CSCE 689: Special Topics in Modern Algorithms for Data Science Fall 2023

Lecture lecture number — date here

Prof. Samson Zhou *Scribe: Tzu-Shen (Jason), Wang*

1 Review of Streaming Algorithms (Statistical Algorithms) of All Previous Classes

The list of streaming algorithms covered in previous classes is as follows:

- Reservoir sampling
- Heavy-hitter
  - Misra-Gries
  - CountMin
  - CountSketch
- Moment estimation — AMS algorithm (F_2 norm)
- Sparse recovery
- Distinct elements estimation

1.1 Reservoir Sampling

We see a stream of elements from $[n]$. we want to uniformly sample one of the positions of the stream

Example 1. The stream is: $[47, 72, 10, 1014, 33, 51]$, with length 7. The reservoir sampling problem is to randomly pick an element in the stream with uniform probability, i.e., $\frac{1}{7}$ in this example.

1.2 Heavy-Hitters

Given a set S of m elements from $[n]$. Denote f_i as the frequency of element i, where $i \in [n]$. Let L_p be the norm of the frequency vector:

$$L_p = (f_1^p + f_2^p + ... + f_n^p)^{1/p}.$$

The goal of the heavy-hitters problem is that with threshold ε, output elements i, where $f_i > \varepsilon L_p$, and do not output elements j, where $f_j < \frac{\varepsilon}{2} L_p$.

1.3 Frequency Moments (L_pNorm)

Given a set S of m elements from $[n]$. Denote f_i as the frequency of element i, where $i \in [n]$. Let F_p be the frequency moment of the vector:

$$F_p = f_1^p + f_2^p + ... + f_n^p.$$

The goal is that given an accuracy parameter ε, output a (1+ε)-approximate to F_p.

1.4 Distinct Elements (F_0 Estimation)

Given a set S of m elements from $[n]$. Denote f_i as the frequency of element i, where $i \in [n]$. Let F_0 be the frequency moment of the vector:

$$F_0 = |\{i : f_i \neq 0\}|.$$

The goal is that given an accuracy parameter ε, output a (1+ε)-approximate to F_0.

1.5 Spare Recovery

Given an insertion-deletion stream of length $m = \Theta(n)$, and we are promised that there are at most k nonzero coordinates. The goal is to recover the k nonzero coordinates and their frequencies.

2 Geometric Streaming algorithm

2.1 Graph Theory

- We have a graph G with vertex set V and edge set E.
- We let $V = [n]$, so each vertex is an integer from 1 to n.
- For each edge $e \in E$, we write it as $e = (u, v)$ for $u, v \in [n]$.

To simplify, we have the settings as follows:

- No self-loops
- No multi-edges
- Undirected edges
- Unweighted edge (or each edge has the same weight 1)

2.2 Matching

A matching, denoted M, where $M \subseteq E$, s.t. for every two edges $\in$ M shares a common vertex. i.e. $\nexists\ e_1 = (u_1, v_1) \wedge e_2 = (u_2, v_2)$ where $e_1, e_2 \in M$ and $u_1 == u_2 \vee v_1 == v_2$.

Definition (Maximal Matching)**.** A matching M of G such that any additional edges would no longer be a matching. I.e., Adding any edge $e \in G$ that is not already in M to M will not be a matching.

Definition (Maximum Matching)**.** Finding a matching M of G with the largest possible number of edges.

Definition (Alternating Path)**.** Given a matching M of G, an alternating path P is a path of edges that alternate between edges in M and edges not in M

Definition (Augmenting Path)**.** Given a matching M of G, an augmenting path is any alternating path that does not start and end at a vertex in the matching.

Example 2. Suppose there are edges: $(1, 2), (2, 3), (4, 5), (6, 1), (7, 1)$ in graph G and a matching M composed of edges $(1, 2), (4, 5)$. Then the path $6 \to 1 \to 2 \to 3$ is an augmenting path.

Claim 1. Flipping all the edges in an augment path increases the matching size.

Claim 2. If a matching is not a maximum matching, there exists an augmenting path to the matching.

Hopcroft and Karp (1973) [2] and Edmonds (1965) [1] show finding augment paths can be done in polynomial time.

3 Semi-streaming Model.

We have graph $G = (V = [n], E)$, where $|E| = m$. Suppose that we are only allowed to use $O(n\text{poly}(\log n))$ space, which is enough to store a matching, but not enough to store G, since $m = O(n^2)$. Kapralov (2013) [3] showed no one-pass semi-streaming algorithm for maximum matching can achieve approximation better than $\dfrac{e}{e-1} \approx 1.582$.

As for maximal matching, we can just run a greedy algorithm that when each edge comes in, if it does not violate the matching property (i.e., having a common vertex with an edge in the matching), then we add it to the matching. The correctness of this greedy algorithm is intuitive, since each edge that is not in the match, the algorithm has already checked that adding it will violate the matching property.

Claim 3. Each maximal matching is a 2-approximation to maximum matching.

Each edge e of the maximum matching M^* must be incident to some edge $e^{'}$ of any maximal matching $M^{'}$, but each edge $e^{'} \in M^{'}$ can be incident to at most 2 edges of M^*.

References

[1] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965.

[2] John E. Hopcroft and Richard M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973.

[3] Michael Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1679–1697, 2013.