Sketching Graphs

Kook Jin Ahn *University of Pennsylvania*
Sudipto Guha *University of Pennsylvania*
Andrew McGregor *University of Massachusetts*

Linear Sketches

Linear Sketches

- Random linear projection $M: R^n \rightarrow R^k$ that preserves properties of any $v \in R^n$ with high probability where $k \ll n$.

Linear Sketches

- Random linear projection M: $R^n \rightarrow R^k$ that preserves properties of any $v \in R^n$ with high probability where $k \ll n$.

$$\begin{pmatrix} \\ \\ v \\ \\ \\ \end{pmatrix}$$

Linear Sketches

- Random linear projection M: $R^n \rightarrow R^k$ that preserves properties of any $v \in R^n$ with high probability where $k \ll n$.

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

Linear Sketches

- Random linear projection M: $R^n \rightarrow R^k$ that preserves properties of any $v \in R^n$ with high probability where $k \ll n$.

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

Linear Sketches

- Random linear projection $M: R^n \rightarrow R^k$ that preserves properties of any $v \in R^n$ with high probability where $k \ll n$.

$$\begin{pmatrix} & M & \end{pmatrix} \begin{pmatrix} \\ v \\ \\ \end{pmatrix} = \begin{pmatrix} Mv \end{pmatrix} \longrightarrow \text{answer}$$

Linear Sketches

- Random linear projection $M: R^n \to R^k$ that preserves properties of any $v \in R^n$ with high probability where $k \ll n$.

$$\begin{pmatrix} & & M & & \end{pmatrix} \begin{pmatrix} \\ v \\ \\ \end{pmatrix} = \begin{pmatrix} Mv \end{pmatrix} \longrightarrow \text{answer}$$

- *__Many Results:__* Estimating norms, entropy, support size, quantiles, heavy hitters, fitting histograms and polynomials, ...

Linear Sketches

- Random linear projection $M: R^n \to R^k$ that preserves properties of any $v \in R^n$ with high probability where $k \ll n$.

$$\begin{pmatrix} & & M & & \end{pmatrix} \begin{pmatrix} \\ \\ v \\ \\ \\ \end{pmatrix} = \begin{pmatrix} Mv \end{pmatrix} \longrightarrow \text{answer}$$

- *Many Results:* Estimating norms, entropy, support size, quantiles, heavy hitters, fitting histograms and polynomials, ...
- *Rich Theory:* Related to compressed sensing and sparse recovery, dimensionality reduction and metric embeddings, ...

Sketching Graphs?

Sketching Graphs?

- ***Question:*** Are there sketches for structured objects like graphs?

Sketching Graphs?

- *Question:* Are there sketches for structured objects like graphs?

$$\left(\quad A_G \quad \right)$$

Sketching Graphs?

- *Question:* Are there sketches for structured objects like graphs?

$$\begin{pmatrix} & M & \end{pmatrix} \begin{pmatrix} & & \\ & A_G & \\ & & \end{pmatrix}$$

Sketching Graphs?

? *Question:* Are there sketches for structured objects like graphs?

$$\begin{pmatrix} & M & \end{pmatrix} \begin{pmatrix} \\ & A_G & \\ \\ \end{pmatrix} = \begin{pmatrix} & MA_G & \end{pmatrix}$$

Sketching Graphs?

? *Question:* Are there sketches for structured objects like graphs?

$$\begin{pmatrix} & M & \end{pmatrix} \begin{pmatrix} \\ & A_G & \\ \\ \end{pmatrix} = \begin{pmatrix} & MA_G & \end{pmatrix} \longrightarrow \text{answer}$$

Sketching Graphs?

- *Question:* Are there sketches for structured objects like graphs?

$$\begin{pmatrix} & M & \end{pmatrix} \begin{pmatrix} \\ & A_G & \\ \\ \end{pmatrix} = \begin{pmatrix} & MA_G & \end{pmatrix} \longrightarrow \text{answer}$$

- *Example:* Project $O(n^2)$-dimensional adjacency matrix A_G to $\tilde{O}(n)$ dimensions and still determine if graph is bipartite?

Sketching Graphs?

? *Question:* Are there sketches for structured objects like graphs?

$$\begin{pmatrix} & M & \end{pmatrix} \begin{pmatrix} & & \\ & A_G & \\ & & \end{pmatrix} = \begin{pmatrix} & MA_G & \end{pmatrix} \longrightarrow \text{answer}$$

- *Example:* Project O(n^2)-dimensional adjacency matrix A_G to Õ(n) dimensions and still determine if graph is bipartite?

! No cheating! Assume M is finite precision etc.

Why? Graph Streams

Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

[Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

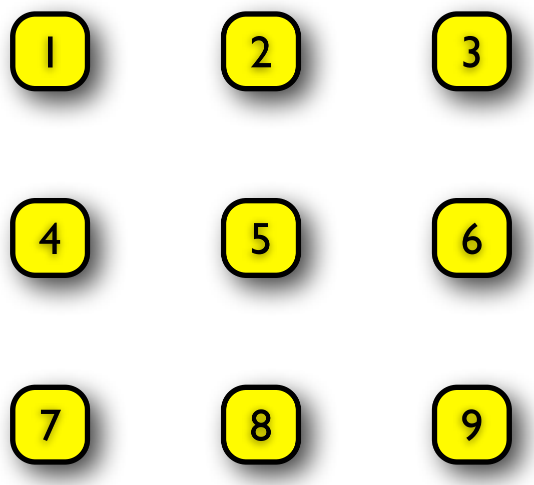
- *Example:* Connectivity is easy if edges are only inserted...

Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!
- *Example:* Connectivity is easy if edges are only inserted...



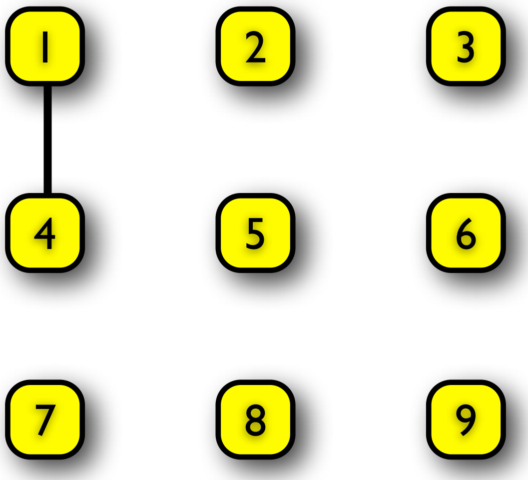
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



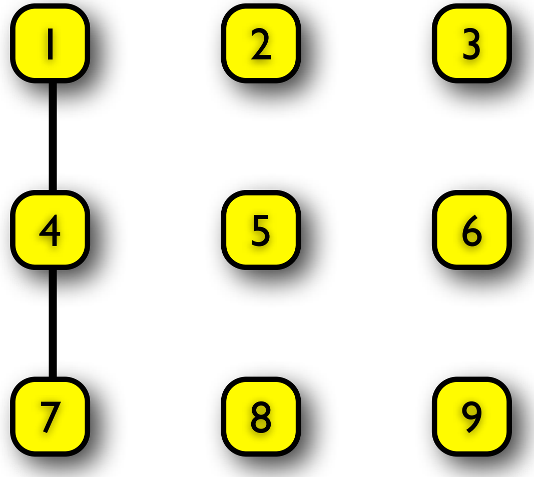
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



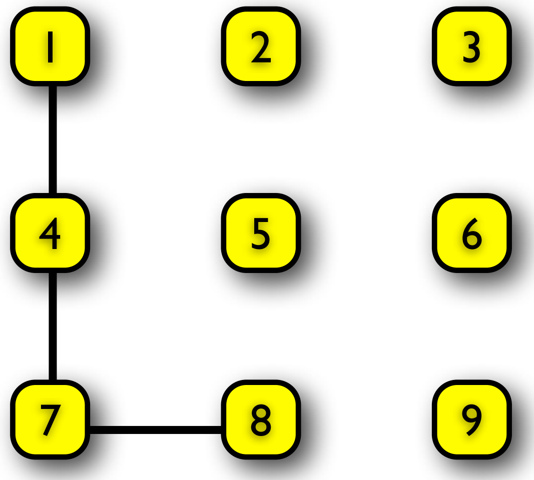
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, ..., e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



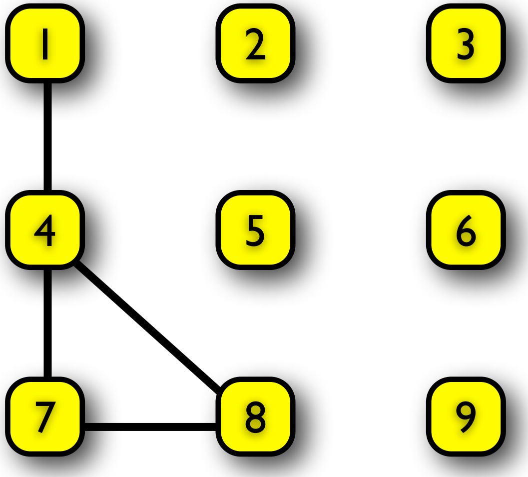
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, ..., e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



