CPSC 768:
Scalable and Private Graph Algorithms

Lecture 7: Streaming Maximum Matching

Quanquan C. Liu
quanquan.liu@yale.edu

Announcements

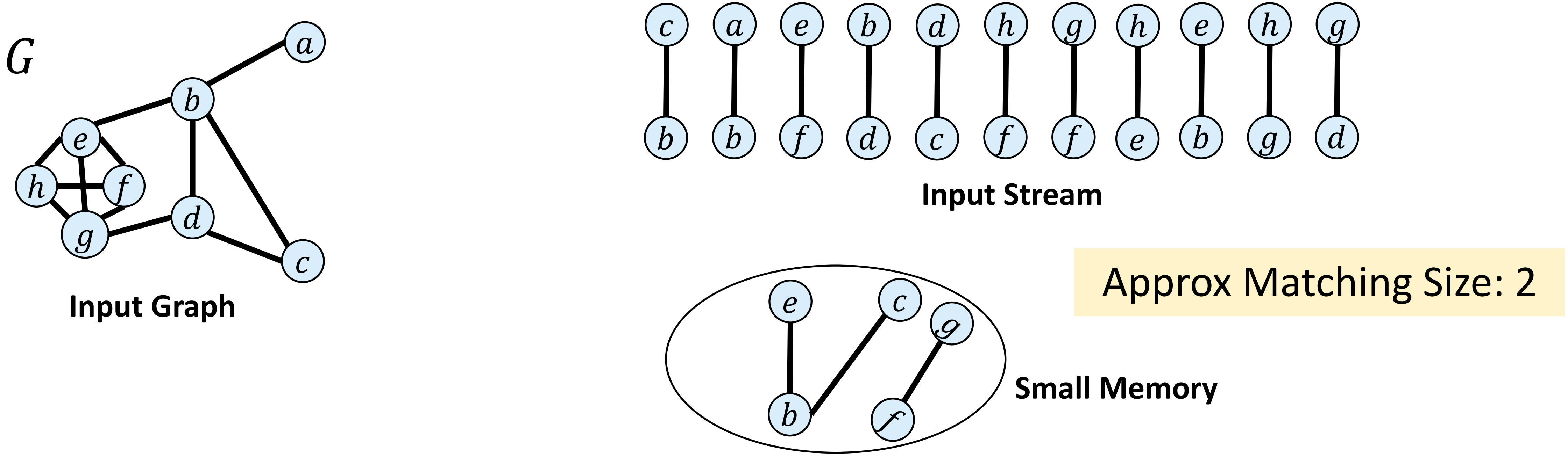
- Check the latest announcement on Canvas:
    - Scheduling lectures
    - Link for joining CPSC 768 Slack

Last Time: Maximum Matching in Bounded Arboricity Graphs

- **Problem**: Given an insertion-only arbitrary-order stream of edges, find an approximate size of the maximum matching in the graph using small space

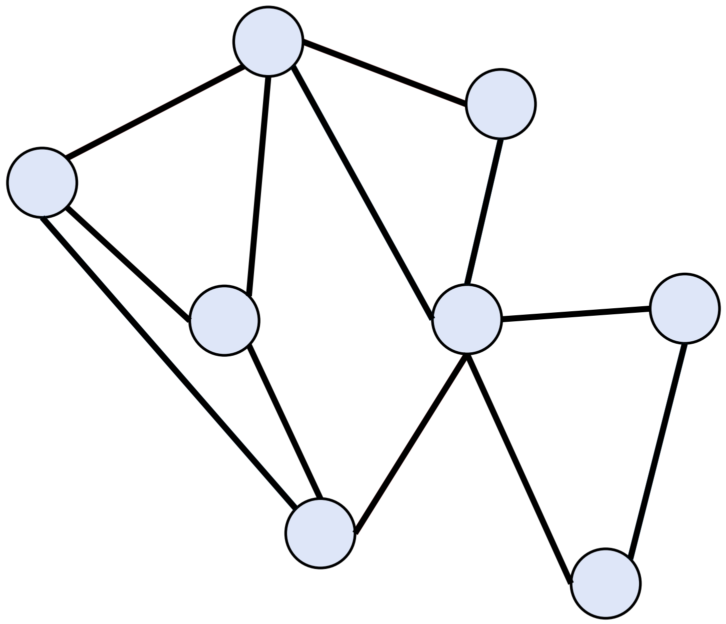
Last Time: Maximum Matching in Bounded Arboricity Graphs

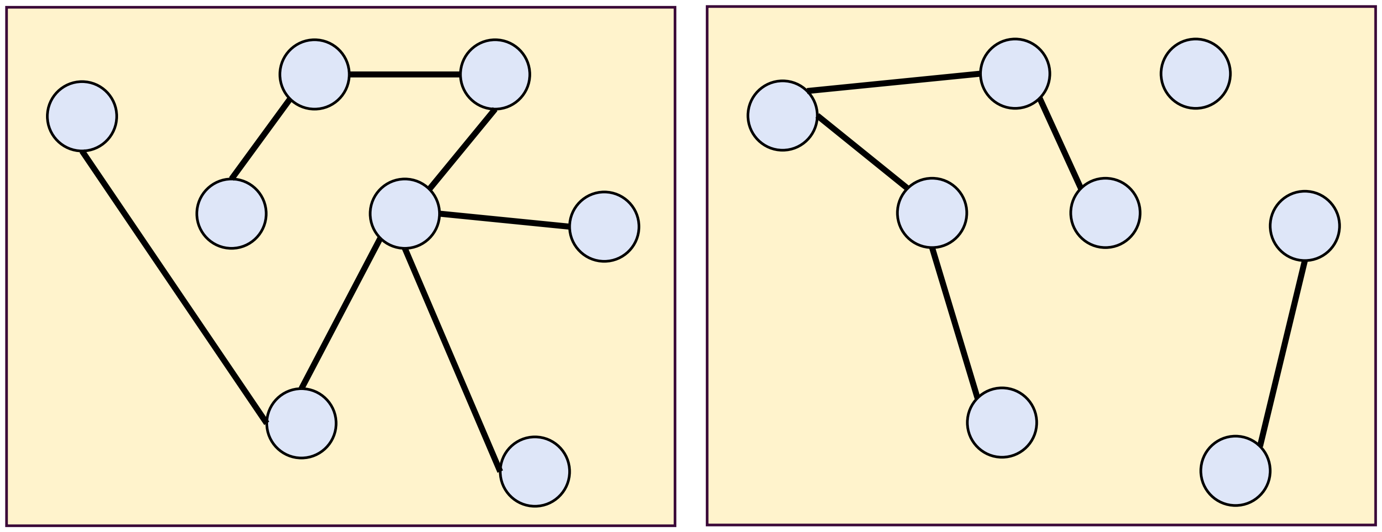
- **Problem**: Given an insertion-only arbitrary-order stream of edges, find an approximate size of the maximum matching in the graph using small space



