Homomorphic Sketches

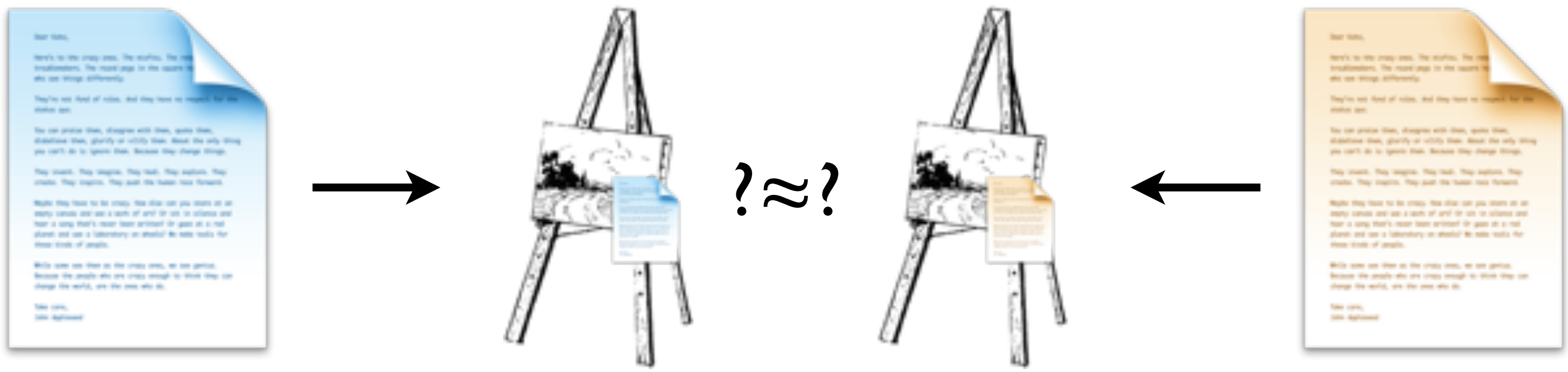
Shrinking Big Data without Sacrificing Structure

Andrew McGregor
University of Massachusetts

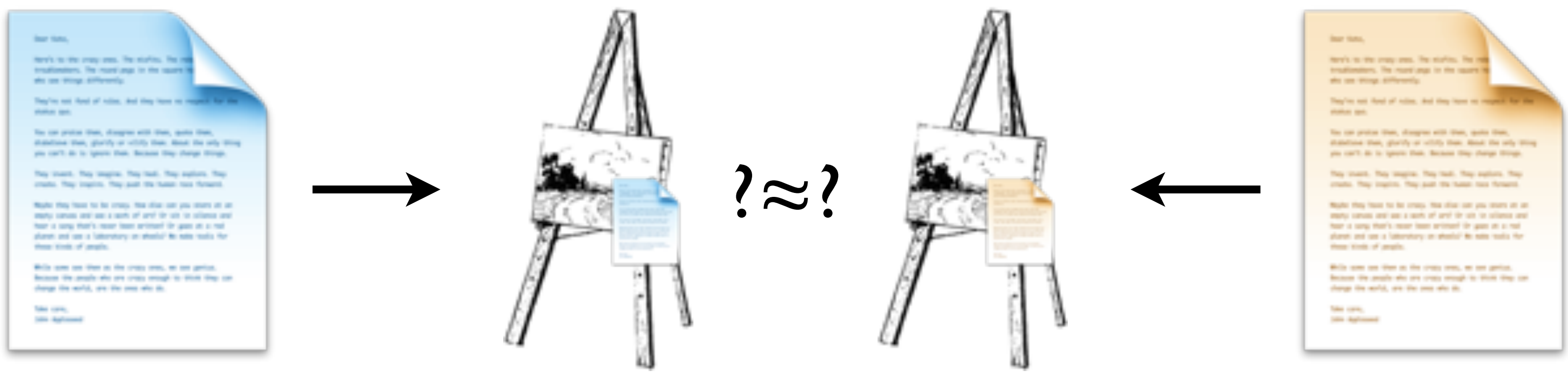
Can test whether two n bit files are identical by comparing O(log n) bit *fingerprints* of each file.

Can test whether two n bit files are identical by comparing O(log n) bit *fingerprints* of each file.

Can test whether two n bit files are identical by comparing O(log n) bit *fingerprints* of each file.



More generally, can construct *sketches* of files to estimate Hamming distance between the files.



More generally, can construct *sketches* of files to estimate Hamming distance between the files.

Many results such as distinct elements, entropy, frequency moments, quantiles, histograms, linear regression, clustering, shape approximation...

Basic Idea: Treat file as vector; use *linear projections* to reduce dimension while preserving properties.

Extensive theory with connections to compressed sensing, metric embeddings; *widely applicable* since parallelizable and suitable for stream processing.

Most existing work concerns numerical statistics of data such as frequency and feature vectors...

$$= \begin{bmatrix} \\ v \\ \\ \end{bmatrix}$$

Basic Idea: Treat file as vector; use *linear projections* to reduce dimension while preserving properties.

Extensive theory with connections to compressed sensing, metric embeddings; *widely applicable* since parallelizable and suitable for stream processing.

Most existing work concerns numerical statistics of data such as frequency and feature vectors...

$$\begin{bmatrix} \\ \\ v \\ \\ \\ \end{bmatrix}$$

Basic Idea: Treat file as vector; use *linear projections* to reduce dimension while preserving properties.

Extensive theory with connections to compressed sensing, metric embeddings; *widely applicable* since parallelizable and suitable for stream processing.

Most existing work concerns numerical statistics of data such as frequency and feature vectors...

$$\begin{bmatrix} & & \\ & M & \\ & & \end{bmatrix} \begin{bmatrix} \\ \\ v \\ \\ \\ \end{bmatrix}$$

Basic Idea: Treat file as vector; use *linear projections* to reduce dimension while preserving properties.

Extensive theory with connections to compressed sensing, metric embeddings; *widely applicable* since parallelizable and suitable for stream processing.

Most existing work concerns numerical statistics of data such as frequency and feature vectors...

$$\begin{bmatrix} & & M & & \end{bmatrix} \begin{bmatrix} \\ v \\ \\ \end{bmatrix} = \begin{bmatrix} Mv \end{bmatrix} =$$

Basic Idea: Treat file as vector; use *linear projections* to reduce dimension while preserving properties.

Extensive theory with connections to compressed sensing, metric embeddings; *widely applicable* since parallelizable and suitable for stream processing.

Most existing work concerns numerical statistics of data such as frequency and feature vectors...

$$\begin{bmatrix} & & M & & \end{bmatrix} \begin{bmatrix} \\ v \\ \\ \end{bmatrix} = \begin{bmatrix} Mv \end{bmatrix} =$$

Basic Idea: Treat file as vector; use *linear projections* to reduce dimension while preserving properties.

Extensive theory with connections to compressed sensing, metric embeddings; *widely applicable* since parallelizable and suitable for stream processing.

$$\begin{bmatrix} & M & \end{bmatrix}\begin{bmatrix} \\ v \\ \\ \end{bmatrix} = \begin{bmatrix} Mv \end{bmatrix} =$$

Basic Idea: Treat file as vector; use *linear projections* to reduce dimension while preserving properties.

Extensive theory with connections to compressed sensing, metric embeddings; *widely applicable* since parallelizable and suitable for stream processing.

Most existing work concerns numerical statistics of data such as frequency and feature vectors...

Is it possible to analyze richer combinatorial and group-theoretic structure via linear sketches?

Is it possible to analyze richer combinatorial and group-theoretic structure via linear sketches?

Can we make compression "homomorphic" and run algorithms on sketched data?

Is it possible to analyze richer combinatorial and group-theoretic structure via linear sketches?

Can we make compression "homomorphic" and run algorithms on sketched data?

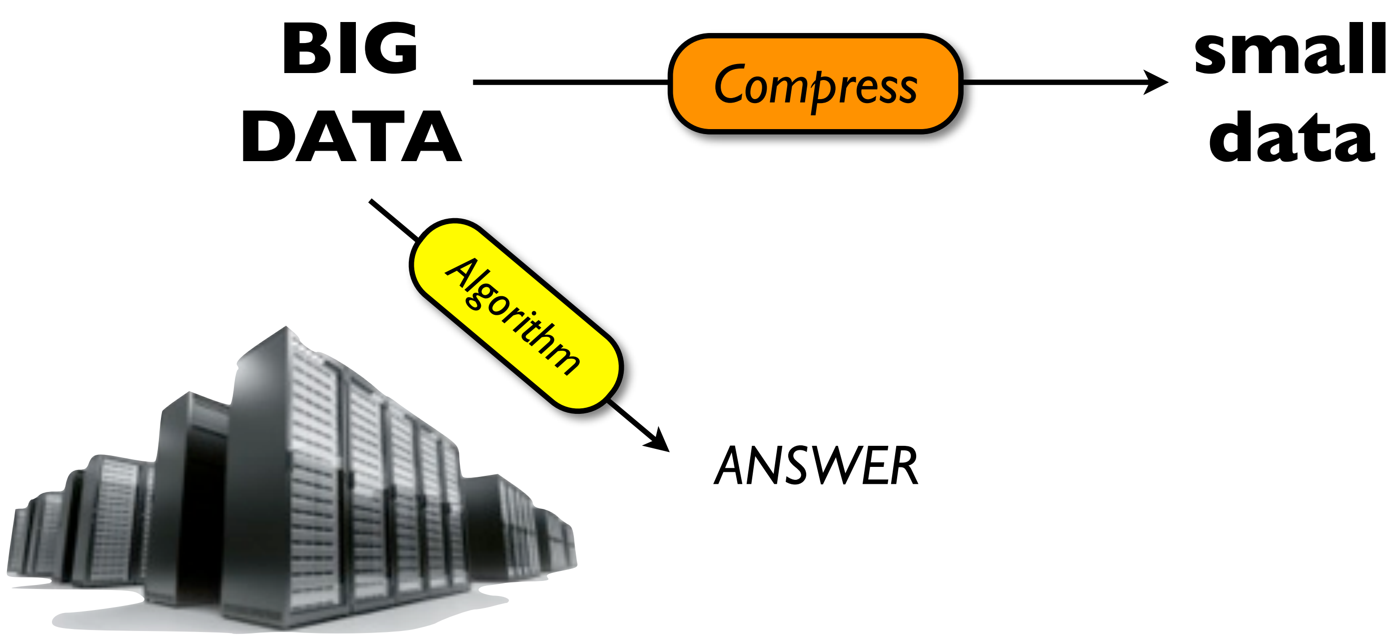
BIG DATA

Is it possible to analyze richer combinatorial and group-theoretic structure via linear sketches?

Can we make compression "homomorphic" and run algorithms on sketched data?