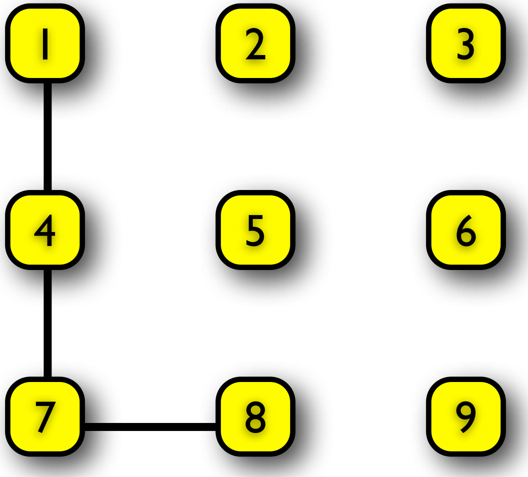
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



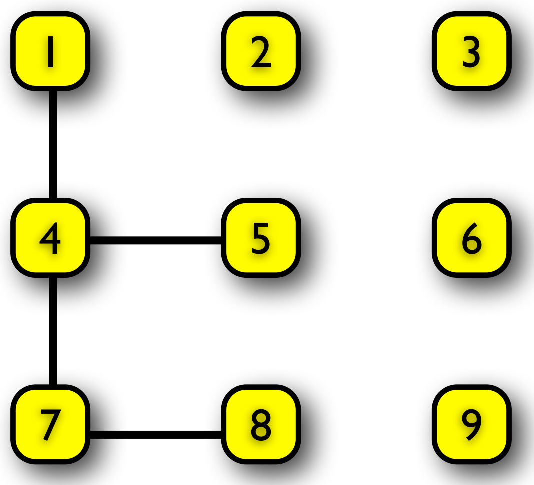
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



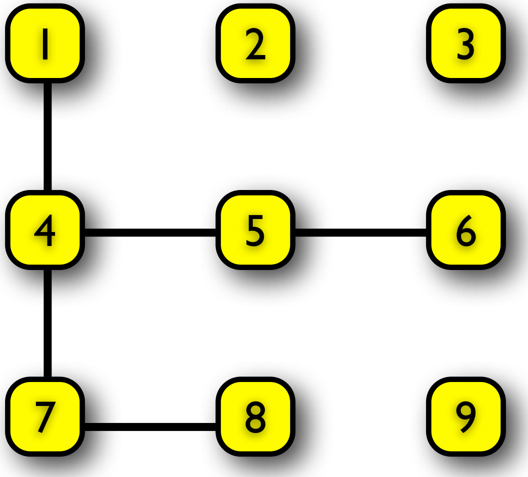
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



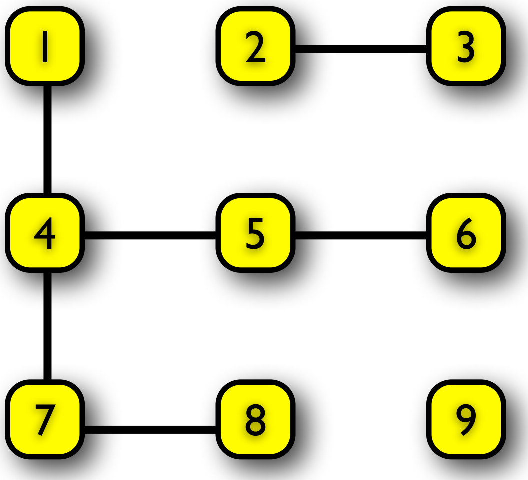
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



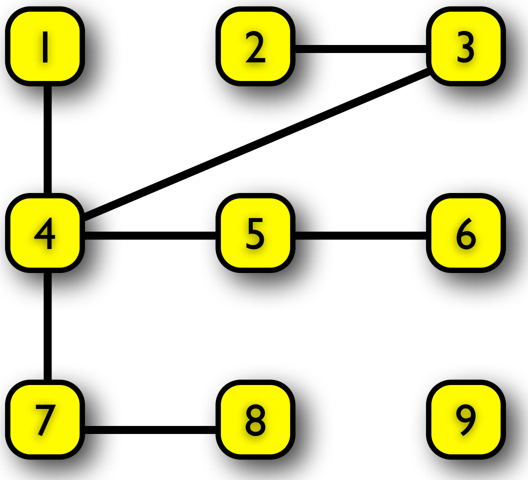
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, ..., e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



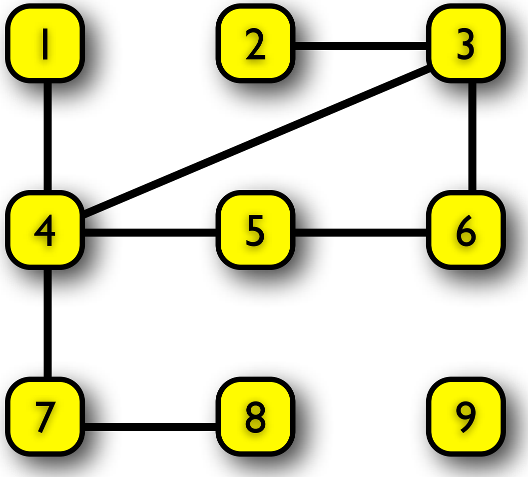
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



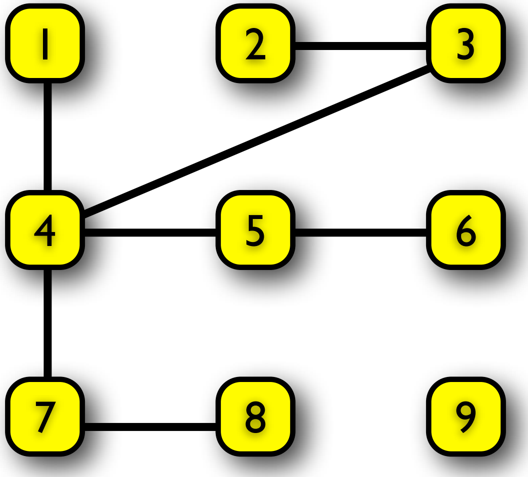
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



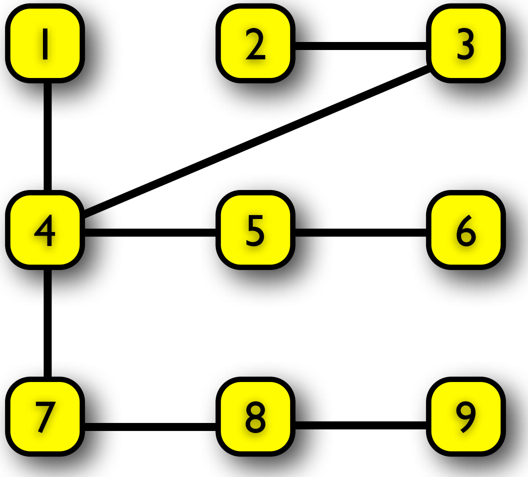
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



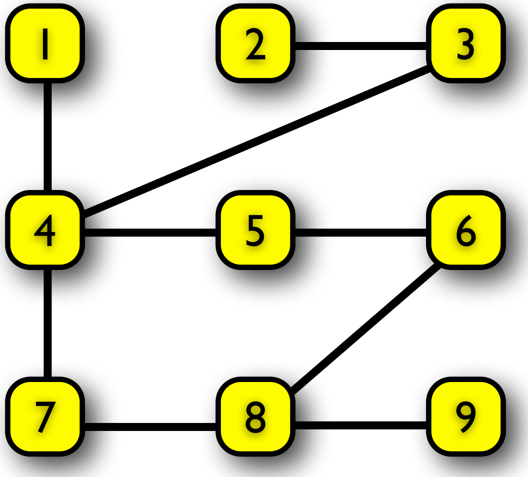
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



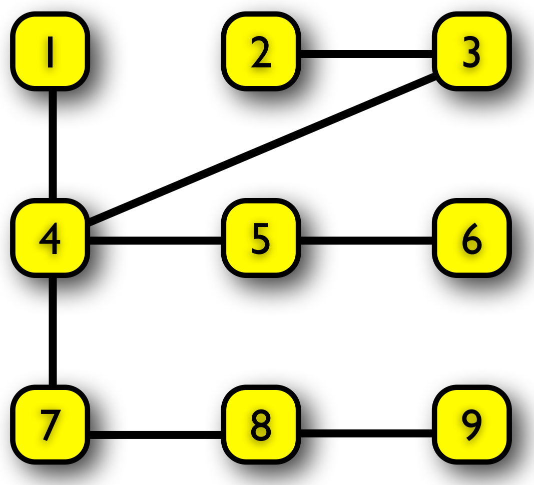
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, ..., e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



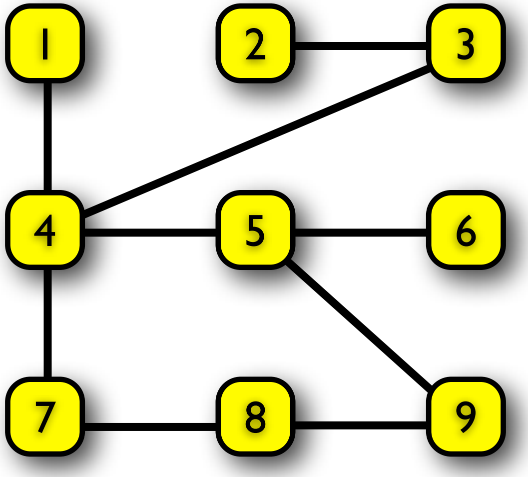
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



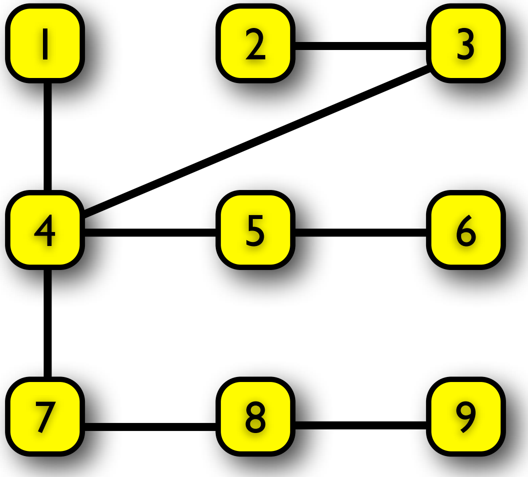
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, \ldots, e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...



