Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2017

Prof. R. Wattenhofer
Sebastian Brandt

Principles of Distributed Computing
Exercise 1

1 Vertex Coloring

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).

a) What is the message complexity, i.e., the total number of messages the algorithm sends in the worst case?

Hint: Note that the "undecided" messages sent in Line 6 are actually not needed. A node could just as well send no message at all. Therefore neglect these messages in your analysis!

b) Let m be the number of edges. Can you come up with an exact number of messages being sent?

2 TDMA

You have learned in the lecture about coloring and TDMA (time division multiple access). We will now present two applications to show you the relevance of coloring. Consider some nodes, which are placed as shown in Figure 1. These nodes exchange messages using some kind of wireless communication.

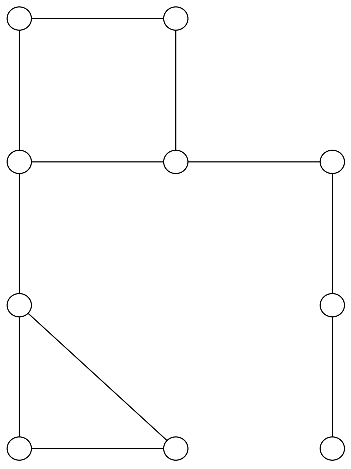

Figure 1: The wireless network

a) If two neighboring nodes send a message at the same time, then none will receive a message (because the messages interfere). We now want to allocate these nodes slots such that they can communicate without any interference. Hence, your task is to color the nodes of this specific instance with as few colors as possible to maximize the amount of messages that can be exchanged.

b) You have upgraded to a faster wireless communication method. But it is a bit more susceptible to errors. In addition to two neighboring nodes not being allowed to send a message simultaneously, there is another constraint. If two (or more) of my neighbors send a message, then I can no longer decode either message correctly. How can you model this? How many time colors do you need now?

c) We now present a different area in which coloring is useful. 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.

3 Coloring Rings and Trees

The combination of Algorithms 1.17 and 1.21 in the lecture notes colors any (directed) tree consisting of n nodes with 3 colors in $O(\log^* n)$ rounds. It consists of two phases: In the first phase (Algorithm 1.17), the initial coloring consisting of all node IDs is reduced to 6 colors, in the second phase (Algorithm 1.21), the 6 colors are further reduced to 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. 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 1.17 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!).

In the following, we want to overcome this problem, and make Algorithms 1.17 and 1.21 work even if the nodes are unaware of n. To make our lives easier we try to find a solution for the ring topology before we tackle the problem on trees. Formally, a ring is a graph $G = (V, E)$, where $V = \{v_1, \ldots, v_n\}$ and $E = \{\{v_i, v_j\} \mid j = i + 1 \pmod{n}\}$. You can assume that G is a *directed* ring, i.e., nodes can distinguish between "left" and "right".[1]

a) Show how the log-star coloring algorithm for trees can be adapted for rings given that the nodes know n!

b) Now adapt your algorithm from a) so that it also works if the ring nodes do not know n. Preserve the running time of $O(\log^* n)$!

Once the color of node v in the ring is in $\mathcal{R}$, it wants to reduce the numbers of colors used to 3. Show how this reduction phase works even if the first phase may not have terminated in some other parts of the ring![2]

Hint: You can use additional colors to segment the ring, and switch phases locally.

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

[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.

c*) Based on the previous exercise, propose a uniform algorithm that colors any directed tree in $O(\log^* n)$ rounds with at most 3 colors! A distributed algorithm is called *uniform* if it works without the knowledge of the number of nodes n.[3]

[3]Problems marked with an asterisk (*) are hard. Example solutions to these problems will not be provided. However, anybody who solves such a problem will receive a prize!