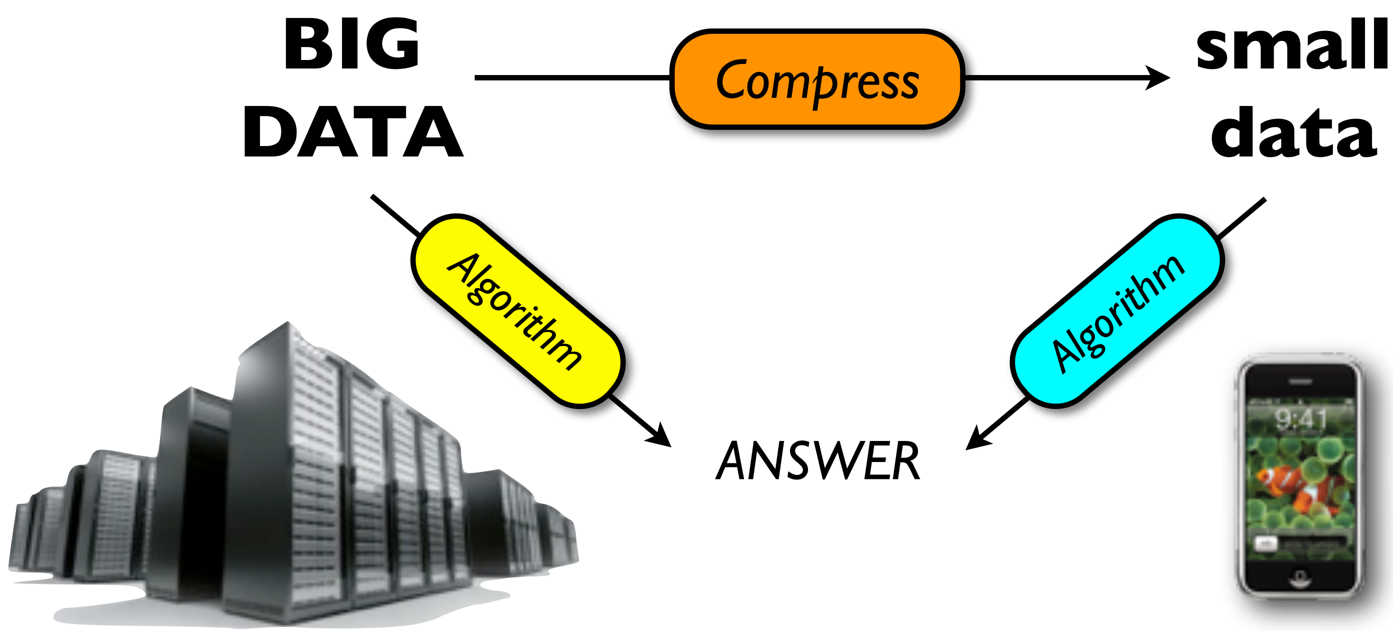
Is it possible to analyze richer combinatorial and group-theoretic structure via linear sketches?

Can we make compression "homomorphic" and run algorithms on sketched data?



Is it possible to analyze richer combinatorial and group-theoretic structure via linear sketches?

Can we make compression "homomorphic" and run algorithms on sketched data?



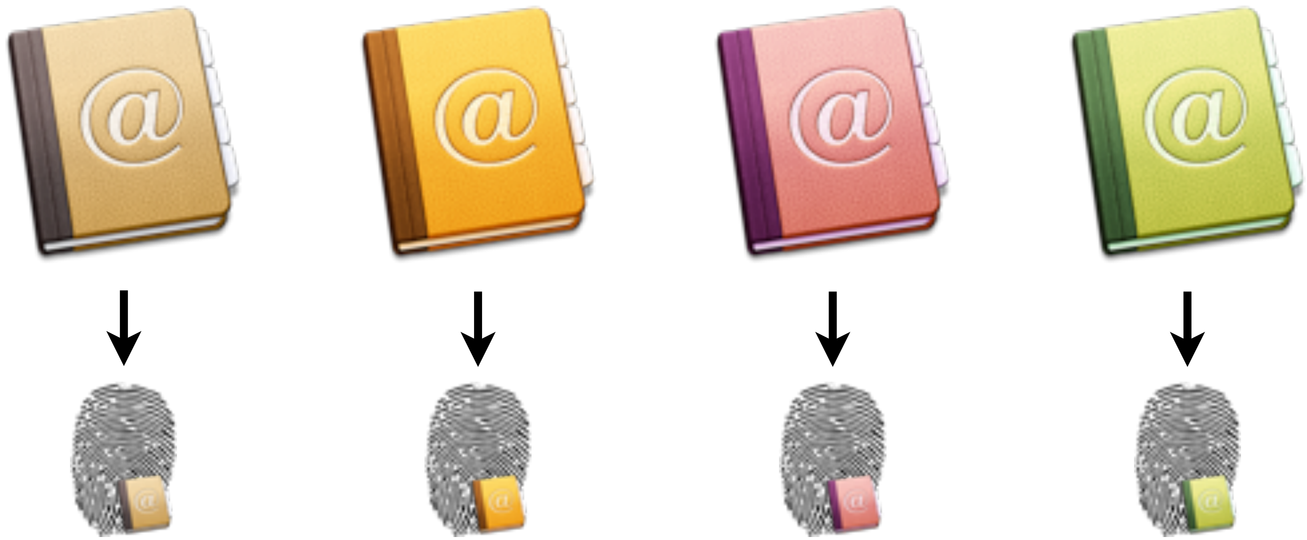
First Result...

Problem: Fingerprint each row of nxn adjacency matrix such that we can check connectivity using fingerprints.

Theorem: Fingerprints of size O(polylog n) bit suffice!

Surprising? Seems impossible if there are bridge edges.

First Result...

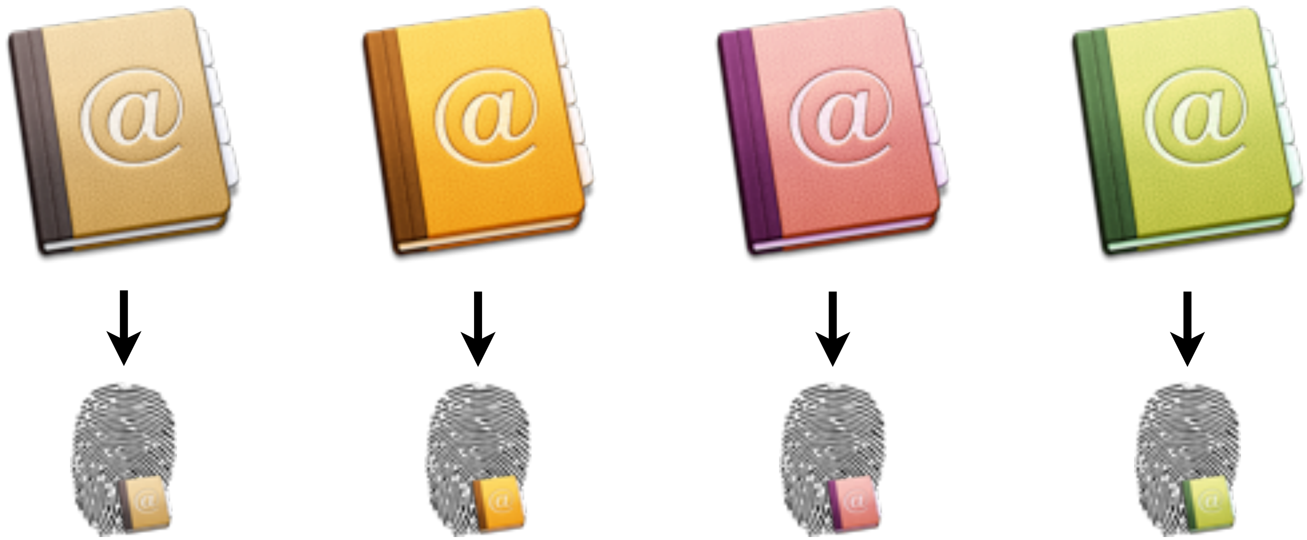


Problem: Fingerprint each row of nxn adjacency matrix such that we can check connectivity using fingerprints.

Theorem: Fingerprints of size O(polylog n) bit suffice!

Surprising? Seems impossible if there are bridge edges.

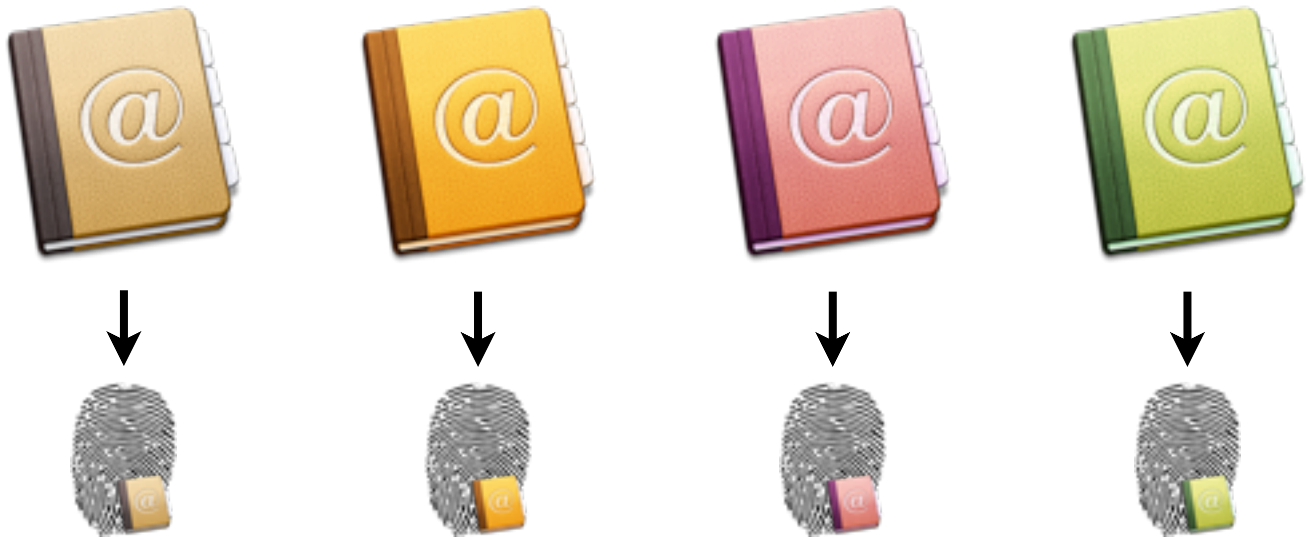
First Result...



Problem: Fingerprint each row of nxn adjacency matrix such that we can check connectivity using fingerprints.

Theorem: Fingerprints of size O(polylog n) bit suffice!

First Result...

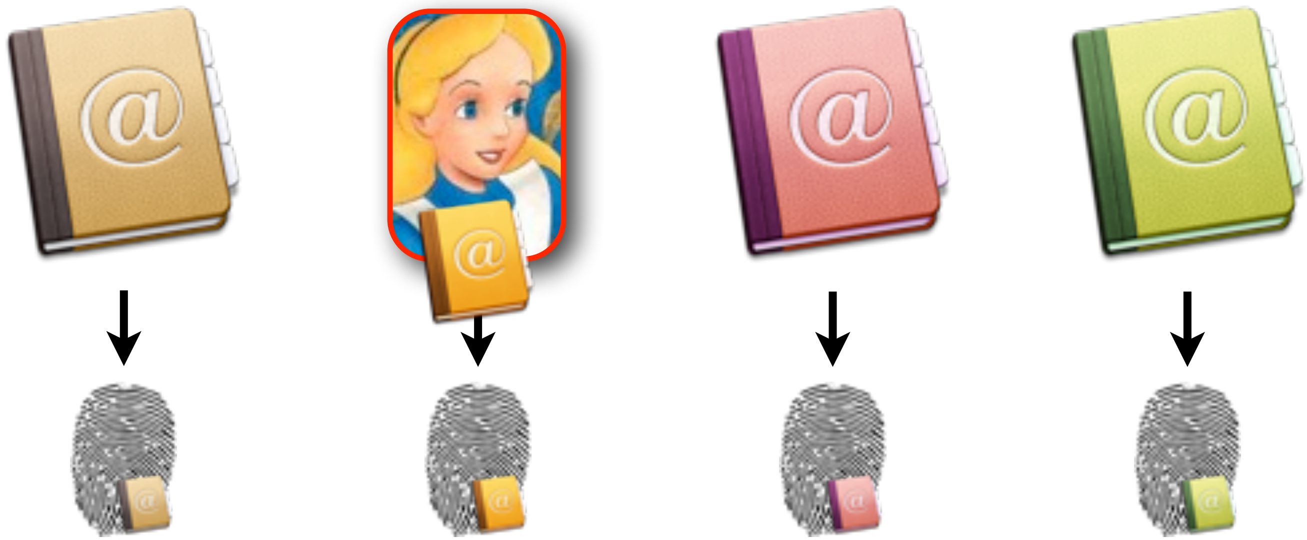


Problem: Fingerprint each row of nxn adjacency matrix such that we can check connectivity using fingerprints.