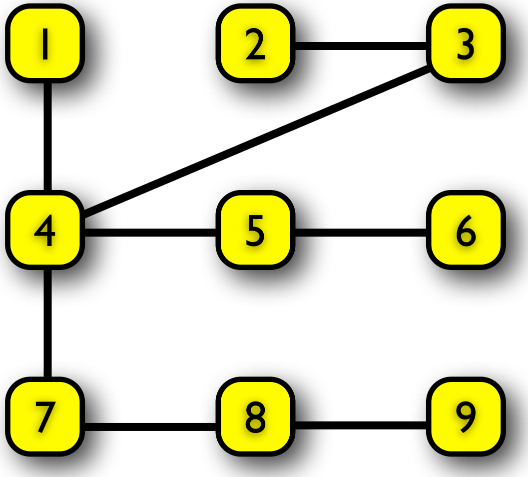
Why? Graph Streams

- In semi-streaming, want to process graph defined by edges $e_1, ..., e_m$ with $\tilde{O}(n)$ memory and reading sequence in order.

  [Muthukrishnan 05; Feigenbaum, Kannan, McGregor, Suri, Zhang 05]

- *Dynamic Graphs:* Work on graph streams doesn't support edge deletions! Work on dynamic graphs stores entire graph!

- *Example:* Connectivity is easy if edges are only inserted...

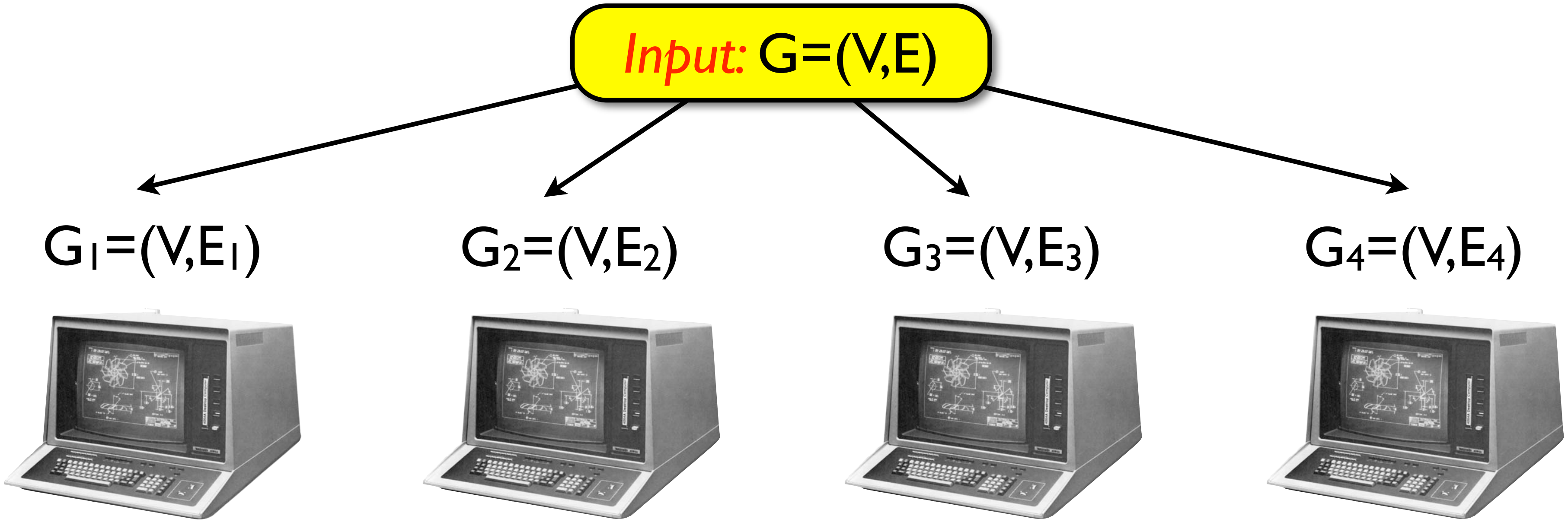


- *Sketches:* To delete e from G: update $MA_G \rightarrow MA_G - MA_e = MA_{G-e}$

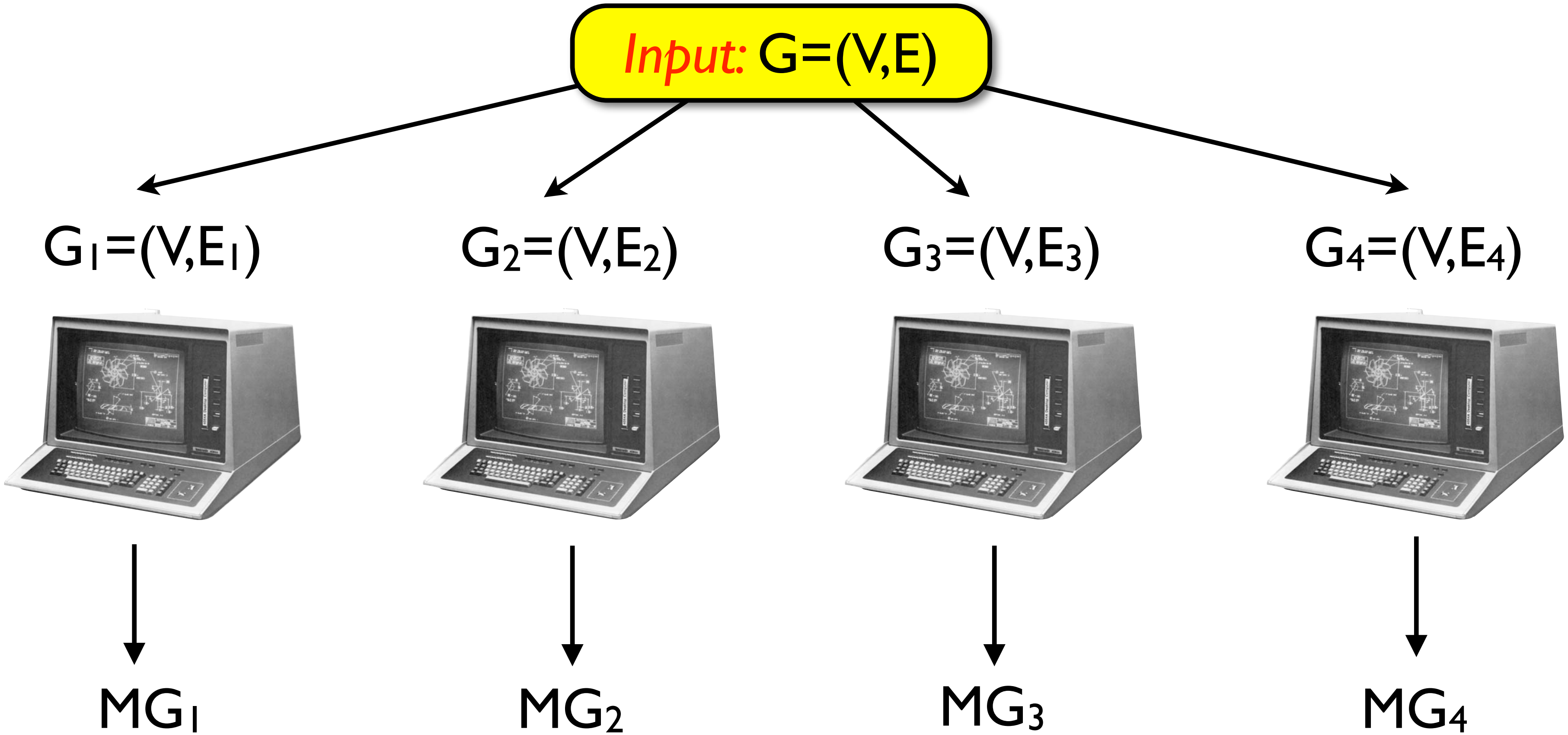
Why? Distributed Processing

Input: G=(V,E)

