Principles of Distributed Computing
Exercise 11: Sample Solution

1 Communication Complexity of Set Disjointness

a) We obtain

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

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. However, a fooling set does not always need to be on a diagonal of the matrix. An example for such a set is

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

and 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 \( \text{DISJ} \). To prove this, we note: If \( y > x \) then there is always an index \( i \) such that \( x_i = y_i = 1 \) and we conclude \( \text{DISJ}(x,y) = 0 \). Second, we note for any two elements \( (x_1,y_1),(x_2,y_2) \) of any fooling set that \( x_1 \neq x_2 \). Otherwise it was \( (x_1,y_j) = (x_2,y_j) \) for \( j \in \{1,2\} \) and thus \( f(x_2,y_1) = f(x_1,y_2) = f(x_1,y_1) = f(x_2,y_2) =: z \) which contradicts the definition of a fooling set. Similarly it is \( y_1 \neq y_2 \).

- For any \((x,y) \in S\) it is \( \text{DISJ}(x,y) = 1 \).
- Now consider any \((x_1,y_1) \neq (x_2,y_2) \in S\). Since \( x_1 \neq x_2 \) and \( y_1 \neq y_2 \), we conclude that either \( y_2 > x_1 \), in which case \( \text{DISJ}(x_1,y_2) = 0 \), or \( y_1 > x_2 \) causing \( \text{DISJ}(x_2,y_1) = 0 \).
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 \( v \).
• 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 \( O(s) \).
• The comment states: Computing an \( H \)-dominating set \( DOM \) takes time \( O(D) = O(1) \).
• 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 \( O(\frac{n \log n}{s}) \).
• 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.
• 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 deriving \( O\left(s + \frac{n \log n}{s}\right) \) as a function of \( s \) we can argue that \( O\left(s + \frac{n \log n}{s}\right) \) is minimal
for \( s = \sqrt{n \log n} \). Thus the runtime of the Algorithm is \( O(\sqrt{n \log n}) \).

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

d) 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”.

e) 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 \( w' \in N_1(u) \).

f) 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).

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.

h) 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 \( u \neq w \neq v \). 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 \( s \leq n - 2 \).