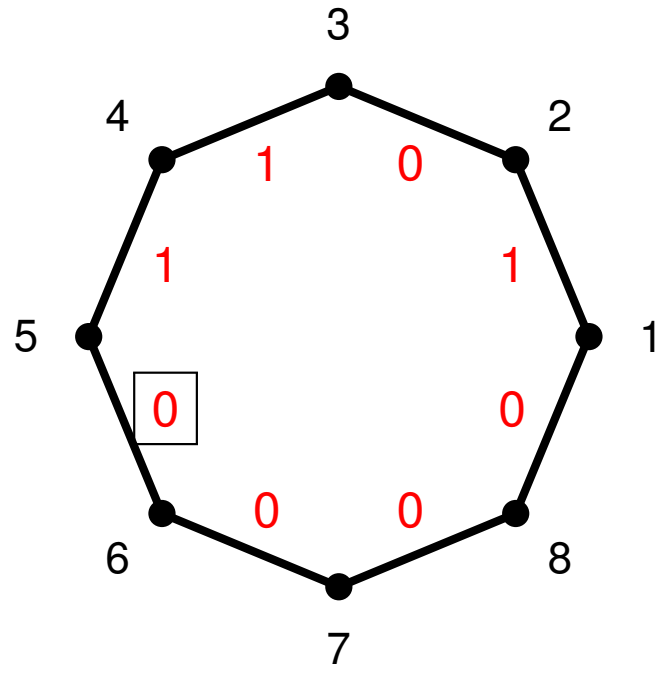


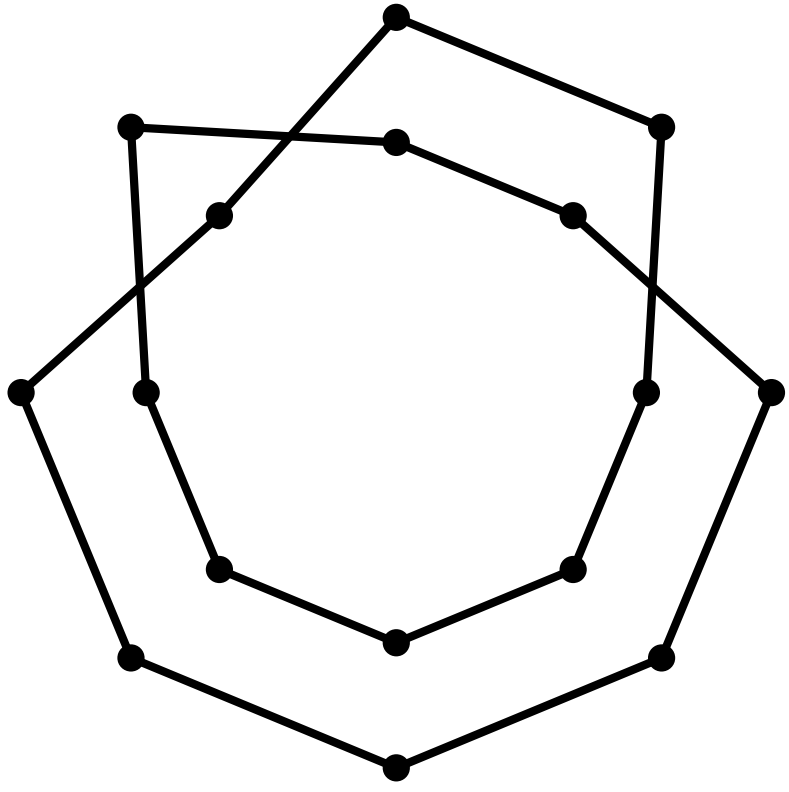


Space complexity of component finding

For every $i \in [n]$ create two vertices i_0 and i_1.

For $\{i,j\}$ insert $\begin{cases} \{i_0,j_0\},\{i_1,j_1\} & \text{if } x_{ij}=0 \\ \{i_0,j_1\},\{i_1,j_0\} & \text{o.w.} \end{cases}$

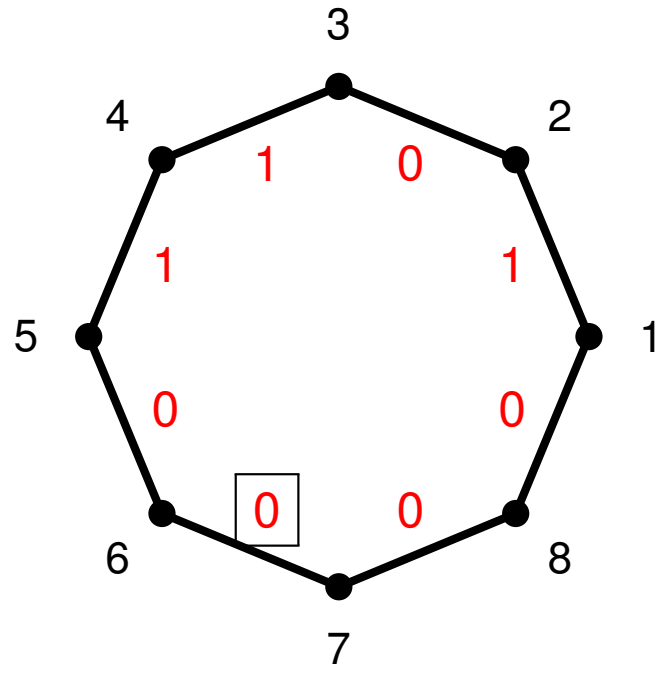


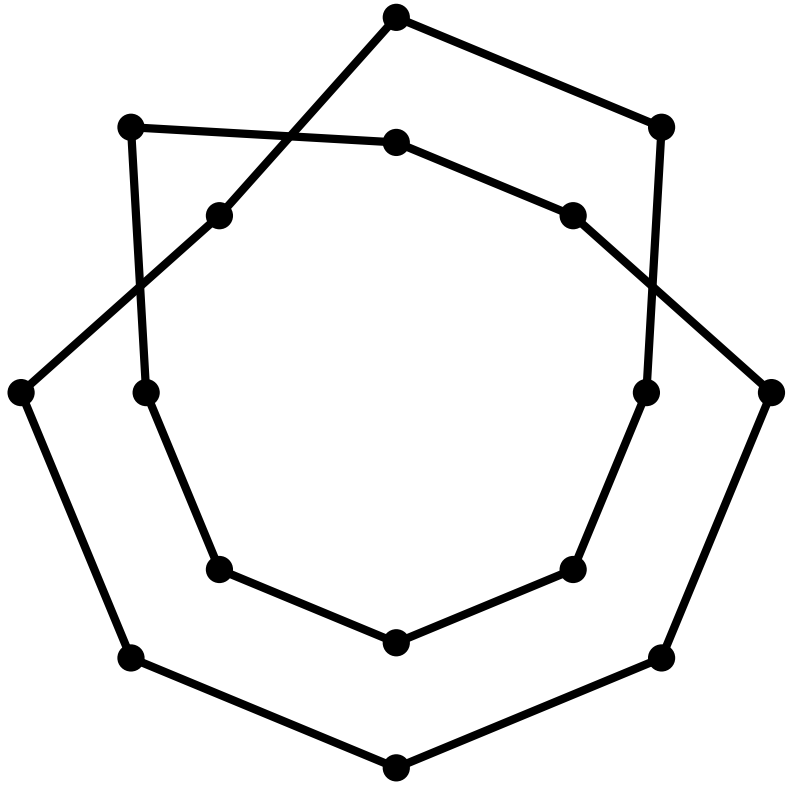


Space complexity of component finding

For every $i \in [n]$ create two vertices i_0 and i_1.