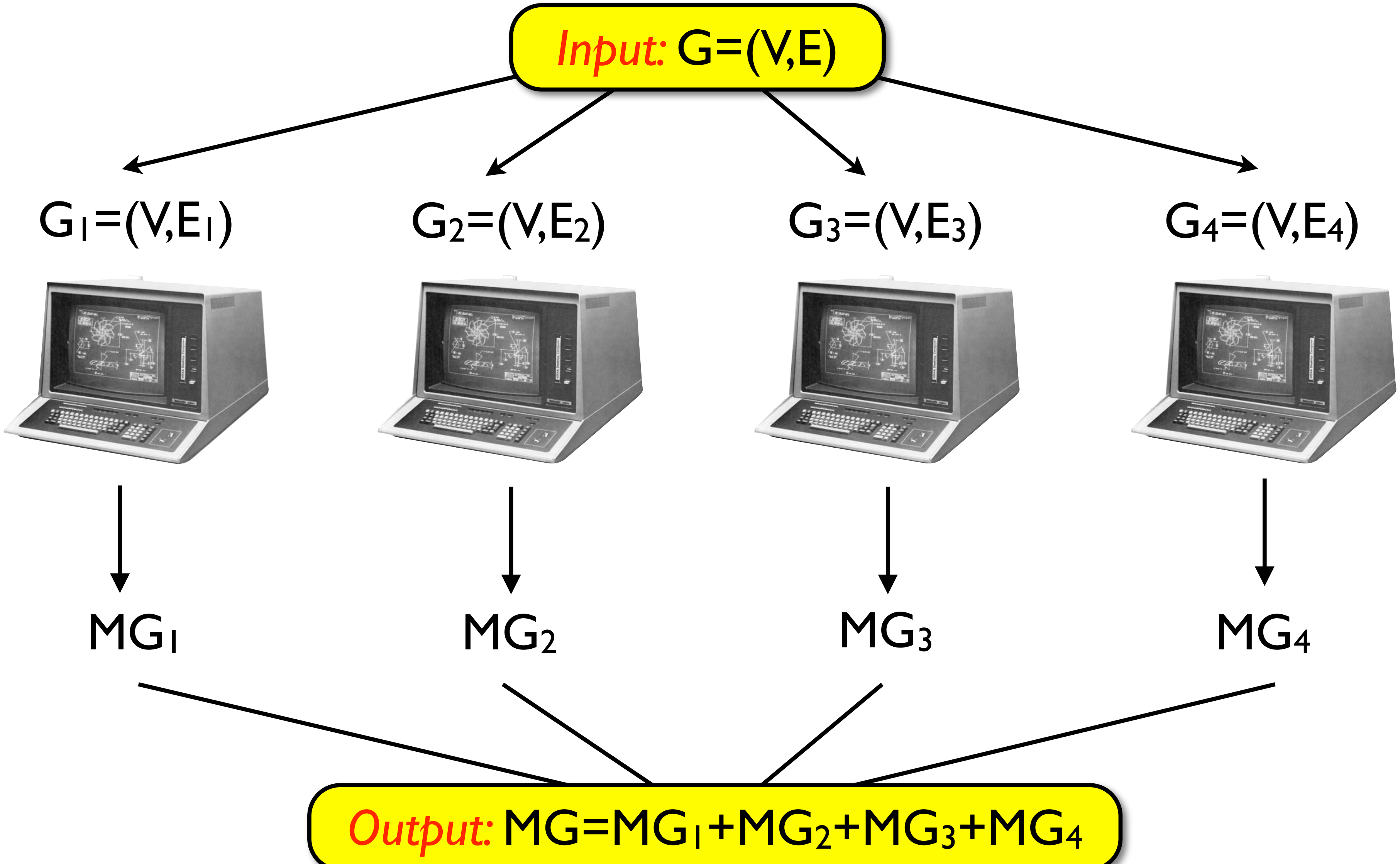
Why? Distributed Processing



Why? Distributed Processing



Why? Distributed Processing



a) Connectivity
b) Applications

a) Connectivity

b) Applications

Connectivity

Connectivity

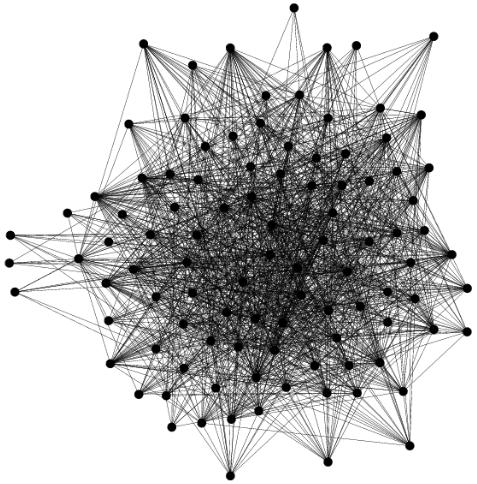
- *Thm:* Can check connectivity with $O(n\log^3 n)$-size sketch.

Connectivity

- *Thm:* Can check connectivity with $O(n\log^3 n)$-size sketch.
- *Main Idea:* *a)* Sketch! *b)* Run Algorithm in Sketch Space

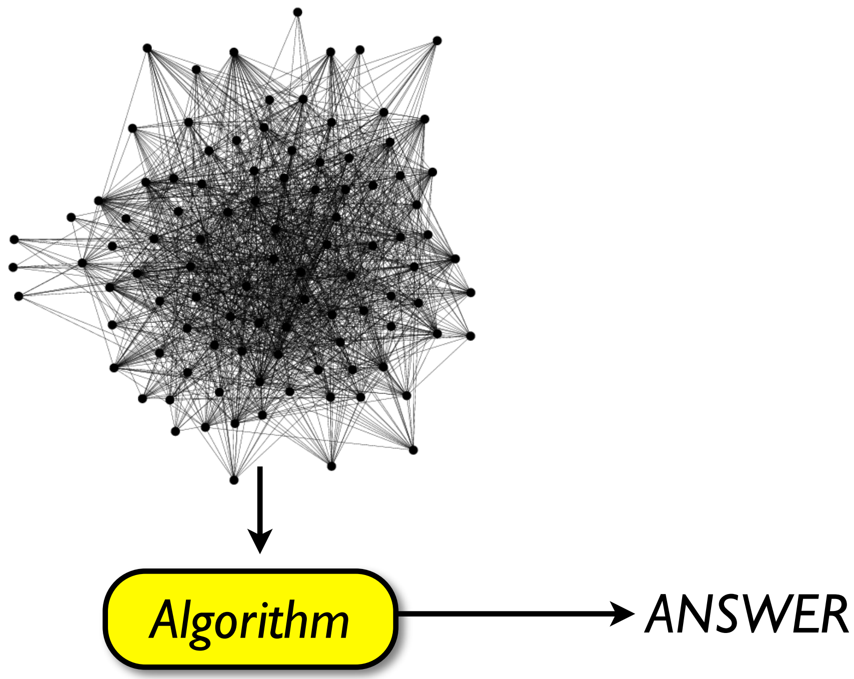
Connectivity

- *Thm:* Can check connectivity with $O(n \log^3 n)$-size sketch.
- *Main Idea:* *a)* Sketch! *b)* Run Algorithm in Sketch Space



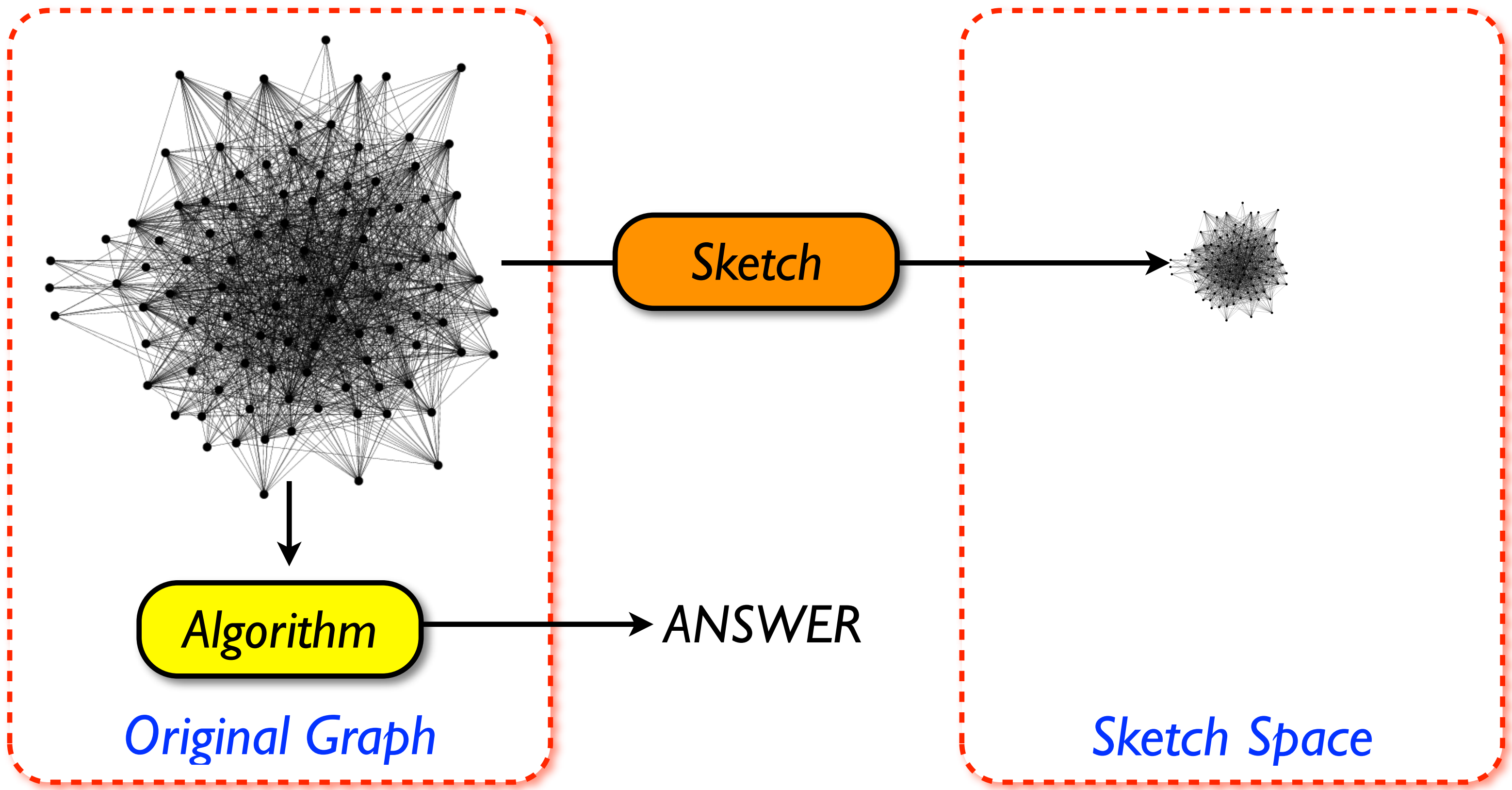
Connectivity

- *Thm:* Can check connectivity with $O(n\log^3 n)$-size sketch.
- *Main Idea:* *a)* Sketch! *b)* Run Algorithm in Sketch Space



