Albert-Ludwigs-Universität, Inst. für Informatik
Prof. Dr. Fabian Kuhn

Distributed Systems, Summer Term 2020
Exercise Sheet 7

1 Maximal matching

In the following, we are given a graph $G = (V, E)$ of maximum degree Δ, where *nodes* are colored with c colors, and the goal is to produce a maximal matching. A maximal matching is a subset of edges $X \subseteq E$ satisfying the following:

- For all e_1, e_2 in X, it holds that e_1 and e_2 are not incident to the same node, that is, they do not share endpoints. Hence, for each node it holds that at most one incident edge is in the matching.
- Adding any additional edge of $E \setminus X$ to X would violate the above constraint.

Hence, we are interested in a subset of edges that are independent such that this subset cannot be extended.

1. Consider the case where $c = 2$, that is, the graph is bipartite and properly colored with two colors, black and white. Assume that nodes know the value of Δ and c. Show that maximal matching can be solved in $O(\Delta)$ rounds. Hint: it can be solved in 2Δ rounds. Spoiler hint: see the footnote[1].
2. Assume that c and Δ are known to each node. Show that, for any value of c, this problem can be solved in $O(c\ \Delta)$.
3. Show that this problem can be solved in $O(c\ \Delta)$ even in the case where c and Δ are unknown to the nodes.

Sample Solution

1. **Proposals Algorithm:** Each node v numbers its incident edges from 1 to $\deg(v)$ in an arbitrary way—we call these numbers *ports*. The idea is that, each white node that is not matched sends a matching proposal to one of its neighbors, processing neighbors in order of its ports. Each black node that have got one or more proposals, accepts exactly one and rejects the others—breaking ties using their ports. In other words, at round 1 each white node will propose to the neighbor connected to its port 1 (some proposals will be accepted and some may be rejected). Next, each white node that is still unmatched will send proposals through the port-2 edge, and so on. In this way, each time that we add an edge to the matching we are sure that it does not share any endpoint with other edges in the matching (white nodes propose only to one neighbor at a time and black nodes accept only one proposal and reject the others). So in $O(\Delta)$ rounds, each white node has proposed to all its neighbors and has got an answer. We argue that after $O(\Delta)$ such rounds we get a maximal matching: if a white node ends up unmatched, it means that it has got rejection from all its neighbors, which means that all its neighbors are matched with someone else; if a black node that is unmatched it means that it did not get any proposals, which means that all its white neighbors got matched with someone else.

[1] White nodes can try to „propose" to each black neighbor, by trying one neighbor at a time. Black nodes can accept the first proposal and reject all the others.

2. We go through color classes and for each color class i we run the Proposals Algorithm considering the nodes of color i as white nodes and the other as black nodes. Since for each color class we spend $O(\Delta)$ rounds, we obtain a running time of $O(c\Delta)$.

3. The crucial ingredient that makes the previous approaches work is that the nodes that propose in parallel form an independent set. Here we can pick such nodes by considering local minima as white nodes and the others as black nodes. Since Δ is not known, once a node v has finished proposing to all its neighbors, v announces that it is "not active" to its neighbors. Nodes check if they are a local minima on the graph induced by active nodes.

2 Coloring planar graphs

Show how to color a planar graph with $O(1)$ colors in $O(\log n)$ time. Hint: every planar graph satisfies that the average degree of the nodes is less than 6. Hint: use the same idea of the algorithm for unrooted trees presented in the lecture.

Sample Solution

We want to be able to orient edges such that, for each node, we can upper bound the number of out-degree by some constant k. We proceed as follows: we show that at least a constant fraction of the nodes have degree less than some constant k; then we can remove these nodes from the graph and recurse on the remaining graph—notice that, if we remove nodes from a planar graph, the remaining graph is still planar. If we can do this, we are able to decompose the graph in a similar way as in the lecture in $O(\log n)$ rounds. If we can do that, then we can use the same algorithm shown in the lecture that colors the graph with $3^k = O(1)$ colors in $O(\log^* n)$ rounds, obtaining a running time of $O(\log n) + O(\log^* n) = O(\log n)$ rounds.
Claim: Less then $n/2$ nodes have degree greater than 11.
Proof: Suppose at least $n/2$ nodes have degree ≥ 12. Then we get that the average degree in the graph is $\frac{12n/2}{n} = 6$, which is a contradiction since we know that in a planar graph the average degree is strictly less than 6. □
Our claim implies that at least $n/2$ nodes have degree ≤ 11, which means that we can decompose the graph as in the lecture and orient the edges such that each node has out-degree at most 11, and this can be done in $O(\log n)$ rounds. Now we can color the graph with the algorithm seen in the lecture using $3^{11} = O(1)$ colors in $O(\log^* n)$ rounds.