For $\{i,j\}$ insert $\begin{cases} \{i_0,j_0\},\{i_1,j_1\} & \text{if } x_{ij}=0 \\ \{i_0,j_1\},\{i_1,j_0\} & \text{o.w.} \end{cases}$

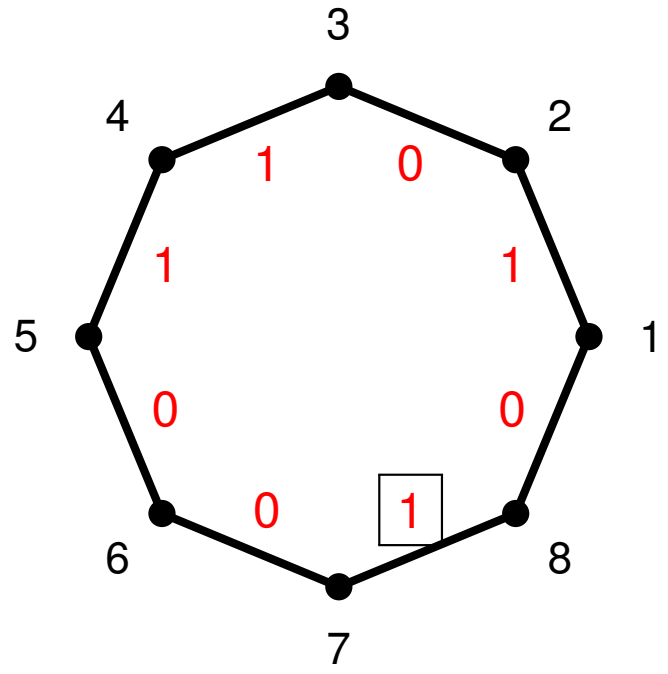


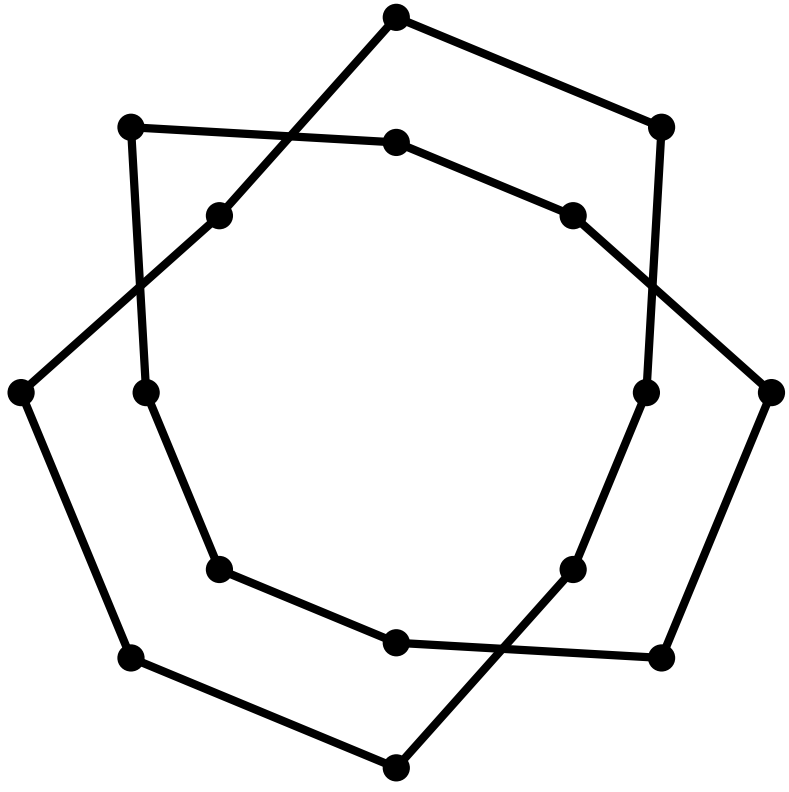


Space complexity of component finding

For every $i \in [n]$ create two vertices i_0 and i_1.

$$\text{For } \{i,j\} \text{ insert } \begin{cases} \{i_0,j_0\},\{i_1,j_1\} & \text{if } x_{ij}=0 \\ \{i_0,j_1\},\{i_1,j_0\} & \text{o.w.} \end{cases}$$