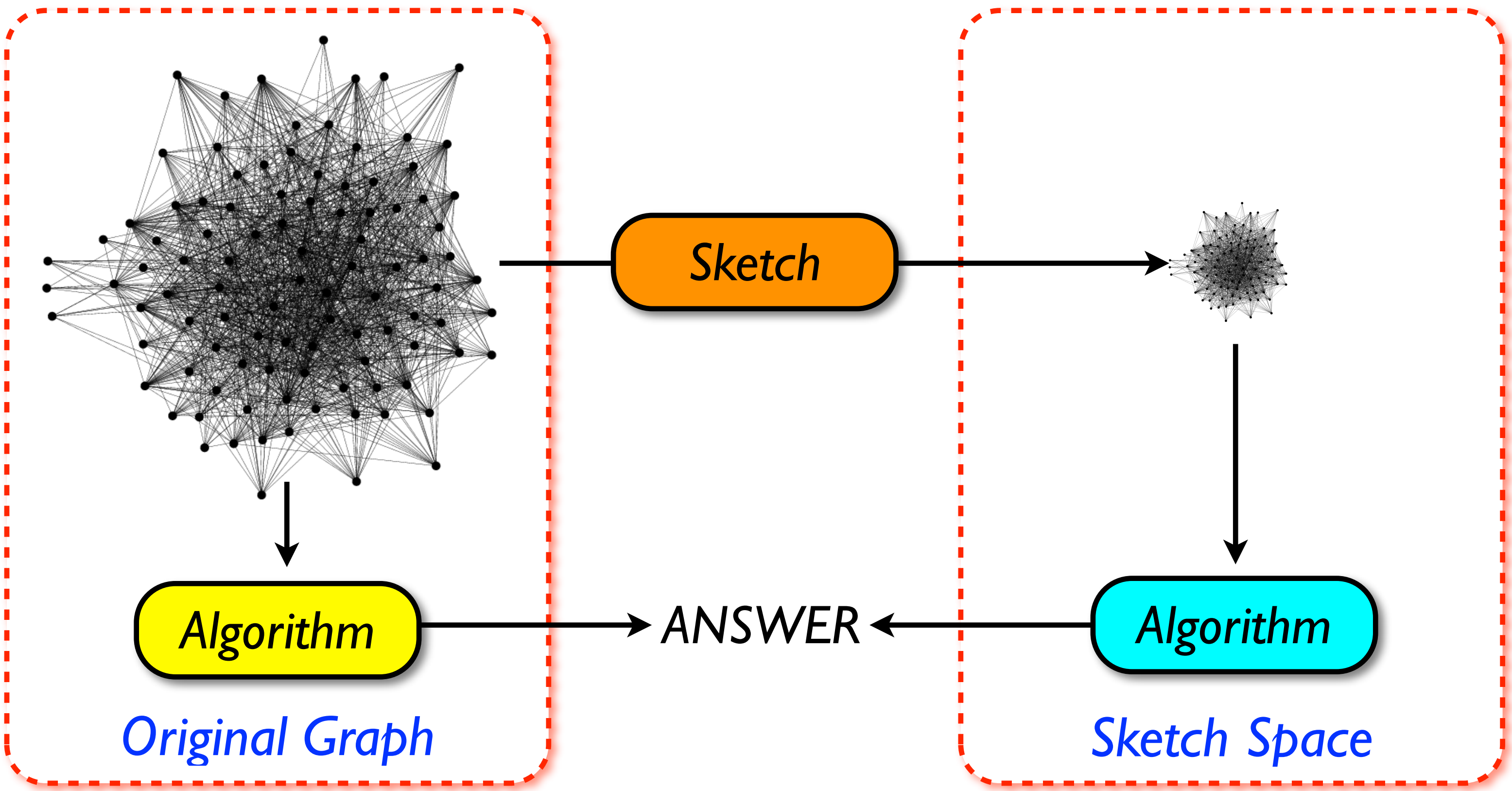
Connectivity

- ***Thm:*** Can check connectivity with $O(n\log^3 n)$-size sketch.
- ***Main Idea:*** *a)* Sketch! *b)* Run Algorithm in Sketch Space



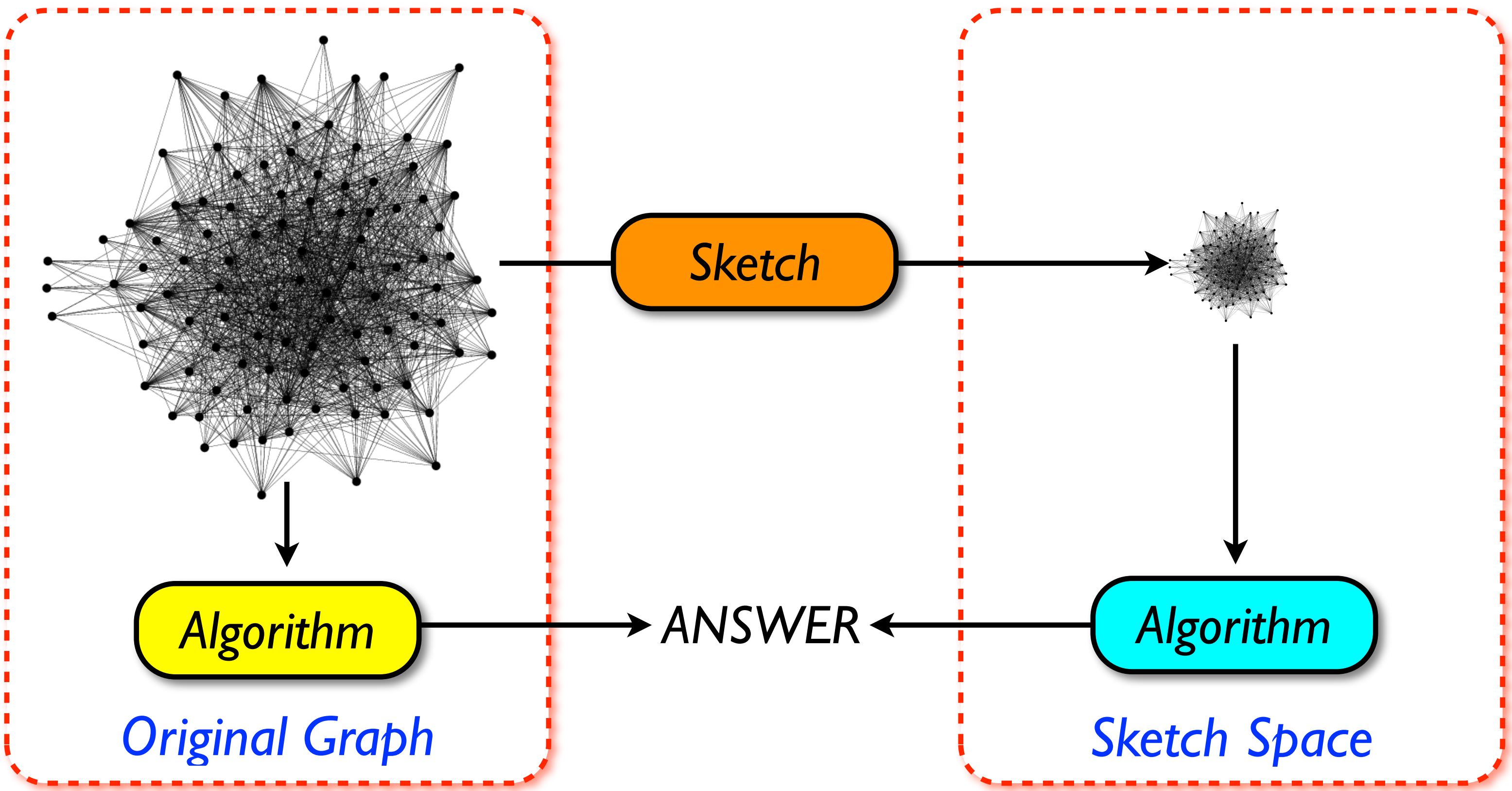
Connectivity

- ***Thm:*** Can check connectivity with $O(n\log^3 n)$-size sketch.
- ***Main Idea:*** *a)* Sketch! *b)* Run Algorithm in Sketch Space



Connectivity

- *Thm:* Can check connectivity with O(nlog^3 n)-size sketch.
- *Main Idea:* *a)* Sketch! *b)* Run Algorithm in Sketch Space



- *Catch:* Sketch must be homomorphic for algorithm operations.

