Principles of Distributed Computing
Exercise 11

1 Communication Complexity of Set Disjointness

In the lecture we studied the communication complexity of the equality function. Now we consider the disjointness function: Alice and Bob are given subsets \( X, Y \subseteq \{1, \ldots, k\} \) and need to determine whether they are disjoint. Each subset can be represented by a string. E.g., we define the \( i^{th} \) bit of \( x \in \{0, 1\}^k \) as \( x_i := 1 \) if \( i \in X \) and \( x_i := 0 \) if \( i \notin X \). Now define disjointness of \( X \) and \( Y \) as:

\[
\text{DISJ}(x, y) := \begin{cases} 
0 & : \text{there is an index } i \text{ such that } x_i = y_i = 1 \\
1 & : \text{else}
\end{cases}
\]

a) Write down \( M^{\text{DISJ}} \) for the \( \text{DISJ} \)-function when \( k = 3 \).

b) Use the matrix obtained in a) to provide a fooling set of size 4 for \( \text{DISJ} \) in case \( k = 3 \).

c) In general, prove that \( CC(\text{DISJ}) = \Omega(k) \).

2 Distinguishing Diameter 2 from 4

In the lecture we stated that when the bandwidth of an edge is limited to \( O(\log n) \), the diameter of a graph can be computed in \( O(n) \). In this problem, we show that we can do faster in case we know that all networks/graphs on which we execute an algorithm have either diameter 2 or diameter 4. We start by partitioning the nodes into sets: Let \( s := s(n) \) be a threshold and define the set of high degree nodes \( H := \{v \in V \mid d(v) \geq s\} \) and the set of low degree nodes \( L := \{v \in V \mid d(v) < s\} \).

Next, we define: An \( H \)-dominating set \( \text{DOM} \) is a subset \( \text{DOM} \subseteq V \) of the nodes such that each node in \( H \) is either in the set \( \text{DOM} \) or adjacent to a node in the set \( \text{DOM} \). Assume in the following, that we can compute an \( H \)-dominating set \( \text{DOM} \) of size \( \frac{n \log n}{s} \) in time \( O(D) \).

a) What is the distributed runtime of Algorithm 2-vs-4 (stated next page)? In case you believe that the distributed implementation of a step is not known from the lecture, find a distributed implementation for this step! **Hint: The runtime depends on \( s \) and \( n \).**

b) Find a function \( s := s(n) \) such that the runtime is minimized (in terms of \( n \)).

c) Prove that if the diameter is 2, then Algorithm 2-vs-4 always returns 2.

Now assume that the diameter of the network is 4 and that we know vertices \( u \) and \( v \) with distance 4 to each other.
Algorithm 1 “2-vs-4”. Input: $G$ with diameter 2 or 4 Output: diameter of $G$

1: if $L \neq \emptyset$ then
2: choose $v \in L$ \hspace{1cm} \triangleright \text{We know: This takes } O(D).
3: compute a BFS tree from each vertex in $N_1(v)$
4: else
5: compute an $H$-dominating set $\text{DOM}$ \hspace{1cm} \triangleright \text{Use: Assumption}
6: compute a BFS tree from each vertex in $\text{DOM}$
7: end if
8: if all BFS trees have depth 2 or 1 then
9: return 2
10: else
11: return 4
12: end if

d) Prove that if the algorithm performs a BFS from at least one node $w \in N_1(u)$ it decides “the diameter is 4”.

e) In case $L \neq \emptyset$: Prove that the algorithm performs a BFS of depth at least 3 from some node $w$. \textbf{Hint: use d)}

f) In case $L = \emptyset$: Prove that the algorithm performs a BFS of depth at least 3 from some node $w$.


g) Give a high level idea, why you think that this does not violate the lower bound of $\Omega(n/\log n)$ presented in the lecture!

h) Assume $s = \frac{n}{2}$. Prove or disprove: If the diameter is 2, then Algorithm 2-vs-4 will always compute some BFS tree of depth exactly 2.