Theorem: Fingerprints of size O(polylog n) bit suffice!

Surprising? Seems impossible if there are bridge edges.

First Result...

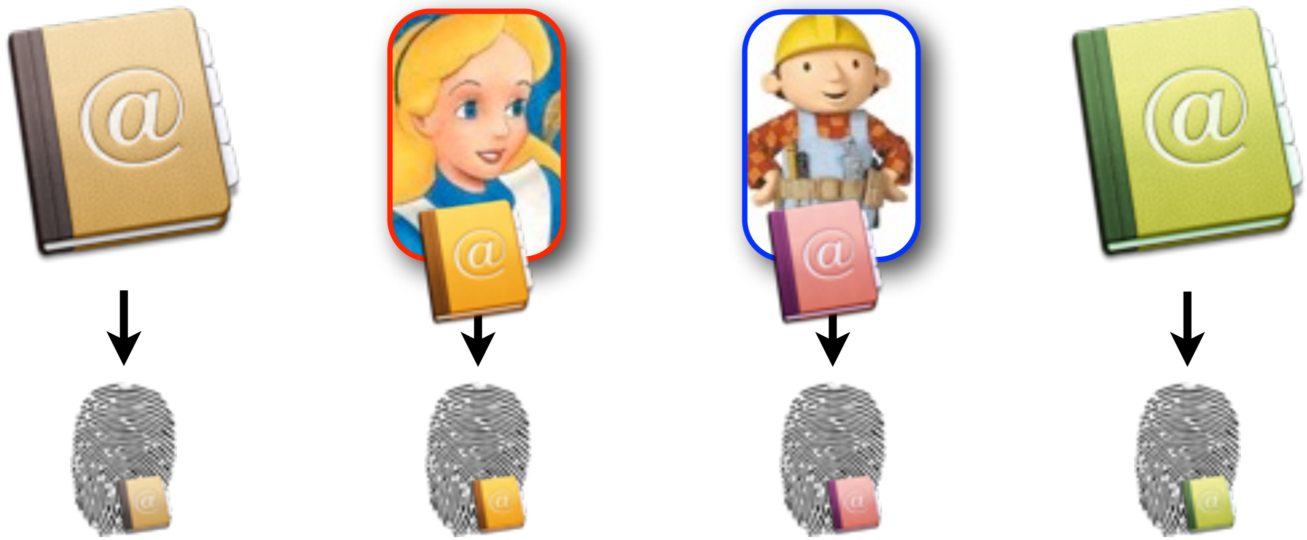


Problem: Fingerprint each row of nxn adjacency matrix such that we can check connectivity using fingerprints.

Theorem: Fingerprints of size O(polylog n) bit suffice!

Surprising? Seems impossible if there are bridge edges.

First Result...

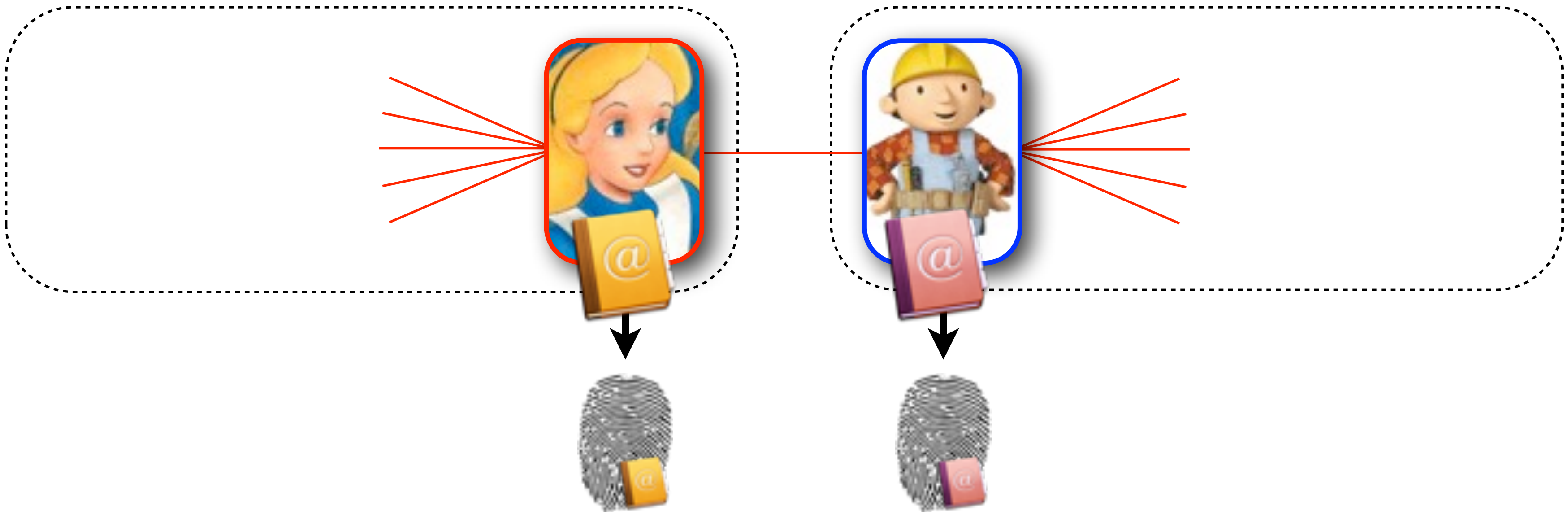


Problem: Fingerprint each row of nxn adjacency matrix such that we can check connectivity using fingerprints.

Theorem: Fingerprints of size O(polylog n) bit suffice!

Surprising? Seems impossible if there are bridge edges.

First Result...



Problem: Fingerprint each row of nxn adjacency matrix such that we can check connectivity using fingerprints.

Theorem: Fingerprints of size O(polylog n) bit suffice!

Surprising? Seems impossible if there are bridge edges.

Second Result...

Second Result...

"The quick brown
fox jumped
over the lazy dog."

"quick brown fox
jumped over the
lazy dog. The"

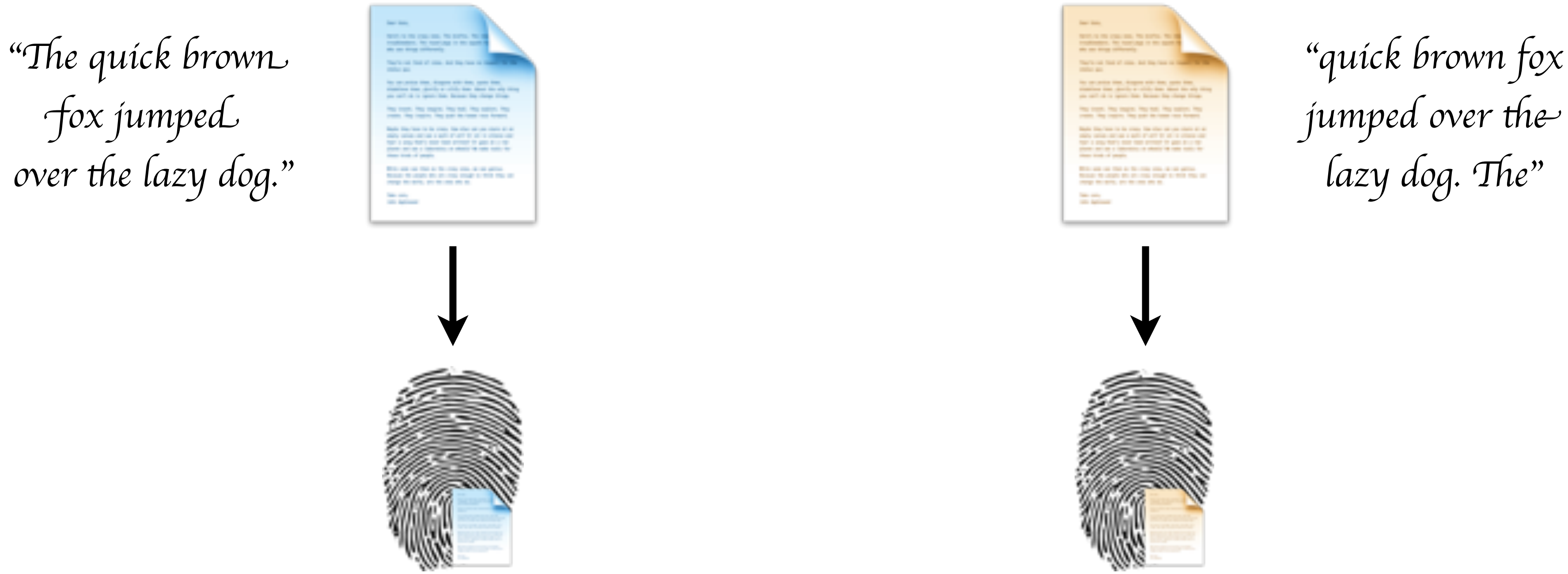
Second Result...

"The quick brown fox jumped over the lazy dog."

"quick brown fox jumped over the lazy dog. The"

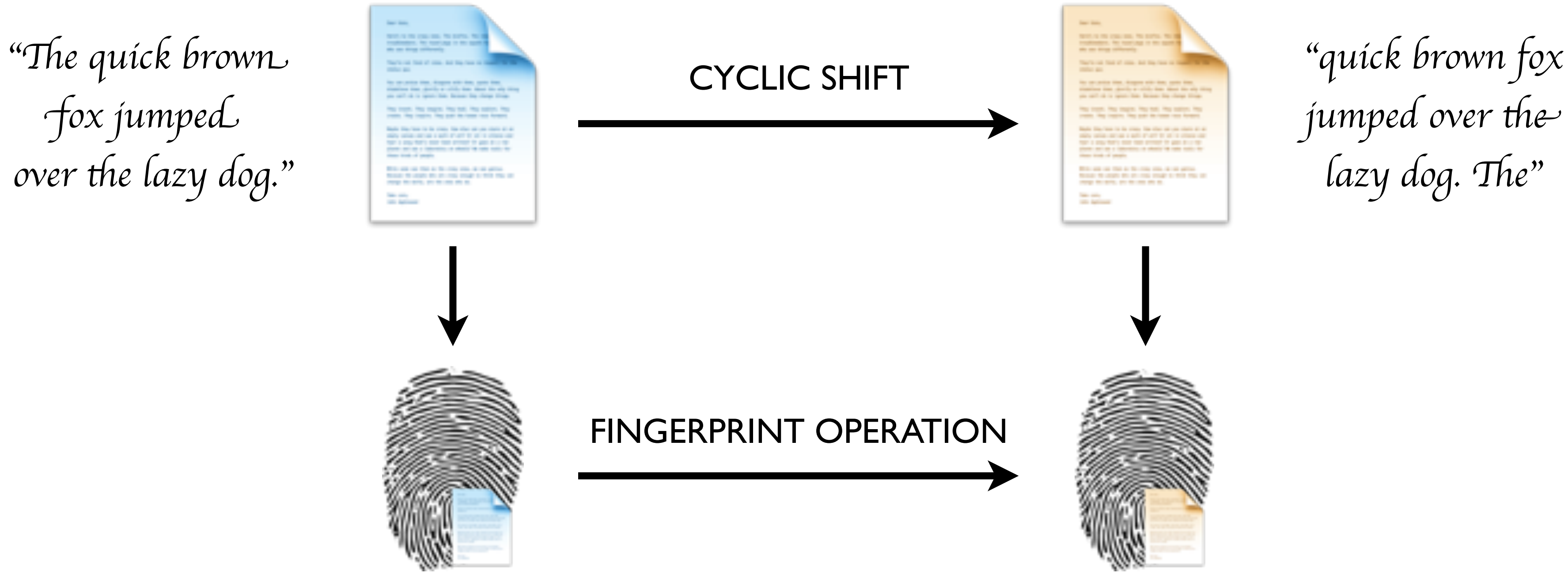
Problem: Fingerprint files such that we can test if files are close under some cyclic rotation.

