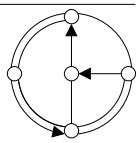
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



FS 2009 Prof. R. Wattenhofer / C. Lenzen / T. Locher

Principles of Distributed Computing
Exercise 1: Sample Solution

1 Vertex Coloring

a) Note that an "undecided" message can be realized by sending nothing at all; thus we do not count such messages. Therefore, each node sends exactly two messages to each neighbor, one in the first round and one after assigning a color. Hence, the total number of messages is $4|E|$, as 4 messages are sent over each edge.

b) Yes, the algorithm still works, it could be reformulated in the following way (we assume that each node knows its degree):

Algorithm 1 Asynchronous "$\Delta + 1$"-Coloring

c)
1: **send** node ID to all neighbors.
2: wait until all neighbor IDs have been received and all neighbors with a lower ID have chosen a color
3: choose smallest possible color
4: **send** chosen color to all neighbors

2 Coloring Trees

a) The log-star algorithm for the ring is basically identical to the algorithm for trees. Nodes do not have a parent in the ring, therefore we simply define the left neighbor of any node to be its "parent". Given this definition, we can run the normal log-star algorithm. Using the same argumentation as for trees, it can be shown that no two neighboring nodes choose the same color.

b) We use two additional colors, ℓ and r, to solve the termination problem. Furthermore, we let each node send its color to both neighbors between each round of the log-star algorithm. This way, each node always knows the colors of both neighbors at the beginning of a round of the log-star algorithm.

The "end game" works as follows: Whenever a node reaches a color in $\mathcal{R} \cup \{\ell, r\}$, it sends this information to its neighbors and does not change its color anymore, i.e., it waits until its neighbors have also acquired a color in this range in order to start the color reduction phase. If a node v learns that the color of its left neighbor is in $\mathcal{R}$ (regardless of the color of the right neighbor), and v's color is not in $\mathcal{R}$, then v recolors itself with the color ℓ and waits until both its neighbors have a color in $\mathcal{R} \cup \{\ell, r\}$. If a node v learns that the color of its right neighbor is in $\mathcal{R}$, while the color of its left neighbor and its own color are both not in $\mathcal{R}$, then v recolors itself with the color r and waits until both its neighbors have a color in $\mathcal{R} \cup \{\ell, r\}$. Additionally, as a node v to the right of a node colored ℓ no longer receives new colors, we need the rule that v simply takes an arbitrary color $c \in \mathcal{R}$ as the new color of its parent and computes its new color based on c and its own color in each round.

We now have to prove that no two neighboring nodes use the same color in $\mathcal{R} \cup \{\ell, r\}$. Assume nodes u and v to be both colored ℓ and assume u to be the parent of v. They cannot have reached color ℓ at the same time, since u only adopts this color if its color is not in $\mathcal{R}$, but its parent is in $\mathcal{R}$. This condition cannot hold for v as well. If u reaches color ℓ first, then v will reduce its color according to an arbitrary color from $\mathcal{R}$ in each following round and thus will never choose color ℓ. If node v gets color ℓ, u's color must be in $\mathcal{R}$ and thus u will not change its color again, in particular not to color ℓ, proving that two neighboring nodes cannot be colored ℓ. The same argumentation applies also to the color r. The log-star algorithm itself ensures that no two neighboring nodes get the same color $c \in \mathcal{R}$. Note that a node v to the right of a node colored ℓ acts like the root in a tree and can thus simply choose an arbitrary color $c \in \mathcal{R}$ in each round without causing any conflicts.

Once a node v colored $c \in \mathcal{R} \cup \{\ell, r\}$ notices that both neighbors also have colors in $\mathcal{R} \cup \{\ell, r\}$, it starts the reduction phase, removing first the colors ℓ and r. Subsequently, the colors are further reduced to the colors $\{0, 1, 2\}$. The nodes need to be synchronized for this procedure, i.e., a node receiving a *reduction message* might have to wait if its other neighbor has not reached the reduction phase. However, after $2 \log^* n + O(1)$ time, where the factor 2 is caused by the additional round of messages between each round of the log-star algorithm, all nodes will be in the reduction phase, enabling a reduction to 3 colors in constant time.

The running time can be reduced to $\log^* n + O(1)$ by simply reordering the steps of the algorithm: Let the nodes send their initial color to both neighbors in a first step. After this step, each round consists of computing the new color, sending them to both neighbors, and receiving the new colors. This ensures that at the beginning of each round, we already know the color of both neighbors. This way, we do not need an additional round of messages between each regular round to figure out if the algorithm is close to termination.