Advanced Topics in Theoretical Computer Science

Lecture 14

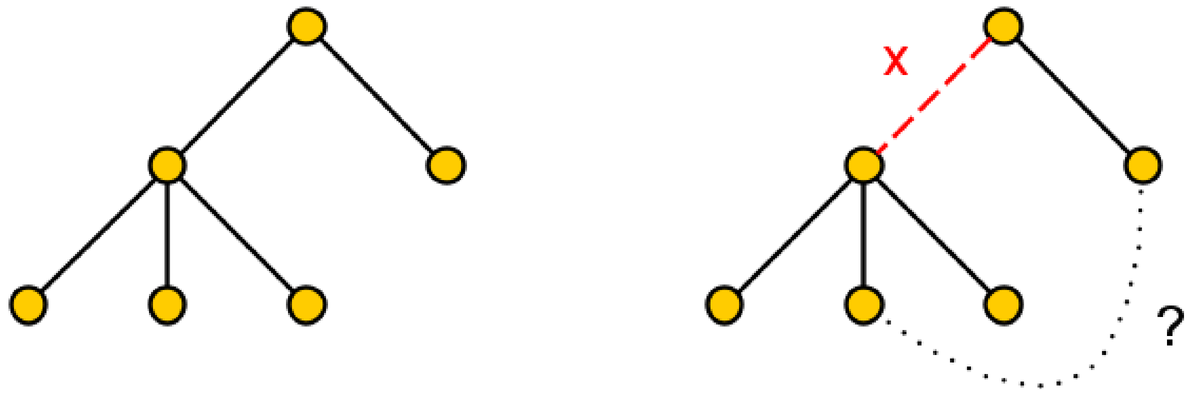
The AGM sketch: Spanning Forests in Insertion-deletion Streams

Connectivity in Insertion-deletion Streams

Insertion-only Streams:

- Maintain a spanning forest
- Semi-streaming space ($O(n \log n)$ space)

Can we maintain a spanning forest in Insertion-deletion Streams?



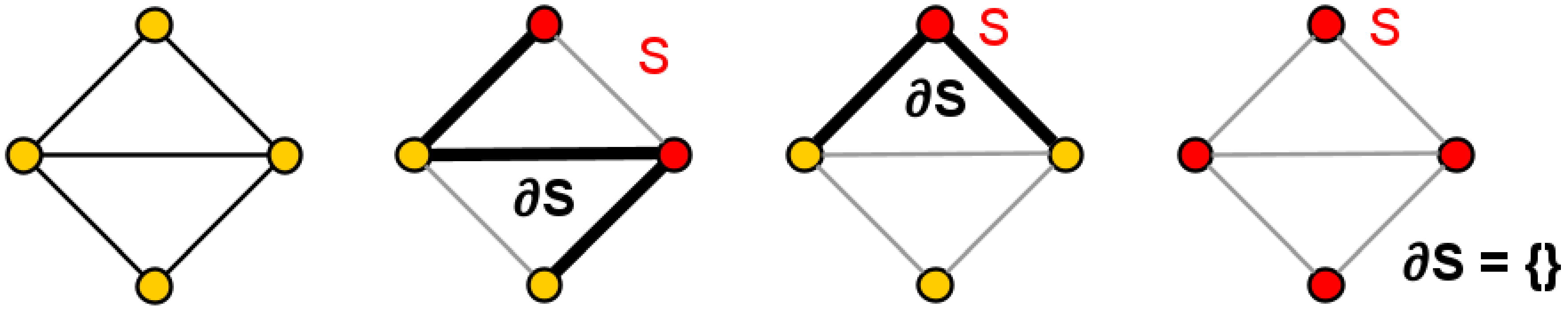
No way of remembering all potential replacement edges…

Boundary Edges

Definition: Let $G = (V, E)$ be a graph. For each $S \subseteq V$, the boundary ∂S is defined as:

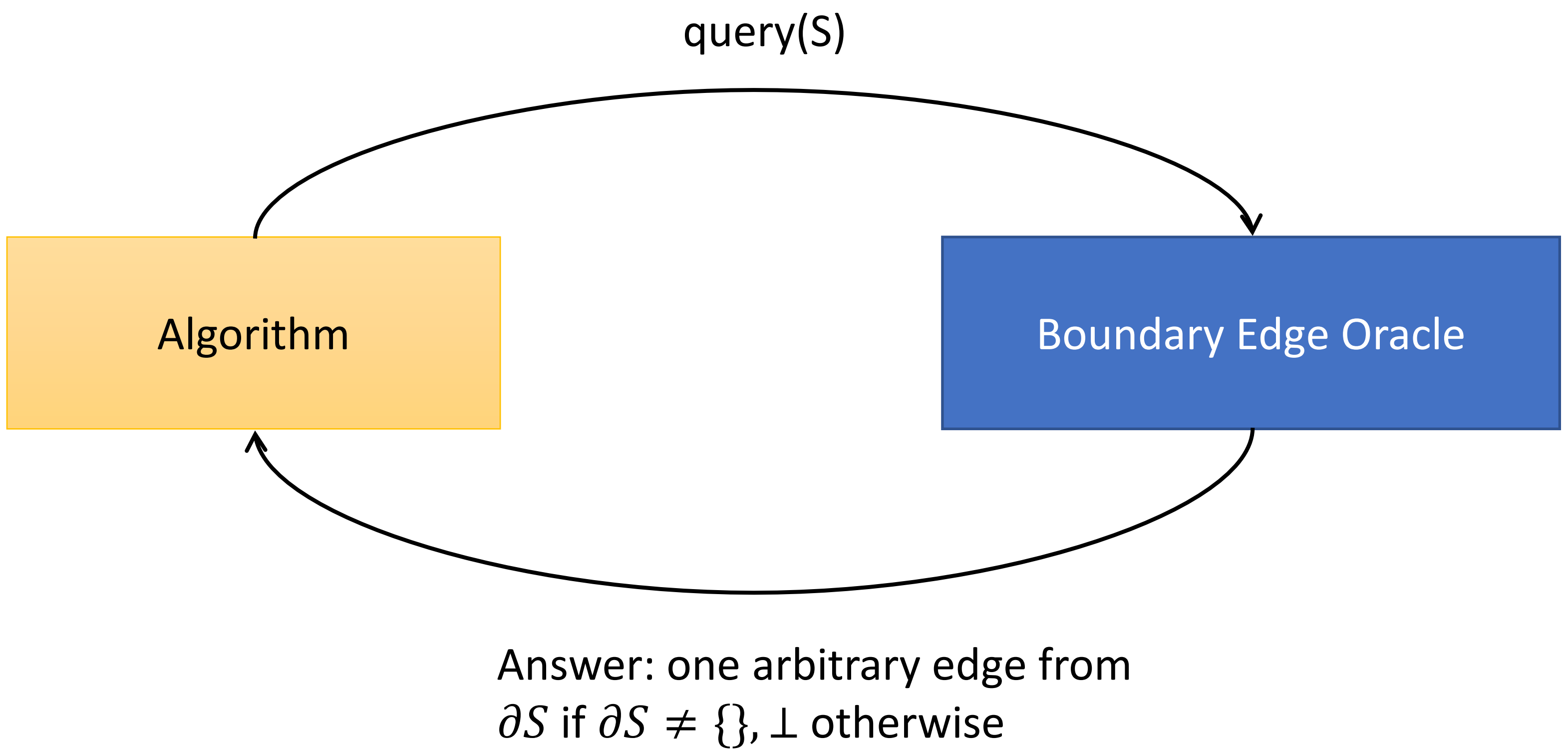
$$\partial S \coloneqq \{e \in E : |e \cap S| = 1\}.$$

"∂S is the set of edges with exactly one endpoint in S"



Spanning Trees via a "Boundary Edge Oracle"

Boundary Edge Oracle:



How can we compute a spanning tree with a Boundary Edge Oracle?

Offline Algorithm (Boruvka's Algorithm)

Algorithm: (input: graph $G = (V, E)$, output: spanning forest in G)

1. $F \leftarrow \{\}$
2. $C \leftarrow \left\{\{v\} : v \in V\right\}$
3. **repeat**
    1. Query boundary edge for each $S \in C$ and collect returned edges in H
    2. $F \leftarrow$ spanning forest in graph $(V, F \cup H)$
    3. $C \leftarrow \{V(T) : T \text{ is a connected component of } (V, F)\}$

    until $H = \{\ \}$
4. **return** F

Offline Algorithm - Analysis

Observation. A component $S \in C$ with $\partial S = \{\,\}$ is a connected component in G**.**

Lemma. Let $S \in C$ be the smallest component such that ∂S is non-empty before iteration i. Then, every component after iteration i is of size at least $2|S|$.

