Homework 4

Algorithms for Big Data: CS498 ABD/ABG, Fall 2020
Due: Monday at 10pm CDT, December 14, 2020

Instructions and Policy:

- Unlike previous homeworks, you need only do **2** problems. (Of course you're encouraged to try and welcome to submit all of them!)
- Each homework can be done in a group of size at most two. Only one homework needs to be submitted per group. However, we recommend that each of you think about the problems on your own first.
- Homework needs to be submitted in pdf format on Gradescope. See `https://courses.engr.illinois.edu/cs374/fa2018/hw-policies.html` for more detailed instructions on Gradescope submissions.
- Follow academic integrity policies as laid out in student code. You can consult sources but cite all of them including discussions with other classmates. Write in your own words. See the site mentioned in the preceding item for more detailed policies.

Problem 1. Minhash and permutations We discussed minhash based on random permutations for Jaccard similarity of sets. It is useful in a variety of settings but requires, in the traditional analysis, that we use several permutations. Here is an alternative approach from a NeurIPS 2012 paper based on a single permutation and bucketing. `https://papers.nips.cc/paper/2012/file/eaa32c96f620053cf442ad32258076b9-Paper.pdf`. Read the paper and summarize it.

Problem 2. LSH We saw LSH and the Indyk-Motwani approach. We covered it a high-level but the practical aspects require some care and subtlety. Read the recent paper `https://arxiv.org/abs/2005.12065` and summarize your understanding.

Problem 3. Describe a semi-streaming algorithm in the strict turnstile model (edges can be inserted and deleted) to check whether a graph is 2-edge connected.

Problem 4. Matchings with additional constraint We saw an algorithm in the semi-streaming model for finding a constant factor approximation to the maximum cardinality and maximum weight matching problem. Now consider the following variant. We are given a graph $G = (V, E)$. Moreover each edge has a color from $\{1, 2, ..., k\}$ and each color i has an integer upper bound b_i. The goal is to find a maximum cardinality matching M which satisfies the additional constraint that the number of edges in M from a color class i is at most b_i. Assume that you are given the b_i values ahead of time and the each edge when it arrives in the stream specifies its end points and its color. Describe a constant factor approximation for this problem in the semi-streaming setting.

Problem 5. Weighted matchings from unweighted In lecture we saw an algorithm for maximum weighted matching in the semi-streaming model. The unweighted case admits a simple greedy

algorithm (maximal matching is 1/2-approximation) while the weighted case is more involved. In this problem you will see another approach which gives a simple reduction to the unweighted case; the advantage is that it is a generic reduction that applies to other problems as well. Assume all edge weights are between 1 and W for some known upper bound W. Partition edges in to $k = O(\log W)$ groups $E_i = \{e \mid w(e) \in w(e) \geq 2^{i-1}\}$. The algorithm runs the unweighted algorithm separately for each E_i (that is, it ignores the weights of edges in E_i). Let M_i be the matching it computes for $G = (V, E_i)$. At the end of the stream it outputs the matching with the maximum weight among the k matchings so the space is $O(n \log W)$.

- Analyze the algorithm and show that it achieves a constant factor approximation for the weighted matching problem. You can obtain a $\frac{1}{4(1+\epsilon)}$-approximation if you choose parameters carefully.
- Show that this approach applies to handle the weighted case of matching with additional constraint from Problem 1.
- Assuming minimum weight of an edge is 1, adapt the algorithm so that it uses $O(n\text{poly}(\log n))$ space without knowledge of W.

Problem 6. Hypergraph matching A hypergraph $G = (V, E)$ consists of set of vertices V and a set hyperedges E. Each hyperedge $e \in E$ is a subset of V, that is $e \subseteq V$. The rank of G is $\max_e |e|$. Graphs are a special case with $r = 2$. $M \subseteq E$ is a matching in a hypergraph if no two hyperedges in M intersect in a vertex. Unlike matchings in graphs finding the maximum cardinality matching in a hypergraph is NP-Hard even when $r = 3$ (the standard NP-Complete problem related to this is the 3-D matching problem). Consider the semi-streaming version of finding an approximate matching in a hypergraph where the edges arrive one by one.

- Obtain a semi-streaming algorithm for the maximum cardinality matching with approximation ratio $1/r$ where r is the rank.
- Obtain an $\Omega(1/r^2)$-approximation for the weighted case.
- **Extra credit:** Obtain an $\Omega(1/r)$-approximation for the weighted case. You can skip the previous two parts if you do this.

Problem 7. In a turnstile stream updating a vector $x \in \mathbb{R}^n$ starting as the 0 vector, an *ϵ-error ℓ_1 sampler* is a streaming algorithm that when queried outputs a pair $(i, \hat{x}_i)$ such that i is output with probability $|x_i|/\|x\|_1$ and $\hat{x}_i = (1 \pm \epsilon)x_i$ (recall that in turnstile streams, each stream update is of the form $x_i \leftarrow x_i + v$ where v can be positive or negative). Pretend we have such an ℓ_1 sampler using space $S(n, \epsilon)$. Now consider the following problem: you see a stream $i_1\ i_2\ \cdots\ i_{n+1}$ with each $i_j \in [n]$. This stream must have at least one duplicate entry due to the pigeonhole principle. Show how to use a 1/2-error ℓ_1 sampler to give a one-pass streaming algorithm that reports at least one duplicate index $i \in [n]$ with probability at least $1 - \delta$. The space of your algorithm should be $O(S(n, 1/2) \cdot \log(1/\delta))$.

Problem 8. We have seen streaming algorithms for ϵ-approximate quantiles. We defined An ϵ-approximate quantile for a quantile $\phi \in (0, 1]$ as an element of rank r where $\phi n - \epsilon n \leq r \leq \phi n + \epsilon n$ where n is number of elements. We define a stronger notion of ϵ-approximate quantiles where we

wish to return an element of rank r where $(1-\epsilon)\phi n \leq r \leq (1+\epsilon)\phi n$. Describe how to compute an ϵ-approximate quantile summary for this stronger notion of approximation.

Problem 9. In lecture we saw the frequent directions algorithms for low rank approximation with parameter k. The algorithm computes n SVDs of $\ell \times d$ matrices where $\ell = k(1 + 1/\epsilon)$ each of which takes $O(d\ell^2)$ time for a total time of $O(ndk^2/\epsilon^2)$. We would like to improve the run time to $O(ndk/\epsilon)$. Consider some constant $c > 1$ and now we will use matrix Q with $c\ell$ rows but we will batch every $(c-1)\ell$ rows of A and compute an SVD only once for each batch. Show that this scheme gives $(1+\epsilon)$-approximation as before but the running time is now reduced to $O(\frac{c^2}{c-1}nd\ell)$.