Distributed algorithms for networks: Exercises

Exercise 1 Independent sets in cycles

Consider a cycle with identifiers in $[1, n^2]$.

1. Prove that one can compute a *maximal* independent set in such a cycle in $O(\log^* n)$ rounds.
2. Prove that one cannot compute a *maximum* independent set in such a cycle in $o(n)$ rounds.

Exercise 2 Coloring rooted trees

In this exercise, unless stated otherwise, the network will be a *rooted tree* of size n, where all edges are oriented upward (towards the parent), with unique identifiers in $[1, n^2]$. The degree of the tree is not bounded (that is, a node can have an arbitrarily large number of children). We assume as usual that the nodes of the tree know the size the tree, n. We will prove that we can 3-color rooted trees efficiently.

1. Give a 0-round algorithm to output a coloring with a very large number of colors. (Very large here means that it can be polynomial in n.)

Remember that Cole-Vishkin algorithm is used to color oriented cycle with a constant number of color, by iterating a "one-round Cole-Vishkin procedure" which uses only the current colors of the node and its successor.

2. Show how to adapt Cole-Vishkin algorithm to work on *paths*, in the same asymptotic complexity as on cycles.
3. Show how to adapt Cole-Vishkin algorithm to work on *rooted trees* in the same asymptotic complexity.
4. Define a class of oriented graphs (where *oriented* only means that every edge is oriented in exactly one direction), strictly containing rooted trees, such that the same algorithm works.

A classic technique to reduce the number of colors from a constant a to a smaller constant b, is color elimination, which removes one color at each round.

5. Explain why, in the setting of this exercise, we cannot use this naive color elimination to get down to 3 colors.
6. Give an algorithm to get from k colors to 3 colors in $O(k)$ rounds. Hint: Consider the process in which every node takes the color of its parent.
7. Putting the pieces together, we get an $O(\log^* n)$ algorithm to 3-color oriented trees. This is in stark contrast with the case of "non-rooted" trees. Explain why the lower bound established during the lectures for non-oriented trees cannot be adapted to rooted trees.
8. Give an algorithm for 3^Δ coloring general graphs of maximum degree Δ in $O(\log^* n)$, without using cover-free families.

Exercise 3 Edge coloring via sinkless orientation

Consider the setting we had for the sinkless orientation lower bound: a bipartite 3-regular tree where the nodes are white or black (such that these colors form a proper 2-coloring). Show a lower bound for computing an edge coloring with four colors.

Exercise 4 Error-sensitive local certification

In this exercise, the nodes have inputs, and the properties we want to check refer to these inputs. An example of such a property (the leader property): every node has an input saying whether it is selected or not, and we want to check that there exists exactly one selected node in the graph.

The *edit distance* from a graph with input to a property is the minimum number of inputs that need to be modified in order to get a graph with input that satisfies the property. For example, for the leader property, if there are three selected nodes then this graph with input is at distance 2 from the leader property.

We consider local certification at distance 1: a node can see the identifiers and certificates of its neighbors, and of itself. We say that a certification is α-sensitive if the following holds. For every graph with inputs:

- If the property is satisfied, there exists a certificate assignment such that all nodes accept.
- If the property is not satisfied, then for every certificate assignment, the number of nodes rejecting is at least α times the edit distance from the graph with inputs to the property.

Here is a list of properties:

- *Neighbor identifiers*: in this property, every node should have as an input the identifier of a neighbor or its own itself.
- *Acyclic pointers*, is a strengthening of the *Neighbor identifiers* property: in addition, if we interpret the inputs as pointers, there shouldn't be a cycle of pointers.
- *Spanning tree*, is a strengthening of the *Acyclic pointers* property: now the pointers should form a spanning tree.

1. Prove that the property of *Neighbor identifiers* admits a 1-sensitive local certification, with certificates of size 0.
2. Prove that the property of *Acyclic pointers* admits a 1-sensitive local certification, with certificates of size $O(\log n)$.
3. Prove that the property *Spanning tree* admits no error-sensitive local certification with constant α, whatever the certificate size is. Hint: Consider a path, where every node is pointing to the nearest extremity.

Exercise 5 Upper bounds for certifying k leaders

We consider graphs where each node has an input label (not to be confused with a certificate) that is either *selected* or *non-selected*. The nodes are equipped with identifiers as usual, and the graph has n nodes.

1. We consider the following property: there are *at most k* selected nodes. Prove that there exists a certification of this property of size $O(k \log n)$, where all nodes get the same certificate.

2. We consider the property that there are *exactly* k selected nodes. Prove that there exists a certification of this property of size $O(k \log n)$ (with no further constraints).

3. Prove that for the property of the previous question, one can actually have a certification of size $O(k + \log n)$.
   Hint: Consider the following labeling: in a tree, with exactly one selected node v, every node is given its distance modulo 3 to v.

Exercise 6 Randomized algorithms don't need IDs

At some point in the lectures, we mentioned that for randomized algorithms that succeed with high probability, identifiers are basically useless. Prove this claim. (You can choose the identifier range as you like, but polynomial in n. Randomized algorithm here should succeed with high probability.)

Exercise 7 A lower bound for certifying treedepth

1. What is the treedepth of the cycle C_n on n vertices?

Let n be an integer and G be the graph obtained from $2n$ disjoint paths on 4 vertices a_i, b_i, c_i, d_i and a'_i, b'_i, c'_i, d'_i for $i \in [1, n]$, joined together to a universal vertex.

Given two matchings M, M', we denote by $G(M, M')$ the graph obtained from G by adding M between the vertices $a_1, \dots, a_n$ and $a'_1, \dots, a'_n$, and adding M' between $d'_1, \dots, d'_n$ and $d_1, \dots, d_n$.

2. Show that $G(M, M')$ has treedepth at most 5 if $M = M'$, and treedepth at least 6 otherwise.

3. Using a reduction from the equality problem, show that certifying that an n-vertex graph has treedepth at most 5 requires $\Omega(\log n)$ bits.

Exercise 8 Improved Rake and compress

We consider the following variant of the rake and compress method seen in the lectures. Let T be a tree. We want to select all the vertices V_1 such that $v \in V_1$ is either of degree 1 or has degree 2 and belongs to a chain of at least three vertices each of them being of degree at most 2.

1. Prove that the number of vertices removed by this procedure is linear in n.
   (Optimize the linear factor that is deleted)

2. Prove that a node can determine in a constant number of rounds if it has to be deleted.

3. We apply the following algorithm:
   Let $T_0 = T$ and $V_0 = V$.
   Repeat until the graph is empty:
   Compute the set V_i of vertices of the form described above.
   $T_{i+1} = T_i \setminus V_i$.

   Prove that the procedure ends after in $O(\log n)$ rounds.

4. Now, we compute an IS as follows from the last set to the first set:

- for every node in V_i with at least one neighbor in V_j with $j > i$, we add it in the IS if it has no neighbor in the IS.
- We greedily complete the IS using a 3-coloring algorithm (we compute a 3-coloring and then try to reduce the colors starting with color 2 and then color 3).

Prove that the independent set I computed by this algorithm satisfies that every connected component of $V \setminus I$ has at size at most 3.

5. Conclude that we can find in $O(\log n)$ rounds an IS in every tree T whose deletion leaves a graph whose connected components have size at most 3.
6. Is it optimal?
   Hint: Prove that we can, in a centralized way, compute an IS whose deletion leaves connected components of size one.