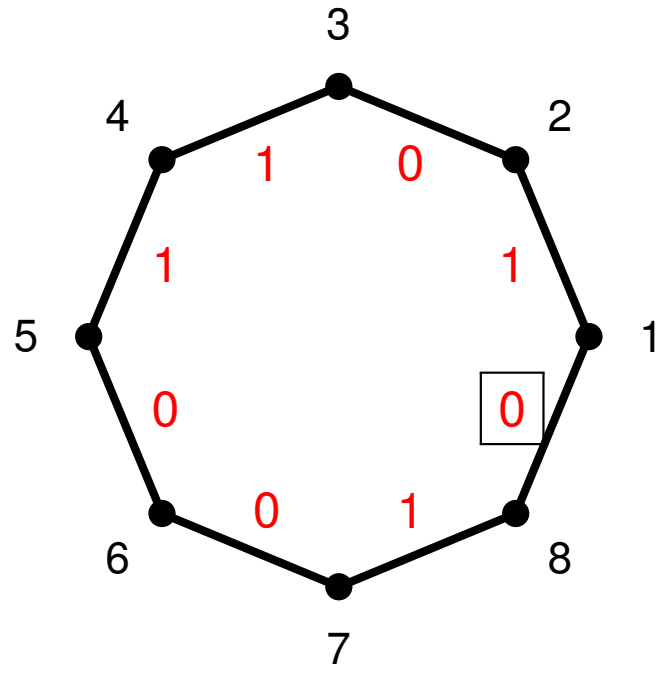


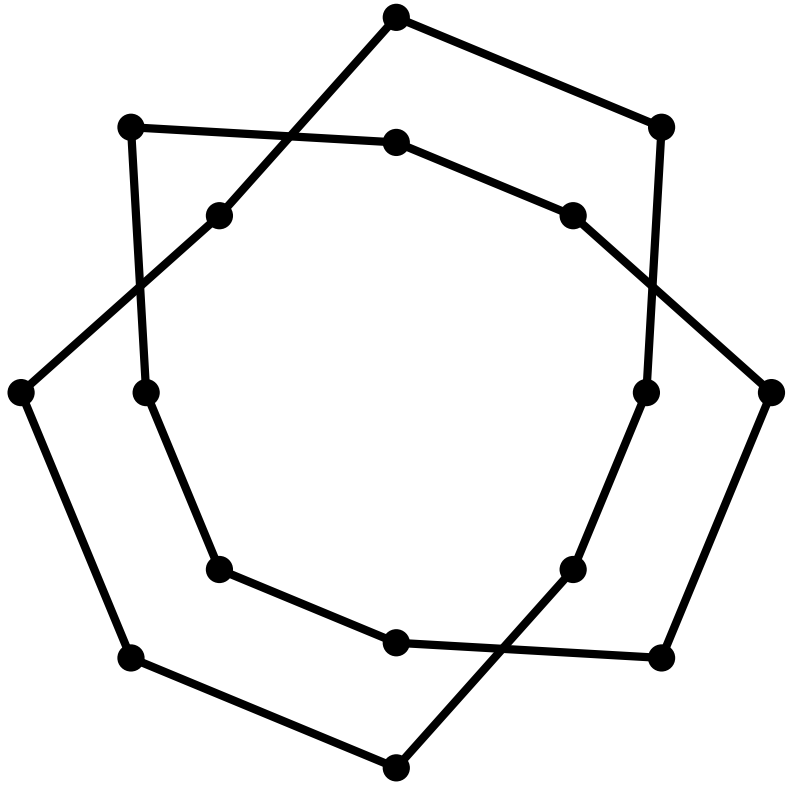


Space complexity of component finding

For every $i \in [n]$ create two vertices i_0 and i_1 (inner and outer).

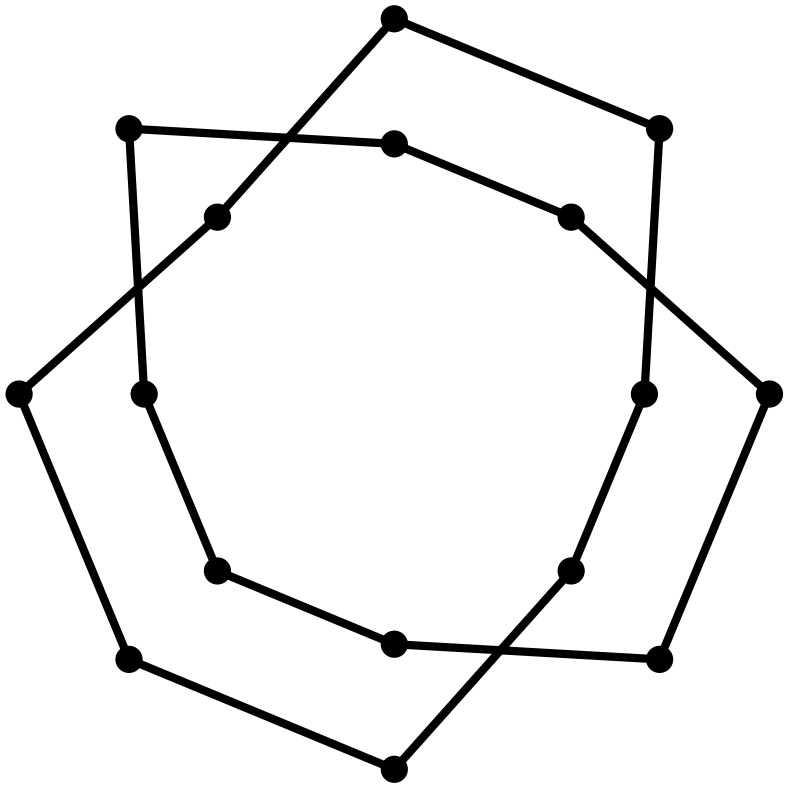
For $\{i,j\}$ insert $\begin{cases} \{i_0,j_0\},\{i_1,j_1\} & \text{if } x_{ij}=0 \\ \{i_0,j_1\},\{i_1,j_0\} & \text{o.w.} \end{cases}$



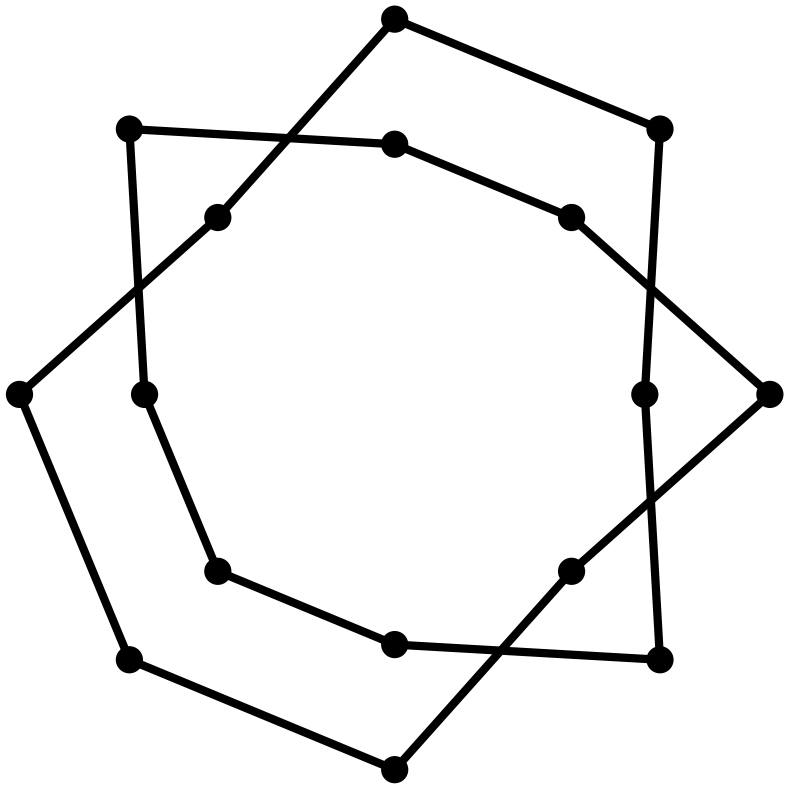


Space complexity of component finding

If parity of cycle C is odd, get two components of size ℓ



If parity of cycle C is odd, get a single component of size 2ℓ



Corollary

Component finding requires $\ell^{\Omega(\ell)}$ space

Peng-Sohler'18: component finding in $\ell^{O(\ell^3)}$ space; we show $\ell^{O(\ell)}$ space suffices, so factorial dependence is tight

Corollary

Component finding requires $\ell^{\Omega(\ell)}$ space

Peng-Sohler'18: component finding in $\ell^{O(\ell^3)}$ space; we show $\ell^{O(\ell)}$ space suffices, so factorial dependence is tight

Corollary

For a constant C generating $C4^{\ell}$ random walks requires $\ell^{\Omega(\ell)}$ space.

