Chapter 11

Locality Lower Bounds

In Chapter 1, we looked at distributed algorithms for coloring. In particular, we saw that rings and rooted trees can be colored with 3 colors in $\log^* n + O(1)$ rounds. In this chapter, we will reconsider the distributed coloring problem. We will look at a classic lower bound by Nathan Linial that shows that the result of Chapter 1 is tight: Coloring rings (and rooted trees) indeed requires $\Omega(\log^* n)$ rounds. In particular, we will prove a lower bound for coloring in the following setting:

- We consider deterministic, synchronous algorithms.

- Message size and local computations are unbounded.

- We assume that the network is a directed ring with n nodes.

- Nodes have unique labels (identifiers) from 1 to n.

Remarks:

- A generalization of the lower bound to randomized algorithms is possible. Unfortunately, we will however not have time to discuss this.

- Except for restricting to deterministic algorithms, all the conditions above make a lower bound stronger. Any lower bound for synchronous algorithms certainly also holds for asynchronous ones. A lower bound that is true if message size and local computations are not restricted is clearly also valid if we require a bound on the maximal message size or the amount of local computations. Similarly also assuming that the ring is directed and that node labels are from 1 to n (instead of choosing IDs from a more general domain) strengthen the lower bound.

- Instead of directly proving that 3-coloring a ring needs $\Omega(\log^* n)$ rounds, we will prove a slightly more general statement. We will consider deterministic algorithms with time complexity r (for arbitrary r) and derive a lower bound on the number of colors that are needed if we want to properly color an n-node ring with an r-round algorithm. A 3-coloring lower bound can then be derived by taking the smallest r for which an r-round algorithm needs 3 or fewer colors.

Algorithm 39 Synchronous Algorithm: Canonical Form

1: In r rounds: **send** complete initial state to nodes at distance at most r
2:   *// do all the communication first*
3: Compute output based on complete information about r-neighborhood
4:   *// do all the computation in the end*

11.1 Locality

Let us for a moment look at distributed algorithms more generally (i.e., not only at coloring and not only at rings). Assume that initially, all nodes only know their own label (identifier) and potentially some additional input. As information needs at least r rounds to travel r hops, after r rounds, a node v can only learn about other nodes at distance at most r. If message size and local computations are not restricted, it is in fact not hard to see, that in r rounds, a node v can exactly learn all the node labels and inputs up to distance r. As shown by the following lemma, this allows to transform every deterministic r-round synchronous algorithm into a simple canonical form.

Lemma 11.1. *If message size and local computations are not bounded, every deterministic, synchronous r-round algorithm can be transformed into an algorithm of the form given by Algorithm 39 (i.e., it is possible to first communicate for r rounds and then do all the computations in the end).*

Proof. Consider some r-round algorithm $\mathcal{A}$. We want to show that $\mathcal{A}$ can be brought to the canonical form given by Algorithm 39. First, we let the nodes communicate for r rounds. Assume that in every round, every node sends its complete state to all of its neighbors (remember that there is no restriction on the maximal message size). By induction, after r rounds, every node knows the initial state of all other nodes at distance at most i. Hence, after r rounds, a node v has the combined initial knowledge of all the nodes in its r-neighborhood. We want to show that this suffices to locally (at node v) simulate enough of Algorithm $\mathcal{A}$ to compute all the messages that v receives in the r communication rounds of a regular execution of Algorithm $\mathcal{A}$.

Concretely, we prove the following statement by induction on i. For all nodes at distance at most $r - i + 1$ from v, node v can compute all messages of the first i rounds of a regular execution of $\mathcal{A}$. Note that this implies that v can compute all the messages it receives from its neighbors during all r rounds. Because v knows the initial state of all nodes in the r-neighborhood, v can clearly compute all messages of the first round (i.e., the statement is true for $i = 1$). Let us now consider the induction step from i to $i + 1$. By the induction hypothesis, v can compute the messages of the first i rounds of all nodes in its $(r - i + 1)$-neighborhood. It can therefore compute all messages that are received by nodes in the $(r - i)$-neighborhood in the first i rounds. This is of course exactly what is needed to compute the messages of round $i + 1$ of nodes in the $(r - i)$-neighborhood. $\qquad\square$

Remark:

- It is straight-forward to generalize the canonical form to randomized algorithms: Every node first computes all the random bits it needs throughout the algorithm. The random bits are then part of the initial state of a node.

