CS-448 Sublinear Algorithms for Big Data Analysis April 2, 2025

Lecture 7

Lecturer: Michael Kapralov

1 Graph sketching

We now show how to use ℓ_0 sampler to obtain an algorithm for dynamic connectivity [1]. Suppose that a graph $G = (V, E)$ is presented as a stream of dynamic edge updates (i.e. edges are inserted or deleted). We would like to design a streaming algorithm that uses $n \log^{O(1)} n$ space and allows listing the connected components of the graph G at the end of the stream, together with a spanning forest.

We start with the following simple (non-streaming) algorithm for finding connected components.

Algorithm 1 ConnectedComponents($G = (V, E)$)

procedure ConnectedComponents($G = (V, E)$)
    Initialize $C_0 := \bigcup_{u \in V} \{u\}$ ▷ Initially all vertices are in connected components by themselves
    for $t = 1$ to T **do** ▷ $T = O(\log n)$ suffices
        $E' \leftarrow \emptyset$
        Each component in C_{t-1} chooses an outgoing edge
        $C_t \leftarrow$new set of components obtained by adding E' to C_{t-1}
    end for
    return C_T
end procedure

Note that if we start with a connected graph, then the number of connected components reduces by a factor of at least 2 in each iteration. Thus, $T = O(\log n)$ iterations suffice to connect the graph. Applying this reasoning to every connected component of a general graph G shows that C_T is the list of connected components at the end of the execution of the algorithm above.

We now show how to implement the algorithm above using a sketch. Recall that the edge incidence matrix of a graph G is a matrix $B \in \mathbb{R}^{\binom{n}{2} \times n}$, where rows are indexed by pairs of vertices and columns by vertices. For a pair of vertices $\{u, v\} \in \binom{V}{2}$ the row $b_{\{u,v\}}$ is zero if $\{u, v\}$ is not an edge, and otherwise has two nonzero entries – one in position u and the other in v. One of the entries equals $+1$, and the other equals -1.

The following claim will be crucial.

Claim 1 *For every $S \subseteq V$ the vector $B \cdot \mathbf{1}_S \in \mathbb{R}^{\binom{n}{2}}$ has entries in the set $\{-1, 0, +1\}$, and the nonzero entries are exactly the edges that cross the cut $(S, V \setminus S)$. Here*

$$(\mathbf{1}_S)_u = \begin{cases} 1 & \text{if } u \in S \\ 0 & o.w. \end{cases}$$

is the indicator vector of the cut S.

Proof The rows corresponding to nonedges are zero, so it suffices to consider the rows that correspond to the edges. If $e = (u, v)$ is an edge, then if both endpoints of e are in S $b_e \cdot \mathbf{1}_S = 0$, since the two nonzeros of b_e in positions u and v have different signs and hence cancel. If both endpoints of e are in $V \setminus S$, then $b_e \cdot \mathbf{1}_S = 0$. Finally, if one endpoint is S and the other outside, we get that $b_e \cdot \mathbf{1}_S$ is either $+1$ or -1. ■

We will need the concept of ℓ_p-samplers, which we now define.

Definition 2 *(ℓ_p sampler) An $(\epsilon, \delta_1, \delta_2)$ ℓ_p sampler is a linear sketching $A \in \mathbb{R}^{m \times n}$ together with a decoding algorithm $D : \mathbb{R}^m \to [n] \cup \{\perp\}$ that satisfy the following conditions for every $x \in \mathbb{R}^n \setminus \{0\}$:*

1. $\Pr[D(Ax) = \perp] \leq \delta_1$ *(i.e. the decoder outputs 'I don't know' with probability at most δ_1)*
2. $\Pr[D(Ax) \text{ fails}] \leq \delta_2$ *(i.e. the decoder fails, without necessarily knowing that, with probability at most δ_2)*
3. *conditioned on $D(Ax)$ not failing and not outputting $\perp$, one has, for every $i \in supp(x)$*

$$\Pr[D(Ax) = i] \in \left[\frac{(1-\epsilon)|x_i|^p}{||x||_p^p}, \frac{(1+\epsilon)|x_i|^p}{||x||_p^p} \right]$$

Remark Note that ℓ_p samplers are usually defined with only one failure probability δ, which can be thought of as setting $\delta_1 = \delta_2 = \delta/2$. In this lecture we will get a slightly stronger construction that allows setting δ_2 inverse polynomially small without losing much in the space complexity.