Kallaugher-K.-Price'22: can generate s walks to precision ε using $(1/\varepsilon)^{\ell} 2^{O(\ell^2)} s$ space

Corollary

Component finding in random order streams (??) requires $\ell^{\Omega(\ell)}$ space

Peng-Sohler'18: component finding in $\ell^{O(\ell^3)}$ space; we show $\ell^{O(\ell)}$ space suffices, so factorial dependence is tight

Corollary

For a constant C generating $C4^{\ell}$ random walks in random order streams (??) requires $\ell^{\Omega(\ell)}$ space.

Kallaugher-K.-Price'22: can generate s walks to precision ε using $(1/\varepsilon)^k 2^{O(k^2)} s$ space

Corollary

Component finding in random order streams (??) requires $\ell^{\Omega(\ell)}$ space

Peng-Sohler'18: component finding in $\ell^{O(\ell^3)}$ space; we show $\ell^{O(\ell)}$ space suffices, so factorial dependence is tight

Corollary

For a constant C generating $C4^{\ell}$ random walks in random order streams (??) requires $\ell^{\Omega(\ell)}$ space.

Kallaugher-K.-Price'22: can generate s walks to precision ε using $(1/\varepsilon)^k 2^{O(k^2)} s$ space

While arrival order in StreamingCycles is random, our reduction creates some amount of correlation...

Hidden batch random order streams

Definition (b-hidden batch random order streams)

Edges of G are partitioned into *batches* of size bounded by b. Batches are ordered randomly, and edges appear in batch-induced order (adversarial within a batch).

(Order of arrival is correlated through external events, unknown to the algorithm)

Hidden batch random order streams

Definition (b-hidden batch random order streams)

Edges of G are partitioned into *batches* of size bounded by b. Batches are ordered randomly, and edges appear in batch-induced order (adversarial within a batch).

(Order of arrival is correlated through external events, unknown to the algorithm)

Theorem

Component finding admits a $(1/\varepsilon)^{O(1/\varepsilon)} \cdot b \cdot \log n$ *space algorithm in* b*-hidden batch random order streams.*

Hidden batch random order streams

Definition (b-hidden batch random order streams)

Edges of G are partitioned into *batches* of size bounded by b. Batches are ordered randomly, and edges appear in batch-induced order (adversarial within a batch).

(Order of arrival is correlated through external events, unknown to the algorithm)

Theorem

Component finding admits a $(1/\varepsilon)^{O(1/\varepsilon)} \cdot b \cdot \log n$ *space algorithm in* b*-hidden batch random order streams.*

From streaming hardness of StreamingCycles:

Corollary

Component finding in random order streams (??) requires $\ell^{\Omega(\ell)}$ *space*

Hidden batch random order streams

Definition (b-hidden batch random order streams)

Edges of G are partitioned into *batches* of size bounded by b. Batches are ordered randomly, and edges appear in batch-induced order (adversarial within a batch).

(Order of arrival is correlated through external events, unknown to the algorithm)

Theorem

Component finding admits a $(1/\varepsilon)^{O(1/\varepsilon)} \cdot b \cdot \log n$ *space algorithm in* b*-hidden batch random order streams.*

From streaming hardness of StreamingCycles:

Corollary

Component finding in 2*-hidden batch random order streams requires* $\ell^{\Omega(\ell)}$ *space*

Space complexity of StreamingCycles

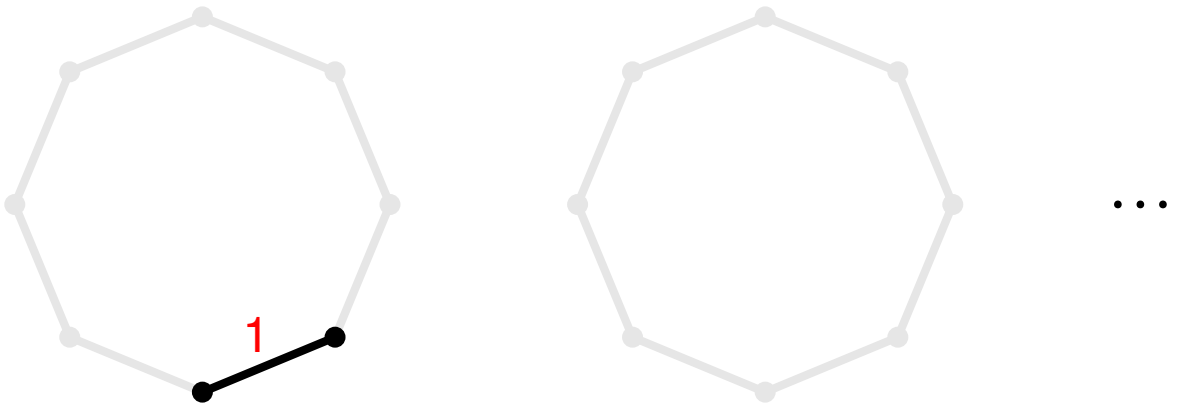
Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream

Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

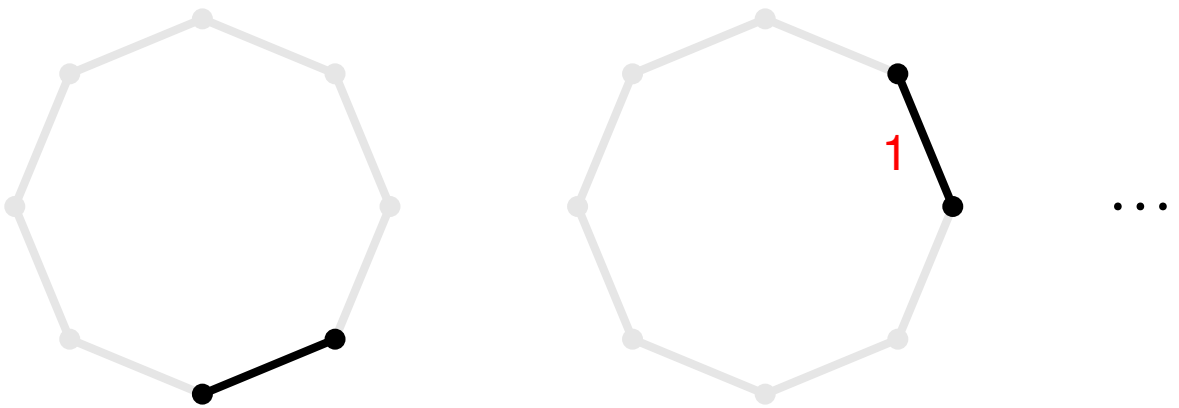
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