Second Result...



Problem: Fingerprint files such that we can test if files are close under some cyclic rotation.

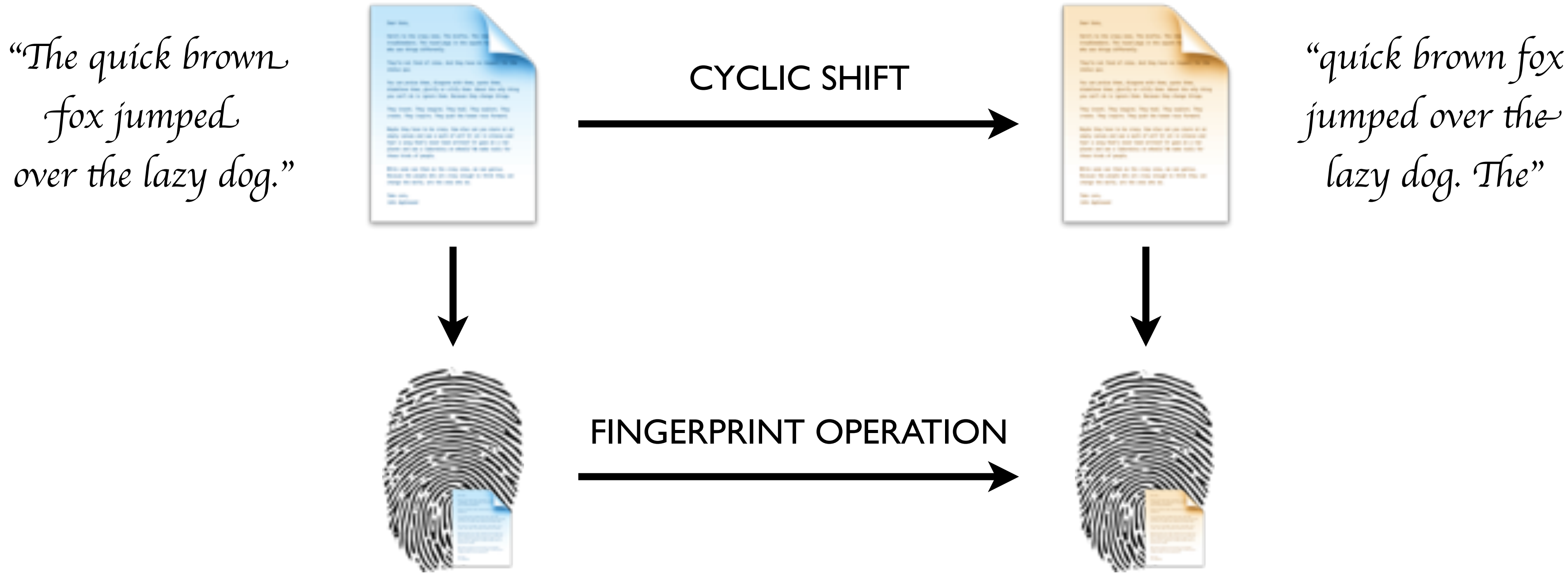
Second Result...



Problem: Fingerprint files such that we can test if files are close under some cyclic rotation.

Theorem: Fingerprints of size ≈ D(n) bits suffice where D(n) is the number of divisors of n.

Second Result...



Problem: Fingerprint files such that we can test if files are close under some cyclic rotation.

Theorem: Fingerprints of size ≈ D(n) bits suffice where D(n) is the number of divisors of n.

Surprising? Fingerprint size isn't monotonic in file size!

I. **Connectivity**

II. **Misalignment**

I. **Connectivity**

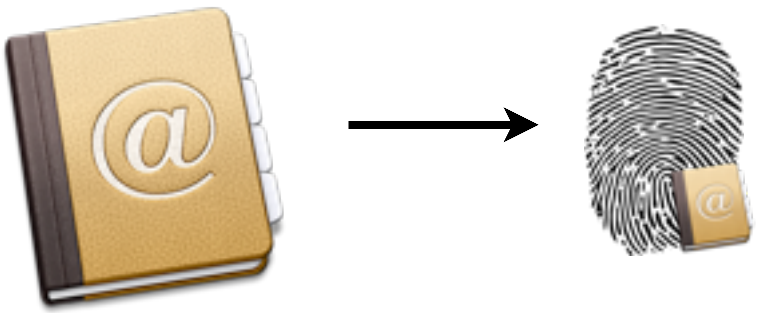
II. **Misalignment**

a) Connectivity via O(polylog n) bit Fingerprints

b) Extension to Estimating Cuts and Eigenvalues

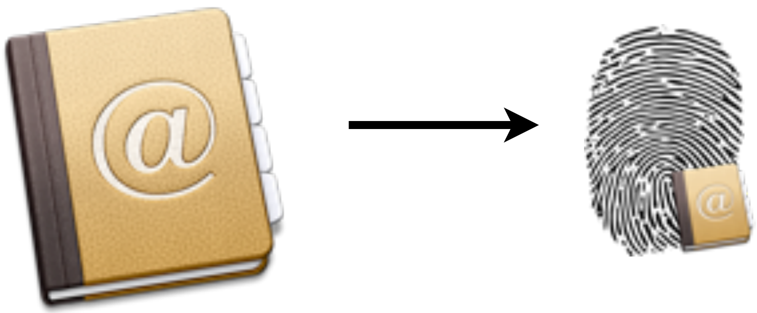
Joint work with Kook Jin Ahn and Sudipto Guha

Sketches for Connectivity



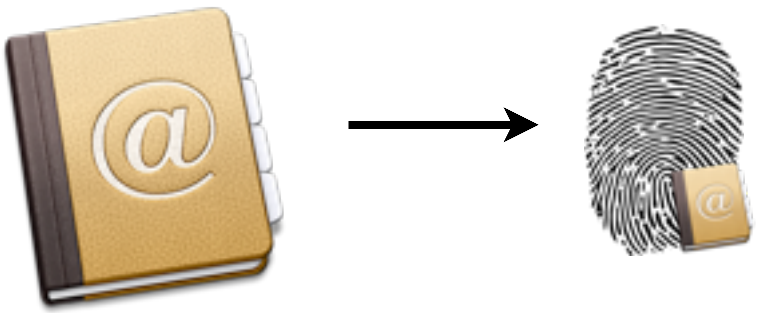
- *Theorem:* Can check k-connectivity w.h.p. using O(k polylog n) bit fingerprint of each adjacency list.

Sketches for Connectivity



- *Theorem:* Can check k-connectivity w.h.p. using O(k polylog n) bit fingerprint of each adjacency list.
- *Corollary:* Can monitor connectivity in a dynamic graph stream where edges are both inserted and deleted.

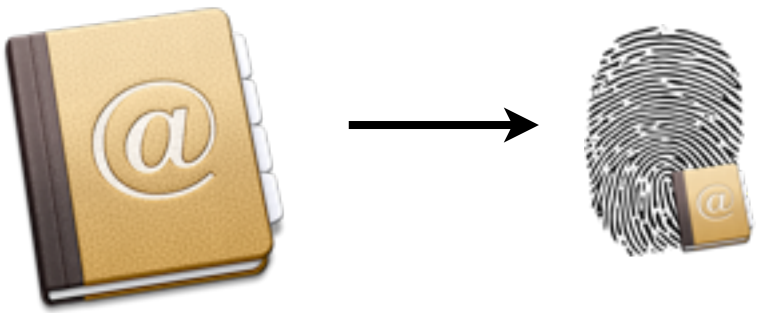
Sketches for Connectivity



- *Theorem:* Can check k-connectivity w.h.p. using O(k polylog n) bit fingerprint of each adjacency list.
- *Corollary:* Can monitor connectivity in a dynamic graph stream where edges are both inserted and deleted.
- Previous stream work assumed no edge deletions.

e.g., [Feigenbaum, Kannan, McGregor, Suri, Zhang 2004, 2005], [McGregor 2005] [Jowhari, Ghodsi 2005], [Zelke 2008], [Sarma, Gollapudi, Panigrahy 2008, 2009] [Ahn, Guha 2009, 2011], [Konrad, Magniez, Mathieu 2012], [Goel, Kapralov, Khanna 2012]

Sketches for Connectivity



- *Theorem:* Can check k-connectivity w.h.p. using O(k polylog n) bit fingerprint of each adjacency list.
- *Corollary:* Can monitor connectivity in a dynamic graph stream where edges are both inserted and deleted.
- Previous stream work assumed no edge deletions.

e.g., [Feigenbaum, Kannan, McGregor, Suri, Zhang 2004, 2005], [McGregor 2005]
[Jowhari, Ghodsi 2005], [Zelke 2008], [Sarma, Gollapudi, Panigrahy 2008, 2009]
[Ahn, Guha 2009, 2011], [Konrad, Magniez, Mathieu 2012], [Goel, Kapralov, Khanna 2012]

- New sliding window graph results presented yesterday.

[Crouch, McGregor, Stubbs 2013]

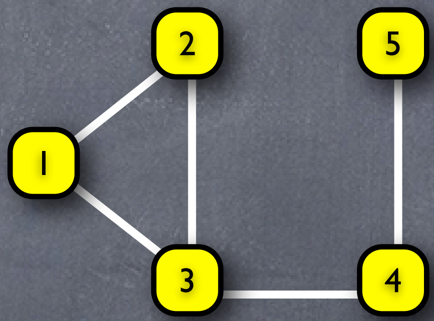
Basic Primitive: Neighborhood Sketches

Basic Primitive: Neighborhood Sketches

- Defn: Let a_i be i^{th} row of signed vertex-edge matrix

Basic Primitive: Neighborhood Sketches

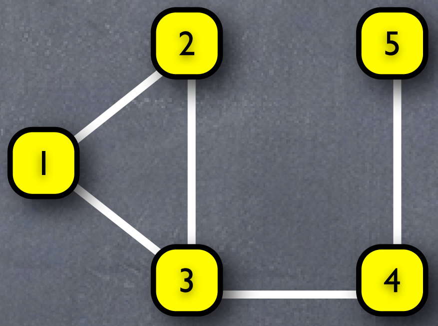
- Defn: Let a_i be i^{th} row of signed vertex-edge matrix



Basic Primitive: Neighborhood Sketches

- Defn: Let a_i be i^{th} row of signed vertex-edge matrix

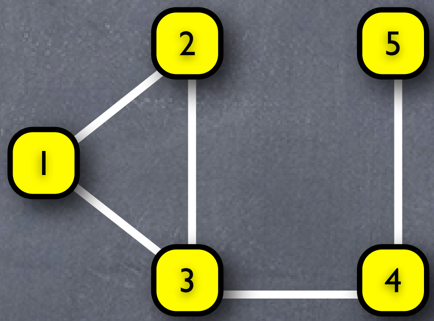
$$
\begin{array}{cccccccccccc}
 & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\
\mathbf{a}_1 = & (1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0)
\end{array}
$$