Arboricity of the Graph

- **Arboricity** of the graph
  - Minimum number of forests to decompose the graph





Properties of Arboricity

- Related to the **density** of the graph

By Nash-Williams Theorem:

$$\boldsymbol{\alpha} = \max_{\boldsymbol{S}} \left\{ \left\lceil \frac{\boldsymbol{m_S}}{\boldsymbol{n_S - 1}} \right\rceil \right\}$$

Properties of Arboricity

- Related to the **density** of the graph

By Nash-Williams Theorem:

$$\boldsymbol{\alpha} = \max_{\boldsymbol{S}} \left\{ \left\lceil \frac{\boldsymbol{m_S}}{\boldsymbol{n_s} - \boldsymbol{1}} \right\rceil \right\}$$

- Every subgraph $S \subseteq G$ **has arboricity** $\leq \alpha$

Properties of Arboricity

- Related to the **density** of the graph

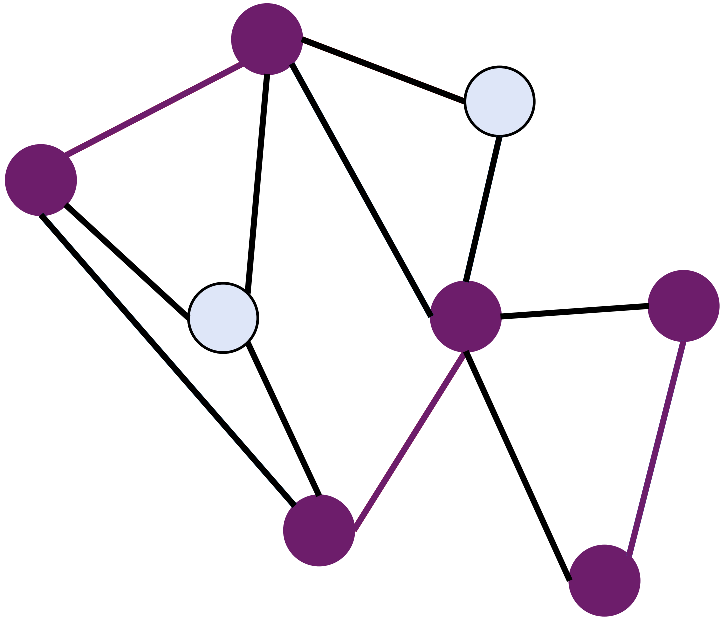
By Nash-Williams Theorem:

$$\boldsymbol{\alpha} = \max_{\boldsymbol{S}} \left\{ \left\lceil \frac{\boldsymbol{m_S}}{\boldsymbol{n_s - 1}} \right\rceil \right\}$$

- Every subgraph $S \subseteq G$ **has arboricity** $\leq \alpha$
- Subgraph S has at most $\alpha \cdot V(S)$ **edges**

Maximum Matching

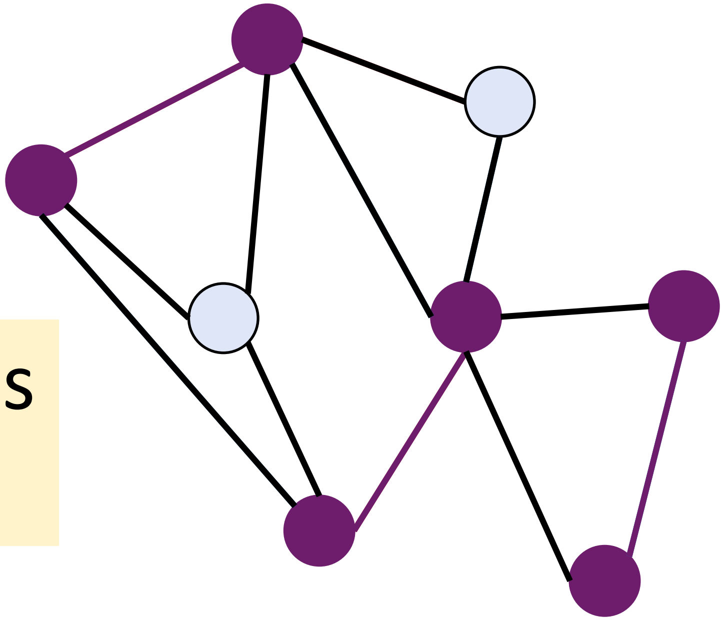
- A **matching** in a graph is a set of edges where no two edges share an endpoint



Maximum Matching

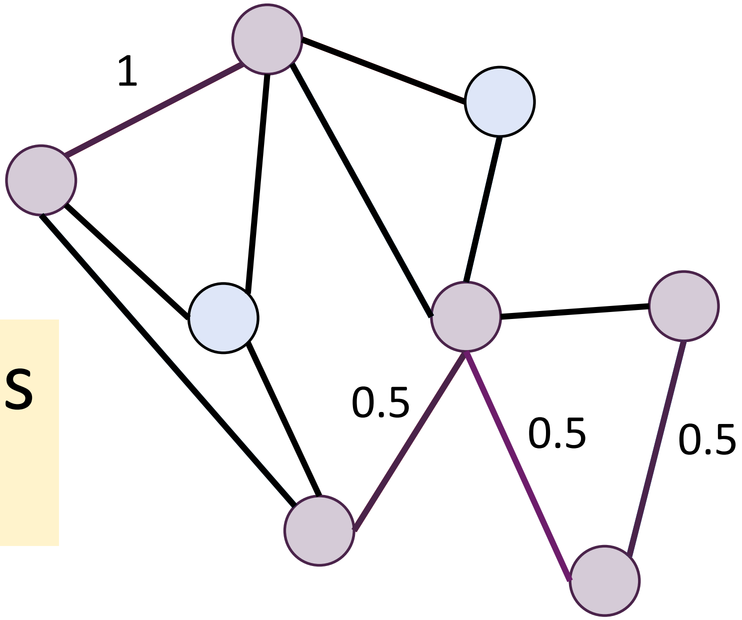
- A **matching** in a graph is a set of edges where no two edges share an endpoint

Maximum Matching has **Maximum Cardinality**



Maximum Matching

- A **matching** in a graph is a set of edges where no two edges share an endpoint