Ingredient 1: Basic Connectivity Algorithm

Ingredient 1: Basic Connectivity Algorithm

- Algorithm (Spanning Forest):

Ingredient 1: Basic Connectivity Algorithm

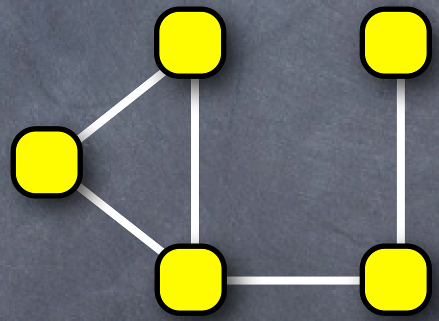
- Algorithm (Spanning Forest):
  1. For each node, select an incident edge

Ingredient 1: Basic Connectivity Algorithm

- Algorithm (Spanning Forest):
  1. For each node, select an incident edge
  2. Contract selected edges. Repeat until no edges.

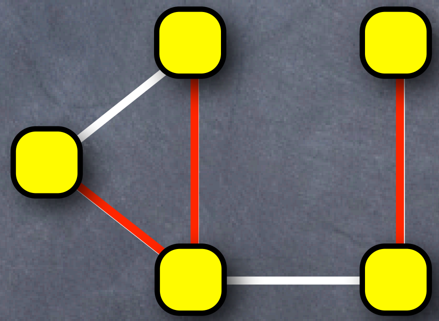
Ingredient 1: Basic Connectivity Algorithm

- Algorithm (Spanning Forest):
  1. For each node, select an incident edge
  2. Contract selected edges. Repeat until no edges.



Ingredient 1: Basic Connectivity Algorithm

- Algorithm (Spanning Forest):
  1. For each node, select an incident edge
  2. Contract selected edges. Repeat until no edges.



Ingredient 1: Basic Connectivity Algorithm

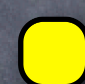
- Algorithm (Spanning Forest):
  1. For each node, select an incident edge
  2. Contract selected edges. Repeat until no edges.

Ingredient 1: Basic Connectivity Algorithm

- Algorithm (Spanning Forest):
  1. For each node, select an incident edge
  2. Contract selected edges. Repeat until no edges.

Ingredient 1: Basic Connectivity Algorithm

- Algorithm (Spanning Forest):
  1. For each node, select an incident edge
  2. Contract selected edges. Repeat until no edges.



Ingredient 1: Basic Connectivity Algorithm

- Algorithm (Spanning Forest):
  1. For each node, select an incident edge
  2. Contract selected edges. Repeat until no edges.

- Lemma: Takes O(log n) steps and selected edges include spanning forest.

Ingredient 2: Graph Representation

Ingredient 2: Graph Representation

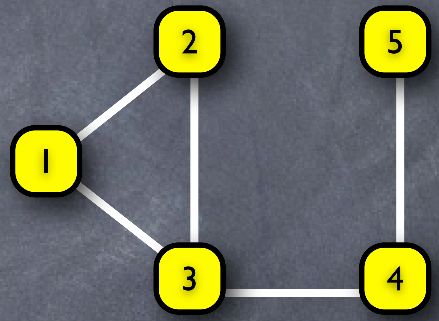
- For node i, let a_i be vector indexed by node pairs. Non-zero entries: $a_i[i,j]=1$ if $j>i$ and $a_i[i,j]=-1$ if $j<i$.

Ingredient 2: Graph Representation

- For node i, let a_i be vector indexed by node pairs. Non-zero entries: $a_i[i,j]=1$ if $j>i$ and $a_i[i,j]=-1$ if $j<i$.
- Example:

Ingredient 2: Graph Representation

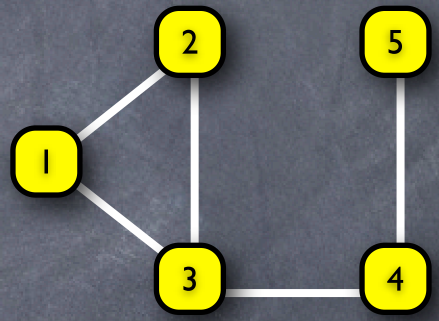
- For node i, let a_i be vector indexed by node pairs. Non-zero entries: $a_i[i,j]=1$ if $j>i$ and $a_i[i,j]=-1$ if $j<i$.
- Example:



Ingredient 2: Graph Representation

- For node i, let a_i be vector indexed by node pairs. Non-zero entries: $a_i[i,j]=1$ if $j>i$ and $a_i[i,j]=-1$ if $j<i$.
- Example:

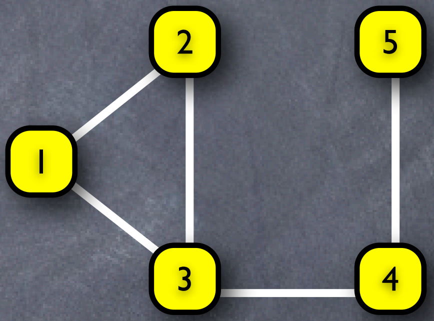
	{1,2}	{1,3}	{1,4}	{1,5}	{2,3}	{2,4}	{2,5}	{3,4}	{3,5}	{4,5}
$\mathbf{a}_1 =$	1	1	0	0	0	0	0	0	0	0



Ingredient 2: Graph Representation

- For node i, let a_i be vector indexed by node pairs. Non-zero entries: $a_i[i,j]=1$ if $j>i$ and $a_i[i,j]=-1$ if $j<i$.
- Example:

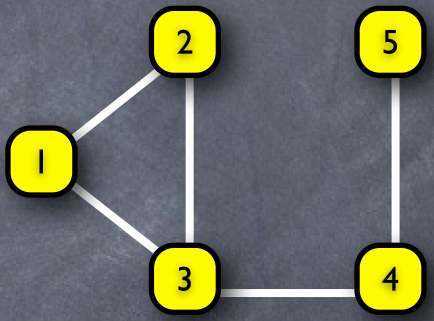
	{1,2}	{1,3}	{1,4}	{1,5}	{2,3}	{2,4}	{2,5}	{3,4}	{3,5}	{4,5}
$\mathbf{a}_1 =$	1	1	0	0	0	0	0	0	0	0
$\mathbf{a}_2 =$	-1	0	0	0	1	0	0	0	0	0



Ingredient 2: Graph Representation

- For node i, let a_i be vector indexed by node pairs. Non-zero entries: $a_i[i,j]=1$ if $j>i$ and $a_i[i,j]=-1$ if $j<i$.
- Example:

	{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



- Lemma: For any subset of nodes $S \subset V$,

$$\text{support}\left(\sum_{i \in S} \mathbf{a}_i\right) = E(S, V \setminus S)$$

Ingredient 3: l_0-Sampling

Ingredient 3: l_0-Sampling

- Lemma: Exists random $C \in R^{d \times m}$ with $d=O(\log^2 m)$ such that for any $a \in R^m$

$$C\mathbf{a} \longrightarrow e \in \text{support}(\mathbf{a})$$

with probability 9/10.

[Cormode, Muthukrishnan, Rozenbaum 05; Jowhari, Saglam, Tardos 11]

Recipe: Sketch & Compute on Sketches

Recipe: Sketch & Compute on Sketches

- Sketch: Apply log n sketches C_i to each a_j
- Run Algorithm in Sketch Space:

Recipe: Sketch & Compute on Sketches

- Sketch: Apply log n sketches C_i to each a_j
- Run Algorithm in Sketch Space:
  - Use $C_1 a_j$ to get incident edge on each node j

Recipe: Sketch & Compute on Sketches

- Sketch: Apply log n sketches C_i to each a_j
- Run Algorithm in Sketch Space:
  - Use $C_1 a_j$ to get incident edge on each node j
  - For i=2 to t:

Recipe: Sketch & Compute on Sketches

- Sketch: Apply log n sketches C_i to each a_j
- Run Algorithm in Sketch Space:
  - Use $C_1 a_j$ to get incident edge on each node j
  - For i=2 to t:
    - To get incident edge on supernode $S \subset V$ use:

$$\sum_{j \in S} C_i \mathbf{a}_j = C_i \left(\sum_{j \in S} \mathbf{a}_j \right)$$