Basic Primitive: Neighborhood Sketches

- **Defn:** Let a_i be i^{th} row of signed vertex-edge matrix

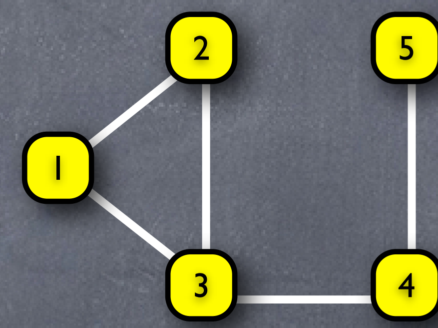
$$
\begin{array}{rcccccccccc}
 & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\
\mathbf{a}_1 = & \left(1 \right. & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left. 0 \right) \\
\mathbf{a}_2 = & \left(-1 \right. & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \left. 0 \right)
\end{array}
$$



Basic Primitive: Neighborhood Sketches

- **Defn:** Let a_i be i^{th} row of signed vertex-edge matrix

$$
\begin{array}{ccccccccccc}
 & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\
\mathbf{a}_1 = & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{a}_2 = & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

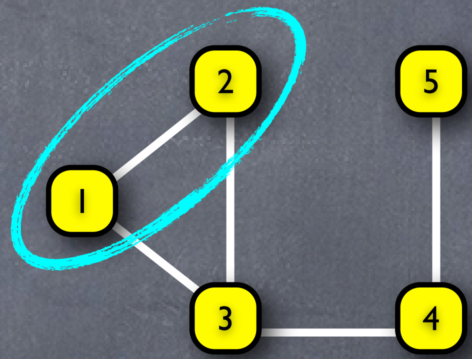


For $S \subset V$, non-zero entries of $\sum_{i \in S} a_i$ equals $E(S, V \setminus S)$

Basic Primitive: Neighborhood Sketches

- Defn: Let a_i be i^{th} row of signed vertex-edge matrix

$$
\begin{array}{ccccccccccc}
 & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\
\mathbf{a}_1 = & \left(\begin{matrix} 1 \end{matrix}\right. & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left.\begin{matrix} 0 \end{matrix}\right) \\
\mathbf{a}_2 = & \left(\begin{matrix} -1 \end{matrix}\right. & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \left.\begin{matrix} 0 \end{matrix}\right)
\end{array}
$$

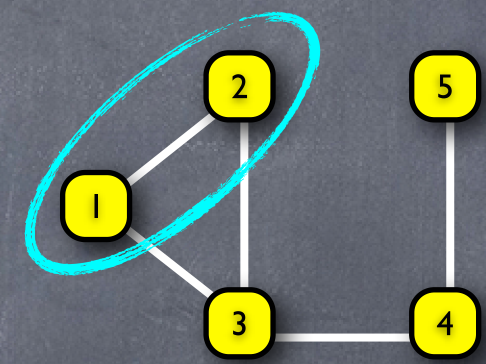


For $S \subset V$, non-zero entries of $\sum_{i \in S} a_i$ equals $E(S, V \setminus S)$

Basic Primitive: Neighborhood Sketches

- Defn: Let a_i be i^{th} row of signed vertex-edge matrix

$$
\begin{array}{rc}
 & \begin{array}{cccccccccc} \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \end{array} \\
\mathbf{a}_1 = & \left(\begin{array}{cccccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \\
\mathbf{a}_2 = & \left(\begin{array}{cccccccccc} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \\
\mathbf{a}_1 + \mathbf{a}_2 = & \left(\begin{array}{cccccccccc} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right)
\end{array}
$$

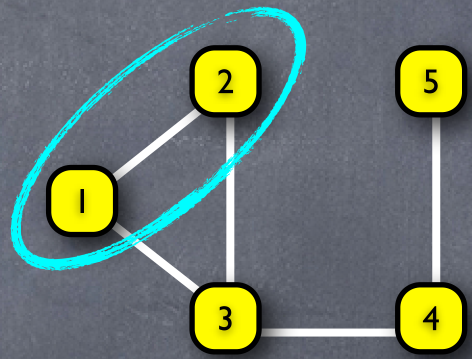


For $S \subset V$, non-zero entries of $\sum_{i \in S} a_i$ equals $E(S, V \setminus S)$

Basic Primitive: Neighborhood Sketches

- Defn: Let a_i be i^{th} row of signed vertex-edge matrix

$$
\begin{array}{rl}
 & \begin{array}{cccccccccc} \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \end{array} \\
\mathbf{a}_1 = & \left(\begin{array}{cccccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \\
\mathbf{a}_2 = & \left(\begin{array}{cccccccccc} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \\
\mathbf{a}_1 + \mathbf{a}_2 = & \left(\begin{array}{cccccccccc} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right)
\end{array}
$$



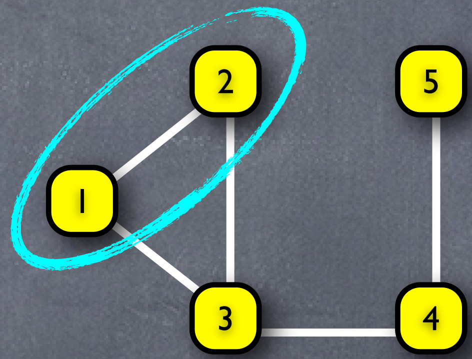
For $S \subset V$, non-zero entries of $\sum_{i \in S} a_i$ equals $E(S, V \setminus S)$

- Fingerprint: Ma_i where M is $\tilde{O}(k)$ dim. proj. such that k non-zero entries of any x can be recovered from Mx.

Basic Primitive: Neighborhood Sketches

- Defn: Let a_i be i^{th} row of signed vertex-edge matrix

$$
\begin{array}{rccccccccccc}
 & \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \\
\mathbf{a}_1 = & (\; 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \;) \\
\mathbf{a}_2 = & (\; -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \;) \\
\mathbf{a}_1 + \mathbf{a}_2 = & (\; 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \;)
\end{array}
$$



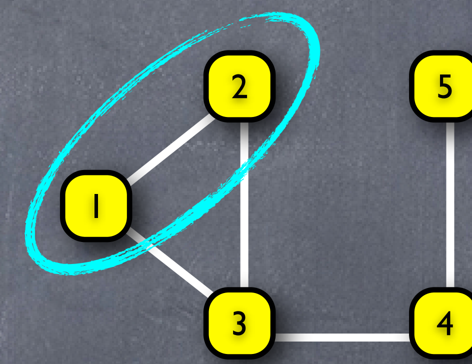
For $S \subset V$, non-zero entries of $\sum_{i \in S} a_i$ equals $E(S, V \setminus S)$

- Fingerprint: Ma_i where M is $\tilde{O}(k)$ dim. proj. such that k non-zero entries of any x can be recovered from Mx.
- Utility: Can find min(all,k) edges across any cut S. Call the set of recovered edges a "k-skeleton".

Basic Primitive: Neighborhood Sketches

- Defn: Let a_i be i^{th} row of signed vertex-edge matrix

$$
\begin{array}{rl}
 & \begin{array}{cccccccccc} \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \end{array} \\
\mathbf{a}_1 = & \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{a}_2 = & \begin{pmatrix} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{a}_1 + \mathbf{a}_2 = & \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}
\end{array}
$$



For $S \subset V$, non-zero entries of $\sum_{i \in S} a_i$ equals $E(S, V \setminus S)$

- Fingerprint: Ma_i where M is $\tilde{O}(k)$ dim. proj. such that k non-zero entries of any x can be recovered from Mx.

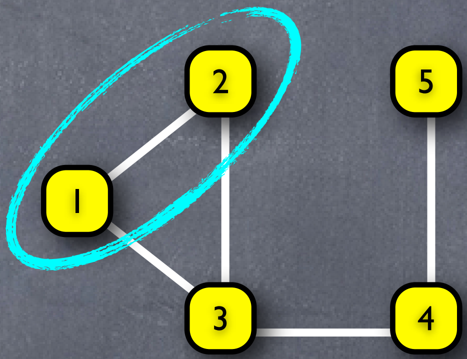
- Utility: Can find min(all,k) edges across any cut S. Call the set of recovered edges a "k-skeleton".

$$\sum_{j \in S} M\mathbf{a}_j = M\Big(\sum_{j \in S} \mathbf{a}_j\Big)$$

Basic Primitive: Neighborhood Sketches

- Defn: Let a_i be i^{th} row of signed vertex-edge matrix

$$
\begin{array}{rc}
 & \begin{array}{cccccccccc} \{1,2\} & \{1,3\} & \{1,4\} & \{1,5\} & \{2,3\} & \{2,4\} & \{2,5\} & \{3,4\} & \{3,5\} & \{4,5\} \end{array} \\
\mathbf{a}_1 = & \left(\begin{array}{cccccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \\
\mathbf{a}_2 = & \left(\begin{array}{cccccccccc} -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \\
\mathbf{a}_1 + \mathbf{a}_2 = & \left(\begin{array}{cccccccccc} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right)
\end{array}
$$



For $S \subset V$, non-zero entries of $\sum_{i \in S} a_i$ equals $E(S, V \setminus S)$

- Fingerprint: Ma_i where M is $\tilde{O}(k)$ dim. proj. such that k non-zero entries of any x can be recovered from Mx.

- Utility: Can find min(all,k) edges across any cut S. Call the set of recovered edges a "k-skeleton".

$$\sum_{j \in S} M\mathbf{a}_j = M\left(\sum_{j \in S} \mathbf{a}_j\right) \longrightarrow \min(\text{all}, k) \text{ edges in } E(S, V \setminus S)$$

Extension to Sparsification

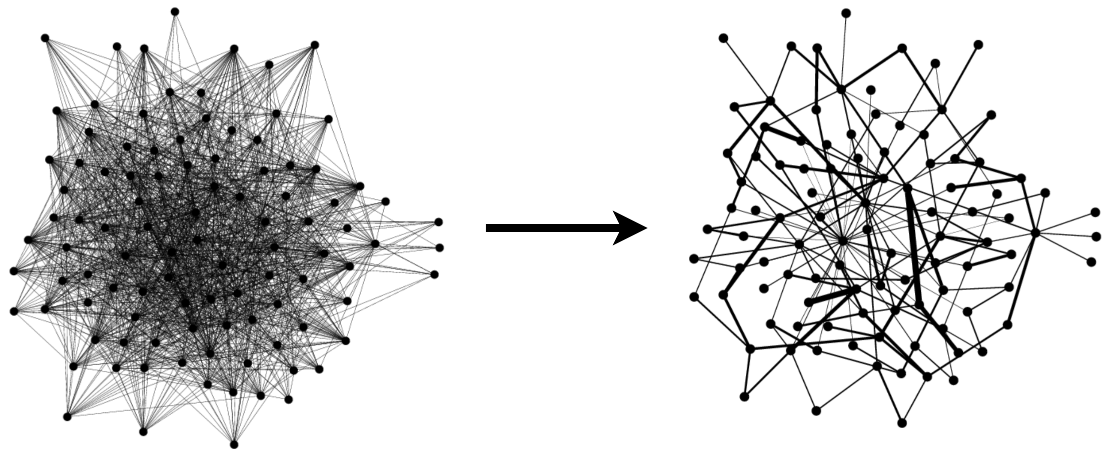


DIAGRAM COURTESY OF NICK HARVEY

Extension to Sparsification

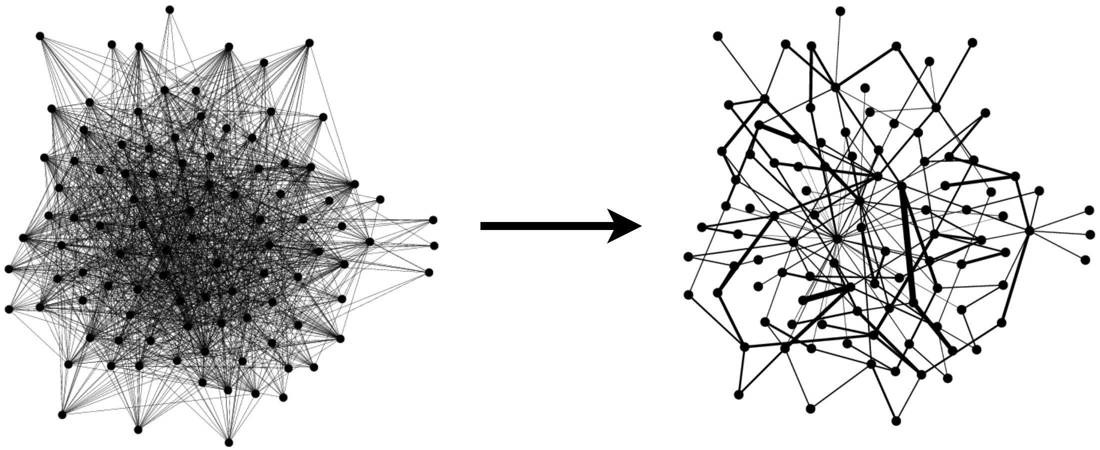


DIAGRAM COURTESY OF NICK HARVEY

- *Theorem:* Can $(1+\varepsilon)$-approximate every graph cut using $O(\varepsilon^{-2}$ polylog n) bit fingerprints of each adjacency list.