Fractional Matching:

$$\forall v \in V: \sum_{e \ni v} f(e) \leq 1$$

Maximum Matching has **Maximum Cardinality**

Strategy for Streaming Algorithms

1. Figure out quantity to approximate and gives approximation of the quantity we want to approximate
2. Approximate the quantity via sampling and prove concentration bounds

Strategy for Streaming Algorithms

1. **Figure out quantity to approximate and gives approximation of the quantity we want to approximate**
2. Approximate the quantity via sampling and prove concentration bounds

Strategy for Streaming Algorithms

1. **Approximate $|E_\alpha|$** to approximate **$M(G)$ the maximum matching size**
   - E_α is set of edges $\{u, v\}$ where **u and v both incident to at most α edges** that show up **later in the stream**

Lemma 1: $M(G) \leq |E_\alpha| \leq (\alpha + 2) \cdot M(G)$

Strategy for Streaming Algorithms

Lemma 1: $M(G) \leq |E_\alpha| \leq (\alpha + 2) \cdot M(G)$

Last time: Proved $|E_\alpha| \leq (\alpha + 2) \cdot M(G)$ via defining fractional matching $Y_e = \frac{1}{\alpha+1}$ if $e \in E_\alpha$ and 0 otherwise

Edmond's Matching Polytope Corollary:

Let $\{Y_e\}_{e \in E}$ be a fractional matching where the **maximum weight** on any edge is $\boldsymbol{\eta}$. Then, $\boldsymbol{\sum_{e \in E} Y_e \leq (1 + \eta) \cdot M(G)}$.

Strategy for Streaming Algorithms

Lemma 1: $M(G) \leq |E_\alpha| \leq (\alpha + 2) \cdot M(G)$

Last time: Proved $|E_\alpha| \leq (\alpha + 2) \cdot M(G)$ via defining fractional matching $Y_e = \frac{1}{\alpha+1}$ if $e \in E_\alpha$ and 0 otherwise.

Thus, $\frac{1}{\alpha+1} \cdot |E_\alpha| \leq \left(1 + \frac{1}{\alpha+1}\right) \cdot M(G) = \frac{\alpha+2}{\alpha+1} \cdot M(G)$ and so

$$|E_\alpha| \leq (\alpha + 2) \cdot M(G)$$

Strategy for Streaming Algorithms

Lemma 1: $M(G) \leq |E_\alpha| \leq (\alpha + 2) \cdot M(G)$

Last time: Proving $M(G) \leq |E_\alpha|$.

Strategy for Streaming Algorithms

Lemma 1: $M(G) \leq |E_\alpha| \leq (\alpha + 2) \cdot M(G)$

Last time: Proving $M(G) \leq |E_\alpha|$.

- Defined $\boldsymbol{B_u}$ for each $u \in V$ as set of **$\boldsymbol{\alpha + 1}$ edges incident to $\boldsymbol{u}$** that arrive **last in the stream**

Strategy for Streaming Algorithms

Lemma 1: $M(G) \leq |E_\alpha| \leq (\alpha + 2) \cdot M(G)$

Last time: Proving $M(G) \leq |E_\alpha|$.

- Defined $\boldsymbol{B_u}$ for each $u \in V$ as set of **$\boldsymbol{\alpha + 1}$ edges incident to $\boldsymbol{u}$** that arrive **last in the stream**
- Defined **good** edge $\{u, v\} \in B_u \cap B_v$

Strategy for Streaming Algorithms

<u>Lemma 1</u>: $M(G) \leq |E_\alpha| \leq (\alpha + 2) \cdot M(G)$

Last time: Proving $M(G) \leq |E_\alpha|$.

- Defined $\boldsymbol{B_u}$ for each $u \in V$ as set of **$\boldsymbol{\alpha + 1}$ edges incident to $\boldsymbol{u}$** that arrive **last in the stream**
- Defined **good** edge $\{u, v\} \in B_u \cap B_v$
- Defined **wasted** edge $\{a, b\} \in B_a \oplus B_b$

E_α is **exactly** set of good edges

Strategy for Streaming Algorithms

E_α is **exactly** set of good edges

- **Heavy vertices H:** set of vertices with degree at least $\alpha + 1$

Strategy for Streaming Algorithms

E_α is **exactly** set of good edges

- **Heavy vertices $\boldsymbol{H}$:** set of vertices with degree at least $\alpha + 1$
- $\boldsymbol{w}$:= number of good edges with **no** endpoints in H

Strategy for Streaming Algorithms

E_α is **exactly** set of good edges

- **Heavy vertices H:** set of vertices with degree at least $\alpha + 1$
- w := number of good edges with **no** endpoints in H
- x := number of good edges with **one** endpoint in H

Strategy for Streaming Algorithms

E_α is **exactly** set of good edges

- **Heavy vertices $\boldsymbol{H}$:** set of vertices with degree at least $\alpha + 1$
- $\boldsymbol{w}$ $:=$ number of good edges with **no** endpoints in H
- $\boldsymbol{x}$ $:=$ number of good edges with **one** endpoint in H
- $\boldsymbol{y}$ $:=$ number of good edges with **two** endpoints in H

Strategy for Streaming Algorithms

E_α is **exactly** set of good edges

- **Heavy vertices $\boldsymbol{H}$:** set of vertices with degree at least $\alpha + 1$
- $\boldsymbol{w}$ ≔ number of good edges with **no** endpoints in H
- $\boldsymbol{x}$ ≔ number of good edges with **one** endpoint in H
- $\boldsymbol{y}$ ≔ number of good edges with **two** endpoints in H
- $\boldsymbol{z}$ ≔ number of **wasted** edges with **two** endpoints in H

Strategy for Streaming Algorithms

E_α is **exactly** set of good edges

- **Heavy vertices $\boldsymbol{H}$:** set of vertices with degree at least $\alpha + 1$
- $\boldsymbol{w}$:= number of good edges with **no** endpoints in H
- $\boldsymbol{x}$:= number of good edges with **one** endpoint in H
- $\boldsymbol{y}$:= number of good edges with **two** endpoints in H
- $\boldsymbol{z}$:= number of **wasted** edges with **two** endpoints in H

$$|E_\alpha| = w + x + y$$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- $\boldsymbol{w}$ ≔ **no** endpoints in H
- $\boldsymbol{x}$ ≔ **one** endpoint in H
- $\boldsymbol{y}$ ≔ **two** endpoints in H
- $\boldsymbol{z}$ ≔ **wasted** edges with **two** endpoints in H