Proof. Let $F \in C$ be an arbitrary component with non-empty boundary. By construction of the algorithm, F is merged with at least one other component T. Hence, the resulting component is of size at least $|F| + |T| \geq |S| + |S| = 2|S|$.

□

Corollary. The size of the smallest component with non-empty boundary doubles in each iteration.

Offline Algorithm – Analysis II

Theorem. Boruvka's algorithm computes a spanning forest and terminates in at most $\log n$ rounds.

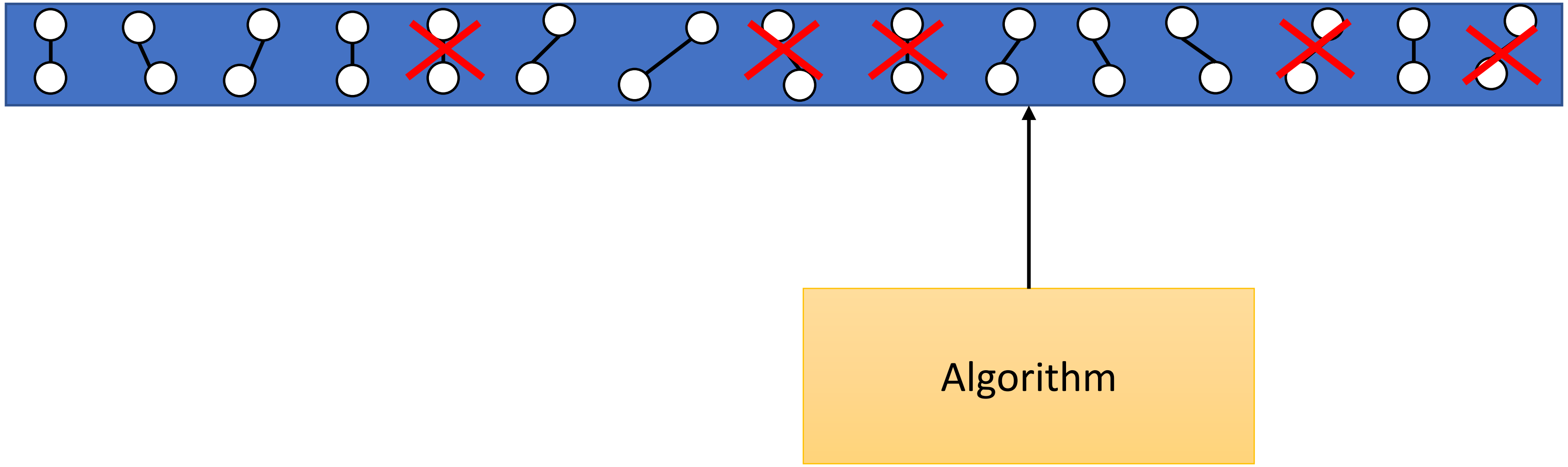
Proof.

Since the size of the smallest component with non-empty boundary doubles in each iteration, the smallest component with non-empty boundary after round i is of size at least 2^i. Since every component is of size at most n, we have:

$$2^i \leq n \ \Rightarrow i \leq \log n.$$

□

Boundary Edge Oracle and Streaming



Strategy:

- While processing the stream: Compute data structure D that is able to answer boundary edge queries
- Use D in a post-processing step to implement Boruvka's algorithm

Recap on l_0-sampling

Turnstile stream:

- Stream describes vector $f \in \{-m, \dots, m\}^n$ by updates to its coordinates ($m \in \mathbb{N}$)
- Initially, $f = (0, 0, \dots, 0)$
- Each item in the stream is an update (j, c), meaning $f_j \leftarrow f_j + c$ ($c \in \{-1, 1\}$)

$\boldsymbol{l_0}$-sampling: [Jowhari, Sağlam, Tardos, 2011**]**

There is a turnstile streaming algorithm with space $O\left(\log^2 n \log \frac{1}{\delta}\right)$ that outputs a uniform random coordinate among the non-zero coordinates of f. It succeeds with proba. $1 - \delta$.