Extension to Sparsification

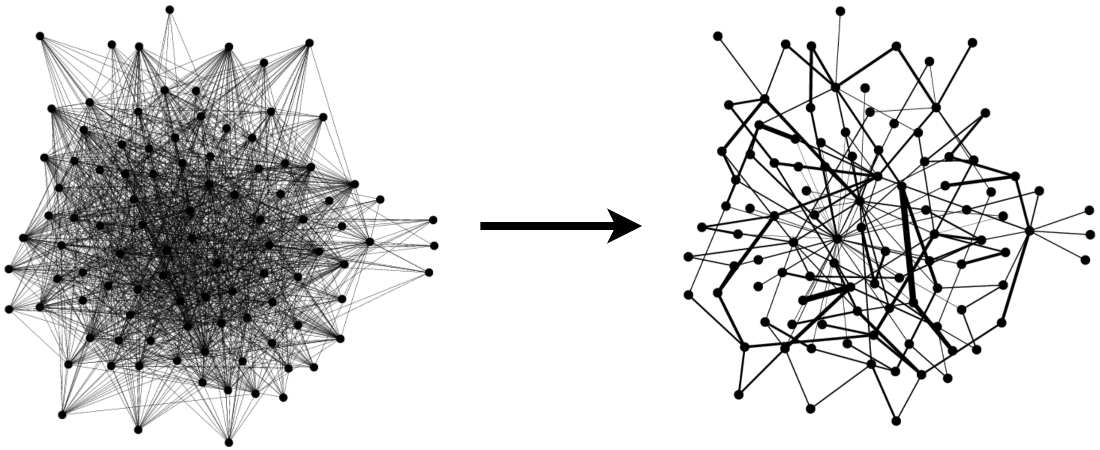


DIAGRAM COURTESY OF NICK HARVEY

- *Theorem:* Can (1+ε)-approximate every graph cut using $O(\varepsilon^{-2}$ polylog n) bit fingerprints of each adjacency list.
- *Theorem:* Can construct a spectral sparsifier H using $O(\varepsilon^{-2} n^{2/3}$ polylog n) bit fingerprints of each adjacency list.

$$\forall x \in \mathbb{R}^n \ : \ (1-\epsilon)x^T L_G x \le x^T L_H x \le (1+\epsilon)x^T L_G x$$

where L_G and L_H are the Laplacians of G and H.

Cut Sparsification

Cut Sparsification

- Thm (Fung et al.) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} \log^2 n/c_e$ where c_e is size of min e cut, then all cuts are preserved up to factor $1+\varepsilon$.

Cut Sparsification

- Thm (Fung et al.) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} \log^2 n / c_e$ where c_e is size of min e cut, then all cuts are preserved up to factor $1+\varepsilon$.
- Algorithm (Edge sampling via k-skeletons)

Cut Sparsification

- Thm (Fung et al.) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} \log^2 n/c_e$ where c_e is size of min e cut, then all cuts are preserved up to factor $1+\varepsilon$.
- Algorithm (Edge sampling via k-skeletons)
  - Let G_i be graph with edges sampled w/p 2^{-i}.

Cut Sparsification

- Thm (Fung et al.) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} \log^2 n/c_e$ where c_e is size of min e cut, then all cuts are preserved up to factor $1+\varepsilon$.
- Algorithm (Edge sampling via k-skeletons)
  - Let G_i be graph with edges sampled w/p 2^{-i}.
  - Return k-skeleton H_i for each G_i where $k= 2\varepsilon^{-2} \log^2 n$

Cut Sparsification

- Thm (Fung et al.) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} \log^2 n/c_e$ where c_e is size of min e cut, then all cuts are preserved up to factor $1+\varepsilon$.
- Algorithm (Edge sampling via k-skeletons)
  - Let G_i be graph with edges sampled w/p 2^{-i}.
  - Return k-skeleton H_i for each G_i where $k= 2\varepsilon^{-2} \log^2 n$
- Thm: e=(u,v) is in some H_i with probability at least p_e

Cut Sparsification

- Thm (Fung et al.) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} \log^2 n/c_e$ where c_e is size of min e cut, then all cuts are preserved up to factor $1+\varepsilon$.
- Algorithm (Edge sampling via k-skeletons)
  - Let G_i be graph with edges sampled w/p 2^{-i}.
  - Return k-skeleton H_i for each G_i where $k= 2\varepsilon^{-2} \log^2 n$
- Thm: e=(u,v) is in some H_i with probability at least p_e
- Proof: Let C be edges in min u-v cut in G.

Cut Sparsification

- Thm (Fung et al.) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} \log^2 n/c_e$ where c_e is size of min e cut, then all cuts are preserved up to factor $1+\varepsilon$.
- Algorithm (Edge sampling via k-skeletons)
  - Let G_i be graph with edges sampled w/p 2^{-i}.
  - Return k-skeleton H_i for each G_i where $k= 2\varepsilon^{-2} \log^2 n$
- Thm: $e=(u,v)$ is in some H_i with probability at least p_e
- Proof: Let C be edges in min u-v cut in G.
  - For $i= -\log p_e$, $E[|C \cap G_i|]=\varepsilon^{-2} \log^2 n$ and whp $|C \cap G_i| \leq k$.

Cut Sparsification

- **Thm (Fung et al.)** Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} \log^2 n/c_e$ where c_e is size of min e cut, then all cuts are preserved up to factor $1+\varepsilon$.
- **Algorithm (Edge sampling via k-skeletons)**
  - Let G_i be graph with edges sampled w/p 2^{-i}.
  - Return k-skeleton H_i for each G_i where $k= 2\varepsilon^{-2} \log^2 n$
- **Thm:** $e=(u,v)$ is in some H_i with probability at least p_e
- **Proof:** Let C be edges in min u-v cut in G.
  - For $i= -\log p_e$, $E[|C\cap G_i|]=\varepsilon^{-2} \log^2 n$ and whp $|C\cap G_i|\leq k$.
  - Hence $e\in H_i$ iff $e\in G_i$ which happens w/p p_e

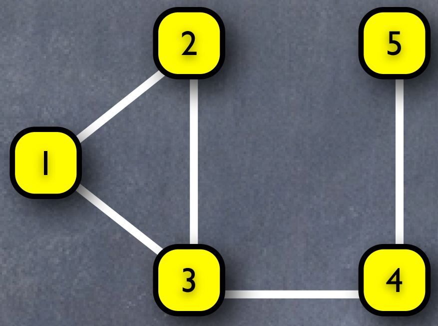
Spectral Sparsification

Spectral Sparsification

- Thm (Spielman-Srivastava) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} r_e \log n$ where r_e is the effective resistance, then preserve spectral properties.

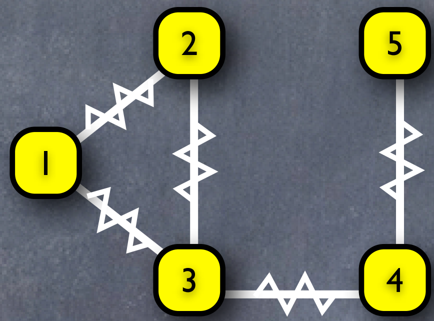
Spectral Sparsification

- Thm (Spielman-Srivastava) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} r_e \log n$ where r_e is the effective resistance, then preserve spectral properties.



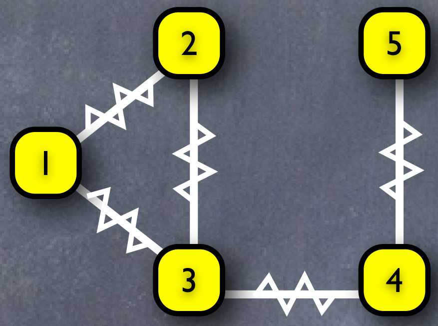
Spectral Sparsification

- Thm (Spielman-Srivastava) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2}\, r_e \log n$ where r_e is the effective resistance, then preserve spectral properties.



Spectral Sparsification

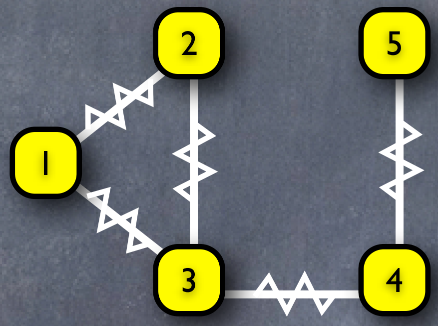
- Thm (Spielman-Srivastava) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} r_e \log n$ where r_e is the effective resistance, then preserve spectral properties.



Effective resistance of (u,v)
is potential difference
when unit of flow injected
at u and extracted at v

Spectral Sparsification

- Thm (Spielman-Srivastava) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} r_e \log n$ where r_e is the effective resistance, then preserve spectral properties.

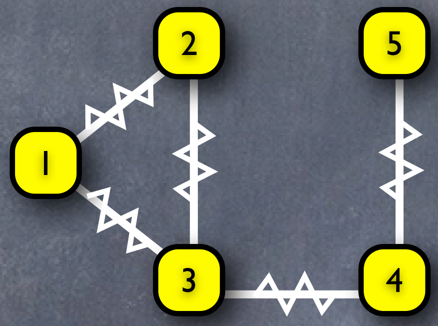


Effective resistance of (u,v) is potential difference when unit of flow injected at u and extracted at v

- Lemma: $1/c_e \le r_e \le O(n^{2/3})/c_e$ for edges in a simple graph.

Spectral Sparsification

- Thm (Spielman-Srivastava) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} r_e \log n$ where r_e is the effective resistance, then preserve spectral properties.

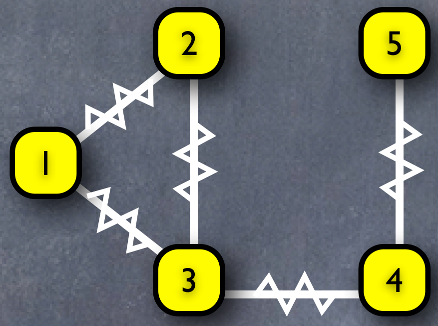


Effective resistance of (u,v) is potential difference when unit of flow injected at u and extracted at v

- Lemma: $1/c_e \leq r_e \leq O(n^{2/3})/c_e$ for edges in a simple graph.
- Proof: Find $O(c_e)$ disjoint paths of length $O(n/\sqrt{c_e})$

Spectral Sparsification

- Thm (Spielman-Srivastava) Sample edge e w/p p_e and weight by $1/p_e$. If $p_e = \varepsilon^{-2} r_e \log n$ where r_e is the effective resistance, then preserve spectral properties.



