

# Sublinear Time and Space Algorithms 2026A – Lecture 5

## Dynamic Geometric Streams and Euclidean MST\*

Robert Krauthgamer

### 1 Dynamic Geometric Streams

**Geometric stream:** The input is a stream of points in  $\mathbb{R}^d$  denoted  $P = \langle p_1, \dots, p_n \rangle$ . We usually think of low dimension, even  $d = 2$ , and a very long stream.

The above is an insertion-only stream, but we may allow deletions of points and then it is called dynamic.

**Representation:** We restrict the points to come from an integer grid  $[\Delta]^d$ . We assume a machine word can store a point and count the number of points, and for convenience also that  $n \leq \Delta^d$  (e.g., the points are distinct). Thus, a word consists of  $O(\log(\Delta^d + n)) = O(d \log \Delta)$  bits.

**Problem definition:** The goal could be to solve some optimization problem over  $P$ , for example in the *diameter problem*, the goal is compute  $\text{diam}(P) = \max_{p, p' \in P} \|p - p'\|$ .

**Theorem 1 (Diameter):** In an insertion-only geometric streams, the diameter problem admits 2-approximation using storage of  $O(1)$  words.

Proof: The algorithm stores the first point seen  $p_1$ , and also the point  $p_j$  farthest from it so far. It outputs the distance between these two points. This is the *radius* of  $P$  around  $p_1$ , and clearly

$$\frac{1}{2} \text{diam}(P) \leq \text{output} \leq \text{diam}(P).$$

**Open:** Similar performance, namely  $O(1)$ -approximation with storage that is polynomial in  $d$ , in dynamic streams?

### 2 Euclidean MST

**MST problem:** In the *Minimum Spanning Tree (MST) problem*, we view the points as forming a complete graph with edge weights representing Euclidean distances, and the goal is compute a tree (or connected subgraph) of minimum weight, denoted  $\text{MST}(P)$ .

---

\*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.

We focus on approximating the value/cost (not reporting a tree).

**Exer:** Prove that  $\text{MST}(P)$  is also the cost of single-linkage clustering, an iterative clustering that starts with each point as its own cluster, then repeatedly connects two clusters that have closest to each other.

**Theorem 2 [Indyk 2004]:** There is a streaming algorithm for  $O(d \log \Delta)$ -approximation of MST using storage of  $(d \log \Delta)^{O(1)}$  bits.

The presentation focuses on the case  $d = 2$  for simplicity, the generalization is straightforward.

The key here is that the storage is polynomial in  $d$ . Another algorithm by [Frahling, Indyk and Sohler, 2008] achieves  $(1 + \varepsilon)$ -approximation using storage of  $(\varepsilon^{-1} \log \Delta)^{O(d)}$  words.

**Idea:** Map the grid into a hierarchical tree, then solve MST in this tree using algorithms for the vector-frequency model.

**Quadtree:** Partition the grid  $[\Delta]^2$  recursively, partitioning each square into 4 equal-size squares. (For general  $d$ , the word square should be replaced by cube or cell.)

This can be represented as a *quadtree*, which is 4-regular tree with  $O(\log \Delta)$  levels, where each tree node at level  $i = 0, \dots, \log \Delta$  is a  $2^i \times 2^i$  square. In particular, the root represents the entire grid, and each leaf represents a  $1 \times 1$  square, i.e., a grid point.

We let tree edges have weights, where an edge between square at level  $i$  and its parent at level  $i + 1$  has weight  $2^{i+1}$ . Why? This is the diameter of the smaller square (or a slight overestimate), and thus enough to travel from the “center” (or say lower-left corner) of the small square to the “center” of its parent square. Note that for general  $d$ , we would use  $\sqrt{d}2^i$ .

**Lemma 3:** Denote the quadtree by  $T$ , and distances in it by  $d_T$ . Recall that grid points are also leaves in  $T$ . Then

$$\forall p, q \in [\Delta]^2, \quad d_T(p, q) \geq \|p - q\|.$$

For intuition, assume for now that tree distances are a good approximation to the grid distances, i.e.,  $d_T(p, q) = \Theta(1) \cdot \|p - q\|$ . We will see later that the approximation is worse and holds only in expectation.