Example: $\boldsymbol{f = (2, -4, 0, 0, 1, 0)}$

Then, the l_0-sampler outputs 1, 2, or 5 each with probability $\frac{1}{3}$ (with success prob. $1 - \delta$).

Insertion-deletion Streams are Turnstile Streams

Insertion-deletion Graph Streams are Turnstile Streams:

- Insertion-deletion graph stream describes vector $f \in \{0,1\}^{\binom{n}{2}}$
- l_0-sampling therefore corresponds to sampling one edge from the input graph

Other Applications of l_0-sampling in Graph Streams:

By considering substreams of the input stream we can sample from…

- The set of edges incident to a specific vertex
- A random edge in a specific induced subgraph
- …

Implementing Boundary Edge Oracle for Singletons

First Iteration of Boruvka's Algorithm:

- For each vertex $v \in V$, compute arbitrary incident edge to v
- How can we implement this step in insertion-deletion streams?

Insertion-deletion Streams:

For each vertex $v \in V$, run an l_0-sampler on edges incident to v in order to sample a random incident edge while processing the stream (see example in previous lecture)!

Running n l_0-samplers requires only semi-streaming space!

Implementing Second Round of Boruvka

Second Iteration of Boruvka's Algorithm:

- First iteration yields a collection of forests of arbitrary sizes
- Let $S \in C$ be an arbitrary forest (subset of vertices)
- How can we process the input stream without knowing S so that we can find a boundary edge from ∂S in a post-processing step?

AGM-Sketch!

Kook Jin Ahn, Sudipto Guha, Andrew McGregor:

Analyzing graph structure via linear measurements. SODA 2012: 459-467

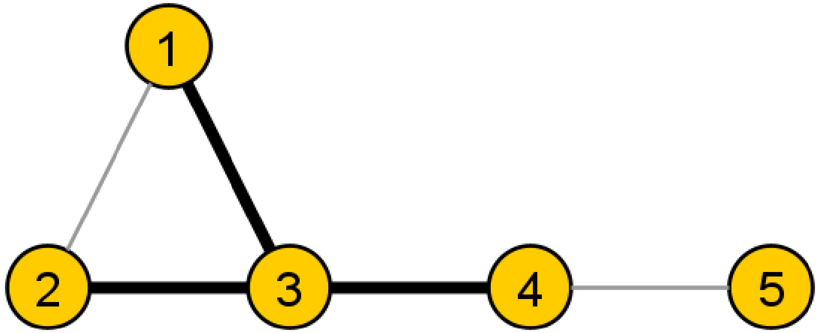
Signed Incidence Matrix

Signed Incidence Matrix $\boldsymbol{B} \in \{-1, 0, 1\}^{n \times \binom{n}{2}}$:

$((x, \{x, y\}) = 1$ if (x, y) is an edge and $x < y$

$((x, \{y, x\}) = -1$ if (x, y) is an edge and $x > y$

$((x, \{y, z\}) = 0$ otherwise



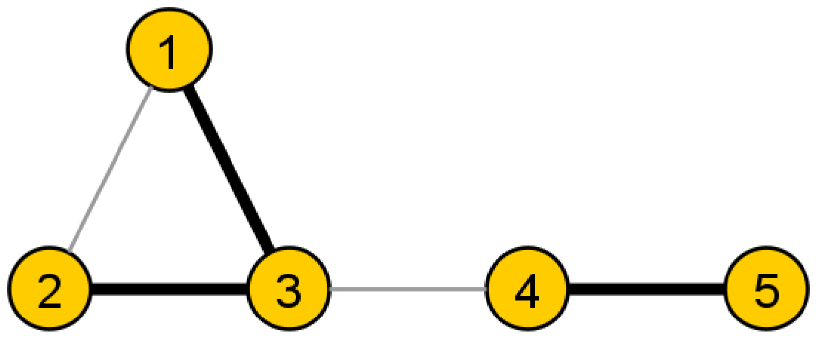
	{1,2}	{1,3}	{1,4}	{1,5}	{2,3}	{2,4}	{2,5}	{3,4}	{3,5}	{4,5}
x_1	1	1	0	0	0	0	0	0	0	0
x_2	−1	0	0	0	1	0	0	0	0	0
x_3	0	−1	0	0	−1	0	0	1	0	0
x_4	0	0	0	0	0	0	0	−1	0	1
x_5	0	0	0	0	0	0	0	0	0	−1

Signed Incidence Matrix and Boundary Edges

Example:

- l_0-sampling of row vector x_3 + row vector x_4

  → Boundary edge of component {3,4}!



