Tutor: Mitja Krebs

Exercise Sheet 2

Coloring

Exercise 1 (Scheduling Lectures)**.** Students can choose the lectures they want to attend. These lectures should not take place at the same time to allow the students to attend the lectures they have chosen. You are now given a list of students and the lectures they want to attend and are supposed to schedule these lectures to as few slots as possible.

- Arnold wants to attend Principles of Distributed Computing, Statistical Learning Theory, and Ubiquitous Computing.
- Berta wants to attend Principles of Distributed Computing and Graph Theory.
- Christina wants to attend Cryptography.
- Don wants to attend Statistical Learning Theory and Ubiquitous Computing.
- Emil wants to attend Graph Theory and Cryptography.
- Flo wants to attend Principles of Distributed Computing and Cryptography.

Exercise 2 (Algorithm "Reduce")**.** In the lecture, a distributed algorithm ("Reduce") for coloring an arbitrary graph with $\Delta+1$ colors in n synchronous rounds was presented (Δ denotes the largest degree, n the number of nodes of the graph).

1. What is the message complexity, i.e., the total number of messages the algorithm sends in the worst case?
2. Let m be the number of edges. Can you come up with an exact number of messages being sent?

Exercise 3 (Coloring Rings)**.** Algorithms 2.14 and 2.19, as discussed in the lecture, color any (rooted) tree with n nodes using 3 colors in $O(\log^* n)$ rounds. The approach consists of two phases:

- Phase 1 (Algorithm 2.14): Reduces the initial coloring based on node IDs to 6 colors.
- Phase 2 (Algorithm 2.19): Further reduces the 6 colors to 3.

In this exercise, we aim to adapt this method to *rings* with n nodes, achieving 3-coloring in $O(\log^* n)$ rounds. A ring is a graph $G = (V, E)$, where $V = \{v_1, \dots, v_n\}$ and $E = \{\{v_i, v_j\} \mid j = i + 1 \mod n\}$. You may assume that G is a *directed ring*, meaning nodes can distinguish their left and right neighbors.[1]

1. Adapt and simplify Algorithms 2.14 and 2.19 to color directed rings. Preserve the running time of $O(\log^* n)$!
2. Modify your solution to also work for *undirected rings*. Preserve the running time of $O(\log^* n)$!

[1]Note that this assumption is stronger than *sense of direction*, which merely requires that nodes can distinguish their neighbors.

Hint: Use the local maxima and minima to segment the ring into subpaths so that within each subpath we have unique identifiers given in an increasing order. Use this ordering to orient each subpath. Then you can apply Algorithms 2.14 and 2.19 in each subpath. Finally, combine the solutions.

3. Note that, in order to decide when to switch from Phase 1 to Phase 2, the nodes running the algorithms actually count $\log^* n$ rounds.[2] However, this is only possible if the nodes are aware of the total number of nodes n. If n is unknown the nodes do not know when the first phase is over: A node v running Algorithm 2.14 cannot simply decide to be done once its color is in $\mathcal{R} = \{0, \ldots, 5\}$ since its parent w might still change its color in the future. Even if the color of w is also in $\mathcal{R}$, w might receive a message from its parent that forces w to change its color once more (potentially to node v's color!).

   Adapt your algorithm for directed rings so that it also works if the nodes do not know n. Preserve the running time of $O(\log^* n)$!

   Hint: Use additional colors to coordinate phase transitions.

[2]Note that a node cannot wait until it knows that all other nodes are ready to start the second phase as this would require time in the order of n.