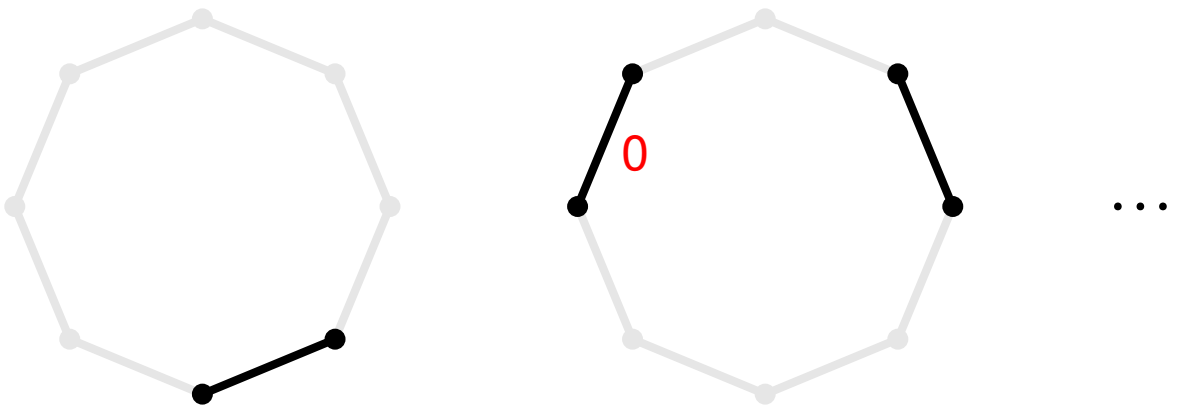
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

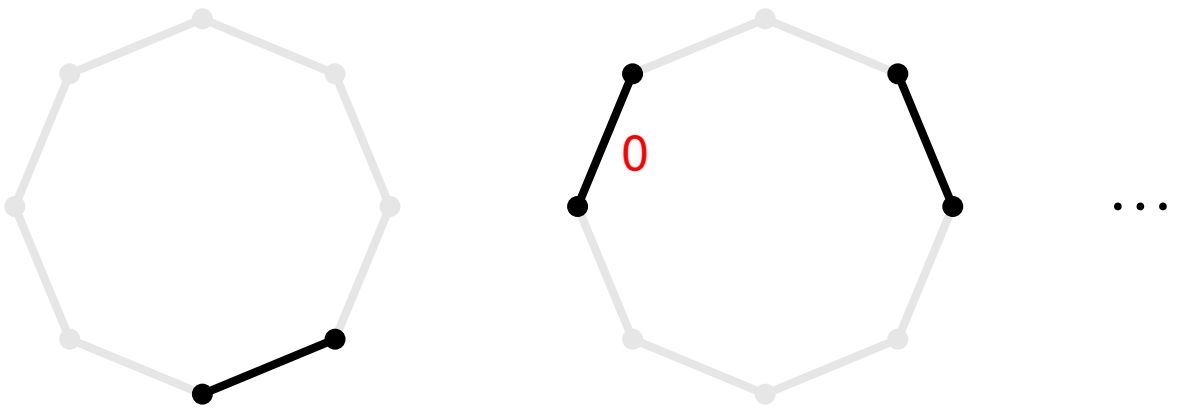
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

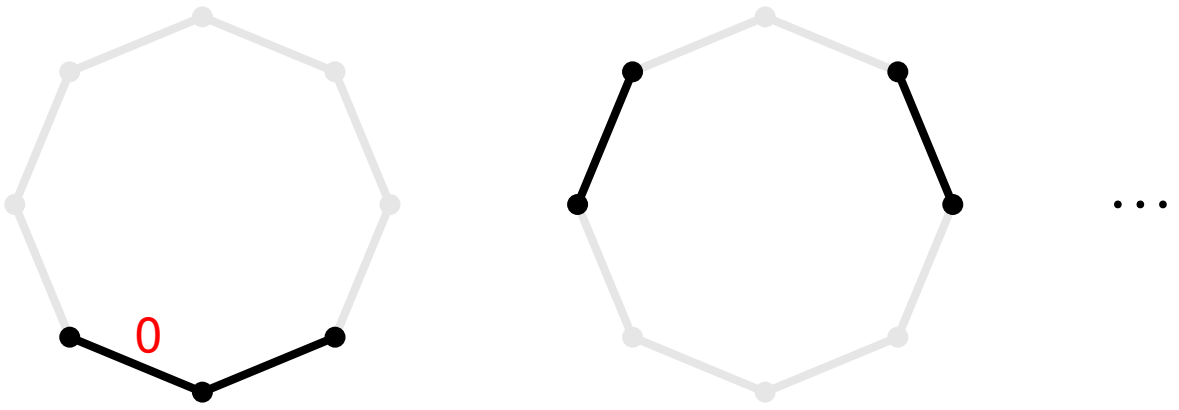
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

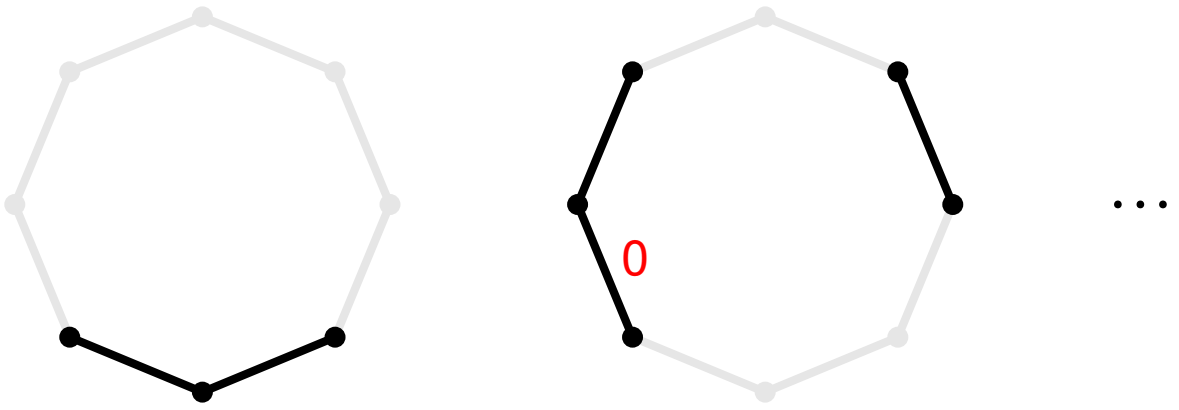
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