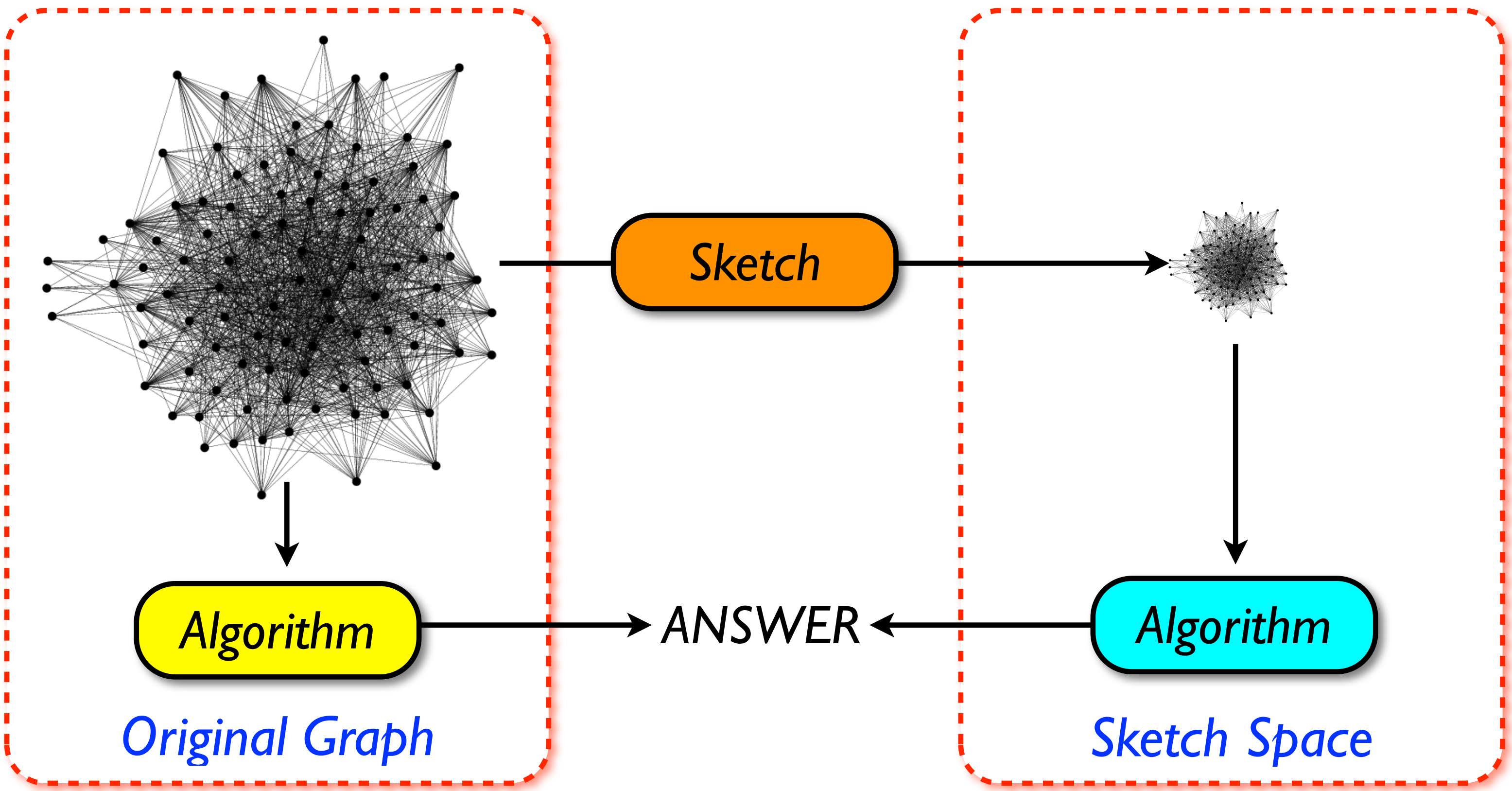
Recipe: Sketch & Compute on Sketches

- Sketch: Apply log n sketches C_i to each a_j
- Run Algorithm in Sketch Space:
  - Use $C_1 a_j$ to get incident edge on each node j
  - For i=2 to t:
    - To get incident edge on supernode $S \subset V$ use:

$$\sum_{j \in S} C_i \mathbf{a}_j = C_i \left(\sum_{j \in S} \mathbf{a}_j \right) \longrightarrow e \in \text{support}(\sum_{j \in S} \mathbf{a}_j) = E(S, V \setminus S)$$

Connectivity

- *Thm:* Can check connectivity with $O(n\log^3 n)$-size sketch.
- *Main Idea:* *a)* Sketch! *b)* Run Algorithm in Sketch-Space



- *Catch:* Sketch must be homomorphic for algorithm operations.

a) Connectivity

b) Applications

a) Connectivity

b) Applications

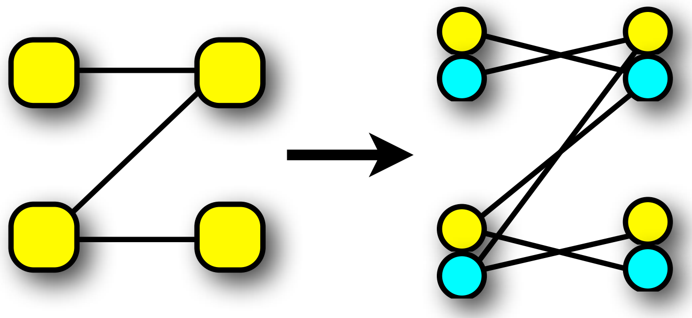
Bipartiteness

Bipartiteness

- *Idea:* Given G, define graph G’ where a node v becomes v_1 and v_2 and edge (u,v) becomes (u_1,v_2) and (u_2,v_1).

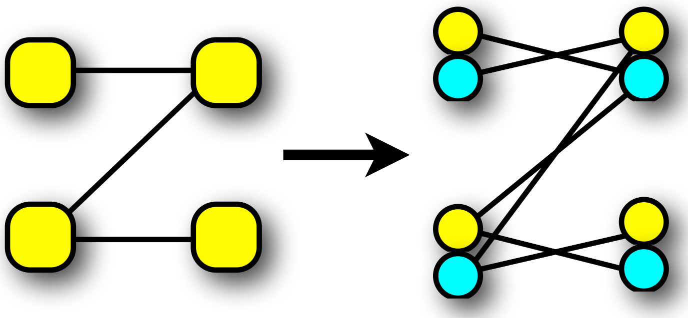
Bipartiteness

- *Idea:* Given G, define graph G' where a node v becomes v_1 and v_2 and edge (u,v) becomes (u_1,v_2) and (u_2,v_1).



Bipartiteness

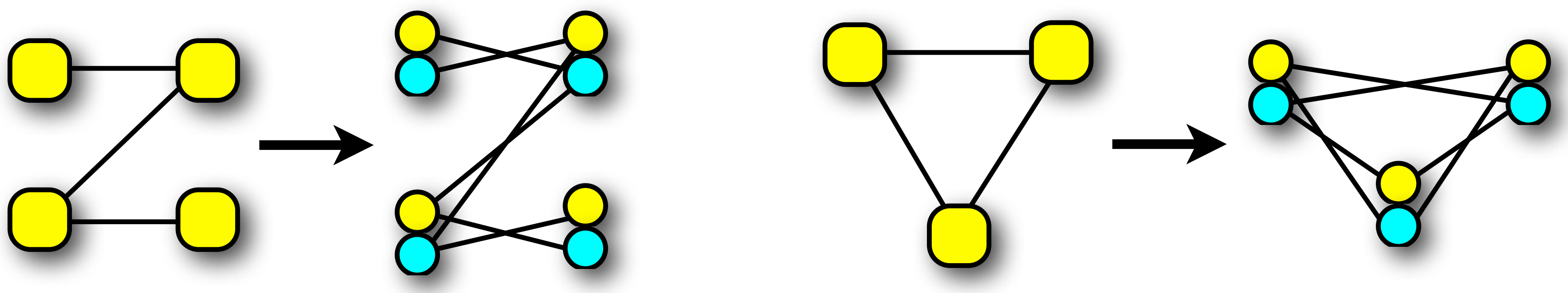
- *Idea:* Given G, define graph G' where a node v becomes v_1 and v_2 and edge (u,v) becomes (u_1,v_2) and (u_2,v_1).



- *Lemma:* Number of connected components doubles iff G is bipartite. Can sketch G' implicitly.

Bipartiteness

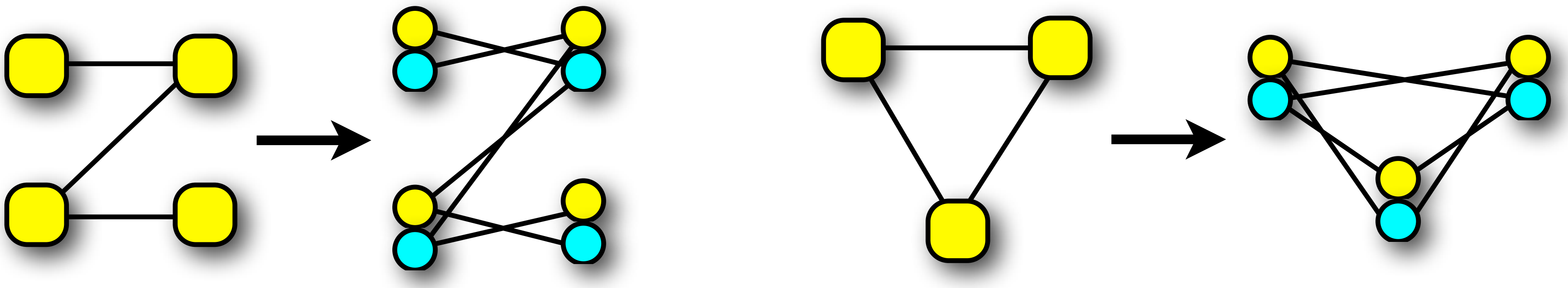
- ***Idea:*** Given G, define graph G' where a node v becomes v_1 and v_2 and edge (u,v) becomes (u_1,v_2) and (u_2,v_1).