3 Coloring unrooted trees

Show that it is possible to 3-color unrooted trees in $O(\log n)$ time. Hint: modify the algorithm that 9-colors unrooted trees presented in the lecture.

Sample Solution

In the lecture we saw how to decompose the graph in $O(\log n)$ components such that nodes in component i have degree 2 in the graph induced by nodes in components $j \geq i$. Since inside each component we have that the degree is at most 2, we can 3-color nodes in each component in parallel spending $O(\log^* n)$ rounds. This coloring is a proper 3-coloring in the graph induced by the nodes in a certain component i, but it may not be a proper coloring in general, but we can fix inconsistencies in the following way. We process components in a top-down manner, and at each step we fix the inconsistencies that a node of component i has with neighbors belonging to a component $j > i$. To do this, we exploit the fact that the degree of these nodes is at most 2, hence there is always a free color to chose. Moreover, in order to avoid creating conflicts with nodes in the same component, we process nodes of a component i by color classes. More precisely, we proceed in the following way. Base case:

nodes in the last components have no "upper" component, so they trivially satisfy the claim, i.e., they have no inconsistencies with nodes in upper components. So suppose nodes in components $j \geq i+1$ have fixed their inconsistencies with neighbors in the upper components. Now we show that we can fix inconsistencies of nodes in component i. We go through the three color classes, and for each node in color class ℓ we see if the node has inconsistencies with nodes in upper components: if yes, these nodes chose a free available color. Notice that it is not an issue if two nodes that are processed in parallel choose the same color, since these nodes form an independent set. After 3 rounds, we have fixed all inconsistencies of nodes in component i. The proof follows by induction. Overall we spend $O(\log n) + O(\log^* n) = O(\log n)$ rounds.

4 Color Reduction

a) Given a graph which is colored with $m > \Delta + 1$ colors, describe a method to recolor the graph in one round using $m - \lfloor \frac{m}{\Delta+2} \rfloor$ colors.

   Hint: Partition the set of colors into sets of size $\Delta + 2$ and recall the color reduction method from the lecture.

b) Show that after $O(\Delta \log(m/\Delta))$ iterations of step a), one obtains a $O(\Delta)$ coloring.

Sample Solution

a) Partition the set of colors into $\lfloor \frac{m}{\Delta+2} \rfloor$ disjoint sets of size $\Delta + 2$ and one set of size at most $\Delta + 1$. From each set C of size $\Delta + 2$, take the largest color and let each node v with this color choose a new color from C that is not among the colors of its neighbors. If a neighbor u of v concurrently chooses a new color, it will not cause a conflict as u chooses from a disjoint color set. So we obtain a new coloring with $m - \lfloor \frac{m}{\Delta+2} \rfloor$ colors.

b) We calculate the number of iterations needed to obtain at most $2(\Delta + 2)$ colors. In one iteration m is reduced to

$$m - \left\lfloor \frac{m}{\Delta+2} \right\rfloor \leq m - \frac{m}{\Delta+2} + 1 = m\left(1 - \left(\frac{1}{\Delta+2} - \frac{1}{m}\right)\right) \overset{m \geq 2(\Delta+2)}{\leq} m\left(1 - \frac{1}{2(\Delta+2)}\right)$$

So we are looking for the minimum t such that

$$m\left(1 - \frac{1}{2(\Delta+2)}\right)^t \leq 2(\Delta+2)$$

For all $x \in \mathbb{R}$ it holds $1 + x \leq e^x$. It follows

$$m\left(1 - \frac{1}{2(\Delta+2)}\right)^t \leq m \cdot e^{-\frac{t}{2(\Delta+2)}} \leq 2(\Delta+2)\ ,$$

so we choose $t = \left\lceil 2(\Delta+2)\ln\left(\frac{m}{2(\Delta+2)}\right)\right\rceil$.

Once we obtained $O(\Delta)$ colors, we can use a) another $O(\Delta)$ times until $\Delta + 1$ colors are left (as long as $m > \Delta + 1$, at least one color is eliminated in each step). This yields an overall runtime of $O(\Delta \log(m/\Delta))$.