- First, figure out $x + y$ by stating some facts

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- $\boldsymbol{w}$:= **no** endpoints in H
- $\boldsymbol{x}$:= **one** endpoint in H
- $\boldsymbol{y}$:= **two** endpoints in H
- $\boldsymbol{z}$:= **wasted** edges with **two** endpoints in H

- First, figure out $x + y$ by stating some facts
  - Number of edges in the $\boldsymbol{B_u}$ **of every $\boldsymbol{u} \in \boldsymbol{H}$ incident to good edge**

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- $\boldsymbol{w}$:= **no** endpoints in H
- $\boldsymbol{x}$:= **one** endpoint in H
- $\boldsymbol{y}$:= **two** endpoints in H
- $\boldsymbol{z}$:= **wasted** edges with **two** endpoints in H

- First, figure out $x + y$ by stating some facts
  - Number of edges in the $\boldsymbol{B_u}$ **of every $\boldsymbol{u} \in \boldsymbol{H}$ incident to good edge**
  - $(\alpha + 1)|H| = x + 2y + z$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- w := **no** endpoints in H
- x := **one** endpoint in H
- y := **two** endpoints in H
- z := **wasted** edges with **two** endpoints in H

- First, figure out $x + y$ by stating some facts
  1. Number of edges in the **B_u of every $u \in H$ incident to good edge**
     - $(\alpha + 1)|H| = x + 2y + z$
  2. Good and wasted edges:

     $z + y \leq \alpha \cdot |H|$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- $\boldsymbol{w}$:= **no** endpoints in H
- $\boldsymbol{x}$:= **one** endpoint in H
- $\boldsymbol{y}$:= **two** endpoints in H
- $\boldsymbol{z}$:= **wasted** edges with **two** endpoints in H

- First, figure out $x + y$ by stating some facts
  1. Number of edges in the **$\boldsymbol{B_u}$ of every $\boldsymbol{u \in H}$ incident to good edge**
     - $(\alpha + 1)|H| = x + 2y + z$
  2. Good and wasted edges:

     $$z + y \leq \alpha \cdot |H|$$

Properties of Arboricity

- Related to the **density** of the graph

By Nash-Williams Theorem:

$$\boldsymbol{\alpha} = \max_{\mathbf{S}} \left\{ \left\lceil \frac{\boldsymbol{m_s}}{\boldsymbol{n_s} - \mathbf{1}} \right\rceil \right\}$$

- Every subgraph $S \subseteq G$ **has arboricity** $\leq \alpha$
- Subgraph S has at most $\alpha \cdot V(S)$ **edges**

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- $\boldsymbol{w}$:= **no** endpoints in H
- $\boldsymbol{x}$:= **one** endpoint in H
- $\boldsymbol{y}$:= **two** endpoints in H
- $\boldsymbol{z}$:= **wasted** edges with **two** endpoints in H

All edges in H:
at most $\alpha \cdot |H|$ of them

- First, figure out $x + y$ by stating some facts
  1. Number of edges in the **$\boldsymbol{B_u}$ of every $\boldsymbol{u \in H}$ incident to good edge**
     - $(\alpha + 1)|H| = x + 2y + z$
  2. Good and wasted edges:
     $z + y \leq \alpha \cdot |H|$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- w := **no** endpoints in H
- x := **one** endpoint in H
- y := **two** endpoints in H
- z := **wasted** edges with **two** endpoints in H

- Hence,
  - $x + 2y + z = (\alpha + 1)H$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- w := **no** endpoints in H
- x := **one** endpoint in H
- y := **two** endpoints in H
- z := **wasted** edges with **two** endpoints in H

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- w := **no** endpoints in H
- x := **one** endpoint in H
- y := **two** endpoints in H
- z := **wasted** edges with **two** endpoints in H

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x + y \geq |H|}$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- w := **no** endpoints in H
- x := **one** endpoint in H
- y := **two** endpoints in H
- z := **wasted** edges with **two** endpoints in H

Define $\boldsymbol{E_L}$ be set of edges with **no endpoints in $\boldsymbol{H}$**

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x + y \geq |H|}$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- w := **no** endpoints in H
- x := **one** endpoint in H
- y := **two** endpoints in H
- z := **wasted** edges with **two** endpoints in H

Define E_L be set of edges with no endpoints in H

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x + y \geq |H|}$
- **Every edge in E_L is good**

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- w := **no** endpoints in H
- x := **one** endpoint in H
- y := **two** endpoints in H
- z := **wasted** edges with **two** endpoints in H

Define $\boldsymbol{E_L}$ be set of edges with **no endpoints in $\boldsymbol{H}$**

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x} + \boldsymbol{y} \geq |\boldsymbol{H}|$
- **Every edge in $\boldsymbol{E_L}$ is good**
- $\boldsymbol{w} = |\boldsymbol{E_L}|$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y$$

- w := **no** endpoints in H
- x := **one** endpoint in H
- y := **two** endpoints in H
- z := **wasted** edges with **two** endpoints in H

Define $\boldsymbol{E_L}$ be set of edges with **no endpoints in $\boldsymbol{H}$**

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x + y \geq |H|}$
- **Every edge in $\boldsymbol{E_L}$ is good**
- $\boldsymbol{w = |E_L|}$
- Therefore,

  $\boldsymbol{x + y + w \geq |H| + |E_L|}$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y \geq |\boldsymbol{H}| + |\boldsymbol{E_L}|$$

- $\boldsymbol{w}$:= **no** endpoints in H
- $\boldsymbol{x}$:= **one** endpoint in H
- $\boldsymbol{y}$:= **two** endpoints in H
- $\boldsymbol{z}$:= **wasted** edges with **two** endpoints in H

Define $\boldsymbol{E_L}$ be set of edges with **no endpoints in $\boldsymbol{H}$**

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x} + \boldsymbol{y} \geq |\boldsymbol{H}|$
- **Every edge in $\boldsymbol{E_L}$ is good**
- $\boldsymbol{w} = |\boldsymbol{E_L}|$
- Therefore,
  $\boldsymbol{x} + \boldsymbol{y} + \boldsymbol{w} \geq |\boldsymbol{H}| + |\boldsymbol{E_L}|$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y \geq |\boldsymbol{H}| + |\boldsymbol{E_L}|$$

- **Relate back to $\boldsymbol{M(G)}$**

Define $\boldsymbol{E_L}$ be set of edges with **no endpoints in $\boldsymbol{H}$**

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x + y \geq |H|}$
- **Every edge in $\boldsymbol{E_L}$ is good**
- $\boldsymbol{w = |E_L|}$
- Therefore,

  $\boldsymbol{x + y + w \geq |H| + |E_L|}$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y \geq |\boldsymbol{H}| + |\boldsymbol{E_L}|$$

- **Relate back to $M(G)$**
  - What is the size of $M(G)$ in relation to $|H|$ and $|E_L|$?