The following result is known:

Theorem 3 *[2] There exists an ℓ_0-sampler with $\epsilon = 0, \delta_1 = \delta$ and $\delta_2 = 1/n^{10}$ that uses $O(\log n \log(1/\delta))$ bits of space.*

Now let $L_1, \ldots, L_T \in \mathbb{R}^{\log^{O(1)} n \times \binom{n}{2}}$ denote independent ℓ_0-samplers for vectors in dimension $\binom{n}{2}$. We get the following algorithm:

Note that if both failure probabilities (δ_1 and δ_2) were less than $1/n^{10}$, say, then Claim 1 Algorithm 2 would directly implement Algorithm 1. As we show below, it still works if δ_1 is a small constant (say, $1/100$) and $\delta_2 = 1/n^{10}$. This setting of parameters yields the asymptotically tight bound of $O(n \log^3 n)$ on the space complexity of dynamic spanning forest computation.

It is important to note that we prepared **independent** sketches L_j for use in the $T = O(\log n)$ iterations of the process. This is because an ℓ_0 sampler is only guaranteed to output a uniformly random element of the support of input x (except for the failure events) when the randomness of the sketch is

Algorithm 2 CONNECTEDCOMPONENTSSKETCH($G = (V, E)$)

procedure CONNECTEDCOMPONENTSSKETCH($G = (V, E)$)
  Initialize $C_0 := \bigcup_{u \in V} \{u\}$ ▷ Initially all vertices are in connected components by themselves
  Prepare $L_j B$ for all $j = 1, \ldots, T$ ▷ $T = O(\log n)$ suffices
  for $t = 1$ to T **do**
    $E' \leftarrow \emptyset$
    for each component $S \subseteq V$ in C_{t-1} **do**
      If Decode($L_t B \mathbf{1}_S$) $\neq \perp$, add output edge to E'
    end for
    $C_t \leftarrow$new set of components obtained by adding E' to C_{t-1}
  end for
  return C_T
end procedure

independent of x, i.e. when x is chosen first, and then the coins are flipped to design the sketch. Using a single sketch L instead of $L_1, \ldots, L_T$ would violate this assumption.

We now show correctness. Suppose that G is connected (if not, repeat the same argument on each connected component). Let $X_i = 1$ if the number of connected components decreased by at least a factor of $2/3$ in round i and $X_i = 0$ otherwise. Since the number of connected components never increases, we have

$$\# \text{ of connected components after round } t \leq n \cdot (2/3)^{\sum_{i=1}^{T} X_i}$$

where $T = C \log n$ is the number of iterations that the algorithm runs for.

Consider iteration i, and let Z_i denote the number of connected components in that iteration. Let A_i denote that number of connected components that receive at least one edge incident on them in that iteration. Note that if $A_i \geq (9/10) Z_i$, we have

$$Z_{i+1} \leq A_i/2 + (Z_i - A_i) \leq (9/10) Z_i/2 + (1/10) Z_i \leq (1/2 + 1/10) Z_i \leq (2/3) Z_i,$$

and also note that if fewer than $1/10$ fraction of supernodes have their sketches fail, we get $A_i \geq (9/10) Z_i$. Finally, it remains to note that by an application of Markov's inequality the probability that at most a $1/10$ fraction of the nodes succeed is at most $1/10$ (i.e. with probability at least $9/10$ at least a $9/10$ fraction of the nodes succeed). Putting the above together, we conclude that $\mathbf{E}[X_i] \geq 9/10$ for every i as long as the number of connected components at step i is larger than 1. Finally, since X_i's are independent, we get by Chernoff bounds $\sum_{i=1}^{T} X_i \geq \log_{3/2} n$ with probability at least $1 - 1/n$ if C is larger than an absolute constant, as required.

