CS:5620 Homework 4 Solution, Fall 2016

(2) The algorithm starts with each node telling all other nodes its ID. This helps nodes elect the node with lowest ID is the "global" leader.

In the GHS algorithm, each node is initially an MST fragment by itself and each node knows that all edges incident on it lead to other fragments. At the start of a typical iteration of the GHS algorithm on a clique, suppose that the following properties hold: (i) each MST fragment has a fragment leader who knows all the nodes in the fragment, (ii) the ID of the fragment leader serves as the "label" for the fragment and each node in a fragment knows this label, (ii) every node knows which of its incident edges connect it to neighbors in other fragments, and (iv) the "global" leader knows all the MST fragments. Note that all four properties are satisfied initially and the following algorithm will ensure that these properties are satisfied after each iteration.

Since the communication network is a clique, an iteration starts with each fragment leader sending a message to each node v in its fragment asking for the lightest edge incident on v connecting it to a different fragment. All nodes respond to their fragment leaders with this information and each fragment leader computes an MWOE for its fragment. The fragment leader then informs the end points of the MWOE that the edge connecting them was chosen as an MWOE. Note that this takes 3 rounds of communication.

Now every fragment leader sends its MWOE to a "global" leader. On receiving all the MWOEs, the "global" leader can merge the fragments into larger fragments, by local computation. The "global" leader then computes the MST fragment leader ID for each (new) fragment. The "global" leader can simply choose the node with lowest ID in a fragment and designate that as the leader of the fragment. Then the "global" leader sends to each node the ID of its (possibly new) MST fragment leader. This completes the work that the "global" leader does in an iteration. Note that this takes 2 rounds of communication.

Finally, each node tells every other node in the network, its fragment label. This helps each fragment leader learn about all the nodes in its fragment. It also helps each node know which of its incident edges lead to other fragments. This takes one final round of communication and this completes the iteration.

So each iteration takes 6 rounds of communication and we know from the analysis of GHS in the Congest model that GHS takes $O(\log n)$ iterations. Thus, this implementation of GHS on a clique takes $O(\log n)$ rounds.

5(a) First use the Cole-Vishkin algorithm to compute a 3-coloring of the oriented tree in $O(\log^* n)$ rounds. It takes $O(1)$ additional rounds to compute a maximal matching of the oriented 3-colored tree. The algorithm proceeds in 3 phases, one for each color. We now describe Phase 1. Before Phase 1 starts all nodes are *active* and all edges are *active*.

Phase 1. Each non-root node v colored 1, *proposes* to its parent u that the edge $\{u, v\}$ should be added to the matching. A node u may receive proposals from a number of children, and it arbitrarily selects one proposal, say from a child v. This edge $\{u, v\}$ is added to the matching M. (Note that if v proposes to u then v is colored 1 and u is not colored 1 and therefore u does not make a proposal in Phase 1.) Nodes incident on matched edges become inactive and then all edges incident on inactive nodes become inactive. At the end of Phase 1, it is easy to see that nodes colored 1 have no active edges left that are connecting them to parents. This is true even when a node's proposal to its parent has been rejected because the parent would have accepted some other child's proposal and become inactive. It is also easy to see that the set M is a matching (though not necessarily a *maximal* matching).

In Phase 2, nodes colored 2 are similarly processed. In particular, each *active* node colored 2 makes a proposal to its parent, provided the parent is also active. At the end of Phase 2, it is easy to see that nodes colored 1 and 2 have no active edges left that are connecting them to parents.

In the final phase (Phase 3), nodes colored 3 are similarly processed and at the end of Phase 3, it is the case that no node has any active edges left that are connecting them to parents. Since every edge connects a node to its parent in an oriented tree, this means that there are no active edges left and therefore we have constructed a maximal matching.

5(b) If the tree is not oriented, we compute (using the algorithm in HW2, Problem 5) an orientation of the tree such that every node has at most two outgoing edges and a 3-coloring of the treee. This takes $O(\log n)$ rounds using a deterministic algorithm.

To go from this 3-coloring to a maximal matching, we use essentially the same idea as in Problem 5(a) above. The algorithm has three phases, and in Phase $i = 1, 2, 3$ we process all nodes of color i. We now describe how Phase 1 differs from Phase 1 in Problem 5(a).

Phase 1. Each node v colored 1 that has a parent, *proposes* to an arbitrarily chosen parent u that the edge $\{u, v\}$ should be added to the matching. Parents behave as before, arbitrarily selecting a proposal. As before, it is easy to see that at this point a node colored 1 has at most one active edge left connecting it to a parent. Each node colored 1 that still have an active parent, will make a proposal to that parent. Parents behave as before, and now we are guaranteed that no node colored 1 has an active edge to a parent.

Note that Phase 1 takes $O(1)$ rounds and similarly Phases 2 and 3 will take $O(1)$ rounds each. Thus, it takes $O(1)$ additional rounds (after the 3-coloring has been computed) to complete the construction of the maximal matching.
