CS 229r: Algorithms for Big Data Fall 2013

Lecture 6 — September 19, 2013

Dr. Karthekeyan Chandrasekaran *Scribe: Martin Camacho*

1 Overview

Today Graph problems in the streaming model. We'll look at the following problems:

- Connectivity
- Number of connected components,
- Spanning forest
- Testing bipartiteness
- k-edge connectivity

The streaming model consists of a stream of edges on n nodes, with additions and deletions. We enforce a space limit: $O(n \operatorname{polylog} n)$.

2 A Sub-Problem

Easy Case Suppose we only *add* edges.

For connectivity, we can store and maintain a spanning forest in $O(n)$ space. (A spanning forest consists of a spanning tree on each connected component).

Reduction If you store all the edges of a graph, you need quadratic space. Non-streaming algorithms for these graphs also use quadratic space. So instead, we'll sketch the graph to get a smaller graph, and simulate these algorithms on the smaller graph.

Sub-problem Given a vector $x \in \{0, 1, -1\}^n$, updated by a turnstile stream $x_i \leftarrow x_i + v$, find an index $i \in \operatorname{supp}(x)$ (such that $x_i \neq 0$) with space limit $O(\operatorname{polylog}(n))$.

Note that the updates are guaranteed to maintain $x_i \in \{0, 1, -1\}$.

Easy Case Suppose $|\operatorname{supp}(x)| = 1$. Maintain $B = \sum_{i=1}^{n} x_i$. At the end, $|B|$ will be the index contained in the support. This takes $O(\log n)$ space.

In the general case, we will maintain substreams such that one of the substreams has support of size 1, and then use the above result.

AMS Sketch For all $\epsilon, \delta \in (0, 1/2)$ and x updated by a turnstile stream, there exists a sketch to estimate $(1 \pm \epsilon)||x||_2$ with probability $\geq 1 - \delta$, using space $O(\log(1/\delta)/\epsilon^2)$.

Solution to the Subproblem Pick a hash function $g : [n] \to [n]$ from a pairwise independent hash family, as define $h : [n] \to [\log n]$ as $h(i) = \text{lsb}(g(i))$, where $\text{lsb}(x)$ is the least significant bit of x. Then $\mathbb{P}(h(i) = t) = 1/2^t$.

Maintain $\log n$ streams defined by indices $I^t = \{i \in [n], h(i) = t\}$. Then $\mathbb{E}\,|I^t| = n/2^t$. Now, let $X^t = x|_{I^t}$, so that $\mathbb{E}\,|\operatorname{supp}(X^t)| = |\operatorname{supp}(x)|/2^t$. Then by expectation one of the streams must be of constant size, say $k = 60$ (we'll see why 60 works below).

We want a substream with support of size 1, though. Define a hash function $q : [n] \to [k]$ from a pairwise independent hash family to generate substreams $I^{h(i)}_{q(i)}$. For each $j \in [k], t \in [\log n]$, maintain two sketches:

1. AMS sketch of $X^t_j = x|_{I^t_j}$ with $\epsilon = 0.1, \delta = 1/\operatorname{poly}(n)$.
2. $B^t_j = \sum_{i \in I^t_j} i x_i$, as in the 'easy case' above.

The probability that the AMS sketch fails for some $j \in [k], t \in [\log n]$ is $\leq (60 \log n)/\operatorname{poly}(n)$, by union bound. Use the AMS sketch to find $j \in [k], t \in [\log n]$ such that the $\operatorname{supp}(X^t_j) = 1$, then output $|B^t_j|$.

2.1 Analysis

Claim $\exists t \in [\log n]$ s.t. $1 \leq |\operatorname{supp}(X^t)| \leq 20$ with probability $\geq 1/2$.

Proof. Fix t, let $Y_1, \ldots, Y_n$ be indicator random variables defined by $Y_i = 1$ if $i \in I_t$ and $Y_i = 0$ otherwise. Then $|\operatorname{supp}(X^t)| = \sum_{i \in \operatorname{supp}(x)} Y_i$. Then $\mathbb{E}\,|\operatorname{supp}(X^t)| = |\operatorname{supp}(x)|/2^t$, and $Var(\sum_{i \in \operatorname{supp}(x)} Y_i) = \sum_{i \in \operatorname{supp}(x)} Var(x_i)$, where the last step followed by pairwise independence of $g(i)$.