Definition 11.2 (r-hop view)**.** *We call the collection of the initial states of all nodes in the r-neighborhood of a node v, the r-hop view of v.*

Remark:

- Assume that initially, every node knows its degree, its label (identifier) and potentially some additional input. The r-hop view of a node v then includes the complete topology of the r-neighborhood (excluding edges between nodes at distance r) and the labels and additional inputs of all nodes in the r-neighborhood.

Based on the definition of an r-hop view, we can state the following corollary of Lemma 11.1

Corollary 11.3. *A deterministic r-round algorithm $\mathcal{A}$ is a function that maps every possible r-hop view to the set of possible outputs.*

Proof. By Lemma 11.1, we know that we can transform Algorithm $\mathcal{A}$ to the canonical form given by Algorithm 39. After r communication rounds, every node v knows exactly its r-hop view. This information suffices to compute the output of node v. $\square$

Remarks:

- Note that the above corollary implies that two nodes with equal r-hop views have to compute the same output in every r-round algorithm.

- For coloring algorithms, the only input of a node v is its label. The r-hop view of a node therefore is its labeled r-neighborhood.

- Since we only consider rings, r-hop neighborhoods are particularly simple. The labeled r-neighborhood of a node v (and hence its r-hop view) in a directed ring is simply a $(2r+1)$-tuple $(\ell_{-r}, \ell_{-r+1}, \ldots, \ell_0, \ldots, \ell_r)$ of distinct node labels where ℓ_0 is the label of v. Assume that for $i > 0$, ℓ_i is the label of the ith clockwise neighbor of v and ℓ_{-i} is the label of the ith counterclockwise neighbor of v. A deterministic coloring algorithm for directed rings therefore is a function that maps $(2r+1)$-tuples of node labels to colors.

- Consider two r-hop views $\mathcal{V}_r = (\ell_{-r}, \ldots, \ell_r)$ and $\mathcal{V}'_r = (\ell'_{-r}, \ldots, \ell'_r)$. If $\ell'_i = \ell_{i+1}$ for $-r \leq i \leq r-1$ and if $\ell'_r \neq \ell_i$ for $-r \leq i \leq r$, the r-hop view $\mathcal{V}'_r$ can be the r-hop view of a clockwise neighbor of a node with r-hop view $\mathcal{V}_r$. Therefore, every algorithm $\mathcal{A}$ that computes a valid coloring needs to assign different colors to $\mathcal{V}_r$ and $\mathcal{V}'_r$. Otherwise, there is a ring labeling for which $\mathcal{A}$ assigns the same color to two adjacent nodes.

11.2   The Neighborhood Graph

We will now make the above observations concerning colorings of rings a bit more formal. Instead of thinking of an r-round coloring algorithm as a function from all possible r-hop views to colors, we will use a slightly different perspective. Interestingly, the problem of understanding distributed coloring algorithms can itself be seen as a classical graph coloring problem.

Definition 11.4 (Neighborhood Graph). *For a given family of network graphs $\mathcal{G}$, the r-neighborhood graph $\mathcal{N}_r(\mathcal{G})$ is defined as follows. The node set of $\mathcal{N}_r(\mathcal{G})$ is the set of all possible labeled r-neighborhoods (i.e., all possible r-hop views). There is an edge between two labeled r-neighborhoods $\mathcal{V}_r$ and $\mathcal{V}'_r$ if $\mathcal{V}_r$ and $\mathcal{V}'_r$ can be the r-hop views of two adjacent nodes.*

Lemma 11.5. *For a given family of network graphs $\mathcal{G}$, there is an r-round algorithm that colors graphs of $\mathcal{G}$ with c colors iff the chromatic number of the neighborhood graph is $\chi(\mathcal{N}_r(\mathcal{G})) \leq c$.*

Proof. We have seen that a coloring algorithm is function that maps every possible r-hop view to a color. If two r-hop views $\mathcal{V}_r$ and $\mathcal{V}'_r$ can be the r-hop views of two adjacent nodes u and v (for some labeled graph in $\mathcal{G}$), every coloring algorithm must assign different colors to $\mathcal{V}_r$ and $\mathcal{V}'_r$. Hence, a coloring algorithm assigns a color to every node of the neighborhood graph $\mathcal{N}_r(\mathcal{G})$. The algorithm must assign different colors two adjacent neighborhood graph nodes (i.e., if the corresponding r-hop views can be r-hop views of neighboring nodes).   □