Define $\boldsymbol{E_L}$ be set of edges with **no endpoints in $\boldsymbol{H}$**

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x + y \geq |H|}$
- **Every edge in $\boldsymbol{E_L}$ is good**
- $\boldsymbol{w = |E_L|}$
- Therefore,

  $\boldsymbol{x + y + w \geq |H| + |E_L|}$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y \geq |\boldsymbol{H}| + |\boldsymbol{E_L}|$$

- **Relate back to $\boldsymbol{M(G)}$**
  - What is the size of $M(G)$ in relation to $|H|$ and $|E_L|$?
  - Every edge in E_L could be in matching

Define $\boldsymbol{E_L}$ be set of edges with **no endpoints in $\boldsymbol{H}$**

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x + y \geq |H|}$
- **Every edge in $\boldsymbol{E_L}$ is good**
- $\boldsymbol{w = |E_L|}$
- Therefore,

  $$\boldsymbol{x + y + w \geq |H| + |E_L|}$$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y \geq |\boldsymbol{H}| + |\boldsymbol{E_L}|$$

- **Relate back to $\boldsymbol{M(G)}$**
  - What is the size of $M(G)$ in relation to $|H|$ and $|E_L|$?
  - Every edge in E_L could be in matching
  - Remaining edges incident to H
    - At most one edge incident to each $u \in H$

- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x} + \boldsymbol{y} \geq |\boldsymbol{H}|$
- **Every edge in $\boldsymbol{E_L}$ is good**
- $\boldsymbol{w} = |\boldsymbol{E_L}|$
- Therefore,

  $\boldsymbol{x} + \boldsymbol{y} + \boldsymbol{w} \geq |\boldsymbol{H}| + |\boldsymbol{E_L}|$

Strategy for Streaming Algorithms

$$|E_\alpha| = w + x + y \geq |\boldsymbol{H}| + |\boldsymbol{E_L}| \geq \boldsymbol{M}(\boldsymbol{G})$$

- **Relate back to $\boldsymbol{M}(\boldsymbol{G})$**
  - What is the size of $M(G)$ in relation to $|H|$ and $|E_L|$?
  - Every edge in E_L could be in matching
  - Remaining edges incident to H
    - At most one edge incident to each $u \in H$
- Hence,
  - $x + 2y + z = (\alpha + 1)H$
  - $-(z + y) \geq -\alpha \cdot |H|$
  - $\boldsymbol{x} + \boldsymbol{y} \geq |\boldsymbol{H}|$
- **Every edge in $\boldsymbol{E_L}$ is good**
- $\boldsymbol{w} = |\boldsymbol{E_L}|$
- Therefore,
  $\boldsymbol{x} + \boldsymbol{y} + \boldsymbol{w} \geq |\boldsymbol{H}| + |\boldsymbol{E_L}|$

Strategy for Streaming Algorithms

- **Most of our work is proving:**

Lemma 1: $M(G) \leq |E_\alpha| \leq (\alpha + 2) \cdot M(G)$

Strategy for Streaming Algorithms

1. Figure out quantity to approximate and gives approximation of the quantity we want to approximate
2. **Approximate the quantity via sampling and prove concentration bounds**

Approximating $|E_\alpha|$

- Let $\boldsymbol{G_t}$ be the graph defined by the prefix of the stream consisting of first $\boldsymbol{t}$ **edges**

Approximating $|E_\alpha|$

- Let $\boldsymbol{G_t}$ be the graph defined by the prefix of the stream consisting of first $\boldsymbol{t}$ **edges**
  - Let $\boldsymbol{E_\alpha^t}$ be the set of good edges in this prefix

Approximating $|E_\alpha|$

- Let $\boldsymbol{G_t}$ be the graph defined by the prefix of the stream consisting of first $\boldsymbol{t}$ **edges**
  - Let $\boldsymbol{E_\alpha^t}$ be the set of good edges in this prefix
  - Let $\boldsymbol{E^*} = \max_{\boldsymbol{t}} (|\boldsymbol{E_\alpha^t}|)$

Then, $\boldsymbol{M(G) \leq E^* \leq (\alpha + 2) \cdot M(G)}$
since $E^* \geq |E_\alpha|$ and $M(G_t) \leq M(G)$

Approximating $|E_\alpha|$

- Let $\boldsymbol{G_t}$ be the graph defined by the prefix of the stream consisting of first $\boldsymbol{t}$ **edges**
  - Let $\boldsymbol{E_\alpha^t}$ be the set of good edg
  - Let $\boldsymbol{E^* = \max_t (|E_\alpha^t|)}$

Question: does $|E_\alpha^t|$ ever drop as t increases?

Then, $\boldsymbol{M(G) \leq E^* \leq (\alpha + 2) \cdot M(G)}$ since $E^* \geq |E_\alpha|$ and $M(G_t) \leq M(G)$

Approximating E^*

Theorem: Can approximate E^* to $(1+\varepsilon)$-approximation in $O\left(\frac{\log(n)}{\varepsilon^2}\right)$ space whp.

Approximating E^*

- Intuition: sample edges from E_α^t to obtain accurate approximation of $|E_\alpha^t|$

Theorem: Can approximate E^* to $(1+\varepsilon)$-approximation in $O\left(\frac{\log(n)}{\varepsilon^2}\right)$ space whp.