**MST inside quadtree  $T$ :** Let  $\text{MST}_T(P)$  denote the cost a tree inside  $T$  that spans  $P$ , viewed as leaves of  $T$ . It is (almost) equivalent to the cost of computing MST under the distances  $d_T$ , see the factor  $\frac{1}{2}$  in Lemma 6.

**Lemma 4:** There is a streaming algorithm that, given a stream of grid points  $P$ , viewed as leaves of  $T$ , computes  $(1 + \varepsilon)$ -approximation of  $\text{MST}_T(P)$  using storage of  $(\varepsilon^{-1} d \log \Delta)^{O(1)}$  bits.

**Algorithm QuadtreeMST:**

Idea: It’s enough to count how many squares are non-empty at each level

1. For each level  $i = 0, \dots, \log \Delta$ , initialize a Distinct Elements (DE) algorithm over the squares (tree nodes) at level  $i$ , and denote its frequency vector  $x^{(i)} \in \mathbb{R}^{(\Delta/2^i)^2}$ .
2. Update (insert/delete  $p$ ): update the square containing  $p$  at each DE algorithm, i.e., each  $x^{(i)}$ .

3. Output: compute DE estimates  $\tilde{z}_i \in (1 \pm \varepsilon)\|x^{(i)}\|_0$ , and report  $\sum_i \mathbf{1}_{\{\tilde{z}_i \geq 1.5\}} \tilde{z}_i \cdot 2^{i+1}$ .

Remark: Make sure each DE algorithm has small error probability, say at most  $1/(8 \log \Delta)$ .

**Proof of Lemma 4:** Was seen in class.

**Shifted quadtree:** Suppose that we translate  $P$  by a random amount, namely pick uniformly random  $r \in [\Delta]^d$  and add it to all of  $P$ , and then compute a partitioning/quadtree on  $[2\Delta]^d$ . Equivalently, compute a recursive partitioning of  $[2\Delta]^d$  and translate its squares by  $-r$ .

**Probabilistic tree metric:** Each leaf of  $T$  still represents a  $1 \times 1$  square, i.e., one grid point. Thus, for every  $p, q \in P$ , their tree distance  $d_T(p, q)$  is a random variable.

**Lemma 5:** Let  $T$  be a randomly shifted quadtree. Then Lemma 3 still holds and

$$\forall p, q \in [\Delta]^d, \quad \mathbb{E}_T[d_T(p, q)] \leq O(d \log \Delta) \|p - q\|.$$

**Proof:** Was seen in class.

So far we analyzed pairwise distances (in  $T$ ). Let's now analyze the MST value.

**Lemma 6:**  $\text{MST}_T(P) \geq \frac{1}{2} \text{MST}(P)$ . (For a randomly shifted quadtree, this holds with probability 1.)

**Exer:** Prove this lemma using what we proved before for pairs  $p, q \in [\Delta]^d$ . Think why the constant  $\frac{1}{2}$  is needed.

**Lemma 7:** Let  $T$  be a randomly shifted quadtree. Then  $\mathbb{E}_T[\text{MST}_T(P)] \leq O(d \log \Delta) \cdot \text{MST}(P)$ .

**Exer:** Prove this lemma.

Hint: Think of the optimal tree for  $P$  (in  $\mathbb{R}^d$ ) and use linearity of expectation.

**Proof of Theorem 2:** The two lemmas above show that with high probability,  $2 \text{MST}_T(P)$  is an  $O(d \log \Delta)$ -approximation for  $\text{MST}(P)$ . We saw earlier how to  $(1 + \varepsilon)$ -approximates  $\text{MST}_T(P)$  using storage  $(\varepsilon^{-1} d \log(\Delta))^{O(1)}$  bits, and we can use it with  $\varepsilon = 0.1$ . Altogether, we obtain a streaming algorithm for Euclidean MST, which proves the theorem.

**Exer:** Verify that the algorithm extends to deletions, assuming that the DE algorithm supports deletions.

**Exer:** Use similar ideas for the minimum bichromatic matching problem (aka earthmover distance), where the points in  $P$  are colored, exactly half in blue and half in red, i.e.,  $P = R \cup B$ , each point arrives with its color (and in arbitrary order), and the goal is to compute a minimum-weight perfect matching between  $R$  and  $B$ .

Hint: Reduce the problem to estimating  $\|x^{(i)}\|_1$  for each level  $i$ .