Next, $Var(y_i) = \mathbb{E}(y_i^2) - \mathbb{E}(y_i)^2 \leq \mathbb{E}(y_i^2) = 1/2^t$. Then $Var|supp(X^t)| \leq |\operatorname{supp}(x)|/2^t = \mathbb{E}\,|\operatorname{supp}(X^t)|$. Fix t such that $6 \leq \mathbb{E}\,|\operatorname{supp}(X^t)| \leq 12$.

Now, use Chebyshev:

$$P_\gamma\left(\left||\operatorname{supp}(X^t)| - \mathbb{E}\,|\operatorname{supp}(X^t)|\right| > 5\right) \leq \frac{Var}{25} \leq \frac{\mathbb{E}}{25} \leq \frac{12}{25} < \frac{1}{2}.$$

So with probability $\geq 1/2$, $1 \leq |\operatorname{supp}(X^t)| \leq 17 < 20$. □

Claim $\exists j \in [60]$ s.t. $|\operatorname{supp}(X^t_j)| = 1$ with probability $\geq 2/3$.

Proof. Fix $e \in \operatorname{supp}(X^t)$, and say $j = q(e)$. Then $\mathbb{P}(\text{collision at } j) \leq 19/60 < 1/3$. □

Finally, we analyze the probability of failure: $\mathbb{P}(\text{failure}) \leq 1/2 + 1/3 = 5/6$, so maintain $\log n$ sketches and one of them succeeds with probability $\geq 1 - 1/\operatorname{poly}(n)$.

Our total space is $O(\log^3 n)$.

3 Back to Graph Problems

3.1 Connectivity

First, if you had $O(n^2)$ space, how would you measure connectivity? Repeat until there are no more edges:

1. Pick an edge incident to each vertex
2. Contract the chosen edges

Output the number of remaining supernodes.

The number of rounds of this algorithm is $O(\log n)$, since in each round, the number of vertices in each connected component is at least halved.

Representation We will use a **node-edge incidence** vector for each node. For node i, keep a vector

$$a_i[j,k] = \begin{cases} 1 & \text{if } (j,k) \in E, i = j, j < k \\ -1 & \text{if } (j,k) \in E, i = j, j > k \\ 0 & \text{otherwise} \end{cases}$$

Given a set S, let $\mathbb{E}(S, V \setminus S)$ be the set of edges which crosses S. In particular, we can write $E(S, V \setminus S) = \text{supp}(\sum_{i \in S} a_i)$.

Now, given the sketch Π from the subproblem, use Πa_i to find an edge adjacent to i. In round $r = 2, \ldots, \log n$, to get an edge incident to supernode S, maintain $\Pi \sum_{i=S} a_i = \sum_{i \in S} \Pi a_i$.

There are couple of subtle issues in doing this, however. We need to use one sketch per round: $\Pi_1, \ldots, \Pi_{\log n}$ in order to avoid introducing dependencies.

The total space is thus $O(n \log^5 n)$.

We have now proved the following theorem.

Theorem 1. *There is a single-pass algorithm that uses space $O(n \operatorname{polylog} n)$ to solve connectivity as well as the number of connected components.*

We can also use this algorithm to find a spanning forest, since the contracted edges contain a spanning forest. Since we only contracted a linear number of edges (one per vertex), we can find a spanning forest in $O(n \log n)$ space (it takes $O(\log n)$ space to store an edge).

3.2 k-edge connectivity

Claim. For $i = 1, \ldots, k$, let F_i be a spanning forest in $E \setminus \cup_{j=1}^{i-1} F_j$. G is k-edge connected iff $\cup_{i=1}^{k} F_i$ is k-edge connected.

Algorithm Use sketches $I_1, \ldots, I_k$ for spanning forests. For $i = 1, \ldots, k$:

1. Pick a spanning forest F_i in $E \setminus \cup_{j=1}^{i-1} F_j$
2. Update $I_{i+1}, \ldots, I_k$ to delete the edges in F_i

After this we have a graph with kn edges. Use Karger's min-cut algorithm to test k-edge connectivity on $\cup_{i=1}^{k} F_i$.

4 Bibliography.

This material of this lecture was mostly taken from [AGM12].

References

[AGM12] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *SODA*, pages 459–467, 2012.