Network Computing and Efficient Algorithms

Locality Lower Bounds

Xiang-Yang Li and Xiaohua Xu

School of Computer Science and Technology
University of Science and Technology of China (USTC)

September 1, 2021

Locality Lower Bounds

Locality Lower Bounds

$$\boxed{\begin{array}{l} \text{Minimization} \rightarrow \text{Lower bounds} \rightarrow \Omega(f(n)) \\ \text{Maximization} \rightarrow \text{Upper bounds} \rightarrow O(f(n)) \end{array}}$$

Locality Lower Bounds

$$\boxed{\begin{array}{l} \text{Minimization} \rightarrow \text{Lower bounds} \rightarrow \Omega(f(n)) \\ \text{Maximization} \rightarrow \text{Upper bounds} \rightarrow O(f(n)) \end{array}}$$

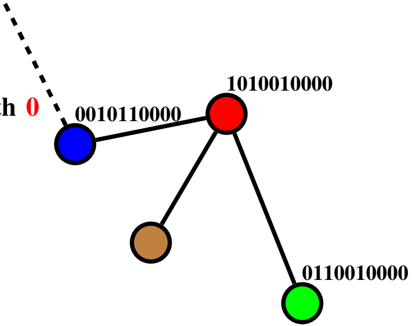
Each node execute the same code;
Different only in terms of neighborhoods.

Locality Lower Bounds

Minimization $\rightarrow$ Lower bounds $\rightarrow$ $\Omega(f(n))$
Maximization $\rightarrow$ Upper bounds $\rightarrow$ $O(f(n))$

Recall: Tree Coloring

We count the bit positions from right to left, starting with 0

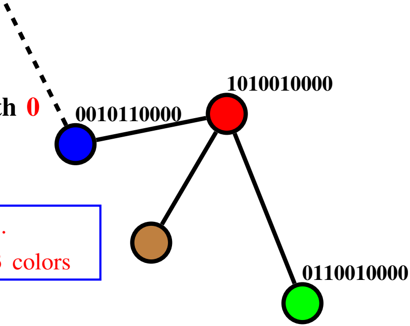


Round 1

Grand-parent	0010110000	
Parent	1010010000	$\rightarrow$ 01010
Child	0110010000	$\rightarrow$ 10001

Recall: Tree Coloring

We count the bit positions from right to left, starting with 0



$\log^*(n)$ time ,down to 6 colors …

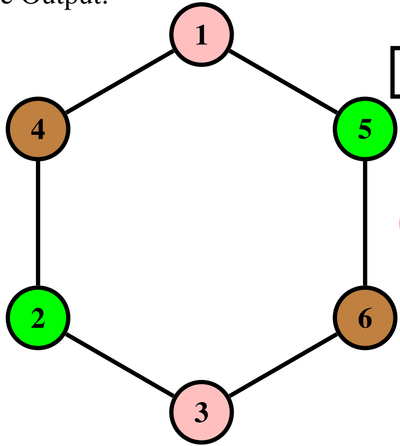
…and then shift-down: down to 3 colors

Round 1

Grand-parent	0010110000		
Parent	1010010000	$\rightarrow$	01010
Child	0110010000	$\rightarrow$	10001

Ring Coloring

Possible Output:



$\log^*(n)$ time for 3 colors.

Can we do even faster?

Algorithm for trees can be adapted!

Problem

- **Lower bound of distributed coloring problem:**
  - Coloring rings (and rooted trees) with 3 or less colors indeed requires $\Omega(\log^* n)$ rounds.
  - How to prove?

Problem

- **Lower bound of distributed coloring problem:**
  - Coloring rings (and rooted trees) with 3 or less colors indeed requires $\Omega(\log^* n)$ rounds.
  - How to prove?
- **Assumptions:**
  - Deterministic, synchronous algorithms.
  - Message size and local computations are unbounded.
  - Network is a directed ring with n nodes.
  - Nodes have unique labels (identifiers) from 1 to n.

Problem

- **Lower bound of distributed coloring problem:**
  - Coloring rings (and rooted trees) with 3 or less colors indeed requires $\Omega(\log^* n)$ rounds.
  - How to prove?
- **Assumptions:**
  - Deterministic, synchronous algorithms.
  - Message size and local computations are unbounded.
  - Network is a directed ring with n nodes.
  - Nodes have unique labels (identifiers) from 1 to n.
- **All the conditions above make a lower bound stronger.**

Canonical Form for Synchronous Alg.

What can a distributed algorithm do or learn in r rounds?

1. Initially, all nodes only know their own ID
2. As information needs at least r rounds to travel r hops, a node can only learn about r-loop neighborhood!

Canonical Form for Synchronous Alg.

What can a distributed algorithm do or learn in r rounds?

1. Initially, all nodes only know their own ID
2. As information needs at least r rounds to travel r hops, a node can only learn about r-loop neighborhood!

