Principles of Distributed Computing
Exercise 6

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 $X, Y \subseteq \{1, \dots, k\}$ and need to determine whether they are disjoint. Each subset $Z \subseteq \{1, \dots, k\}$ can be represented by a string of bits $z \in \{0,1\}^k$, where the i^{th} bit of z is 1 if and only if $i \in Z$. Now, we can define the disjointness of x and y as:

$$DISJ(x, y) := \begin{cases} 0, & \text{if there is an index } i \text{ such that } x_i = y_i = 1 \\ 1, & \text{otherwise.} \end{cases}$$

a) Write down M^{DISJ} for function $DISJ$ when $k = 3$. **Bonus**, for fun: How does M^{DISJ} look in general? Can you spot any patterns?

b) Use the matrix obtained in **a)** to provide a fooling set of size 4 for $DISJ$ when $k = 3$.

c) Prove that if S is a fooling set and $(x_1, y_1), (x_2, y_2)$ are two different elements of S, then $x_1 \neq x_2$ and $y_1 \neq y_2$.

d) Prove that $CC(DISJ) = \Omega(k)$.

2 Distinguishing Diameter 2 from 4

In the lecture we stated that when the bandwidth of each 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 much faster in case we know that all networks/graphs on which we execute our algorithm have either diameter 2 or diameter 4. We start by partitioning the nodes of our graph $G = (V, E)$ into two sets: let $s := s(n)$ be a threshold to be determined later 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 a dominating set $\mathcal{DOM} \subseteq V$ to be a subset of nodes such that each node in the graph is either in $\mathcal{DOM}$ or is adjacent to a node in the $\mathcal{DOM}$. For this problem we assume that if all nodes in G have degree at least s, then one can compute a dominating set $\mathcal{DOM}$ of size at most $\frac{n \log n}{s}$ in time $O(D)$.
Note: We define $N_1(v)$ as the closed neighborhood of node v (v and its adjacent nodes).

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.**

Algorithm 1 "2-vs-4"

Input: Graph G with diameter 2 or 4.
Output: Diameter of G.

if $L \neq \emptyset$ **then**
    Choose $v \in L$. ▷ We know: this takes time $O(D)$.
    Compute a BFS tree from each node in $N_1(v)$.
else
    Compute a dominating set $\mathcal{DOM}$ of size at most $\frac{n \log n}{s}$. ▷ Use: Assumption
    Compute a BFS tree starting from each node in $\mathcal{DOM}$.
end if
if all BFS trees have depth 1 or 2 **then**
    return 2
else
    return 4
end if

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 s and t are vertices with distance 4 to each other.

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

e) Assuming $L \neq \emptyset$, prove that the algorithm performs a BFS of depth at least 3 from some node w. **Hint:** use **d)**.

f) Assuming $L = \emptyset$, prove that the algorithm performs a BFS of depth at least 3 from some node w.

We have now proven that Algorithm 2-vs-4 is always correct in distinguishing graphs of diameter 2 from graphs of diameter 4.

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) Assuming $s = n/2$, prove or disprove: if the diameter is 2, then Algorithm 2-vs-4 will always compute some BFS tree of depth exactly 2.