Your Name	October 16, 2024
6.5240 Problem Set 3	
Collaborators: *Firstname Lastname*	Due: October 18, 2024, 10pm

Homework guidelines: You may work with other students, as long as (1) they have not yet solved the problem, (2) you write down the names of all other students with which you discussed the problem, and (3) you write up the solution on your own. No points will be deducted, no matter how many people you talk to, as long as you are honest. If you already knew the answer to one of the problems (call these "famous" problems), then let us know that in your solution writeup – it will not affect your score, but will help us in the future. It's ok to look up famous sums and inequalities that help you to solve the problem, but don't look up an entire solution.

1. The *arboricity* of an undirected graph $G = (V, E)$ is a measure of "uniform sparsity" of G. There are multiple equivalent definitions of arboricity. Here, we mention one key definition for our purpose:

   Definition 1 *A graph G is said to have arboricity $\alpha(G) = \alpha$ if*

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

   where $E(S)$ denotes the set of edges with both endpoints in S.

   In this problem, we design a sublinear query algorithm for estimating arboricity in the query model. Namely, we obtain an algorithm that given $G = (V, E)$ via the general query model plus random edge samples, and a parameter $\varepsilon \in (0, 1)$, outputs an estimate $\tilde{\alpha}$ such that

   $$\Pr[|\tilde{\alpha} - \alpha(G)| > \varepsilon \cdot \alpha(G)] < 1/10.$$

   a) Suppose we are told that G has n vertices and m edges. Sample each edge of G independently with probability $p := \frac{100}{\varepsilon^2} \cdot \log n \cdot \frac{n}{m}$, to get a subgraph H of G. Prove that

   $$\Pr[|p^{-1} \cdot \alpha(H) - \alpha(G)| > \varepsilon \cdot \alpha(G)] < 1/10,$$

   where the probability is over the choice of H.

   b) Use the above result to obtain an algorithm with $O(\varepsilon^{-2} \cdot n \log n)$ queries to the graph for approximating the arboricity problem. (You do not need to bound the runtime of the algorithm or the number of samples).

2. **LCA for coloring.** In the following parts, assume that all input graphs start out with unique IDs in $[n]$.

   a) Given a graph of maximum degree at most Δ, show that the edges can be partitioned into at most Δ oriented forests where each node has outdegree at most 1, the roots have outdegree 0, and edges point along the path to a root. Moreover, show that given a vertex v and index i, we can compute the outgoing edge (if it exists) from vertex v in the i^{th} forest of the partition, in $O(\Delta)$ sequential time.

   b) Present a distributed algorithm for 6-coloring trees. Assume that the tree can be viewed as a rooted tree in which children know who their parent is. For full credit, your algorithm should run in $k = O(\log^\star n)$ rounds (here, $\log^\star n$ denotes the number of times the logarithm function must be applied to n to produce a value less than or equal to 1). Hint: Consider algorithms in which a node u looks at its parent v and recolors itself based on the location of the first bit which differs between u and v.

   c) Given a graph G along with a c-coloring of the nodes (assume you can query the coloring of a node in 1 step), show how to find an MIS in c distributed rounds. Note: unlike in Luby's algorithm, this gives a deterministic approach to get a MIS.

   d) Present an LCA for 6^Δ-coloring graphs with maximum degree at most Δ.

3. **Spanner LCA for general graphs.** In class, we gave an LCA for the spanner problem that works for graphs of max degree at most $n^{3/4}$. Show how to construct an LCA for the spanner problem for any graph. Recall that the algorithm can probe for the degree of u, the i^{th} neighbor of u, or the existence and position of v in u's neighbor list; all such probes cost unit time. For full credit, your runtime should still be $\tilde{O}(n^{3/4})$ per query.

4. **Property testing of the clusterability of a set of points.** Let X be a set of points in an arbitrary metric space. Assume that one can compute the distance between any pair of points in one step. Say that X is (k, b)-diameter clusterable if X can be partitioned into k subsets, which we call "clusters," such that the maximum distance between any pair of points in a cluster is b. Say that X is ε-far from (k, b)- diameter clusterable if at least $\varepsilon|X|$ points must be deleted from X in order to make it (k, b)-diameter clusterable. Show how to distinguish the case where X is (k, b)-diameter clusterable from the case where X is ε-far from $(k, 2b)$-diameter clusterable. Your algorithm should use $\text{poly}(k, 1/\varepsilon)$ queries. Note that it is possible to get an algorithm which uses $O((k^2 \log k)/\varepsilon)$ queries.