Sublinear Time and Space Algorithms 2022B – Lecture 7 Euclidean MST (cont'd) and ℓ_0-sampling*

Robert Krauthgamer

1 Euclidean MST (cont'd)

We want to complete the algorithm for Euclidean MST from last class. Previously, we analyzed the distance between a pair of points p, q (how the quadtree distorts their Euclidean distance), and now we analyze the MST of n points.

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

Exer: Prove this lemma using what we proved before for pairs $p, q \in [\Delta]^d$.

Lemma: Let T be a randomly shifted quadtree. Then with high probability $\mathrm{MST}_T(P) \leq O(d \log \Delta) \cdot \mathrm{MST}(P)$.

Proof: Was seen in class, using linearity of expectation to bound $\mathbb{E}_T\left[\mathrm{MST}_T(P)\right.$.

Putting it together: These two lemmas show that with high probability, $2\,\mathrm{MST}_T(P)$ is an $O(d \log \Delta)$-approximation for $\mathrm{MST}(P)$. We saw earlier how to $(1+\varepsilon)$-approximates $\mathrm{MST}_T(P)$ 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: Use similar ideas for the minimum bichromatic matching problem (aka earthmover distance), where the points in P are colored, half in blue and half in red, i.e., $P = R \cup B$, 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.

Another Euclidean MST algorithm [Frahling, Indyk and Sohler, 2008]: There is a streaming algorithm for $(1+\varepsilon)$-approximation of MST using storage of $(\varepsilon^{-1} \log \Delta)^{O(d)}$ bits.

*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.

2 ℓ_0-sampling

Problem Definition (ℓ_p-sampling): Let $x \in \mathbb{R}^n$ be the frequency vector of the input stream. The goal is to draw a random index from $[n]$ where each i has probability $\frac{|x_i|^p}{\|x\|_p^p}$.

We will see today the case $p = 0$, where the goal is to draw a uniformly random i from the set $\mathrm{supp}(x) = \{i \in [n] : x_i \neq 0\}$.

Algorithms may have some errors either in the probabilities being approximately correct (e.g., $\pm\delta$) and/or that with some probability the algorithm gives a wrong answer (returns FAIL or a sample not according to the desired distribution).

Framework for ℓ_0-sampling [following Cormode and Firmani, 2014]:

(A) Subsample the coordinates of x with geometrically decreasing rates

(B) Detect if the resulting vector y is 1-sparse

(C) If y is 1-sparse, recover its nonzero coordinate.

(A) Subsampling:

The algorithm chooses a random hash function $h : [n] \to [\log n]$, such that for each $i \in [n]$,

$$\Pr[h(i) = l] = 2^{-l}, \qquad \forall l \in [\log n].$$

(The probabilities do not sum up to 1, and in the remaining probability we can set $h(i)$ to nil, i.e., no level.)

For each "level" $l \in [\log n]$, create a virtual stream for the coordinates in $h^{-1}(l)$, formally defined as $y^{(l)} \in \mathbb{R}^n$ which is obtained from x by zeroing out coordinates outside $h^{-1}(l)$.

Observe that y is obtained from x by a linear map.

Lemma: If $x \neq 0$, then there exists $l \in [\log n]$ for which $\Pr[|\mathrm{supp}(y)| = 1] = \Omega(1)$.

Proof: Was seen in class.

Exer: Show that whenever $\mathrm{supp}(y)$ contains only one coordinate, that coordinate is indeed drawn uniformly from $\mathrm{supp}(x)$.

Exer: Show how to achieve a similar guarantee (for $|\mathrm{supp}(y)| = 1$) using a hash function h that is only pairwise independent. (However, now the "surviving" coordinate might be non-uniform.)

The success probability (getting $|\mathrm{supp}(y)| = 1$) can be increased to $1 - \delta$ by $O(\log \frac{1}{\delta})$ repetitions. The overall result is a $O(\log n \log \frac{1}{\delta})$ virtual streams y.

(C) Sparse recovery (of a 1-sparse vector): Suppose $y \in \mathbb{R}^n$ (which is $y^{(l)}$ from above) is 1-sparse. How can we find which coordinate i is nonzero?

Compute $A = \sum_i y_i$ and $B = \sum_i i \cdot y_i$ and report their ratio B/A.

For 1-sparse vector the output is always correct, as this step is deterministic.

Notice that A, B form a linear sketch whose size (dimension) is 2 words. Thus, they can be easily maintained over the virtual stream y (and also over the original stream x), even in the presence of deletions.

(B) Detection (if a vector is 1-sparse):

Lemma: There is a linear sketch to detect whether y is 1-sparse, using $O(\log n)$ words and achieving one-sided error probability $1/n^3$ (i.e., if $|\text{supp}(y)| = 1$ it always accepts, otherwise it accepts with probability at most $1/n^3$).

Proof: Was seen in class, using the AMS sketch to test if ℓ_2 norm is zero.

Exer: Show how to improve the storage to $O(1)$ words by a more direct approach.

Hint: Use a linear map (of y) with random coefficients from $[-n^3, n^3]$.

Overall Algorithm:

The algorithm goes over all virtual streams (all levels and all repetitions) in a fixed order, and reports the first coordinate that is recovered successfully and passes the detection test. If none of them succeeded, it reports FAIL.

Storage: The total storage is $O(\log^2 n \log \frac{1}{\delta})$ words, not including randomness.

However, using limited randomness in the subsampling (necessary to reduce randomness) might introduce some bias to the uniform probabilities.

Variations of this approach: Detection and recovery of vectors with sparsity $s = 1/\varepsilon$ instead of $s = 1$, using k-wise independent hashing in the subsampling, or using Nisan's pseudorandom generator to reduce storage.

Theorem [Jowhari, Saglam and Tardos, 2011]: There is a streaming algorithm with storage $O(\log^2 n \log \frac{1}{\delta})$ bits, that with probability at most δ reports FAIL, with probability at most $1/n^2$ reports an arbitrary answer, and with the remaining probability produces a uniform sample from $\text{supp}(x)$.