Lemma 8.2

Any deterministic synchronous r-round algorithm can be transformed into Canonical Form:

ALGORITHM 8.1 SYNCHRONOUS ALGORITHM: CANONICAL FORM()

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

Canonical Form for Synchronous Alg.

What can a distributed algorithm do or learn in r rounds?

1. Initially, all nodes only know their own ID
2. As information needs at least r rounds to travel r hops, a node can only learn about r-loop neighborhood!

Lemma 8.2

Any deterministic synchronous r-round algorithm can be transformed into Canonical Form:

ALGORITHM 8.1 SYNCHRONOUS ALGORITHM: CANONICAL FORM()

1: In r rounds: **send** complete initial stat to nodes at distance at most r

2: ▷ do all the communication first

3: Compute output based on the complete information about r-neighborhood

4: ▷ do all the computation in the end

In other words: we can emulate any local algorithm by making all communication first and then do all local computations! Why?

Canonical Form for Synchronous Alg.

What can a distributed algorithm do or learn in r rounds?

1. Initially, all nodes only know their own ID
2. As information needs at least r rounds to travel r hops, a node can only learn about r-loop neighborhood!

Lemma 8.2

Any deterministic synchronous r-round algorithm can be transformed into Canonical Form:

ALGORITHM 8.1 SYNCHRONOUS ALGORITHM: CANONICAL FORM()

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

In other words: we can emulate any local algorithm by making all communication first and then do all local computations! Why?

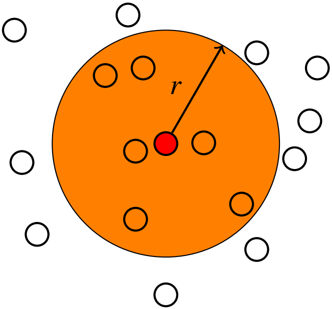
Example "leader election":
Whether nodes only forward highest ID so far or whether all information is collected first and later selected does not make a difference!

We can do all communication forst and then do all local computations!

How to prove this?

Let A be any r-round algorithm.
We can show that the canonical forn algorithm C can compute all possible messages that A may send as well.
By induction over distance of nodes
…if we can compute messages of first i rounds in $(r-i+1)$-neighborhood, we have all information to compute first $(i+1)$ round message in $(r-i)$-neighborhood.

So first trivial: Can compute all first messages in r-neighborhood



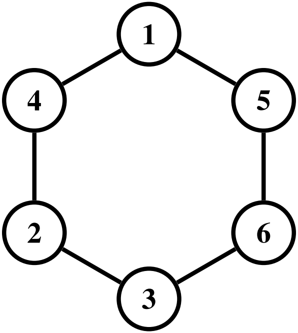
r-hop view

Definition 8.3 (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.

How do r-hop views of our rings look like?

E.g.,1-hop view of 4?



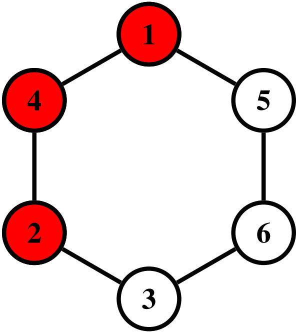
r-hop view

Definition 8.3(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.

How do r-hop views of our rings look like?

E.g.,1-hop view of 4?



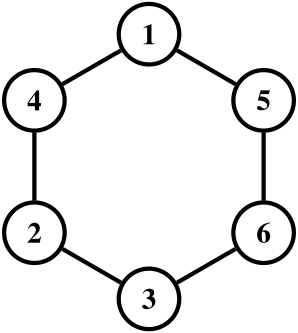
r-hop view

Definition 8.3(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.

How do r-hop views of our rings look like?

E.g.,2-hop view of 4?



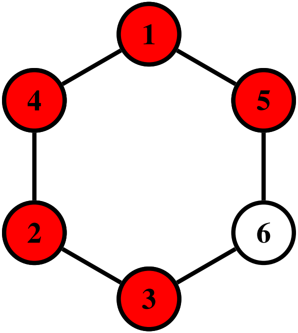
r-hop view

Definition 8.3(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.

How do r-hop views of our rings look like?

E.g.,2-hop view of 4?



Definition 8.3(r-hop view).

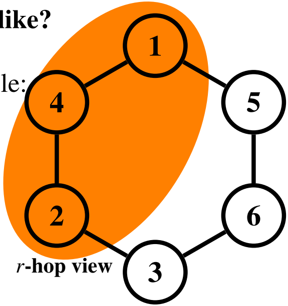
We call the collection of the initial states of all nodes in the r-neighborhood of a node v the r-hop view of v.

How do r-hop views for our rings look like?
Generally:

The r-hop view of a ring is a $(2r+1)$ tuple:

$$(l_{-r}, l_{-r+1}, \ldots, l_0, \ldots, l_r)$$

where l_0 is ID/label of considered node v.



