CS368: Algorithmic Techniques for Big Data	Spring 2019-2020

Problem Set 3

Prof. Moses Charikar — *Due:* ***May 24, 2020, 11:59pm***

Policy: You are permitted to discuss and collaborate on the homework but you must write up your solutions on your own or as a group of two. Furthermore, you need to cite your collaborators and/or any sources that you consulted. No late submissions are allowed. There will be no late days. All homework submissions are subject to the Stanford Honor Code. For all assignments, we are allowing group submissions for groups of 1 or 2.

Submission: We will use Gradescope for the homework submissions. Go to www.gradescope.com to either login or create a new account using your stanford.edu account. Use the course code **MK7NNR** to register for CS368. You must use LaTeX, LyX, Microsoft Word, or a similar editor to typeset your write-up. If you are working as a group of two, only one group member needs to submit the assignment. When submitting, please remember to add all group member names on Gradescope.

Length of submissions: Please include as much of the calculations that show that you understand everything that is going through the answer. As a rule of thumb after you have solved the problem, try to identify what are the main steps taken and critical points of a proof and include them. Unnecessary long answers to questions will be penalized. The points next to each question are indicative of the hardness/length of the proof.

1 Lower bounds for existence of triangles [25 points]

Consider a graph stream describing an unweighted, undirected n-vertex graph G. Prove that $\Omega(n^2)$ space is required to determine, in one pass, whether or not G contains a triangle, even with randomization allowed.

2 Lower bounds for exact computation of F_2 [20 points]

Prove that computing F_2 exactly, in one pass with randomization allowed, requires $\Omega(\min\{m, n\})$ space. Construct an appropriate "hard stream" of length m with universe size n, where $m = \Theta(n)$, and show that $\Omega(n)$ space is required on this stream. Then extend the result to multiple passes, with randomization allowed. The lower bound for p passes should be $\Omega(\min\{m, n\}/p)$.

3 Spanners for Weighted Graphs [45 points]

Recall that the distance estimation problem asks us to process a streamed graph G so that, given any $x, y \in V(G)$, we can return a t-approximation of $d_G(x, y)$, i.e., an estimate $\hat{d}(x, y)$ with the property:

$$d_G(x, y) \leq \hat{d}(x, y) \leq t \cdot d_G(x, y).$$

Here, t is a fixed integer known beforehand. In class, we solved this using space $\tilde{O}(n^{1+2/t})$ by computing a subgraph H of G that happened to be a t-spanner.

(a) **[20 points]** Now suppose that the input graph is edge-weighted, with weights being integers in $[W]$. Each token in the input stream is of the form (u, v, w_{uv}), specifying an edge (u, v) and its weight $w_{uv} \in [W]$. Distances in G are defined using weighted shortest paths, i.e.,

$$d_{G,w}(x, y) := \min \left\{ \sum_{e \in \pi} w_e : \pi \text{ is a path from x to y} \right\}.$$

Give an algorithm that processes G using space $\tilde{O}(n^{1+2/t} \log W)$ so that, given $x, y \in V(G)$, we can then return a $(2t)$- approximation of $d_{G,w}(x, y)$. Give careful proofs of the quality and space guarantees of your algorithm.

(b) **[25 points]** In class, we saw that even for the unweighted case, space $\Omega(n^{1+2/t})$ is necessary to preserve all distances up to a factor of t. What if we only care about the max distance of a connected graph? The diameter of a graph $G = (V, E)$ is defined as $\text{diam}(G) = \max\{d_G(x, y) : x, y \in V\}$, i.e., the largest vertex-to-vertex distance in the graph. A real number $\hat{d}$ satisfying

$$\text{diam}(G) \le \hat{d} \le \alpha \cdot \text{diam}(G)$$

is called an α-approximation to the diameter. Suppose that $1 \le \alpha < 1.5$. Prove that, in the vanilla graph streaming model, a 1-pass randomized algorithm that α- approximates the diameter of a connected graph must use $\Omega(n)$ space. How does the result generalize to p passes?

4 L_0 Sampling with Pairwise Independent Hash Functions [45 points]

We revisit the problem of L_0 sampling in which we uniformly sample an element l from the support S of an input vector x. Recall, we are in the general framework of sketching where we maintain a sketch Ax under increments and decrements of coordinates of x where x is a n-dimensional vector and estimate some desired function from the sketch. Formally, we want the probability $\Pr(l = i)$ that i is returned satisfy

$$\Pr(l = i) \in \left((1 - O(\epsilon)) \frac{|x_i|^0}{\|x\|_0}, (1 + O(\epsilon)) \frac{|x_i|^0}{\|x\|_0} \right),$$

where $|x_i|^0 = 0$ if $x_i = 0$ and $|x_i|^0 = 1$ otherwise. Note $|S| = \|x\|_0$.

In class, we saw an L_0 sampling sketch using fully random hash functions. In this problem, we consider another L_0 sampling sketch using a family H of pairwise independent hash functions $h : [n] \to [m]$. Recall that pairwise independence means for any $x_1, x_2 \in [n]$ and values $y_1, y_2 \in [m]$,

$$\Pr_{h \in H}(h(x_1) = y_1 \wedge h(x_2) = y_2) = \Pr_{h \in H}(h(x_1) = y_1) \Pr_{h \in H}(h(x_2) = y_2),$$

and, also, for any $x \in [n]$ and $y \in [m]$, $\Pr_{h \in H}(h(x) = y) = \frac{1}{m}$.