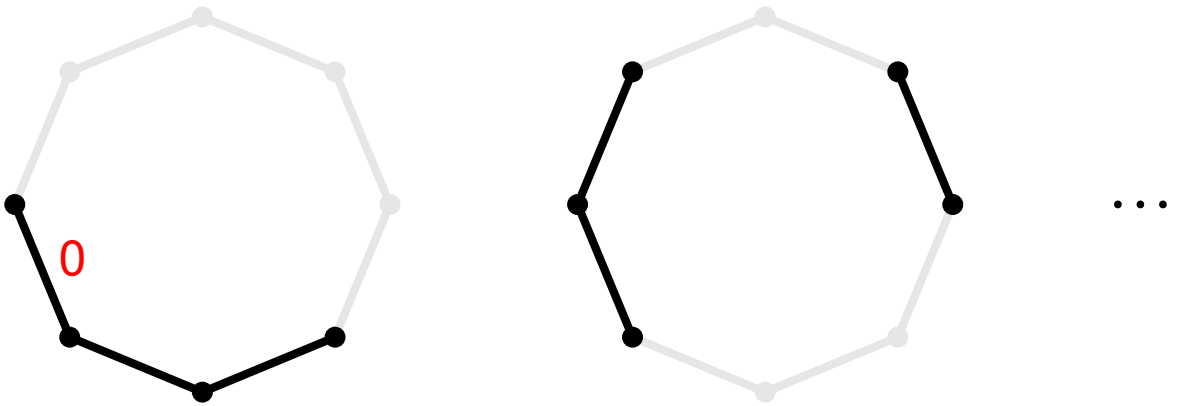
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

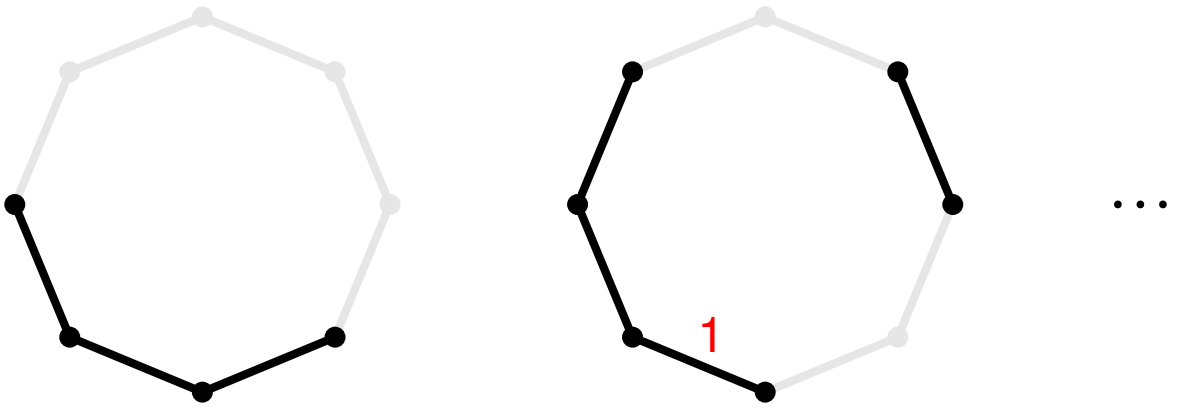
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

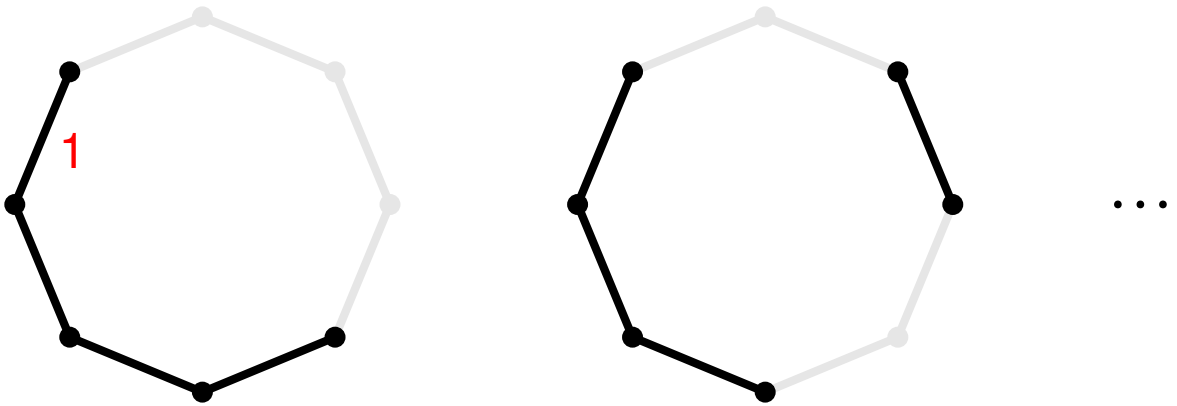
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

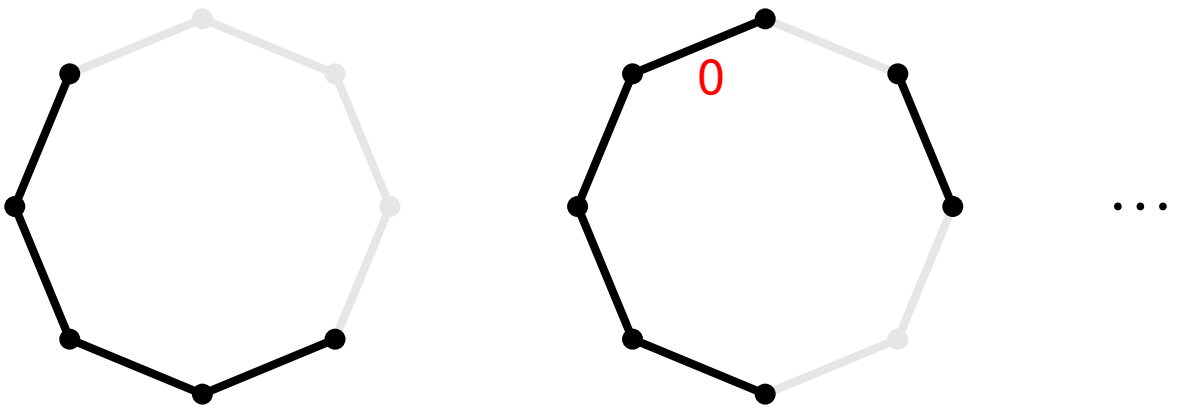
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