2 ℓ_0-samplers

In what follows we design a slightly less efficient version of ℓ_0 samplers than what is provided by Theorem 3. First note that if x is 1-sparse (i.e. contains exactly one nonzero element), we can recover it exactly using techniques from previous lecture, and if x is not 1-sparse, we can subsample the universe $[n]$

at a sequence of geometric rates, and run our 1-sparse solution on one of the geometric scales. We will need several primitives to execute on this plan. We design the primitives below.

Checking that $x \neq 0$. This can be accomplished using a constant number of dot products of x with a random sign vector. More precisely, we use the AMS sketch with precision $\epsilon = 1/2$ and desired failure probability. We can ensure that the failure probability is at most δ' with $O(\log(1/\delta'))$ rows.

Recovering a 1-sparse vector. In this case in order to recover x, it suffices to store two dot products (x, u) and (x, v), where $u_j = 1$ for all $j \in [n]$, and $v_j = j$ for every $j \in [n]$. We use the notation $[n] = \{1, 2, \ldots, n\}$. Given $\alpha := (x, u)$ and $\beta := (x, v)$, our reconstruction procedure proceeds by first checking if $\alpha \neq 0$. If $\alpha = 0$, we conclude that x is the zero vector and output nothing. If $\alpha \neq 0$, we let $j^* := \beta/\alpha$ and conclude that the only nonzero entry of x is entry j^*, with a value of α.

Reducing from the case of general sparsity to the case of 1-sparse x. Now suppose that x is not 1-sparse. Consider a hash function $h : [n] \to \{1, 2, \ldots, \log_2 n\}$ that hashes every $i \in [n]$ to bucket $j \in \{1, 2, \ldots, \log_2 n\}$ with probability 2^{-j} for all j, and disregards the item with remaining probability $\sum_{j > \log_2 n} 2^{-j}$.

For each $b \in \{1, 2, \ldots, \log_2 n\}$ define $y^b \in \mathbb{R}^n$ by

$$y_i^b = \begin{cases} x_i & \text{if } h(i) = b \\ 0 & \text{o.w.} \end{cases}$$

Note that for every $b \in \{1, 2, \ldots, \log_2 n\}$ we have $\mathbf{E}_h[|\text{supp}(y^b)|] = 2^{-b}|\text{supp}(x)|$, so if $b = \log_2 |\text{supp}(x)|$, we should expect y^b to be about 1-sparse! We now make this precise. Let b be such that $||x||_0 \leq 2^b \leq 2||x||_0$. We claim that y^b is 1-sparse with a constant probability:

$$\begin{aligned}
\Pr\left[y^b \text{ is 1-sparse}\right] &= \sum_{i \in \text{supp}(x)} \Pr\left[h(i) = b \text{ and } h(i') \neq b \text{ for all } i' \in \text{supp}(x) \setminus \{i\}\right] \\
&= \sum_{i \in \text{supp}(x)} \Pr\left[h(i) = b\right] \prod_{i' \in \text{supp}(x) \setminus \{i\}} \Pr\left[h(i') \neq b\right] \quad \text{(by independence of } h\text{)} \\
&= \sum_{i \in \text{supp}(x)} 2^{-b}(1 - 2^{-b})^{||x||_0 - 1} \\
&= (1/2)(1 - 2^{-b})^{2^{b+1} - 1} \quad (\text{since } ||x||_0 \leq 2^b \leq 2||x||_0) \\
&\geq (1/2)(1 - 2^{-b})^{2^{b+1}} \quad (\text{since } 1 - 2^{-b} < 1)
\end{aligned}$$

We now show that the expression on the last line above is lower bounded by a constant for all $b \geq 1$. Indeed, we have

$$(1 - 2^{-b})^{2^{b+1}} = ((1 - 2^{-b})^{2^b})^2 \geq (1/2)^2 = 1/4,$$

since $(1 - 1/n)^n$ is monotone increasing in n, and the minimum is achieved at $n = 1/2$ (i.e. $b = 1$) in our case. Putting the bounds above together, we get

$$\Pr\left[y^b \text{ is 1-sparse}\right] \geq 1/8.$$

Now we can run 1-sparse recovery on the y^b's. Since one of them will be 1-sparse, recovery will succeed. There is one problem, however: recovery may fail on vectors that are not actually 1-sparse without alerting us to the fact that it failed. We need a way to test whether recovery was successful.