Approximating E^* Algorithm

- Intuition: sample edges from E_α^t to obtain accurate approximation of $|E_\alpha^t|$
- For each sampled edge $e = \{u, v\}$, **store c_e^u and c_e^v** for degrees of u and v in the rest of the stream

Approximating E^* Algorithm

- Intuition: sample edges from E_α^t to obtain accurate approximation of $|E_\alpha^t|$
- For each sampled edge $e = \{u, v\}$, **store $\boldsymbol{c_e^u}$ and $\boldsymbol{c_e^v}$** for degrees of u and v in the rest of the stream
  - If either $\boldsymbol{c_e^u}$ or $\boldsymbol{c_e^v}$ exceeds α delete $\{u, v\}$

Approximating E^* Algorithm

1. Initialize $S \leftarrow \emptyset$, $p = 1$, estimate = 0

Approximating E^* Algorithm

1. Initialize $S \leftarrow \emptyset$, $p = 1$, estimate = 0
2. For each $\{u, v\}$ in stream:

Approximating E^* Algorithm

1. Initialize $S \leftarrow \emptyset$, $p = 1$, estimate = 0
2. For each $\{u, v\}$ in stream:
   a) With probability p add $S \leftarrow S \cup \{u, v\}$, initialize counters

Add new sampled edges

Approximating E^* Algorithm

1. Initialize $S \leftarrow \emptyset$, $p = 1$, estimate = 0
2. For each $\{u, v\}$ in stream:
    a) With probability p add $S \leftarrow S \cup \{u, v\}$, initialize counters
    b) For each edge $e' \in S$, if e' shares endpoint w with e:

Approximating E^* Algorithm

1. Initialize $S \leftarrow \emptyset$, $p = 1$, estimate = 0
2. For each $\{u, v\}$ in stream:
   a) With probability p add $S \leftarrow S \cup \{u, v\}$, initialize counters
   b) For each edge $e' \in S$, if e' shares endpoint w with e:
      i. Increment $c_{e'}^{w}$

Check the counters of previously sampled edges

Approximating E^* Algorithm

1. Initialize $S \leftarrow \emptyset$, $p = 1$, estimate = 0
2. For each $\{u, v\}$ in stream:
   a) With probability p add $S \leftarrow S \cup \{u, v\}$, initialize counters
   b) For each edge $e' \in S$, if e' shares endpoint w with e:
      i. Increment $c_{e'}^{w}$
      ii. If $c_{e'}^{w} > \alpha$, remove e' and corresponding counters from S

Remove edge if it is no longer in E_{α}^{t}

Approximating E^* Algorithm

1. Initialize $S \leftarrow \emptyset$, $p = 1$, estimate = 0
2. For each $\{u, v\}$ in stream:
   a) With probability p add $S \leftarrow S \cup \{u, v\}$, initialize counters
   b) For each edge $e' \in S$, if e' shares endpoint w with e:
      i. Increment $c_{e'}^{w}$
      ii. If $c_{e'}^{w} > \alpha$, remove e' and corresponding counters from S
   c) If $|S| > 80\, \varepsilon^{-2} \log n$:

Approximating E^* Algorithm

1. Initialize $S \leftarrow \emptyset$, $p = 1$, estimate = 0
2. For each $\{u, v\}$ in stream:
   a) With probability p add $S \leftarrow S \cup \{u, v\}$, initialize counters
   b) For each edge $e' \in S$, if e' shares endpoint w with e:
      i. Increment $c_{e'}^{w}$
      ii. If $c_{e'}^{w} > \alpha$, remove e' and corresponding counters from S
   c) If $|S| > 80\,\varepsilon^{-2} \log n$:
      i. Set $p \leftarrow \frac{p}{2}$

If you used too much space, reduce sampling rate

Approximating E^* Algorithm

1. Initialize $S \leftarrow \emptyset$, $p = 1$, estimate = 0
2. For each $\{u, v\}$ in stream:
   a) With probability p add $S \leftarrow S \cup \{u, v\}$, initialize counters
   b) For each edge $e' \in S$, if e' shares endpoint w with e:
      i. Increment $c_{e'}^{w}$
      ii. If $c_{e'}^{w} > \alpha$, remove e' and corresponding counters from S
   c) If $|S| > 80\, \varepsilon^{-2} \log n$:
      i. Set $p \leftarrow \frac{p}{2}$
      ii. Remove each edge in S with probability ½

Resample previous samples

Approximating E^* Algorithm

1. Initialize $S \leftarrow \emptyset$, $p = 1$, estimate = 0
2. For each $\{u, v\}$ in stream:
   a) With probability p add $S \leftarrow S \cup \{u, v\}$, initialize counters
   b) For each edge $e' \in S$, if e' shares endpoint w with e:
      i. Increment $c_{e'}^{w}$
      ii. If $c_{e'}^{w} > \alpha$, remove e' and corresponding counters from S
   c) If $|S| > 80\,\varepsilon^{-2} \log n$:
      i. Set $p \leftarrow \frac{p}{2}$
      ii. Remove each edge in S with probability ½
   d) Estimate $\leftarrow \max($estimate, $|S|/p)$

Update estimate of E^*

Approximating E^* Algorithm

Theorem: Can approximate E^* to $(1+\varepsilon)$-approximation in $O\left(\frac{\log(n)}{\varepsilon^2}\right)$ space whp.

- Proof: Let $\boldsymbol{\tau} = \frac{\mathbf{40 \log n}}{\boldsymbol{\varepsilon^2}}$ and level i (starting with $i = 2$) be
  $$\mathbf{2^{i-1}} \cdot \boldsymbol{\tau} \leq |\boldsymbol{E_\alpha^t}| < \mathbf{2^i} \cdot \boldsymbol{\tau}$$
- Define level $i = 1$ to be $\mathbf{0} \leq |\boldsymbol{E_\alpha^t}| < \mathbf{2} \cdot \boldsymbol{\tau}$

Approximating E^* Algorithm

Theorem: Can approximate E^* to $(1+\varepsilon)$-approximation in $O\left(\frac{\log(n)}{\varepsilon^2}\right)$ space whp.

- Proof: Let $\boldsymbol{\tau} = \frac{\mathbf{40 \log n}}{\boldsymbol{\varepsilon^2}}$ and level i (starting with $i = 2$) be
  $$\mathbf{2^{i-1} \cdot \tau \leq |E_\alpha^t| < 2^i \cdot \tau}$$
- Define level $i = 1$ to be $\mathbf{0 \leq |E_\alpha^t| < 2 \cdot \tau}$
- Edge e is **sampled in level i** with probability $\frac{1}{2^i}$ for $i \geq 2$

Approximating E^* Algorithm

- Proof: Let $\boldsymbol{\tau} = \frac{\mathbf{40 \log n}}{\boldsymbol{\varepsilon^2}}$ and level i (starting with $i = 2$) be $\mathbf{2^{i-1} \cdot \tau \leq |E_\alpha^t| < 2^i \cdot \tau}$
- Define level $i = 1$ to be $\mathbf{0 \leq |E_\alpha^t| < 2 \cdot \tau}$
- Edge e is **sampled in level $\boldsymbol{i}$** with probability $\frac{1}{2^i}$ for $i \geq 2$
- For any level i, let's show the probability we get $(1 + \varepsilon)$-approx. of E_α^t

