Principles of Distributed Computing  
Exercise 9: Sample Solution

1 Communication Complexity of Set Disjointness

a) We obtain

\[
M_{\text{DISJ}} = \begin{pmatrix}
000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
000 & 1 & 1 & 1 & 1 & 1 & 1 & \color{gray}1 \\
001 & 1 & 0 & \color{gray}1 & 0 & 1 & 0 & 1 \\
010 & 1 & 1 & 0 & 0 & 1 & \color{gray}1 & 0 & 0 \\
011 & 1 & 0 & 0 & 0 & \color{gray}1 & 0 & 0 & 0 \\
100 & 1 & 1 & 1 & \color{gray}1 & 0 & 0 & 0 & 0 \\
101 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
110 & 1 & \color{gray}1 & 0 & 0 & 0 & 0 & 0 & 0 \\
111 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix} \leftarrow x
\]

b) When \( k = 3 \), a fooling set of size 4 for \( \text{DISJ} \) is, e.g.,

\[ S_1 := \{(111,000), (110,001), (101,010), (100,011)\}. \]

Entries in \( M_{\text{DISJ}} \) corresponding to elements of \( S_1 \) are marked dark gray. Note that a fooling set need not be on a diagonal of the matrix. E.g.

\[ S_2 := \{(001,110), (010,001), (011,100), (100,010)\}, \]

marked light gray in \( M_{\text{DISJ}} \).

c) In general, \( S := \{(x,\overline{x}) \mid x \in \{0,1\}^k\} \) is a fooling set for \( \text{DISJ} \). First, we note that for any two elements \((x_1,y_1),(x_2,y_2)\) of any fooling set \( x_1 \neq x_2 \). Otherwise we would have \((x_1,y_j) = (x_2,y_j)\) for \( j \in \{1,2\} \) and thus \( f(x_2,y_1) = f(x_1,y_1) = f(x_1,y_2) = f(x_2,y_2) =: z \), contradicting the definition of a fooling set. Similarly \( y_1 \neq y_2 \).

- For any \((x,y) \in S\), \( \text{DISJ}(x,y) = 1 \), by our definition of \( S \).
- Now consider any \((x_1,y_1) \neq (x_2,y_2) \in S\). Since \( x_1 \neq x_2 \), then either \( x_1 \) has some element that \( x_2 \) does not, or \( x_2 \) has some element that \( x_1 \) does not (or both). Wlog \( x_1 \) has some element that \( x_2 \) does not. But then \( x_1 \) and \( y_2 = \overline{x_2} \) are not disjoint so that \( \text{DISJ}(x_1,y_2) = 0 \).

So \( S \) is indeed a fooling set. And The size of \( S \) is \( 2^k \), so \( k \) is a lower bound for the CC by the result from the lecture.
2 Distinguishing Diameter 2 from 4

a) Choosing \( v \in L \) takes \( O(D) \): Use any leader election protocol from the lecture. E.g., the node with smallest ID in \( L \) can be elected as a leader. Then this node will be \( v \). Note that during the leader election protocol if after \( D \) rounds no messages are received, then the nodes can conclude that all nodes are in \( H \).

b) Computing a BFS tree from a vertex usually takes \( O(D) \). Since in our setting all graphs are guaranteed to have constant diameter, the time required for this is \( O(1) \). As node \( v \) is in \( L \), at most \( |N_1(v)| \leq s \) executions of BFS are performed. These can be started one after each other and yield a complexity of \( O(s) \).

c) The comment states: Computing an \( H \)-dominating set \( DOM \) takes time \( O(D) = O(1) \).

d) Since \( |DOM| \leq \frac{n \log n}{s} \), the time complexity of computing all BFS trees from each vertex in \( DOM \) (one after each other) is \( O(\frac{n \log n}{s}) \).

e) Checking whether all trees have depth of at most 2 can be done in \( O(D) = O(1) \) as well: Each node knows its depth in any of the computed trees. If its depth is 3 or 4, it floods “diameter is 4” to the graph. If a node gets such a message from several neighbors, it only forwards it to those from which it did not receive it yet. If any node did not receive message “diameter is 4” after 4 rounds, it decides that the diameter is 2. Otherwise it decides that the diameter is 4. This decision will be consistent among all nodes.

f) By adding all these runtimes, we conclude that the total time complexity of Algorithm 2-vs-4 is \( O \left( s + \frac{n \log n}{s} \right) \).

g) Using the triangle inequality we obtain that \( d(w, v) \geq d(u, v) - d(u, w) = 3 \) thus the BFS tree of \( w \) has at least depth 3. Therefore Algorithm 2-vs-4 decides “diameter is 4”.

h) Let \( w \) be the leader elected in step 2 of Algorithm 2-vs-4. If the BFS started in \( w \) has depth at least 3, we are done. In the other case it is \( d(u, w) \leq 2 \). Using d) we conclude that \( d(u, w) = 2 \). Let \( w' \) be a node that connects \( u \) to \( w \). Since \( w' \in N_1(w) \), Algorithm 2-vs-4 executes a BFS from \( w' \). Then we apply d) using that \( w' \in N_1(u) \).

i) Since \( DOM \) is a dominating set for \( H = V \setminus L = V \), it follows immediately that the algorithm executes a BFS from a node \( w \in DOM \cap N_1(u) \neq \emptyset \). Now apply d).

j) A careful look into the construction of family \( G \) reveals that we essentially showed an \( \Omega(n/\log n) \) lower bound to distinguish diameter 2 from 3. Since the graphs considered here cannot have diameter 3, the studied algorithm does not contradict this lower bound. Suppose we had to decide between diameter 2 and 3 (instead of 2 and 4) and we try using this exact algorithm. Indeed if the algorithm finds a BFS tree of depth greater than 2, then the diameter is 3. However, if all BFS trees found are diameter 2 or less, the diameter could still be 3.

k) Consider a clique (with \( n \) nodes, \( n \) large enough) and remove an arbitrary edge \( (u, v) \). Since \( d(u, v) = 2 \), the graph has diameter 2. We have \( L = \emptyset \) and \( \{w\} \) is an \( H \)-dominating set for all \( u \neq w \). If \( DOM = \{w\} \), then Algorithm 2-vs-4 executes exactly one BFS (from \( w \)) which has depth 1 which disproves the claim. Note that this proof works for all \( s \leq n - 2 \).