Sublinear Time and Space Algorithms 2022B – Lecture 8
Streaming of Graphs and Connectivity in Dynamic Graphs*

Robert Krauthgamer

1 Streaming of Graphs

Basic model: Consider an input stream that represents a graph $G = (V, E)$ as a sequence of edges on the vertex set $V = [n]$. Denote $m = |E|$.

It can be viewed as an insertion-only stream of edges. We may allow deletions of edges, and then it is called a dynamic graph stream.

Semi-streaming: The usual aim is space requirement $\tilde{O}(n)$, which can generally be much smaller than the trivial bound $O(m)$ of storing the current graph explicitly (but without extra workspace an algorithm may need).

For many problems, $\Omega(n)$ storage is required (even to get approximate answers).

Connectivity: Determine whether the graph G is connected (or even which pairs $u, v \in V$ are connected).

In the insertions-only model, it can be solved with storage requirement $O(n)$ words, by maintaining a spanning forest...

Distances: Maintain all the distances in the graph, i.e., given a query $u, v \in V$ report their distance.

Theorem: Can be solved within approximation $2k - 1$ (for integer $k \geq 1$) in the insertions-only model with storage requirement $O(n^{1+1/k})$ words.

The idea is to use a greedy spanner construction by [Althofer, Das, Dobkin, Joseph and Soares, 1993].

Proof: Create and store a subgraph G' as follows. When an edge (u, v) arrives, check if the distance between its endpoints in G' is $d_{G'}(u, v) \leq 2k - 1$. If it is not, then add the edge to G' (otherwise, do nothing).

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

It is not difficult to verify that

$$\forall u, v \in V, \qquad d_G(u,v) \le d_{G'}(u,v) \le (2k-1) d_G(u,v).$$

The bound on the number of edges in G' follows by a theorem from extremal graph theory, because its girth (length of shortest cycle) is $g \ge 2k+1$.

Exer: Show how to 2-approximate maximum matching and vertex-cover using space of $O(n)$ words.

2 Connectivity in Dynamic Graphs

Dynamic graph model: The input stream contains insertions and deletions of edges to G. Recall that we assume $V = [n]$.

The tool of choice is linear sketching, where decrements are supported by definition.

Motivations:

a) updates to the graph like removing hyperlinks or un-friending

b) the graph is distributed (each site contains a subset of the edges), and their linear sketches can be easily combined

Theorem [Ahn, Guha and McGregor, 2012]: There is a streaming algorithm with storage $\tilde{O}(n)$ that determines whp whether the graph is connected (In fact, it computes a spanning forest and can determine which pairs of vertices are connected.)

Idea: To grow (increase) connected components, we need to find an outgoing edge from each current component. Using ℓ_0-sampling and especially its linear-sketch form, we can pick an outgoing edge from an arbitrary set. Informally, if we already have a connected component $Q \subset V$, then we will use a method where edges inside Q get canceled, and outgoing edges survive.

Notation: Let $N = \binom{n}{2}$. and for each vertex v define a vector $x^v \in \mathbb{R}^N$ where coordinate $\{i,j\}$ for $i < j$ is given by

$$x^v_{\{i,j\}} = \begin{cases} +1 & \text{if } (i,j) \in E \text{ and } v = i \\ -1 & \text{if } (i,j) \in E \text{ and } v = j \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm AGM:

Update (on a stream/dynamic graph G):

For each vertex v, create a virtual stream for $x^v \in \mathbb{R}^N$ and maintain an ℓ_0-sampler for this x^v (using the same coins, as these are linear sketches that can be added).

Repeat the above $\log n$ times independently (i.e., $\log n$ "levels" of samplers for each $v \in V$).

Output (to determine connectivity):

Initialize a partition $\Pi = \{\{1\}, \dots, \{n\}\}$ where each vertex is in a separate connected component.

Now repeat for $l = 1, \dots, \log n$:

1. For each connected component $Q \in \Pi$, sum the samplers (more precisely, their sketches) for all $v \in Q$ from level l, to obtain a sampler for $\sum_{v \in Q} x^v$. Then activate the sampler to pick a coordinate from $[N]$ (which we will see is a random outgoing from Q).

2. Use the $|Q|$ sampled edges to merge connected components and update Π

Output "connected" if all the vertices are merged into one connected component.

Analysis: To simplify the analysis, we assume henceforth that G is connected (see below), and that the samplers are perfect (i.e. ignore their polynomially-small error probability).

Exer: Extend the analysis to the case that G is not connected, to determine whether $s, t \in V$ given at query time, are connected.

Claim 1: If the number of connected components at the beginning of an iteration is $k > 1$ (and the samplers succeed in producing outgoing edges), then their number at the end of the iteration is at most $k/2$.

Exer: prove this claim.

Claim 2: Fix a set $Q \subset V$. Then $\sum_{v \in Q} x^v$ is nonzero only in coordinates $\{i, j\}$ corresponding to an edge outgoing from Q, i.e., $|Q \cap \{i, j\}| = 1$.

Proof: Was seen in class.

Corollary 3: Fix a set $Q \subset V$. Then summing ℓ_0-samplers of x^v over all $v \in Q$ (assuming these samplers use a linear sketch) creates an ℓ_0-sampler for $\sum_{v \in Q} x^v$ that reports an outgoing edge from Q.

Storage: The main storage is for ℓ_0-samplers for every vertex. Each one requires $O(\log^4 n)$ bits (in the construction seen in class), and we need fresh randomness in each of the $O(\log n)$ iterations (levels), to avoid potential dependencies. Thus the total storage is $O(n \log^5 n)$ bits.