Approximating E^* Algorithm

- Proof: Let $\boldsymbol{\tau} = \frac{\mathbf{40 \log n}}{\boldsymbol{\varepsilon^2}}$ and level i (starting with $i = 2$) be
  $$\mathbf{2^{i-1} \cdot \tau \leq |E_\alpha^t| < 2^i \cdot \tau}$$
- Define level $i = 1$ to be $\mathbf{0 \leq |E_\alpha^t| < 2 \cdot \tau}$
- Edge e is **sampled in level i** with probability $\frac{1}{2^i}$ for $i \geq 2$
- For any level i, let's show the probability we get $(1 + \varepsilon)$-approx. of E_α^t
  - Let S_i^t be the number of edges sampled in level i after t updates

Approximating E^* Algorithm

- Proof: Let $\boldsymbol{\tau} = \frac{\mathbf{40 \log n}}{\boldsymbol{\varepsilon}^2}$ and level i (starting with $i = 2$) be
$$\mathbf{2}^{\boldsymbol{i-1}} \cdot \boldsymbol{\tau} \leq |\boldsymbol{E}_{\boldsymbol{\alpha}}^{\boldsymbol{t}}| < \mathbf{2}^{\boldsymbol{i}} \cdot \boldsymbol{\tau}$$
- Define level $i = 1$ to be $\mathbf{0} \leq |\boldsymbol{E}_{\boldsymbol{\alpha}}^{\boldsymbol{t}}| < \mathbf{2} \cdot \boldsymbol{\tau}$
- Edge e is **sampled in level $\boldsymbol{i}$** with probability $\frac{1}{2^i}$ for $i \geq 2$
- For any level i, let's show the probability we get $(1 + \varepsilon)$-approx. of E_α^t
  - Let S_i^t be the number of edges sampled in level i after t updates

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2 \mu}{3})$

Approximating E^* Algorithm

- Proof: Let $\boldsymbol{\tau} = \frac{\mathbf{40 \log n}}{\boldsymbol{\varepsilon}^2}$ and level i (starting with $i = 2$) be
$$\mathbf{2}^{\boldsymbol{i-1}} \cdot \boldsymbol{\tau} \leq |\boldsymbol{E}_{\boldsymbol{\alpha}}^{\boldsymbol{t}}| < \mathbf{2}^{\boldsymbol{i}} \cdot \boldsymbol{\tau}$$
- Define level $i = 1$ to be $\mathbf{0} \leq |\boldsymbol{E}_{\boldsymbol{\alpha}}^{\boldsymbol{t}}| < \mathbf{2} \cdot \boldsymbol{\tau}$
- Edge e is **sampled in level $\boldsymbol{i}$** with probability $\frac{1}{2^i}$ for $i \geq 2$
- For any level i, let's show the probability we get $(1 + \varepsilon)$-approx. of E_α^t
  - Let S_i^t be the number of edges sampled in level i after t updates

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \geq 2$

Approximating E^* Algorithm

- Proof: Let $\boldsymbol{\tau} = \frac{\mathbf{40 \log n}}{\boldsymbol{\varepsilon^2}}$ and level i (starting with $i = 2$) be
  $$\mathbf{2^{i-1}} \cdot \boldsymbol{\tau} \leq |\boldsymbol{E_\alpha^t}| < \mathbf{2^i} \cdot \boldsymbol{\tau}$$
- Define level $i = 1$ to be $\mathbf{0} \leq |\boldsymbol{E_\alpha^t}| < \mathbf{2} \cdot \boldsymbol{\tau}$
- Edge e is **sampled in level *i*** with probability $\frac{1}{2^i}$ for $i \geq 2$
- For any level i, let's show the probability we get $(1 + \varepsilon)$-approx. of E_α^t
  - Let S_i^t be the number of edges sampled in level i after t updates

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \geq 2$
- $\Pr\left[\left||S_i^t - \mu\right| \geq \varepsilon \cdot \mu\right] \leq 2\exp(\frac{-\varepsilon^2 |E_\alpha^t| \cdot p_i}{3})$
  $\leq 2 \exp\left(-\frac{\varepsilon^2 \cdot 40 \log n}{3 \cdot \varepsilon^2}\right) = \frac{1}{\mathrm{poly}(n)}$

Approximating E^* Algorithm

- Proof: Let $\boldsymbol{\tau} = \frac{\mathbf{40 \log n}}{\boldsymbol{\varepsilon^2}}$ and level i (starting with $i = 2$) be
$$\mathbf{2^{i-1}} \cdot \boldsymbol{\tau} \leq |\boldsymbol{E_\alpha^t}| < \mathbf{2^i} \cdot \boldsymbol{\tau}$$
- Define level $i = 1$ to be $\mathbf{0} \leq |\boldsymbol{E_\alpha^t}| < \mathbf{2} \cdot \boldsymbol{\tau}$
- Edge e is **sampled in level $\boldsymbol{i}$** with probability $\frac{1}{2^i}$ for $i \geq 2$
- For any level i, let's show the probability we get $(1 + \varepsilon)$-approx. of E_α^t
  - Let S_i^t be the number of edges sampled in level i after t updates

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \geq 2$
- $\Pr\left[\left||S_i^t - \mu\right| \geq \varepsilon \cdot \mu\right] \leq 2\exp(\frac{-\varepsilon^2 |E_\alpha^t| \cdot p_i}{3})$

$$\leq 2 \exp\left(-\frac{\varepsilon^2 \cdot 40 \log n}{3 \cdot \varepsilon^2}\right) = \frac{1}{\text{poly}(n)}$$

- What do you notice about the above calculation?

Approximating E^* Algorithm

- Proof: Let $\boldsymbol{\tau} = \frac{\mathbf{40 \log n}}{\boldsymbol{\varepsilon^2}}$ and level i (starting with $i = 2$) be
  $$\mathbf{2^{i-1}} \cdot \boldsymbol{\tau} \leq |\boldsymbol{E_\alpha^t}| < \mathbf{2^i} \cdot \boldsymbol{\tau}$$
- Define level $i = 1$ to be $\mathbf{0} \leq |\boldsymbol{E_\alpha^t}| < \mathbf{2} \cdot \boldsymbol{\tau}$
- Edge e is **sampled in level *i*** with probability $\frac{1}{2^i}$ for $i \geq 2$
- For any level i, let's show the probability we get $(1+\varepsilon)$-approx. of E_α^t
  - Let S_i^t be the number of edges sampled in level i after t updates

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \geq 2$
- $\Pr\left[\left||S_i^t - \mu\right| \geq \varepsilon \cdot \mu\right] \leq 2\exp(\frac{-\varepsilon^2 |E_\alpha^t| \cdot p_i}{3})$
  $\leq 2\exp\left(-\frac{\varepsilon^2 \cdot 40 \log n}{3 \cdot \varepsilon^2}\right) = \frac{1}{\mathrm{poly}(n)}$
- What do you notice about the above calculation?