Checking that recovery succeeded. Fix b. Suppose that we run 1-sparse recovery on y^b and it outputs (j^*, α^*), i.e. claims that $y^b = \tilde{y}$, where $\tilde{y}$ is a 1-sparse vector with value α^* in coordinate j^*. We need to test whether $\tilde{y} - y^b = 0$ so that the test is correct with probability $1 - 1/n^{10}$, say. We can do that using another copy of the AMS sketch, where we set $\epsilon = 1/2$ and $\delta' = 1/n^{10}$. Indeed, just store Ay^b, where A is the corresponding AMS sketch matrix. Once we recover $\tilde{y}$, compute $A\tilde{y}$, and run the AMS decoding primitive on $Ay^b - A\tilde{y}$. since the randomness of the AMS sketch is independent of the randomness that we used to generate $\tilde{y}$, the AMS guarantee holds.

AMS sketch $(1+\epsilon)$-approximates $||x||_2$ using $\frac{1}{\epsilon^2} \log^{O(1)} n$ space.

We now summarize our sketch. For each $b \in \{1, 2, \ldots, \log_2 n\}$ we maintain

1. (u, y^b), where u is the all-ones vector;
2. (v, y^b), where $v_j = j$ for all $j \in [n]$;
3. an AMS sketch of y^b with $\epsilon = 1/2$ and $\delta' = 1/16$;
4. an AMS sketch of y^b with $\epsilon = 1/2$ and $\delta' = 1/n^{10}$.

The decoding works as follows. For each $b \in \{1, \ldots, \log_2 n\}$ run 1-sparse recovery on y^b if it is nonzero (test using the first AMS sketch). If recovery outputs a vector $\tilde{y}$, use the second sketch to test for correctness. If recovery was correct, output the result and stop.

We claim that this primitive outputs a uniformly random element of $\text{supp}(x)$ with probability at least 1/128. This follows by noting that if for a bucket b we have that y^b is 1-sparse, then sparse recovery succeeds and returns the only nonzero element of y^b, which is a uniformly random element of $\text{supp}(x)$. Furthermore, by our derivations above there exists a value of b such that

$$\Pr\left[y^b \text{ is 1-sparse}\right] \geq 1/8.$$

Sparse recovery is invoked on that particular y^b if the AMS sketch reports that $y^b \neq 0$, which occurs with probability at least $1/8 - 1/16 = 1/16$. Finally, note that sparse recovery may be invoked on a non-sparse input, but the result is then reported with probability at most $\log_2 n / n^{10} < 1/n^9$ by a union bound over $\log_2 n$ buckets that sparse recovery could be run on. Thus, our primitive outputs a uniformly random element with probability at least $1/8 - 1/16 - 1/n^9 > 1/32$, outputs an incorrect answer with probability at most $1/n^9$ and outputs $\perp$ otherwise.We have thus obtained a $(0, \delta_1, \delta_2)$-ℓ_0 sampler with $\delta_1 = 1 - 1/32$ and $\delta_2 \leq 1/n^9$. We can always decrease δ_1 by independent repetition of the construction above, getting a $(0, \delta_1, O(\log(1/\delta_1))/n^9)$-$\ell_0$-sampler with factor $O(\log(1/\delta_1))$ more space.

The space complexity is $O(\log^3 n)$ bits for a single repetition of our construction (the AMS sketch with $1/n^{10}$ failure probability is the bottleneck), and hence we get $O(\log^3 n \log(1/\delta_1))$ bits after independent repetition. Finally, the hash function h can be implemented in small space using Nisan's PRG at the cost of another $\log n$ factor in space complexity.

References

[1] Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 459–467, 2012.

[2] Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, June 12-16, 2011, Athens, Greece*, pages 49–58, 2011.