Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
Mohamad Ahmadi, Hamid Ghodselahi June 15, 2016

Network Algorithms, Summer Term 2016
Problem Set 7

hand in by Wednesday, June 22, 2016

Exercise 1: Coloring Rings

In Chapter 1, we proved that a ring can be colored with 3 colors in $\log^* n + O(1)$ rounds. Clearly, a ring can only be (legally) colored with 2 colors if the number of nodes is even.

1. Prove that, even if the nodes in a directed ring know that the number of nodes is even, coloring the ring with 2 colors requires $\Omega(n)$ rounds![1]
2. Since coloring a ring with 2 colors apparently takes a long time, we again resort to the problem of coloring rings using 3 colors. Assume that a *maximal independent set* (MIS) has already been constructed on the ring, i.e., each node knows whether it is in the independent set or not. Give an algorithm to color the ring with 3 colors in this scenario! What is the time complexity of your algorithm? Deduce from this a lower bound for computing a MIS!

Exercise 2: Coloring unrooted trees

In the coloring chapter we have seen that any **rooted** tree consisting of n nodes can be 3-colored in $O(\log^* n)$ rounds. In contrast, we showed a lower bound of $\Omega(\log n / \log\log n)$ for coloring **unrooted** trees with a constant number of colors.

The goal of this exercise is to almost match the lower bound, that is, we show that unrooted trees can be 3-colored in time $O(\log n)$. We will develop the algorithm in four steps:

1) Assume the nodes V of an unrooted tree $T = (V, E)$ are partitioned into two sets V_0 and V_1 such that all the nodes in V_0 have degree at most 2 and the nodes in V_1 have degree at least 3. Further, suppose we are given valid 3-colorings of the forests $T[V_0]$ and $T[V_1]$ induced by V_0 and V_1.

   Provide an algorithm that merges these two colorings into a valid 3-coloring of T in time $\mathcal{O}(1)$.

2) Show that in any tree with n nodes, at least $\frac{n}{2}$ of the nodes have degree at most 2.

3) Use part 1) and 2) to provide a recursive algorithm for 3-coloring unrooted trees in $O(\log^* n \cdot \log n)$ rounds. What is the recurrence relation of your algorithm?

   Hint: *There is a distributed algorithm that finds a $(\Delta + 1)$-coloring in any graph with maximum degree at most Δ and n nodes in $\mathcal{O}(\Delta^2 + \log^* n)$ rounds (Tutorial Session 1). If the maximum degree Δ is constant this implies a 3-coloring algorithm with time complexity $\mathcal{O}(\log^*(n))$. You can use this result as a black box.*

4) Describe how to parallelize parts of your algorithm to improve the runtime from $\mathcal{O}(\log^* n \log n)$ to $\mathcal{O}(\log n)$.

[1] As in the lecture, the message size and local computations are unbounded and all nodes have unique identifiers from 1 to n.