CS-E4510 Distributed Algorithms / Jukka Suomela
exam, 19 October 2022

Material. You can bring one A4 paper (two-sided), with any content you want. No other material or equipment is allowed in the exam.

Instructions. There are three questions; please ***try to answer something in each of them***. If you cannot solve a problem entirely, please try to solve at least some special case or simpler version of the problem (see the hints for suggestions), or failing that, please at least explain what you tried and what went wrong. Do not spend too much time with one problem; the problems are not listed in order of difficulty and they do not depend on each other.

For question 1, we expect a ***mathematical proof*** (it can be short and informal). For questions 2–3, it is sufficient to give a ***brief, informal description*** of the algorithm, and a ***brief, informal explanation*** of why it solves the problem correctly. You do not need to specify algorithms in the state-machine formalism, or give a complete proof of correctness. You are free to refer to algorithms and results that we discussed in the lectures, course material, and exercises; there is no need to repeat any of their details. In all questions it is enough that a friendly, cooperative reader can understand your idea correctly and see why it makes sense.

Definitions. ***List k-coloring*** is the following graph problem: The input is a graph $G = (V, E)$ in which each node $v \in V$ is given a set of colors $L(v) \subseteq \{1, 2, \dots\}$ with exactly k elements. As output each node has to be labeled with a value $f(v) \in L(v)$ such that for each edge $\{u, v\} \in E$ we have $f(u) \neq f(v)$. That is, we need to produce a proper vertex coloring, but for each node we have given a specific list of possible colors.

Note that if you set $L(v) = \{1, 2, \dots, k\}$ for every node $v \in V$, this is exactly the familiar problem of vertex coloring with k colors. However, in the list k-coloring different nodes may have different color palettes.

As usual, n denotes the number of nodes in the input graph.

Question 1: Graph theory. Prove both of these claims:

(a) ***List* 2-*coloring is not always solvable in a* 3-*cycle.*** That is, let $G = (V, E)$ be a cycle with 3 nodes and 3 edges. Then we can choose the sets $L(v)$ with 2 elements each so that it is impossible to choose any valid colors $f(v) \in L(v)$ from these sets.

(b) ***List* 2-*coloring is always solvable in a* 4-*cycle.*** That is, let $G = (V, E)$ be a cycle with 4 nodes and 4 edges. Then no matter how we choose the sets $L(v)$ with 2 elements each, it is always possible to find valid colors $f(v) \in L(v)$ from these sets.

Hint: consider separately two different cases: (1) all lists are equal; (2) otherwise there is a pair of adjacent nodes with different lists.

Question 2: PN. Give a deterministic distributed algorithm that solves ***list* 2-*coloring*** in the following setting in the PN model (any running time is fine):

- Graph family: ***trees*** (i.e., connected acyclic graphs).
- Local inputs: each node v is given its set $L(v)$ with 2 elements; furthermore, exactly one node is marked as a ***leader*** (there is exactly one node that gets the input "you are the leader" and all other nodes get the input "you are not the leader").
- Local outputs: each node v outputs its color $f(v) \in L(v)$.

Hint: Solve the problem first in a path in which exactly one endpoint is marked as a leader.

Question 3: LOCAL. Give a deterministic distributed algorithm that solves ***list* 3-*coloring*** in the following setting in the LOCAL model in $\boldsymbol{o(n)}$ rounds:

- Graph family: ***cycle graphs*** (i.e., connected 2-regular graphs).
- Local inputs: each node v gets as input its set $L(v)$ with 3 elements, its own unique identifier, and the value of n. You can assume that the unique identifiers are $\{1, 2, \ldots, n\}$.
- Local outputs: each node v outputs its color $f(v) \in L(v)$.

Hints: Please note that the running time has to be $o(n)$, this is little-o, not big-O. So for example $\log n$ or $\log^* n$ is good, while n or $n/100$ is not enough. Also note that we did not promise that the cycle is directed—but if you cannot solve the general case, please try to solve at least the case of a directed cycle. If you cannot design a deterministic algorithm, try to design a randomized algorithm.