COSC-373: HOMEWORK 3

LEE JIAEN, HYERY YOO, AND ALEXANDER LEE

Question 1. (a) We first use the LE/BFS/SSSP protocol from class to find the leader, BFS tree, and let the leader know the distance from the furthest node/nodes which is labeled as “Dist”.

(b) Each node then does the following:

```
$local\_counter \leftarrow t_v$
$second\_counter \leftarrow 0$
$status \leftarrow inactive$
if I'm the leader $reset\_cntdown \leftarrow Dist$
other nodes $reset\_cntdown \leftarrow 0$
foreach $round\ i$ do
    if $i = 0$ then
        if I'm the leader, set $status$ = active and broadcast message
         "$reset\_cntdown - 1$" to children in BFS tree
    if $received\ messages$ then
        set $reset\_cntdown$ = message
        set $status$ = active
        broadcast message "$reset\_cntdown - 1$" to children in BFS tree
    if $status = active$ then
        if $second\_counter = reset\_cntdown$ then
            set $local\_counter = 0$
            set $status$ = inactive
            halt
        else
            $second\_counter$ ++
    $local\_counter$ ++
return
```

(Correctness) The protocol above uses the LE/BFS/SSSP protocol discussed in class to elect a leader, find a BFS tree, and notify the leader of the distance from the leader to the furthest node/nodes in the BFS tree in $O(D)$ rounds. We would find the distance from the leader to the furthest node by using SSSP to get all of the nodes' distances from the leader and then have the leaves of the BFS tree send their distance to their parent nodes. At each of the next rounds, each node sends their distance to their parent node and each parent node (internal node) waits for all its children to send a message. Once it has received messages from all its children, it takes the max distance, and sends it to its parent node. Eventually, the leader will receive messages from all of its children and note the maximum distance it encounters as "Dist"and set it's *reset_cntdown*=Dist. The leader then knows that it will take "Dist" (which can be at most D) rounds to get from the leader to the furthest node/nodes in the BFS tree. Therefore, the leader decides to wait "Dist" rounds before resetting and then lets its children know to wait one less round and then those nodes let their children know to wait one less round and so on. The message that a node receives tells the node how many rounds to wait until resetting, i.e., how many rounds it will take until we reach the furthest node/nodes. Furthermore, *second_counter* and *reset_cntdown* help each node keep track of when to reset and *second_counter* only starts incriminating after the node has received a message and updated its *status* to active. In $r_0 = O(D)$, we would have reached every node in the BFS tree and the furthest node/nodes from the leader in the BFS tree will be told to wait 0 rounds to reset. The furthest node/nodes will then have $second\_counter = reset\_cntdown$ and immediately reset *local_counter* to 0. In the same round, every other node in the BFS tree would also have $second\_counter = reset\_cntdown$ and reset *local_counter* to 0. BFS, ensures that we reach every node in the graph. Thus, the above protocol is correct because all nodes would reset *local_counter* to 0 at the same time in $r_0 = O(D)$.

(Runtime) All nodes reset after the furthest node/nodes received a message to reset. When we start from the leader, the furthest node/nodes receive a message to reset in at most D rounds. If we have less than D rounds, then the furthest away nodes in the BFS tree could potentially not be notified of the reset and fail to reset at the same time as the other nodes. Thus, the algorithm above will take $O(D)$ rounds. We know that the LE/BFS/SSSP algorithm will also take $O(D)$ rounds. Therefore, this protocol will take $O(D)$ rounds.

(Congestion) We already know that the CONGEST protocol LE/BFS/SSSP from class sends less than $O(n \log n)$ bits. Furthermore, the algorithm above has nodes only send a message containing a constant that indicates how many rounds to wait until resetting, which only requires $O(1)$ bits. Therefore, this protocol is a CONGEST protocol.

Question 2. TODO

Question 3. (a) We proceed with devising a greedy algorithm that is guaranteed to produce a proper $\Delta + 1$ coloring of G. The algorithm executes as follows:

(1) Initialize an empty set C, which represents the set of colors.
(2) Assign any vertex in V the color 1 and add 1 to C.
(3) Choose an uncolored vertex $v \in V$. Assign v the smallest color in C that has not been assigned to v's colored neighbors. If v's colored neighbors have been assigned all the colors in C, add a new color $\max\{C\} + 1$ to C and assign it to v.
(4) Repeat step (2) until all vertices are colored.

Since the algorithm assigns an uncolored vertex $v \in V$ with the lowest color in C that has not been assigned to v's neighbors or assigns v with a new color not yet in C, no two neighboring vertices are assigned the same color. Furthermore, because the algorithm only adds a new color to C when all neighbors of v have been assigned all the colors in C and v has at most Δ neighbors, the algorithm adds at most $\Delta + 1$ colors to C. Thus, only $\Delta+1$ colors are required. Therefore, the algorithm produces a proper $\Delta+1$ coloring of G.

(b) Consider the graph G_k of $\Delta + 1$ vertices, where each vertex is connected to every other vertex. In this case, every vertex has a degree of Δ, so the maximum degree is Δ. Now, suppose towards a contradiction that we have a Δ proper coloring of G_k. Thus, at most Δ colors have been assigned to $\Delta+1$ vertices of G_k. By the Pigeonhole Principle, at least two vertices in G_k have been assigned the same color. Since every vertex is connected to every other vertex, these two vertices with the same color are neighbors. This is a contradiction since we assumed that we have a Δ proper coloring of G_k, where no two neighboring vertices are assigned the same color. Therefore, G_k does not admit a proper Δ coloring.

(c) Each phase i:

- If received message "halted" from neighbor in phase $i - 1$
  - Remove neighbor from set of neighbors
- Execute protocol Π
- If node is in the MIS found from Π
  - Output i, send message "halted" to neighbors, and halt

(Correctness) In each phase i, protocol Π finds a MIS in the network of remaining nodes that have not halted. The nodes found in the MIS from phase i output color i. Neighboring nodes will not output the same color since they belong to different maximal independent sets.

(Runtime) After each phase if a node is not in a MIS, then at least one of its neighbors becomes part of an MIS, by the maximality condition. As such, after each phase, a node's degree decreases by at least one. Since a node has at most Δ neighbors, at most Δ phases are needed so that all it's neighbors are assigned a color. Hence, each node requires at most $\Delta+1$ phases to output a color i. Since each phase runs in $T(n)$ rounds, the protocol thus computes a property $\Delta+1$ coloring of G in $O((\Delta+1)T(n)) = O(\Delta T(n) + T(n)) = O(\Delta T(n))$ rounds.

(Congestion) Since the above protocol uses the CONGEST protocol Π and apart from in protocol Π, nodes only send the messages "halted" to neighbors, which only require $O(1)$ bits, the protocol above is therefore a CONGEST protocol.