- ***Lemma:*** Number of connected components doubles iff G is bipartite. Can sketch G' implicitly.

Bipartiteness

- *Idea:* Given G, define graph G' where a node v becomes v_1 and v_2 and edge (u,v) becomes (u_1,v_2) and (u_2,v_1).



- *Lemma:* Number of connected components doubles iff G is bipartite. Can sketch G' implicitly.
- *Thm:* Can check bipartiteness with $O(n\log^3 n)$-size sketch.

Minimum Spanning Tree

Minimum Spanning Tree

- *Idea:* If n_i is the number of connected components if we ignore edges with weight greater than $(1+\varepsilon)^i$, then:

$$w(\mathrm{MST}) \leq \sum_i \epsilon(1+\epsilon)^i n_i \leq (1+\epsilon) w(\mathrm{MST})$$

Minimum Spanning Tree

- *Idea:* If n_i is the number of connected components if we ignore edges with weight greater than $(1+\varepsilon)^i$, then:

$$w(\mathrm{MST}) \leq \sum_i \epsilon(1+\epsilon)^i n_i \leq (1+\epsilon)w(\mathrm{MST})$$

- *Thm:* Can $(1+\varepsilon)$ approximate MST in one-pass dynamic semi-streaming model.

Minimum Spanning Tree

- *Idea:* If n_i is the number of connected components if we ignore edges with weight greater than $(1+\varepsilon)^i$, then:

$$w(\mathrm{MST}) \leq \sum_i \epsilon(1+\epsilon)^i n_i \leq (1+\epsilon) w(\mathrm{MST})$$

- *Thm:* Can $(1+\varepsilon)$ approximate MST in one-pass dynamic semi-streaming model.

- *Thm:* Can find exact MST in dynamic semi-streaming model using O(log n/log log n) passes.

k-Connectivity

k-Connectivity

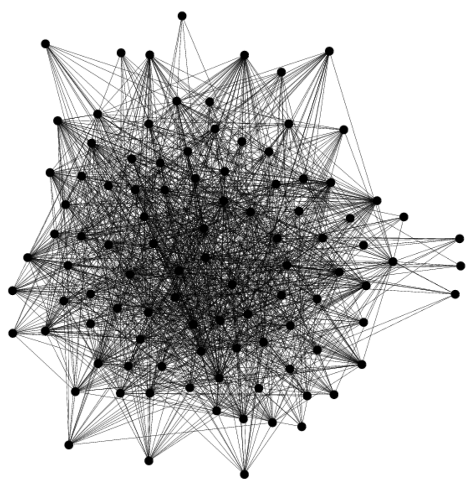
- A graph is k-connected if every cut has size $\geq$ k.

k-Connectivity

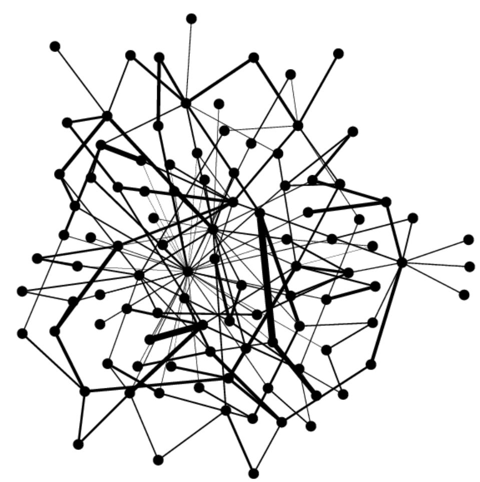
- A graph is k-connected if every cut has size ≥ k.
- *Thm:* Can check k-connectivity with $O(nk\log^3 n)$-size sketch.

k-Connectivity

- A graph is k-connected if every cut has size ≥ k.
- *Thm:* Can check k-connectivity with $O(nk\log^3 n)$-size sketch.
- *Extension:* There exists a $O(\varepsilon^{-2} n\log^5 n)$-size sketch with which we can approximate all cuts up to a factor $(1+\varepsilon)$.



Original Graph



Sparsifier Graph

Ingredient 1: Basic Algorithm

Ingredient 1: Basic Algorithm

- Algorithm (k-Connectivity):

Ingredient 1: Basic Algorithm

- Algorithm (k-Connectivity):

  1. Let F_1 be spanning forest of $G(V,E)$

Ingredient 1: Basic Algorithm

- Algorithm (k-Connectivity):
  1. Let F_1 be spanning forest of $G(V,E)$
  2. For $i=2$ to k:
     2.1. Let F_i be spanning forest of $G(V,E-F_1-\ldots-F_{i-1})$

Ingredient 1: Basic Algorithm

- Algorithm (k-Connectivity):
  1. Let F_1 be spanning forest of $G(V,E)$
  2. For $i=2$ to k:
     2.1. Let F_i be spanning forest of $G(V,E-F_1-\ldots-F_{i-1})$
- Lemma: $G(V,F_1+\ldots+F_k)$ is k-connected iff $G(V,E)$ is.

Ingredient 2: Connectivity Sketches

Ingredient 2: Connectivity Sketches

- Sketch: Simultaneously construct k independent sketches $\{M_1A_G, M_2A_G, \ldots M_kA_G\}$ for connectivity.

Ingredient 2: Connectivity Sketches

- Sketch: Simultaneously construct k independent sketches $\{M_1A_G, M_2A_G, \ldots M_kA_G\}$ for connectivity.
- Run Algorithm in Sketch Space:
  - Use M^1A_G, to find a spanning forest F_1 of G

Ingredient 2: Connectivity Sketches

- Sketch: Simultaneously construct k independent sketches $\{M_1A_G, M_2A_G, \ldots M_kA_G\}$ for connectivity.
- Run Algorithm in Sketch Space:
  - Use M^1A_G, to find a spanning forest F_1 of G
  - Use $M^2A_G - M^2A_{F1} = M^2(A_G - A_{F1}) = M^2(A_{G-F1})$ to find F_2

Ingredient 2: Connectivity Sketches

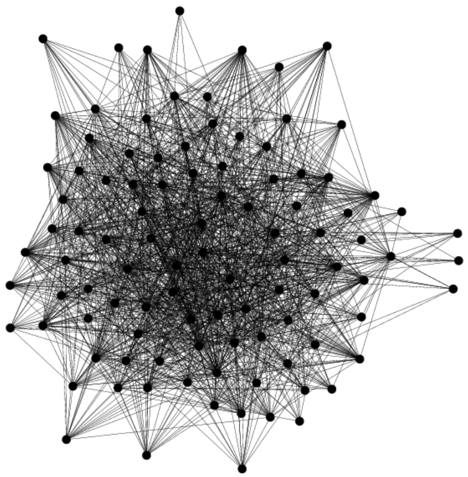
- Sketch: Simultaneously construct k independent sketches $\{M_1A_G, M_2A_G, \ldots M_kA_G\}$ for connectivity.
- Run Algorithm in Sketch Space:
  - Use M^1A_G, to find a spanning forest F_1 of G
  - Use $M^2A_G - M^2A_{F1} = M^2(A_G - A_{F1}) = M^2(A_{G-F1})$ to find F_2
  - Use $M^3A_G - M^3A_{F1} - M^3A_{F2} = M^3(A_{G-F1-F2})$ to find F_3

Ingredient 2: Connectivity Sketches

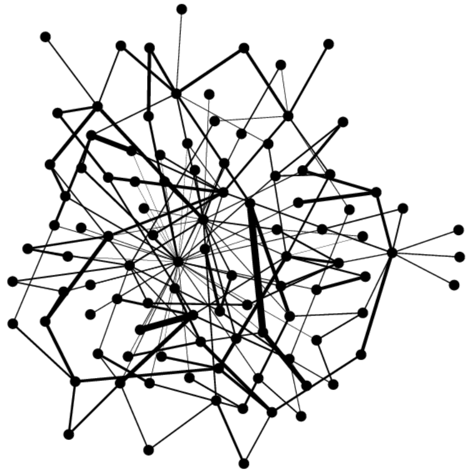
- Sketch: Simultaneously construct k independent sketches $\{M_1A_G, M_2A_G, \ldots M_kA_G\}$ for connectivity.
- Run Algorithm in Sketch Space:
  - Use M^1A_G, to find a spanning forest F_1 of G
  - Use $M^2A_G-M^2A_{F1}=M^2(A_G-A_{F1})=M^2(A_{G-F1})$ to find F_2
  - Use $M^3A_G-M^3A_{F1}-M^3A_{F2}=M^3(A_{G-F1-F2})$ to find F_3
  - etc.

k-Connectivity

- A graph is k-connected if every cut has size ≥ k.
- *Thm:* Can check k-connectivity with $O(nk\log^3 n)$-size sketch.
- *Extension:* There exists a $O(\varepsilon^{-2} n\log^4 n)$-size sketch with which we can approximate all cuts up to a factor $(1+\varepsilon)$.



Original Graph



Sparsifier Graph

Summary

- *Graph Sketches:* Initiates the study of linear projections that preserve structural properties of graphs. Application to dynamic-graph streams and are embarrassingly parallelizable.
- *Properties:* Connectivity, sparsifiers, spanners, bipartite, minimum spanning trees, small cliques, matchings, ...

ありがとう！