ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2017

Prof. R. Wattenhofer
Sebastian Brandt

Principles of Distributed Computing
Sample Solution to Exercise 1

1 Vertex Coloring

a) If the nodes omit the "undecided" messages 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.

If we want to express the maximal message complexity in terms of n, we have to consider the topology that maximizes $|E|$, the clique in particular. A clique has $n(n-1)/2$ edges. Hence the message complexity is $2n(n-1)$ in the worst case.

2 TDMA

a) The resulting coloring is depicted in Figure 1. Note that the clique of size 3 needs at least 3 colors. Hence, our solution is optimal.

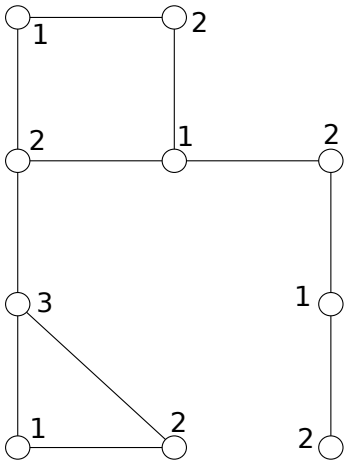

Figure 1: The slots for the wireless network

b) We now have to add additional edges to model the new interferences. The resulting coloring is depicted in Figure 2. The clique of size 4 needs at least 4 colors. Hence, our solution is optimal.

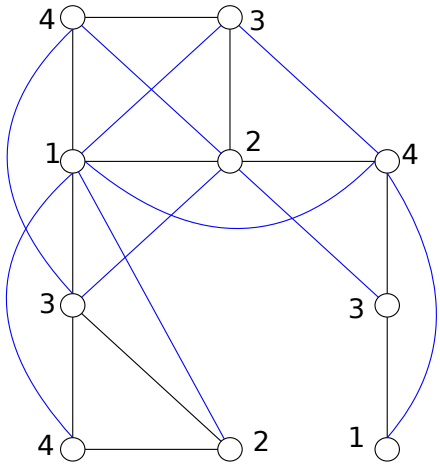

Figure 2: The slots for the wireless network with improved communication. The additional constraints are shown in blue.

c) Every pair of lectures that is selected by a student cannot take place at the same time. Thus, they cannot be colored with the same color. This leads us to the solution shown in Figure 3. The graph can be colored with 3 colors.

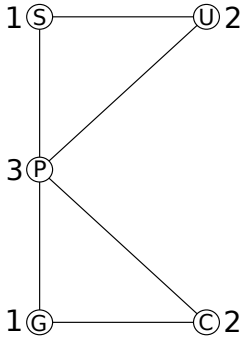

Figure 3: The resulting graph with P being shorthand for Principles of Distributed Computing, S for Statistical Learning Theory, U for Ubiquitous Computing, G for Graph Theory, and C for Cryptography. Note that 3 colors are sufficient.

3 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. Note that we can omit the "shift" step, as all "children" (i.e., the right neighbor) always have 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 ("6-color").

The algorithm works as follows for a node v: As long as neither v nor one of its neighbors has a color in $\mathcal{R} \cup \{\ell, r\}$, it executes Algorithm 6-color. If 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

Algorithm 1 Synchronous "3"-Coloring on Ring

```
1:  send c_v to both neighbors
2:  while c_v ∉ R ∪ {ℓ, r} do
3:    if c_ℓ ∈ R then
4:      c_v := ℓ;
5:    else if c_r ∈ R then
6:      c_v := r;
7:    else if c_ℓ = ℓ then
8:      do one step of Algorithm 6-color (Lines 6–10) with an arbitrary color c ∈ R as parent
        color
9:    else
10:     do one step of Algorithm 6-color (Lines 6–10)
11:   send c_v to both neighbors
12: wait until both neighbors' colors are in R ∪ {ℓ, r}
13: while c_v ∉ {0, 1, 2} do
14:   x := (roundnumber mod 5) + 3;
15:   if (c_v = x) ∨ (c_v = ℓ ∧ x = 6) ∨ (c_v = r ∧ x = 7) then
16:     choose smallest available color c_v ∈ {0, 1, 2}
17:     send c_v to both neighbors
```

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 (Line 8).

Inside the first while-loop, v eventually reaches a color in $\mathcal{R} \cup \{\ell, r\}$. After that, node v waits until its neighbors have also acquired a color in this range in order to start the color reduction phase. In each round of the color reduction phase (Lines 13–17), one color of $\{3, 4, 5, \ell, r\}$ is replaced by a legal color in $\{0, 1, 2\}$. As all nodes know the absolute number of rounds that have passed so far, this can be done without interference.

To prove correctness, we have to show that when reaching Line 13 of the algorithm no two neighboring nodes u and v have picked the same color in $\mathcal{R} \cup \{\ell, r\}$. We show this for ℓ in more detail:

Lemma. Two neighbors u and v cannot both be colored ℓ.

Proof. Let u be the parent of v w.l.o.g. A node only adopts ℓ if its color is not in $\mathcal{R}$, but its parent is in $\mathcal{R}$. This condition cannot hold for two neighboring nodes at the same time. Hence, u and v cannot reach color ℓ in the same round.

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, it will never choose color ℓ.

If v reaches color ℓ first, 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 logic of Algorithm 6-color 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.

The running time of the algorithm is $O(\log^* n)$.