Approximating E^* Algorithm

- True if $\mu \geq \frac{40 \log n}{3\varepsilon^2}$

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \geq 2$
- $\Pr\left[\left||S_i^t - \mu\right| \geq \varepsilon \cdot \mu\right] \leq 2\exp(\frac{-\varepsilon^2 |E_\alpha^t| \cdot p_i}{3})$
  $\leq 2\exp\left(-\frac{\varepsilon^2 \cdot 40 \log n}{3 \cdot \varepsilon^2}\right) = \frac{1}{\text{poly}(n)}$
- What do you notice about the above calculation?

Approximating E^* Algorithm

- True if $\mu \geq \frac{40 \log n}{3\varepsilon^2}$
- What if $\mu < \frac{40 \log n}{3\varepsilon^2}$?
  - Thoughts?

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2\mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \geq 2$
- $\Pr\left[\left||S_i^t - \mu\right| \geq \varepsilon \cdot \mu\right] \leq 2\exp(\frac{-\varepsilon^2|E_\alpha^t| \cdot p_i}{3})$

  $\leq 2\exp\left(-\frac{\varepsilon^2 \cdot 40 \log n}{3 \cdot \varepsilon^2}\right) = \frac{1}{\text{poly}(n)}$
- What do you notice about the above calculation?

Approximating E^* Algorithm

- True if $\mu \geq \frac{40 \log n}{3\varepsilon^2}$
- What if $\mu < \frac{40 \log n}{3\varepsilon^2}$?
  - Thoughts?

Need to prove that p_i matches the level whp

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \geq 2$
- $\Pr\left[\left||S_i^t| - \mu\right| \geq \varepsilon \cdot \mu\right] \leq 2\exp(\frac{-\varepsilon^2 |E_\alpha^t| \cdot p_i}{3})$

  $\leq 2 \exp\left(-\frac{\varepsilon^2 \cdot 40 \log n}{3 \cdot \varepsilon^2}\right) = \frac{1}{\mathrm{poly}(n)}$
- What do you notice about the above calculation?

Approximating E^* Algorithm

- Take the union bound over $t \leq n^2$, then with probability at least $1 - \frac{1}{\text{poly}(n)}$:

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \geq 2$
- $\Pr\left[\left||S_i^t - \mu\right| \geq \varepsilon \cdot \mu\right] \leq 2\exp(\frac{-\varepsilon^2 |E_\alpha^t| \cdot p_i}{3})$

  $\leq 2\exp\left(-\frac{\varepsilon^2 \cdot 40 \log n}{3 \cdot \varepsilon^2}\right) = \frac{1}{\text{poly}(n)}$

Approximating E^* Algorithm

- Take the union bound over $t \leq n^2$, then with probability at least $1 - \frac{1}{\text{poly}(n)}$:
  - $\frac{S_i^t}{p_i} = |E_\alpha^t| \pm \varepsilon \cdot |E_\alpha^t|$ for all t

- Multiplicative Chernoff Bound:
  - $\Pr[|X \ - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \geq 2$
- $\Pr\left[\left||S_i^t \ - \mu\right| \geq \varepsilon \cdot \mu\right] \leq 2\exp(\frac{-\varepsilon^2 |E_\alpha^t| \cdot p_i}{3})$

  $\leq 2\exp\left(-\frac{\varepsilon^2 \cdot 40 \log n}{3 \cdot \varepsilon^2}\right) = \frac{1}{\text{poly}(n)}$

Approximating E^* Algorithm

- Take the union bound over $t \leq n^2$, then with probability at least $1 - \frac{1}{\text{poly}(n)}$:
  - $\frac{S_i^t}{p_i} = |E_\alpha^t| \pm \varepsilon \cdot |E_\alpha^t|$ for all t

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \geq \varepsilon \cdot \mu] \leq 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \geq 2$
- $\Pr\left[\left||S_i^t - \mu\right| \geq \varepsilon \cdot \mu\right] \leq 2\exp(\frac{-\varepsilon^2 |E_\alpha^t| \cdot p_i}{3})$

  $\leq 2\exp\left(-\frac{\varepsilon^2 \cdot 40 \log n}{3 \cdot \varepsilon^2}\right) = \frac{1}{\text{poly}(n)}$

Theorem: Can approximate E^* to $(1+\varepsilon)$-approximation in $O\left(\frac{\log(n)}{\varepsilon^2}\right)$ space whp.

Approximating E^* Algorithm

- Take the union bound over $t \le n^2$, then with probability at least $1 - \frac{1}{\text{poly}(n)}$:
  - $\frac{S_i^t}{p_i} = |E_\alpha^t| \pm \varepsilon \cdot |E_\alpha^t|$ for all t

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \ge \varepsilon \cdot \mu] \le 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- $\mu = p_i \cdot |E_\alpha^t|$ for $i \ge 2$
- $\Pr[|S_i^t - \mu| \ge \varepsilon \cdot \mu] \le 2\exp(\frac{-\varepsilon^2 |E_\alpha^t| \cdot p_i}{3})$

  $\le 2\exp(\frac{\quad}{3 \cdot \varepsilon^2}) \le \frac{1}{\text{poly}(n)}$

Lemma 1: $M(G) \le E^* \le (\alpha + 2) \cdot M(G)$

Theorem: Can approximate E^* to $(1+\varepsilon)$-approximation in $O\left(\frac{\log(n)}{\varepsilon^2}\right)$ space whp.

Approximating E^* Algorithm

- Take the union bound over $t \le n^2$, then with probability at least $1 -$ ...
  pol...
  - ...
    all i

- Multiplicative Chernoff Bound:
  - $\Pr[|X - \mu| \ge \varepsilon \cdot \mu] \le 2\exp(-\frac{\varepsilon^2 \mu}{3})$
- ...

Theorem: Can approximate $\boldsymbol{M(G)}$ to $\boldsymbol{(2+\alpha)(1+\varepsilon)}$**-approximation** in $\boldsymbol{O\left(\frac{\log(n)\cdot\log(\alpha)}{\varepsilon^2}\right)}$ **space whp.**

On Wednesday

- Streaming Bipartite Matching using **Auction Algorithms**
  - Another more intuitive way to solve maximum matching in bipartite graphs than Hungarian algorithm or maxflow!