Instead of directly defining the neighborhood graph for directed rings, we define directed graphs $\mathcal{B}_{k,n}$ that are closely connected to the neighborhood graph. Let k and n be two positive integers and assume that $n \geq k$. The node set of $\mathcal{B}_{k,n}$ contains all k-tuples of increasing node labels ($[n] = \{1, \ldots, n\}$):

$$V[\mathcal{B}_{k,n}] \;=\; \big\{(\alpha_1, \ldots, \alpha_k) : \alpha_i \in [n], i < j \to \alpha_i < \alpha_j\big\} \qquad (11.1)$$

For $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k)$ and $\vec{\beta} = (\beta_1, \ldots, \beta_k)$ there is a directed edge from $\vec{\alpha}$ to $\vec{\beta}$ iff

$$\forall i \in \{1, \ldots, k-1\} : \beta_i = \alpha_{i+1}. \qquad (11.2)$$

Lemma 11.6. *Viewed as an undirected graph, the graph $\mathcal{B}_{2r+1,n}$ is a subgraph of the r-neighborhood graph of directed n-node rings with node labels from $[n]$.*

Proof. The claim follows directly from the observations regarding r-hop views of nodes in a directed ring from Section 11.1. The set of k-tuples of increasing node labels is a subset of the set of k-tuples of distinct node labels. Two nodes of $\mathcal{B}_{2r+1,n}$ are connected by a directed edge iff the two corresponding r-hop views are connected by a directed edge in the neighborhood graph. Note that if there is an edge between $\vec{\alpha}$ and $\vec{\beta}$ in $\mathcal{B}_{k,n}$, $\alpha_1 \neq \beta_k$ because the node labels in $\vec{\alpha}$ and $\vec{\beta}$ are increasing.   □

To determine a lower bound on the number of colors an r-round algorithm needs for directed n-node rings, it therefore suffices to determine a lower bound on the chromatic number of $\mathcal{B}_{2r+1,n}$. To obtain such a lower bound, we need the following definition.

Definition 11.7 (Diline Graph)**.** *The directed line graph (diline graph) $\mathcal{DL}(G)$ of a directed graph $G = (V, E)$ is defined as follows. The node set of $\mathcal{DL}(G)$ is $V[\mathcal{DL}(G)] = E$. There is a directed edge $\big((w, x), (y, z)\big)$ between $(w, x) \in E$ and $(y, z) \in E$ iff $x = y$, i.e., if the first edge ends where the second one starts.*

Lemma 11.8. *If $n > k$, the graph $\mathcal{B}_{k+1,n}$ can be defined recursively as follows:*

$$\mathcal{B}_{k+1,n} = \mathcal{DL}(\mathcal{B}_{k,n}).$$

Proof. The edges of $\mathcal{B}_{k,n}$ are pairs of k-tuples $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k)$ and $\vec{\beta} = (\beta_1, \ldots, \beta_k)$ that satisfy Conditions (11.1) and (11.2). Because the last $k - 1$ labels in $\vec{\alpha}$ are equal to the first $k - 1$ labels in $\vec{\beta}$, the pair $(\vec{\alpha}, \vec{\beta})$ can be represented by a $(k + 1)$-tuple $\vec{\gamma} = (\gamma_1, \ldots, \gamma_{k+1})$ with $\gamma_1 = \alpha_1$, $\gamma_i = \beta_{i-1} = \alpha_i$ for $2 \leq i \leq k$, and $\gamma_{k+1} = \beta_k$. Because the labels in $\vec{\alpha}$ and the labels in $\vec{\beta}$ are increasing, the labels in $\vec{\gamma}$ are increasing as well. The two graphs $\mathcal{B}_{k+1,n}$ and $\mathcal{DL}(\mathcal{B}_{k,n})$ therefore have the same node sets. There is an edge between two nodes $(\vec{\alpha}_1, \vec{\beta}_1)$ and $(\vec{\alpha}_2, \vec{\beta}_2)$ of $\mathcal{DL}(\mathcal{B}_{k,n})$ if $\vec{\beta}_1 = \vec{\alpha}_2$. This is equivalent to requiring that the two corresponding $(k + 1)$-tuples $\vec{\gamma}_1$ and $\vec{\gamma}_2$ are neighbors in $\mathcal{B}_{k+1,n}$, i.e., that the last k labels of $\vec{\gamma}_1$ are equal to the first k labels of $\vec{\gamma}_2$. $\square$

The following lemma establishes a useful connection between the chromatic numbers of a directed graph G and its diline graph $\mathcal{DL}(G)$.

