

# Sublinear Time and Space Algorithms 2026A – Lecture 7

## 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, i.e., find all connected components).

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

**Distances:** Compute the distances in the graph, i.e., given as query  $u, v \in V$  report their distance.

**Theorem:** All distances can be computed within approximation  $2k - 1$  (for integer  $k \geq 1$ ) in insertions-only streams 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. Initially  $G'$  is empty. When an edge  $(u, v)$  arrives, check if the distance between its endpoints  $u, v$  in the current graph  $G'$  is  $d_{G'}(u, v) > 2k - 1$ . If it is, add the edge to  $G'$  (otherwise, do nothing).

As explained in class, it is not difficult to verify that in the final  $G'$ ,

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

---

\*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.

The number of edges in  $G'$  is bounded by  $O(n^{1+1/k})$  by a theorem from extremal graph theory, because its girth (length of shortest cycle) is  $g \geq 2k + 1$ .

**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  $\ell_0$ -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  $N = \binom{n}{2}$ . For each vertex  $v$  define a vector  $x^v \in \mathbb{R}^N$  where coordinate  $\{i, j\}$  for  $i < j$  is given by

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

The above defines  $n$  frequency vectors, where each  $x^v$  is affected only by edges incident to  $v$ .

**Claim 1:** For every  $Q \subset V$ , the support (nonzero coordinates) of  $\sum_{v \in Q} x^v$  is exactly the set of edges between  $Q$  and  $V \setminus Q$ .

**Proof:** Was seen in class.

**Algorithm AGM:**

Update (on a stream/dynamic graph  $G$ ):

1. for each vertex  $v \in V$ , create a virtual stream for  $x^v \in \mathbb{R}^N$  and maintain an  $\ell_0$ -sampler for this  $x^v$  (using the same coins for all  $v$ , as these are linear sketches that may be added).
2. repeat the above  $\log n$  times independently (i.e.,  $\log n$  “levels” of samplers for each  $v \in V$ ).

Output (to determine connectivity):

3. initialize a partition  $\Pi = \{\{1\}, \dots, \{n\}\}$  where each vertex is in a separate connected component.
4. repeat for  $l = 1, \dots, \log n$ :
  5. for each connected component  $Q \in \Pi$ , sum the level  $l$  samplers for all  $v \in Q$ , to obtain a sampler for  $\sum_{v \in Q} x^v$ . Then activate the sampler to obtain a coordinate from  $[N]$  (which we will see is a random outgoing from  $Q$ ).
6. Use the  $|\Pi|$  sampled edges to merge connected components and update  $\Pi$
7. Output “connected” if  $|\Pi| = 1$  (all vertices are in 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  $\delta + 1/n^2$ ).

**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 2:** Fix a set  $Q \subset V$ . Then summing  $\ell_0$ -samplers of  $x^v$  over all  $v \in Q$  (assuming these samplers use a linear sketch and same coins) creates an  $\ell_0$ -sampler for  $\sum_{v \in Q} x^v$  that reports an outgoing edge from  $Q$ .

**Proof:** Follows from Claim 1 since the sketch is linear.

**Claim 3:** 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  $k/2$ .

**Proof:** Was seen in class.

**Storage:** The main storage is for  $\ell_0$ -samplers for each of the  $n$  vertices and each of the  $\log n$  levels. Each sampler requires  $O(\log^4 n)$  bits (in the construction seen in class). Thus the total storage is  $O(n \log^5 n)$  bits.