CS 514: Advanced Algorithms II – Sublinear Algorithms **Rutgers: Fall 2021**

Problem set 3

Due: 11:59PM, December 12, 2021

Problem 1. In the *set disjointness* communication problem, Alice and Bob are given subsets $A, B \subseteq [N]$, respectively. The goal is to determine whether or not $A \cap B = \emptyset$.

(i) Prove that the deterministic communication complexity of this problem is $\mathrm{D}(\text{set-disjointness}) = \Omega(N)$. **(10 points)**

Remark. One can in fact prove that even randomized communication complexity of this problem is $\mathrm{R}(\text{set-disjointness}) = \Omega(N)$; this requires a much more involved proof and we do not cover it.

(ii) Use this in a reduction to prove that any *constant pass* deterministic streaming algorithm for undirected connectivity requires $\Omega(n)$ bits of space. **(10 points)**

Problem 2. In Lecture 10, we designed a streaming algorithm for the k-center clustering problem when points $p_1, \ldots, p_n \in \{1, \ldots, \Delta\}^d$ are arriving one by one in the stream. For any $\varepsilon \in (0, 1)$, the algorithm achieves a $(2+\varepsilon)$-approximation by storing $O(k \cdot \frac{\log D}{\varepsilon})$ points where $D = \sqrt{d} \cdot \Delta$ is the maximum possible value for the optimum solution. Our goal in this problem is to improve the space complexity of this algorithm at a cost of increasing its approximation ratio by a constant factor.

Design a streaming algorithm for the k-center clustering problem that achieves an $O(1)$-approximation by storing only $O(k)$ points throughout the stream. **(20 points)**

Bonus part: Can you reduce the approximation ratio to $(2+\varepsilon)$-approximation again by storing only $O(k/\varepsilon \cdot \log(1/\varepsilon))$ points instead? **(+25 points)**

Hint: The original approach in the lecture was based on two steps: (i) Designing an $O(k)$-space intermediate streaming algorithm that given a parameter $\tau \in [1, D]$, either outputs a clustering C with cost at most $2 \cdot \tau$, or outputs that the optimal solution has cost more than τ; (ii) then we did a simple geometric search by running the algorithm above for $O(\frac{\log D}{\varepsilon})$ choices of $\tau \in \{1, (1+\varepsilon), (1+\varepsilon)^2, \ldots, D\}$ *in parallel*.

Modify the second step by performing the geometric search *sequentially* by updating the current guess for τ on the fly whenever it is smaller than the optimum value.

Problem 3. Design a single-pass semi-streaming algorithm for finding a minimum spanning tree (MST) of the input *weighted* undirected graph $G = (V, E)$ where the weight of each edge is revealed at the same time with the edge in the stream.

You may assume that G is connected and all edge-weights are distinct. **(20 points)**

Problem 4. In Lecture 11, we mentioned the following result:

- **Palette Sparsification Theorem:** Let $G = (V, E)$ be an n-vertex graph with maximum degree Δ. Suppose for every vertex $v \in V$, we *independently and uniformly at random* sample $O(\log n)$ colors $L(v)$ from $\{1, \ldots, \Delta + 1\}$. Then, with high probability, there is a proper coloring of G in which the color of every vertex v is chosen from $L(v)$.

Based on this theorem, we showed how to design a semi-streaming algorithm for $(\Delta + 1)$ coloring. Let us now consider sublinear time algorithms from the first half of the course.

Use the palette sparsification theorem to give an $\widetilde{O}(n^{3/2})$ *query* algorithm for $(\Delta+1)$ coloring problem in the general query model (defined in Lecture 2). Note that for this problem, we only focus on the *query complexity* of the algorithms and not their time complexity. **(20 points)**

Hint: Give an $\widetilde{O}(n^{3/2})$ query and time algorithm for finding the *conflict graph* defined in the context of the Palette Sparsification Theorem; then apply this theorem to finalize the proof.

Problem 5. Recall that a graph $G = (V, E)$ is said to have **arboricity** $\alpha(G) = \alpha$ if

$$\alpha = \max_{S \subseteq V,\ |S|>1} \left\lceil \frac{|E(S)|}{|S|-1} \right\rceil,$$

where $E(S)$ denotes the set of edges with both endpoints in S. In Problem set 1, we considered designing a query algorithm for estimating arboricity of a given graph.

In this problem, the goal is to extend this result to the dynamic streaming model. Design a semi-streaming algorithm (using graph sketching ideas) that given a graph $G = (V, E)$ specified in a dynamic stream and a parameter $\varepsilon \in (0, 1)$, outputs an estimate $\widetilde{\alpha}$ such that:

$$\Pr\left(|\widetilde{\alpha} - \alpha(G)| > \varepsilon \cdot \alpha(G)\right) < 1/10.$$

(20 points)