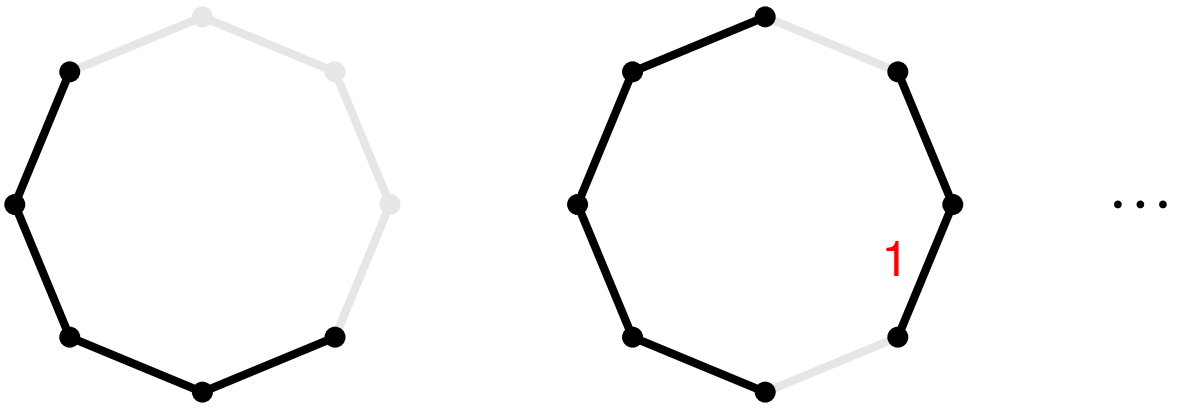
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

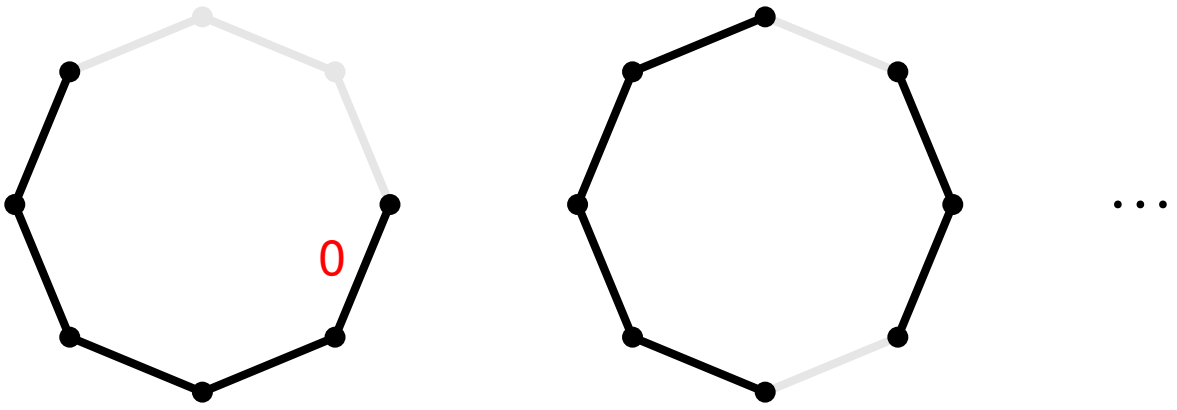
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

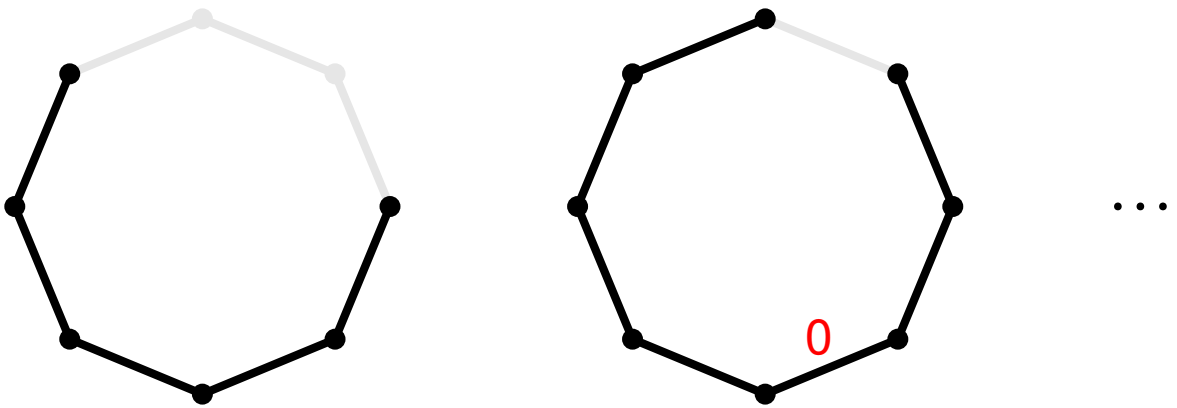
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

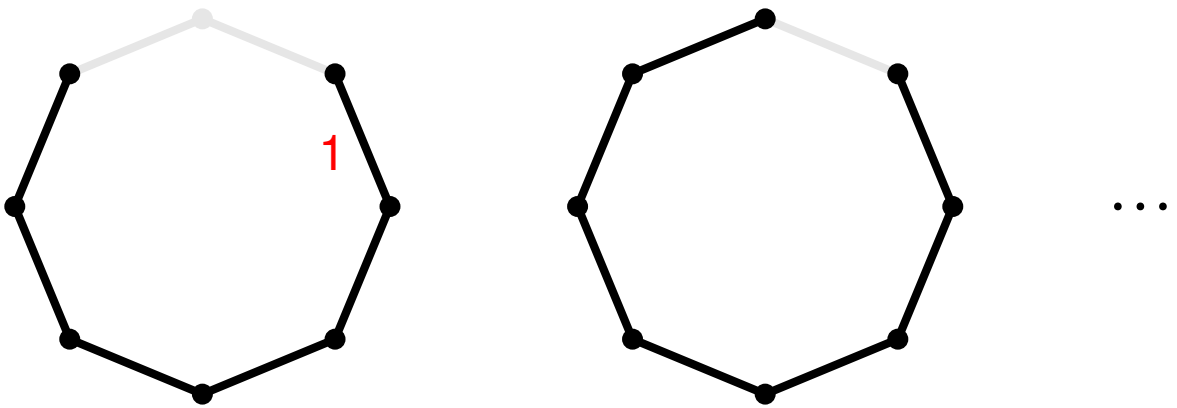
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