	{1,2}	{1,3}	{1,4}	{1,5}	{2,3}	{2,4}	{2,5}	{3,4}	{3,5}	{4,5}
x_1	1	1	0	0	0	0	0	0	0	0
x_2	−1	0	0	0	1	0	0	0	0	0
x_3	0	−1	0	0	−1	0	0	1	0	0
x_4	0	0	0	0	0	0	0	−1	0	1
x_5	0	0	0	0	0	0	0	0	0	−1

Boundary edge of $\boldsymbol{S}$: l_0-sampling of sum of rows associated to vertices in S!

l_0-sampler is a Linear Sketch

l_0-sampling: [Jowhari, Sağlam, Tardos, 2011**]**

There is a turnstile algorithm with space $O\left(\log^2 n \log\frac{1}{\delta}\right)$ that outputs unif. random coordinate among the non-zero coordinates of f. It succeeds with proba. $1-\delta$.

↓ Closer look

Theorem. There exists a random matrix $A \in \mathbb{R}^{O(\log^3 n)\times n}$ s.t. for any $f \in \mathbb{R}^n$, with probability at least $1-\frac{1}{\mathrm{poly}\, n}$, we can learn i for some $f_i \neq 0$. $A\,f$ is a linear sketch.

Streaming Interpretation:

1. Choose random matrix A and set $f' = 0$
2. Upon arrival of update $(j, c) \in [n] \times \{-1, 1\}$, compute $f' \leftarrow f' + c\,A\,e_j$, where e_j is the jth unit vector
3. Upon completion, we can extract a non-zero coordinate of f from f'

l_0-sampler is a Linear Sketch

Useful Properties of Linear Sketches:

1. **Union Bound**: Suppose that we have multiple vectors $f_1, f_2, \ldots, f_t$ then we can determine non-zero elements from everyone of them from $Af_1, Af_2, \ldots, Af_t$ with probability at least $1 - \frac{t}{\text{poly } n}$.

2. **Linearity**: Given Af_1 and Af_2, we can find a non-zero entry from $f_1 + f_2$ since $A(f_1 + f_2) = Af_1 + Af_2$.

Final Algorithm

1. Sample random l_0-sampling matrices $A_1, A_2, \dots, A_{\log n}$
2. Let x_v denote the row vector in the signed incidence matrix B associated to vertex v
3. **While processing the stream:** For every vertex $v \in V$ compute
$$A_1 x_v, A_2 x_v, \dots, A_{\log n} x_v$$
4. **Post-processing:** Emulate Boruvka's algorithm (using Union Bound & Linearity)

   1st round: Can find an incident edge to every vertex v from $A_1 x_v$
   t^{th} round: Suppose we need to find an incident edge from component S. Then we compute the sketch:
$$\sum_{v \in S} A_t x_v = A_t \sum_{v \in S} x_v$$

   and find a boundary edge to S

Analysis

Space:

- Overall, we store $n \log n$ l_0-samplers
- Setting $\delta = \frac{1}{n^3}$ in each l_0-sampler yields overall success probability of at least $1 - \frac{1}{n}$. (union bound)
- This requires only semi-streaming space!

Correctness:

- Observe that in each iteration i of Boruvka, the sketch $A_i x_v$ of any vertex v is needed only once!
- We therefore do not reuse sketches, which would increase the error probability

Summary

AGM Sketch:

- For a long time it was not clear that a spanning forest can be computed using space $o(n^2)$ in insertion-deletion streams
- The AGM sketch is simple but was a surprise to many researchers

Summary Algorithm:

One pass semi-streaming algorithm in insertion-deletion streams for computing a spanning forest (and deciding connectivity)