Sublinear Time Algorithms	November 10, 2022
Homework 5	
Lecturer: Ronitt Rubinfeld	*Due Date: November 30, 2022*

1. In the following parts, assume that all input graphs start out with unique IDs.

   (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 $k = O(\log^* n)$ rounds (here, $\log^* n$ denotes the number of times the logarithm function must be applied to n to produce a value less than or equal to 1). Note that this gives an LCA for 6-coloring trees which runs in $2^{O(\log^* n)}$ probes.

   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 an 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.

2. 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. For full credit, your runtime should still be $O(n^{3/4})$ per query.

   Hint: (1) Handle the nodes that have degree between $\sqrt{n}$ and $n^{3/4}$ with a different setting of parameters for determining centers. (2) For nodes of degree at least $n^{3/4}$, partition the edges into groups of size $n^{3/4}$, and add a rule 3 edge (u, v) whenever v introduces u to a new cluster within its partition (this will allow more edges in the final graph, but show that it won't destroy the sparsity of the spanner).