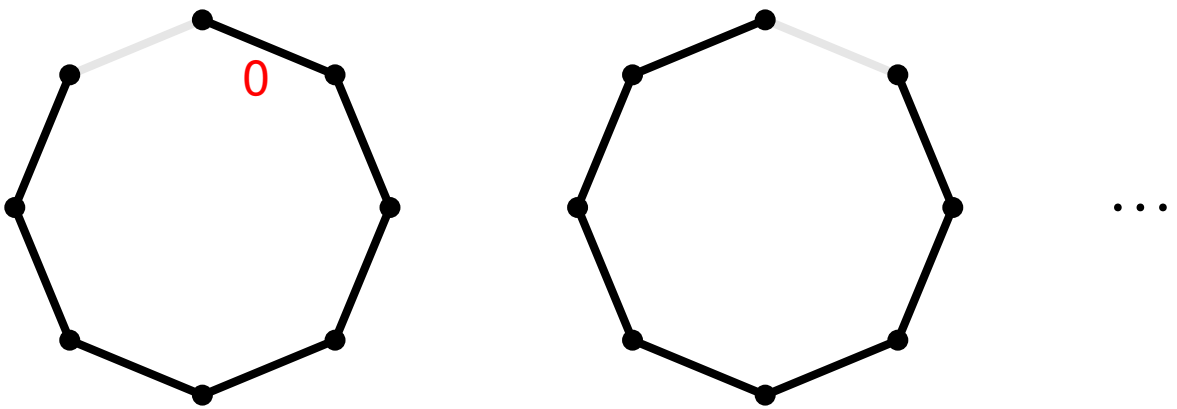
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

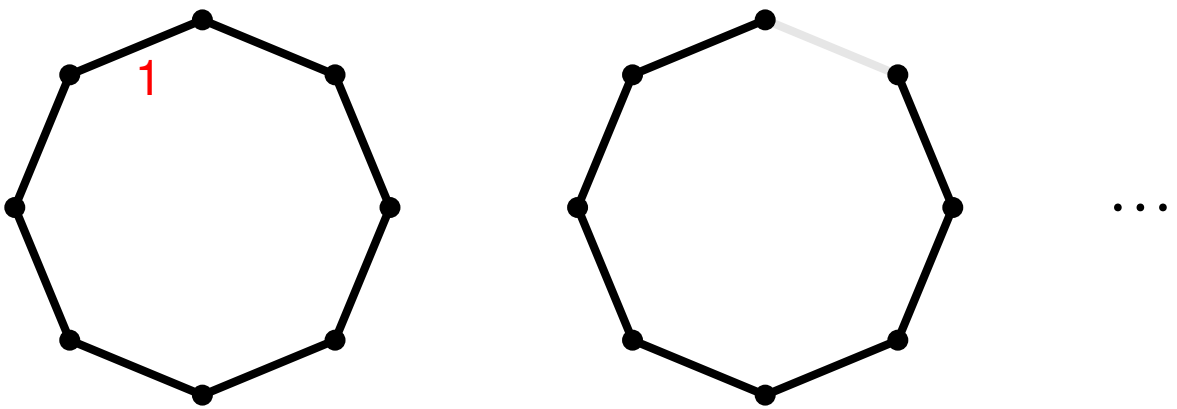
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

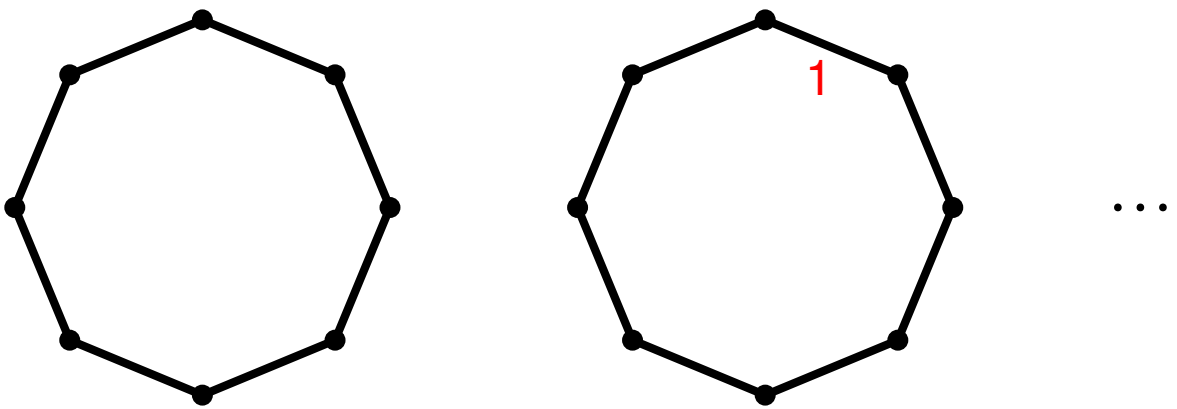
Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Space complexity of StreamingCycles

Edges of n/ℓ disjoint cycles of length ℓ presented in a stream in a random order, annotated with random bits

Goal: output $(C, \sum_{e \in C} x_e)$ for some cycle C at the end of stream



Challenges

Many recent works applying Fourier analysis to multi-party problems with a constant number of parties

Challenges

Many recent works applying Fourier analysis to multi-party problems with a constant number of parties

Cannot partition into fewer than $n^{\Omega(1/\ell)}$ players, otherwise reveal answer

Challenges

Many recent works applying Fourier analysis to multi-party problems with a constant number of parties

Cannot partition into fewer than $n^{\Omega(1/\ell)}$ players, otherwise reveal answer

Cannot lose even polynomially in the number of players ($n^{1/\ell} \gg \ell^{\ell}$ when $\ell = \mathrm{poly}(\log\log n)$, say)

Challenges

Many recent works applying Fourier analysis to multi-party problems with a constant number of parties

Cannot partition into fewer than $n^{\Omega(1/\ell)}$ players, otherwise reveal answer

Cannot lose even polynomially in the number of players ($n^{1/\ell} \gg \ell^{\ell}$ when $\ell = \mathrm{poly}(\log\log n)$, say)

Analyse algorithm's knowledge on a 'per edge' basis

Open problems

Approximate MAX-CUT value in $n^{0.99}$ queries to the graph? Or in $n^{0.99}$ space in a $\mathrm{poly}(\log n)$ number of passes?

Open problems

Approximate MAX-CUT value in $n^{0.99}$ queries to the graph? Or in $n^{0.99}$ space in a $\text{poly}(\log n)$ number of passes?

Does matching size estimation require $\Omega(\log^2 n)$ space?

Open problems

Approximate MAX-CUT value in $n^{0.99}$ queries to the graph? Or in $n^{0.99}$ space in a $\mathrm{poly}(\log n)$ number of passes?

Does matching size estimation require $\Omega(\log^2 n)$ space?

Fourier analytic lower bounds where sketching is/should be optimal (e.g., lower bounds for spanners in sketching)?

Open problems

Approximate MAX-CUT value in $n^{0.99}$ queries to the graph? Or in $n^{0.99}$ space in a $\text{poly}(\log n)$ number of passes?

Does matching size estimation require $\Omega(\log^2 n)$ space?

Fourier analytic lower bounds where sketching is/should be optimal (e.g., lower bounds for spanners in sketching)?

Thank you!

Factorial Lower Bounds for (Almost) Random Order Streams

(joint work with Ashish Chiplunkar, John Kallaugher and Eric Price)