Effective resistance of (u,v) is potential difference when unit of flow injected at u and extracted at v

- Lemma: $1/c_e \leq r_e \leq O(n^{2/3})/c_e$ for edges in a simple graph.
- Proof: Find $O(c_e)$ disjoint paths of length $O(n/\sqrt{c_e})$
- Corollary: Increasing sampling probability by $O(n^{2/3})$ in cut sparsification, also preserves spectral properties.

I. **Connectivity**

II. **Misalignment**

I. **Connectivity**

II. **Misalignment**

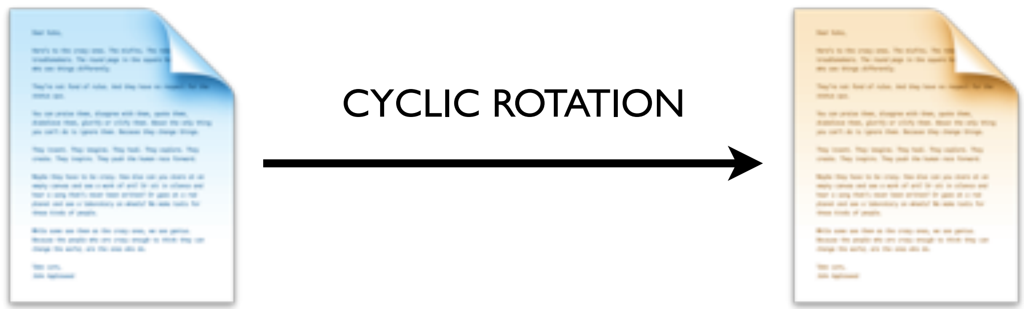
a) Testing Equality with Rotation

b) Matching Lower Bound

Joint work with Alexandr Andoni, Assaf Goldberger, Ely Porat

Fingerprints for Rotation

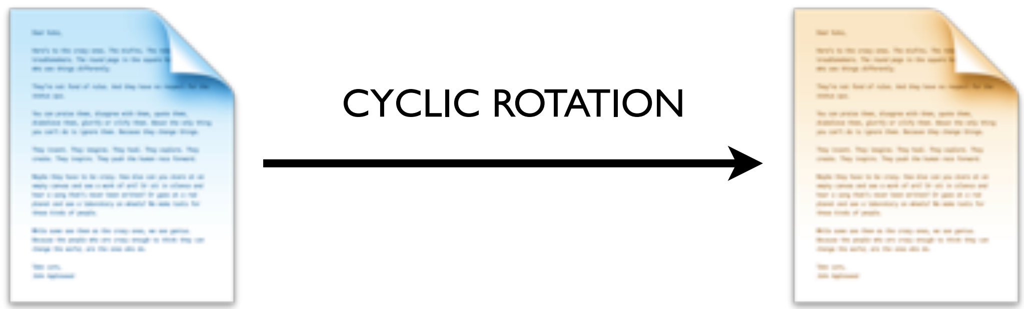
"The quick brown fox jumped over the lazy dog."



"quick brown fox jumped over the lazy dog. The"

Fingerprints for Rotation

“The quick brown fox jumped over the lazy dog.”

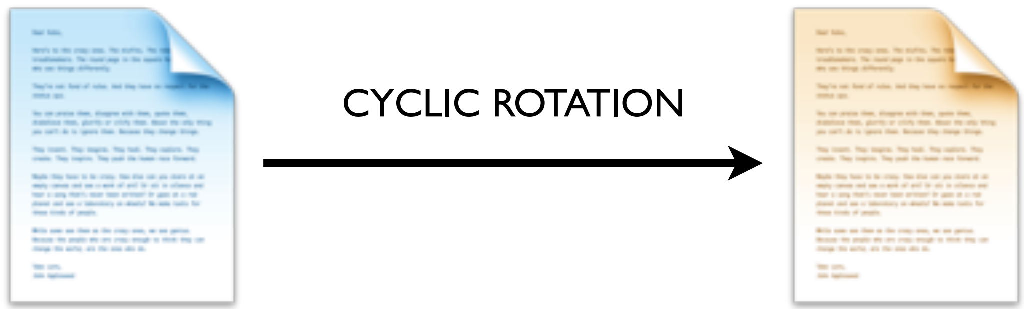


“quick brown fox jumped over the lazy dog. The”

- *Theorem:* There’s a D(n) polylog n bit fingerprint F that is:

Fingerprints for Rotation

"The quick brown fox jumped over the lazy dog."



"quick brown fox jumped over the lazy dog. The"

- *Theorem:* There's a D(n) polylog n bit fingerprint F that is:
  - *Useful:* F(a) and F(b) determine if a, b$\in\mathbb{Z}^n$ are rotations w.h.p.

Fingerprints for Rotation

"The quick brown fox jumped over the lazy dog."

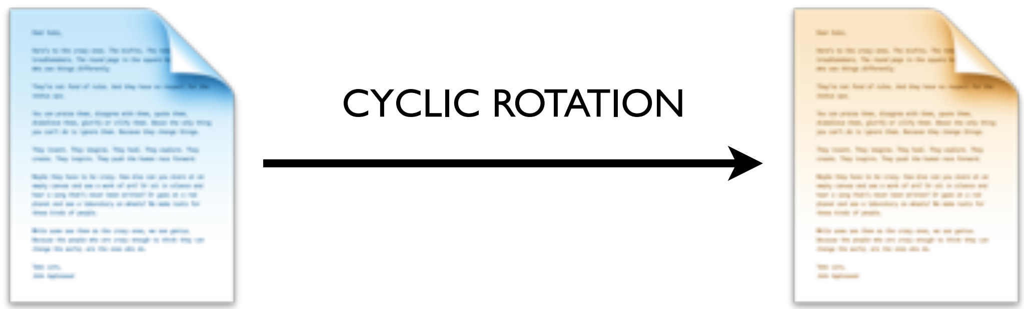
CYCLIC ROTATION

"quick brown fox jumped over the lazy dog. The"

- *Theorem:* There's a D(n) polylog n bit fingerprint F that is:
  - *Useful:* F(a) and F(b) determine if a, b$\in\mathbb{Z}^n$ are rotations w.h.p.
  - *Homomorphic:* From F(a) can construct F(any rotation of a)

Fingerprints for Rotation

"The quick brown fox jumped over the lazy dog."

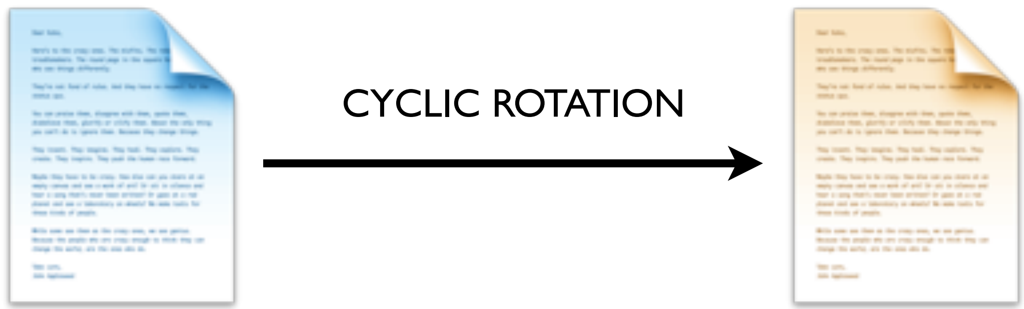


"quick brown fox jumped over the lazy dog. The"

- *Theorem:* There's a D(n) polylog n bit fingerprint F that is:
  - *Useful:* F(a) and F(b) determine if a, b$\in\mathbb{Z}^n$ are rotations w.h.p.
  - *Homomorphic:* From F(a) can construct F(any rotation of a)
  - *Linear:* From F(a) and F(b) can compute F(a+b).

Fingerprints for Rotation

"The quick brown fox jumped over the lazy dog."

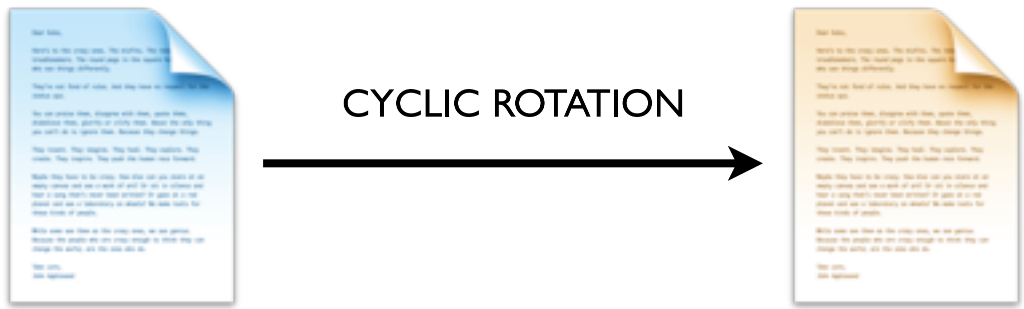


"quick brown fox jumped over the lazy dog. The"

- *Theorem:* There's a D(n) polylog n bit fingerprint F that is:
  - *Useful:* F(a) and F(b) determine if a, b$\in\mathbb{Z}^n$ are rotations w.h.p.
  - *Homomorphic:* From F(a) can construct F(any rotation of a)
  - *Linear:* From F(a) and F(b) can compute F(a+b).
- *Theorem:* Fingerprints with above properties need D(n) bits.

Fingerprints for Rotation

"The quick brown fox jumped over the lazy dog."



"quick brown fox jumped over the lazy dog. The"

- *Theorem:* There's a D(n) polylog n bit fingerprint F that is:
  - *Useful:* F(a) and F(b) determine if $a, b \in \mathbb{Z}^n$ are rotations w.h.p.
  - *Homomorphic:* From F(a) can construct F(any rotation of a)
  - *Linear:* From F(a) and F(b) can compute F(a+b).
- *Theorem:* Fingerprints with above properties need D(n) bits.
- *Extension:* (t + D(n)) polylog n bit fingerprints F(a) and F(b) determine if a,b are within t substitutions of being rotations.

False Start: Fermat's Little Theorem

False Start: Fermat's Little Theorem

- Rabin-Karp: For some p and r, encode $a=a_0a_1a_2...a_{n-1}$ as

False Start: Fermat's Little Theorem

- Rabin-Karp: For some p and r, encode $a=a_0a_1a_2\ldots a_{n-1}$ as
  $f(r,a) = a_0 + a_1 r + a_2 r^2 + \ldots a_{n-1} r^{n-1} \bmod p$

False Start: Fermat's Little Theorem

- Rabin-Karp: For some p and r, encode $a=a_0a_1a_2...a_{n-1}$ as

  $$f(r,a) = a_0 + a_1 r + a_2 r^2 + \ldots a_{n-1} r^{n-1} \bmod p$$

- Fermat's Little Thm: If p=n+1 prime, $r^n=1$ mod p and so,

  $$\begin{aligned} rf(r, a_0a_1 \ldots a_{n-1}) &= a_0 r + a_1 r^2 + a_2 r^3 + \ldots + a_{n-1} r^n \\ &= a_{n-1} + a_0 r + a_1 r^2 + \ldots + a_{n-2} r^{n-1} \\ &= f(r, a_{n-1} a_0 \ldots a_{n-2}) \end{aligned}$$

False Start: Fermat's Little Theorem

- Rabin-Karp: For some p and r, encode $a=a_0a_1a_2...a_{n-1}$ as
  $$f(r,a) = a_0 + a_1 r + a_2 r^2 + \dots a_{n-1} r^{n-1} \bmod p$$
- Fermat's Little Thm: If p=n+1 prime, $r^n=1$ mod p and so,
  $$\begin{aligned} rf(r, a_0 a_1 \dots a_{n-1}) &= a_0 r + a_1 r^2 + a_2 r^3 + \dots + a_{n-1} r^n \\ &= a_{n-1} + a_0 r + a_1 r^2 + \dots + a_{n-2} r^{n-1} \\ &= f(r, a_{n-1} a_0 \dots a_{n-2}) \end{aligned}$$
- So, if b is k-shift of a then $g(r) = r^k f(r,a) - f(r,b) = 0$

