Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. F. Kuhn
Mohamad Ahmadi, Hamid Ghodselahi June 22, 2016

Network Algorithms, Summer Term 2015
Problem Set 7 – Sample Solution

Exercise 1: Coloring Rings

1. Let $n \geq 4$ be even, and $r = n/2 - 2$. Consider the r-neighborhood graph $\mathcal{N}_r(R_n)$ of the ring R_n with n nodes. Note that for $r = n/2 - 2$ the r-neighborhood of a node contains all but three identifiers, ordered according to their occurrence.

   Then it follows from Lemma 7.5 that the ring can be colored legally with two colors in r rounds if and only if $\mathcal{N}_r(R_n)$ is bipartite, i.e., the r-neighborhood contains no odd cycle. However, there is one of length $n - 1$:
   $(1, \dots, n-3), (2, \dots, n-2), (3, \dots, n-1), (4, \dots, n), (5, \dots, n, 1), \dots,$
   $(n, 1, 2 \dots, n-4), (1, \dots, n-3)$.
   Thus no coloring of the ring with 2 colors is possible in less than $n/2 - 1$ rounds.

2. Each node informs its two neighbors whether it is in the MIS or not and additionally sends its identifier. If node v is in the MIS, it sets its color to 1. If v is not in the MIS but both of its neighbors are, then v sets its color to 2. If v has a neighbor w not in the MIS, v chooses color 2 if its identifier is larger than w's identifier, otherwise v chooses the color 3.

   The algorithm only needs one communication round. Correctness follows from the fact that either a node v is in the MIS or at least one of its neighbors is. Thus, a MIS can at best be computed one round faster than a 3-coloring, which implies that computing a MIS costs at least $(\log^* n)/2 - 2$ rounds (since coloring a directed ring with 3 or less colors needs at least $(\log^* n)/2 - 1$ rounds. See Theorem 7.11).

Exercise 2: Coloring Unrooted Trees

1. All the nodes in V_1 keep the color from the 3-coloring of $T[V_1]$. Clearly, this does not create any conflicts in T. For the nodes in V_0, we iterate through all the 3 colors. When considering color x, all nodes that have color x in the 3-coloring of $T[V_0]$, select the minimum possible available color (note that these nodes always form an independent set of T). Given on 3-coloring of $T[V_0]$ and $T[V_1]$, this allows to compute a 3-coloring of T in 3 rounds.

2. We utilize the following fact: the sum of degrees of all nodes in a tree equals twice the number of edges.

   Assume that x denotes the number of nodes with degree 2 and let y be the number of nodes with degree 1. Generally the number of edges in a tree equals $n - 1$. Therefore we have

   $$2x + y + 3(n - x - y) \leq 2(n - 1).$$

   The above inequality implies that $x + 2y \geq n + 2$ and it concludes that $x + y \geq n/2$.

3. The algorithm is defined in recursive steps as follows: In Step 1, the set V is partitioned into two sets V_0^1 and V_1^1 where V_0^1 includes the nodes with degree at most 2 and V_1^1 includes the nodes with degree at least 3. With respect to the part 2 we know $|V_0^1| \geq |V|/2$. The algorithm from the hint is applied on the set V_0^1 and therefore in $O(\log^* n)$ time we have a 3-coloring for the forest induced by the nodes in V_0^1. Now we have the set V_1^1 of nodes with degree at least 3 that are not colored yet.

   Generally in each Step $i \geq 2$, the set V_1^{i-1} is partitioned into two sets V_0^i and V_1^i such that V_0^i includes the nodes with degree at most 2 whose size is at least $|V_1^{i-1}|/2$ (w.r.t. part 2) and V_1^i includes the nodes with degree at least 3 whose size is at most $|V_1^{i-1}|/2$. As Step 1, the algorithm from the hint is applied on the forest induced by the nodes in V_0^i and we get a 3-coloring in time $O(\log^* n)$ for the forest induced by the nodes in V_0^i. Thus at the end of Step $i \geq 1$, the number of nodes that are colored is at least $n - n/2^i$. As a result, it takes $S = O(\log n)$ steps till we get S number of valid 3-colorings that color all the nodes and each step needs $O(\log^* n)$ time.

   Now we need to merge these S 3-colorings into a final valid 3-coloring. In the merging part, we use part 1 and start from the bottom level of recursion tree formed by first part of the recursive algorithm that has been already described. Hence, we do S merges and each merge gives us a 3-coloring in at most 3 rounds regarding to the part 1. Therefore at the end, we get a 3-coloring for T in time $O(\log n \cdot \log^* n)$. Regarding to the described recursive algorithm the recurrence relation is

$$T(n) \leq T(n/2) + c.\log^* n + 3$$

   where c is a constant. Solving the above recurrence relation gives $T(n) = O(\log n \cdot \log^* n)$.

4. We can do the following trick to get rid of the factor $\log^* n$ in the final result. At each Step i, we can parallelize applying the algorithm from the hint on the forest induced by the set V_0^i and solving the problem for the remaining nodes in the set V_1^i. Hence, the recurrence relation is as follows

$$T(n) \leq \max\{T(n/2), c.\log^*\} + 3.$$

   Using one of the tools to solve a recurrence relation, say replacing, therefore $T(n) = O(\log n)$.