Consider the following algorithm for a large enough constant c and a constant c' to be chosen. Assume $\epsilon \in (0, \frac{1}{2})$ and $S \neq \emptyset$.

Algorithm 1

Input: $\epsilon \in (0, \frac{1}{2})$

1. For $j = 1, \ldots, \log \frac{cn}{\epsilon}$ and $k = 1, \ldots, \frac{c'}{\epsilon}$, let $h_j^k : [n] \to \{0, \ldots, 2^j - 1\}$ be hash functions drawn from a pairwise independent hash family.
2. As we read input x (its increments/decrements), maintain the following for each h_j^k:

$D_j^k \in (1 \pm 0.1)\|x_{S_j^k}\|_0$ for $S_j^k = \{i \in S : h_j^k(i) = 0\}$
$C_j^k = \sum_{i \in S_j^k} x_i$
$T_j^k = \sum_{i \in S_j^k} i x_i$

3. Let j^* be the largest j for which $\#\{k : D_j^k \in 1 \pm 0.1\} \geq 1$.
4. Output $T_{j^*}^k / C_{j^*}^k$ for an arbitrary k for which $D_{j^*}^k \in 1 \pm 0.1$.

Strictly speaking, hash functions in Step 1 are being drawn from different hash families with corresponding ranges for different values of j. For simplicity, assume each D_j^k in Step 2 is exactly in the interval $(1 \pm 0.1)\|x_{S_j^k}\|_0$ with error probability of 0. Note 1 ± 0.1 denotes the interval $[0.9, 1.1]$, so $(1 \pm 0.1)\|x_{S_j^k}\|_0$ denotes the interval $[0.9 \cdot \|x_{S_j^k}\|_0, 1.1 \cdot \|x_{S_j^k}\|_0]$.

(a) **[15 points]** Fix arbitrary j and k. For any $i \in S$, note that $\Pr(h_j^k(i) = 0) = \frac{1}{2^j}$. Show that $\Pr(h_j^k(i) = 0 \wedge |S_j^k| = 1) \geq \frac{1}{2^j}\left(1 - \frac{\|x\|_0}{2^j}\right)$. Conclude that $\Pr(h_j^k(i) = 0 \wedge |S_j^k| = 1) \in \left[\frac{1}{2^j}\left(1 - \frac{\|x\|_0}{2^j}\right), \frac{1}{2^j}\right]$. The wedge operator $\wedge$ denotes logical and. *Hint: use the union bound and pairwise independence.*

(b) **[10 points]** Let $\hat{j}$ be the unique integer j such that $\frac{\|x\|_0}{2^j} \in (\frac{\epsilon}{2}, \epsilon]$. For any $j \geq \hat{j}$ and any $i \in S$, show that $\Pr(h_j^k(i) = 0 \mid |S_j^k| = 1) \in \left((1 - O(\epsilon))\frac{1}{\|x\|_0}, (1 + O(\epsilon))\frac{1}{\|x\|_0}\right)$. Note this is a conditional probability.

(c) **[10 points]** For an appropriately chosen c', show that the above algorithm solves the L_0 sampling problem with some constant error probability strictly less than $\dfrac{1}{2}$.

(d) **[10 points]** Using a well-known technique, design an algorithm that solves the L_0 sampling problem with the error probability at most δ for any $\epsilon \in (0, \dfrac{1}{2})$ and $\delta > 0$. What is the overall space requirement in terms of ϵ and δ?

5 Extra Problem: Bipartite Graphs (Do not turn in!)

A graph G is called bipartite if $V(G)$ can be partitioned into two sets S and S^c such that all edges lie between vertices of those two sets, that is, $|E_G(S, S^c)| = |E(G)|$. Equivalently, there exists a valid two coloring of the vertices, where a coloring is valid if there is no monochromatic edge (i.e., with endpoints of the same color). Consider the vanilla graph streaming model (with edge insertions only).

(a) **[15 points]** Give a deterministic algorithm that uses $O(n \log n)$ space and decides whether a graph is bipartite. Give a proof of correctness.

(b) **[20 points]** Show that any randomized one-pass streaming algorithm that decides whether a graph is bipartite requires $\Omega(n)$ space.

(c) **[25 points]** Given a undirected graph G, we define its *bipartite double cover* $\tilde{G} = (\tilde{V}, \tilde{E})$ where $\tilde{V}$ is a vertex set containing two copies v_1, v_2 of every vertex $v \in V(G)$, and $\tilde{E}$ is an edge set containing the edges $\{u_1, v_2\}$ and $\{v_1, u_2\}$ for all edges $\{u, v\} \in E(G)$. Prove that the graph G being bipartite is equivalent to

$$\#\text{Connected Components}(\tilde{G}) = 2 \cdot \#\text{Connected Components}(G)\,.$$

Show how to use this fact to design a streaming algorithm to test whether a graph is bipartite. Give the space requirements of your algorithm.

References

[1] Noam Nisan. Pseudorandom Generators for Space-Bounded Computation. *Combinatorica*, 12(4):449-461, 1992.