False Start: Fermat's Little Theorem

- Rabin-Karp: For some p and r, encode $a=a_0a_1a_2\ldots a_{n-1}$ as
  $$f(r,a) = a_0 + a_1 r + a_2 r^2 + \ldots a_{n-1} r^{n-1} \bmod p$$
- Fermat's Little Thm: If p=n+1 prime, $r^n=1$ mod p and so,
  $$\begin{aligned} rf(r, a_0 a_1 \ldots a_{n-1}) &= a_0 r + a_1 r^2 + a_2 r^3 + \ldots + a_{n-1} r^n \\ &= a_{n-1} + a_0 r + a_1 r^2 + \ldots + a_{n-2} r^{n-1} \\ &= f(r, a_{n-1} a_0 \ldots a_{n-2}) \end{aligned}$$
- So, if b is k-shift of a then $g(r) = r^k f(r,a) - f(r,b) = 0$
- Schwartz-Zippel: If r is random and g non-zero:
  $$P[g(r) = 0] \le (n-1)/p = 1 - O(1/n)$$

False Start: Fermat's Little Theorem

- Rabin-Karp: For some p and r, encode $a = a_0 a_1 a_2 \ldots a_{n-1}$ as
$$f(r,a) = a_0 + a_1 r + a_2 r^2 + \ldots a_{n-1} r^{n-1} \bmod p$$
- Fermat's Little Thm: If p=n+1 prime, $r^n = 1 \bmod p$ and so,
$$\begin{aligned} rf(r, a_0 a_1 \ldots a_{n-1}) &= a_0 r + a_1 r^2 + a_2 r^3 + \ldots + a_{n-1} r^n \\ &= a_{n-1} + a_0 r + a_1 r^2 + \ldots + a_{n-2} r^{n-1} \\ &= f(r, a_{n-1} a_0 \ldots a_{n-2}) \end{aligned}$$
- So, if b is k-shift of a then $g(r) = r^k f(r,a) - f(r,b) = 0$
- Schwartz-Zippel: If r is random and g non-zero:
$$P[g(r) = 0] \leq (n-1)/p = 1 - O(1/n)$$
- Conclusion: No false negatives but likely false positives.

Beyond Schwartz-Zippel

Beyond Schwartz-Zippel

- Evaluate g on roots of x^n-1 but work in larger field

Beyond Schwartz-Zippel

- Evaluate g on roots of x^n-1 but work in larger field
- x^n-1 factorizes as D(n) irreducible polys over rationals:

Beyond Schwartz-Zippel

- Evaluate g on roots of x^n-1 but work in larger field
- x^n-1 factorizes as D(n) irreducible polys over rationals:

$$\begin{aligned} x^{10} - 1 &= \Phi_1(x)\Phi_2(x)\Phi_5(x)\Phi_{10}(x) \\ &= (x-1)(1+x)(1-x+x^2-x^3+x^4)(1+x+x^2+x^3+x^4) \end{aligned}$$

Beyond Schwartz-Zippel

- Evaluate g on roots of x^n-1 but work in larger field
- x^n-1 factorizes as D(n) irreducible polys over rationals:

$$\begin{aligned} x^{10} - 1 &= \Phi_1(x)\Phi_2(x)\Phi_5(x)\Phi_{10}(x) \\ &= (x-1)(1+x)(1-x+x^2-x^3+x^4)(1+x+x^2+x^3+x^4) \end{aligned}$$

- At least one ϕ_i has no shared roots with g:

Beyond Schwartz-Zippel

- Evaluate g on roots of x^n-1 but work in larger field
- x^n-1 factorizes as D(n) irreducible polys over rationals:

$$
\begin{aligned}
x^{10}-1 &= \Phi_1(x)\Phi_2(x)\Phi_5(x)\Phi_{10}(x) \\
&= (x-1)(1+x)(1-x+x^2-x^3+x^4)(1+x+x^2+x^3+x^4)
\end{aligned}
$$

- At least one ϕ_i has no shared roots with g:
  - If ϕ_i shares one root, ϕ_i divides g (Abel's Irred. Thm)

Beyond Schwartz-Zippel

- Evaluate g on roots of x^n-1 but work in larger field
- x^n-1 factorizes as D(n) irreducible polys over rationals:

$$\begin{aligned} x^{10} - 1 &= \Phi_1(x)\Phi_2(x)\Phi_5(x)\Phi_{10}(x) \\ &= (x-1)(1+x)(1-x+x^2-x^3+x^4)(1+x+x^2+x^3+x^4) \end{aligned}$$

- At least one ϕ_i has no shared roots with g:
  - If ϕ_i shares one root, ϕ_i divides g (Abel's Irred. Thm)
  - Can't all divide g because g has degree ≤ n-1

Beyond Schwartz-Zippel

- Evaluate g on roots of x^n-1 but work in larger field
- x^n-1 factorizes as D(n) irreducible polys over rationals:

$$\begin{aligned} x^{10}-1 &= \Phi_1(x)\Phi_2(x)\Phi_5(x)\Phi_{10}(x) \\ &= (x-1)(1+x)(1-x+x^2-x^3+x^4)(1+x+x^2+x^3+x^4) \end{aligned}$$

- At least one ϕ_i has no shared roots with g:
  - If ϕ_i shares one root, ϕ_i divides g (Abel's Irred. Thm)
  - Can't all divide g because g has degree ≤ n-1
- Suffices to test g on an arbitrary root of each ϕ_i

Beyond Schwartz-Zippel

- Evaluate g on roots of x^n-1 but work in larger field
- x^n-1 factorizes as D(n) irreducible polys over rationals:

$$
\begin{aligned}
x^{10} - 1 &= \Phi_1(x)\Phi_2(x)\Phi_5(x)\Phi_{10}(x) \\
&= (x-1)(1+x)(1-x+x^2-x^3+x^4)(1+x+x^2+x^3+x^4)
\end{aligned}
$$

- At least one ϕ_i has no shared roots with g:
  - If ϕ_i shares one root, ϕ_i divides g (Abel's Irred. Thm)
  - Can't all divide g because g has degree ≤ n-1
- Suffices to test g on an arbitrary root of each ϕ_i
- **Bad News:** Can't guarantee g(r) has finite precision.

Beyond Schwartz-Zippel

- Evaluate g on roots of x^n-1 but work in larger field
- x^n-1 factorizes as D(n) irreducible polys over rationals:

$$\begin{aligned} x^{10}-1 &= \Phi_1(x)\Phi_2(x)\Phi_5(x)\Phi_{10}(x) \\ &= (x-1)(1+x)(1-x+x^2-x^3+x^4)(1+x+x^2+x^3+x^4) \end{aligned}$$

- At least one ϕ_i has no shared roots with g:
  - If ϕ_i shares one root, ϕ_i divides g (Abel's Irred. Thm)
  - Can't all divide g because g has degree ≤ n-1
- Suffices to test g on an arbitrary root of each ϕ_i
- Bad News: Can't guarantee g(r) has finite precision.
- Good News: Work modulo a random p. Can show ϕ_i still doesn't share roots with g whp by analyzing resultant.

Lower Bound: Basic Idea

Lower Bound: Basic Idea

- Can recover D(n) bits about a from F(a) by summing the fingerprints of rotations

Lower Bound: Basic Idea

- Can recover D(n) bits about a from F(a) by summing the fingerprints of rotations
- To deduce $\alpha = \sum a_i$ from $F(a_0 a_1 a_2 a_3 a_4 a_5)$

$$F(a_0 a_1 a_2 a_3 a_4 a_5) + F(a_1 a_2 a_3 a_4 a_5 a_0) + \ldots + F(a_5 a_0 a_1 a_2 a_3 a_4) = F(\alpha\alpha\alpha\alpha\alpha\alpha)$$

Lower Bound: Basic Idea

- Can recover D(n) bits about a from F(a) by summing the fingerprints of rotations
- To deduce $\alpha = \sum a_i$ from $F(a_0 a_1 a_2 a_3 a_4 a_5)$

$$F(a_0 a_1 a_2 a_3 a_4 a_5) + F(a_1 a_2 a_3 a_4 a_5 a_0) + \ldots + F(a_5 a_0 a_1 a_2 a_3 a_4) = F(\alpha\alpha\alpha\alpha\alpha\alpha)$$

and compare $F(gggggg)$ for all g until matches.

Lower Bound: Basic Idea

- Can recover D(n) bits about a from F(a) by summing the fingerprints of rotations
- To deduce $\alpha = \sum a_i$ from $F(a_0 a_1 a_2 a_3 a_4 a_5)$

$$F(a_0 a_1 a_2 a_3 a_4 a_5) + F(a_1 a_2 a_3 a_4 a_5 a_0) + \ldots + F(a_5 a_0 a_1 a_2 a_3 a_4) = F(\alpha\alpha\alpha\alpha\alpha\alpha)$$

and compare $F(gggggg)$ for all g until matches.

- To deduce $\beta = a_1 + a_3 + a_5$

$$F(a_0 a_1 a_2 a_3 a_4 a_5) + F(a_2 a_3 a_4 a_5 a_0 a_1) + F(a_4 a_5 a_0 a_1 a_2 a_3) = F(\beta\gamma\beta\gamma\beta\gamma)$$

Lower Bound: Basic Idea

- Can recover D(n) bits about a from F(a) by summing the fingerprints of rotations
- To deduce $\alpha = \sum a_i$ from $F(a_0 a_1 a_2 a_3 a_4 a_5)$

$$F(a_0 a_1 a_2 a_3 a_4 a_5) + F(a_1 a_2 a_3 a_4 a_5 a_0) + \ldots + F(a_5 a_0 a_1 a_2 a_3 a_4) = F(\alpha\alpha\alpha\alpha\alpha\alpha)$$

and compare $F(gggggg)$ for all g until matches.

- To deduce $\beta = a_1 + a_3 + a_5$

$$F(a_0 a_1 a_2 a_3 a_4 a_5) + F(a_2 a_3 a_4 a_5 a_0 a_1) + F(a_4 a_5 a_0 a_1 a_2 a_3) = F(\beta\gamma\beta\gamma\beta\gamma)$$

and compare $F(gg'gg'gg')$ for all g, g'=α-g until matches.

Lower Bound: Basic Idea

- Can recover D(n) bits about a from F(a) by summing the fingerprints of rotations
- To deduce $\alpha = \sum a_i$ from $F(a_0 a_1 a_2 a_3 a_4 a_5)$

$$F(a_0 a_1 a_2 a_3 a_4 a_5) + F(a_1 a_2 a_3 a_4 a_5 a_0) + \ldots + F(a_5 a_0 a_1 a_2 a_3 a_4) = F(\alpha\alpha\alpha\alpha\alpha\alpha)$$

and compare $F(gggggg)$ for all g until matches.

- To deduce $\beta = a_1 + a_3 + a_5$

$$F(a_0 a_1 a_2 a_3 a_4 a_5) + F(a_2 a_3 a_4 a_5 a_0 a_1) + F(a_4 a_5 a_0 a_1 a_2 a_3) = F(\beta\gamma\beta\gamma\beta\gamma)$$

and compare $F(gg'gg'gg')$ for all g, g'=α-g until matches.

- And so on for other divisors of n...

Thanks!

- *Homomorphic Sketches:* Compress using sketches such that we can run algorithms on compressed data directly. Resulting algorithms are *parallelizable* + *streamable*.
- *Graphs:* Dimensionality reduction for preserving structural properties. Enables dynamic graph streaming.
- *Fingerprinting with Misalignments:* Tight bounds on size of fingerprint necessary for testing equality up to rotations.