Lemma 11.9. *For the chromatic numbers $\chi(G)$ and $\chi(\mathcal{DL}(G))$ of a directed graph G and its diline graph, it holds that*

$$\chi\big(\mathcal{DL}(G)\big) \geq \log_2\big(\chi(G)\big).$$

Proof. Given a c-coloring of $\mathcal{DL}(G)$, we show how to construct a 2^c coloring of G. The claim of the lemma then follows because this implies that $\chi(G) \leq 2^{\chi(\mathcal{DL}(G))}$.

Assume that we are given a c-coloring of $\mathcal{DL}(G)$. A c-coloring of the diline graph $\mathcal{DL}(G)$ can be seen as a coloring of the edges of G such that no two adjacent edges have the same color. For a node v of G, let S_v be the set of colors of its outgoing edges. Let u and v be two edges such that G contains a directed edge (u, v) from u to v and let x be the color of (u, v). Clearly, $x \in S_u$ because (u, v) is an outgoing edge of u. Because adjacent edges have different colors, no outgoing edge (v, w) of v can have color x. Therefore $x \notin S_v$. This implies that $S_u \neq S_v$. We can therefore use these color sets to obtain a vertex coloring of G, i.e., the color of u is S_u and the color of v is S_v. Because the number of possible subsets of $[c]$ is 2^c, this yields a 2^c-coloring of G. $\square$

Let $\log^{(i)} x$ be the i-fold application of the base-2 logarithm to x:

$$\log^{(1)} x = \log_2 x, \quad \log^{(i+1)} x = \log_2(\log^{(i)} x).$$

Remember from Chapter 1 that

$$\log^* x = 1 \text{ if } x \leq 2, \quad \log^* x = \min\{i : \log^{(i)} x \leq 2\}.$$

For the chromatic number of $\mathcal{B}_{k,n}$, we obtain

Lemma 11.10. *For all $n \geq 1$, $\chi(\mathcal{B}_{1,n}) = n$. Further, for $n \geq k \geq 2$, $\chi(\mathcal{B}_{k,n}) \geq \log^{(k-1)} n$.*

Proof. For $k = 1$, $\mathcal{B}_{k,n}$ is the complete graph on n nodes with a directed edge from node i to node j iff $i < j$. Therefore, $\chi(\mathcal{B}_{1,n}) = n$. For $k > 2$, the claim follows by induction and Lemmas 11.8 and 11.9.   □

This finally allows us to state a lower bound on the number of rounds needed to color a directed ring with 3 colors.

Theorem 11.11. *Every deterministic, distributed algorithm to color a directed ring with* 3 *or less colors needs at least* $\log^* n/2 - 1$ *rounds.*

Proof. Using the connection between $\mathcal{B}_{k,n}$ and the neighborhood graph for directed rings, it suffices to show that $\chi(\mathcal{B}_{2r+1,n}) > 3$ for all $r > \log^* n/2 - 1$. From Lemma 11.10, we know that $\chi(\mathcal{B}_{2r+1,n}) \geq \log^{(2r)} n$. To obtain $\log^{(2r)} n \leq 2$, we need $r \geq \log^* n/2$. Because $\log_2 3 < 2$, we therefore have $\log^{(2r)} n \leq 3$ if $r \geq \log^* n/2 - 1$.   □

Corollary 11.12. *Every deterministic, distributed algorithm to compute an MIS of a directed ring needs at least* $\log^* n/2 - O(1)$ *rounds.*

Remarks:

- It is straight-forward to see that also for a constant $c > 3$, the number or rounds needed to color a ring with c or less colors is $\log^* n/2 - O(1)$.

- There basically (up to additive constants) is a gap of a factor of 2 between the $\log^* n + O(1)$ upper bound of Chapter 1 and the $\log^* n/2 - O(1)$ lower bound of this chapter. It is possible to show that the lower bound is tight, even for undirected rings (for directed rings, this will be part of the exercises).

- The presented lower bound is due to Nathan Linial. The lower bound is also true for randomized algorithms. The generalization for randomized algorithms was done by Moni Naor.

- The neighborhood graph concept can be used more generally to study distributed graph coloring. It can for instance be used to show that with a single round (every node sends its identifier to all neighbors) it is possible to color a graph with $(1 + o(1))\Delta^2 \ln n$ colors and that every one-round algorithm needs at least $\Omega(\Delta^2/\log^2 \Delta + \log \log n)$ colors.

- Using r-hop views and the fact that nodes with equal r-hop views have to make the same decisions is the basic principle behind almost all locality lower bounds (in fact, we are not aware of a locality a lower bound that does not use this principle). Using this basic technique (but a completely different proof otherwise), it is for instance possible to show that computing an MIS in a general graph requires at least $\Omega(\sqrt{\log n/\log \log n})$ rounds.