Principles of Distributed Computing
Exercise 6: Sample Solution

1 Communication Complexity of Set Disjointness

a) We obtain

\[
M^{\text{DISJ}} = \begin{pmatrix}
000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
000 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
001 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
010 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
011 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
100 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
101 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
110 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
111 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
\uparrow y \\
\end{pmatrix} \xleftarrow{x}
\]

For the Bonus task you can see for instance this short article for a nice visual.

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) If \( x_1 = x_2 \), then we would have \( (x_1, y_j) = (x_2, y_j) \) for \( j \in \{1, 2\} \) and thus \( f(x_1, y_2) = f(x_2, y_2) = f(x_1, y_1) = f(x_1, y_2) = z \), contradicting the definition of a fooling set. Similarly for \( y_1 = y_2 \).

d) \( S := \{(x, \bar{x}) \mid x \in \{0, 1\}^k\} \) is a fooling set for \( \text{DISJ} \):

- For any \( (x, y) \in S \), \( \text{DISJ}(x, y) = 1 \), by our definition of \( S \).
- Now, consider any two distinct elements of \( S \): \((x_1, \bar{x}_1)\) and \((x_2, \bar{x}_2)\). Since \( x_1 \neq x_2 \), either \( x_1 \) has set bit which \( x_2 \) does not, or \( x_2 \) has some set bit which \( x_1 \) does not (or both). Without loss of generality, \( x_1 \) has some set bit which \( x_2 \) does not, but then \( x_1 \) and \( \bar{x}_2 \) are not disjoint, meaning that \( \text{DISJ}(x_1, \bar{x}_2) = 0 \).

The size of \( S \) is \( 2^k \), so \( k \) is a lower bound for the \( CC(\text{DISJ}) \) by the result from the lecture.
2 Distinguishing Diameter 2 from 4

a) Note that \( O(D) = O(1) \), since \( D \leq 4 \) holds for all graphs being considered.

- Choosing \( v \in L \) takes time \( O(1) \): use any leader election protocol from the lectures. E.g., the node with smallest ID in \( L \) can be elected as a leader. This leader node will be node \( v \). Note that, during the leader election protocol, if after 4 rounds no messages are received, then a node can conclude that all nodes are in \( H \), so checking whether \( L \neq \emptyset \) does not need to be done separately.
- Computing a BFS tree from a vertex takes time \( O(D) = O(1) \). Since \( v \in L \), at most \( |N_1(v)| \leq s \) executions of BFS are performed. These can be started one after each other and yield a total time complexity of \( O(s) \).
- The comment states: computing a dominating set \( \mathcal{D} \) takes time \( O(D) = O(1) \).
- Since \( |\mathcal{D}| \leq \frac{n \log n}{s} \), the time complexity of computing all BFS trees from each vertex in \( \mathcal{D} \) (one after each other) is \( O \left( \frac{n \log n}{s} \right) \).
- Checking whether all trees have depth 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.
- By adding all these runtimes, we conclude that the total time complexity of Algorithm 2-vs-4 is \( O \left( s + \frac{n \log n}{s} \right) \).

b) By differentiating \( s + \frac{n \log n}{s} \) as a function of \( s \) we can argue that \( s + \frac{n \log n}{s} \) is minimal for \( s = \sqrt{n \log n} \). Alternatively, one can use the fact that \( a + b \geq 2 \sqrt{ab} \), with equality if and only if \( a = b \), to get that \( s + \frac{n \log n}{s} \geq 2 \sqrt{s \frac{n \log n}{s}} = \sqrt{n \log n} \), with equality if and only if \( s = \frac{n \log n}{s} \iff s = \sqrt{n \log n} \). For this value of \( s \), we get a runtime of \( O(\sqrt{n \log n}) \).

c) Since in this case no BFS tree can have depth larger than 2, the algorithm will always return “diameter is 2”.

d) If \( w = s \), the claim is immediate. Otherwise, using the triangle inequality we have that \( d(s, w) + d(w, t) \geq 4 \iff 1 + d(w, t) \geq 4 \iff d(w, t) \geq 3 \), so the BFS tree of \( w \) has depth at least 3. Therefore, Algorithm 2-vs-4 decides “diameter is 4”.

e) If the BFS started in \( v \) has depth at least 3, then we are done. Otherwise, we have \( d(s, v) \leq 2 \). Using d) we conclude that \( d(s, v) = 2 \). Let \( w \) be a node that connects \( s \) to \( v \). Since \( w \in N_1(v) \), Algorithm 2-vs-4 executes a BFS from \( w \). Then, apply d) using that \( w \in N_1(s) \).

f) Since \( \mathcal{D} \) is a dominating set, it follows that the algorithm executes a BFS from a node \( w \in \mathcal{D} \cap N_1(s) \neq \emptyset \). Now apply d).

g) A careful look into the construction of family \( \mathcal{G} \) 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.
(h) Consider a clique with $n$ nodes, where $n$ should be large enough, and remove an arbitrary edge $(u, v)$ from it. Since $d(u, v) = 2$, the graph has diameter 2. We have that $L = \emptyset$ and that for any $w \notin \{u, v\}$ the set $\{w\}$ is a dominating set. If one such $DOM = \{w\}$ is selected in the algorithm, then Algorithm 2-vs-4 executes exactly one BFS (from $w$), which has depth 1, disproving the claim. Note that this proof works for all $s \leq n - 2$. 