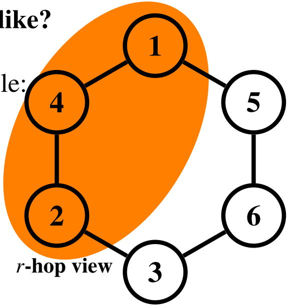
Definition 8.3(r-hop view).

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.

How do r-hop views for our rings look like?
Generally:
The r-hop view of a ring is a $(2r+1)$ tuple:

$$(l_{-r}, l_{-r+1}, \ldots, l_0, \ldots, l_r)$$

where l_0 is ID/label of considered node v.



Corollary 8.4.

A deterministic r-round algorithm A is a function that maps every possible r-hop view to the set of possible outputs.

Ring Coloring

Corollary 8.4.

A deterministic r-round algorithm A is a function that maps every possible r-hop view to the set of possible outputs.

Ring Coloring

Corollary 8.4.

A deterministic r-round algorithm A is a function that maps every possible r-hop view to the set of possible outputs.

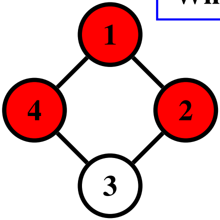
When is a coloring valid?

Ring Coloring

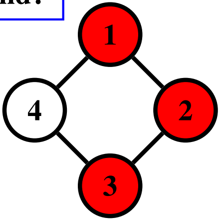
Corollary 8.4.

A deterministic r-round algorithm A is a function that maps every possible r-hop view to the set of possible outputs.

When is a coloring valid?



1-hop view of 1



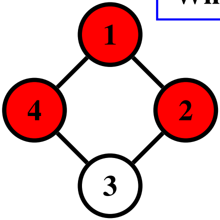
1-hop view of 2

Ring Coloring

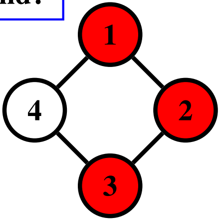
Corollary 8.4.

A deterministic r-round algorithm A is a function that maps every possible r-hop view to the set of possible outputs.

When is a coloring valid?



1-hop view of 1



1-hop view of 2

(4,1,2) and (1,2,3) are 1-hop view of two adjacent nodes. So what?

Corollary 8.4.

A deterministic r-round algorithm A is a function that maps every possible r-hop view to the set of possible outputs.

When is a coloring valid?

Consider two r-hop views:

$$(l_{-r}, l_{-r+1}, \ldots, l_0, \ldots, l_r)$$

$$(l'_{-r}, l'_{-r+1}, \ldots, l'_0, \ldots, l'_r)$$

where $l'_i = l_{i+1}$, for $-r \leq i \leq r-1$ and $l'_i \neq l_{i+1}$ for $-r \leq i \leq r$, so what?
Then the two views can originate from the adjacent nodes in the ring!

Corollary 8.4.

A deterministic r-round algorithm A is a function that maps every possible r-hop view to the set of possible outputs.

When is a coloring valid?

Consider two r-hop views:

$$(l_{-r}, l_{-r+1}, \ldots, l_0, \ldots, l_r)$$

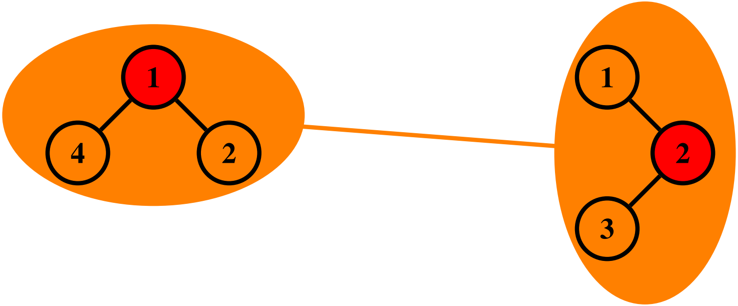
$$(l'_{-r}, l'_{-r+1}, \ldots, l'_0, \ldots, l'_r)$$

where $l'_i = l_{i+1}$, for $-r \leq i \leq r-1$ and $l'_i \neq l_{i+1}$ for $-r \leq i \leq r$, so what? Then the two views can originate from the adjacent nodes in the ring!

So every algorithm needs to assign different colors to the two views!

Neighborhood Graph

- **Nodes: any possible neighborhoods**
- **Edges: conflicting neighborhoods are connected (when?)**
- **Coloring:**
  - The same neighborhoods have the same color.
  - Conflicting ones with different colors



Neighborhood Graph

- **Nodes: any possible neighborhoods**
- **Edges: conflicting neighborhoods are connected (when?)**
- **Coloring:**
  - The same neighborhoods have the same color.
  - Conflicting ones with different colors

Definition 8.5 (Neighborhood Graph).

For a given **famliy of network graphs** G, the r-neighborhood graph $N_r(G)$ is defined as follows. The node set of $N_r(G)$ 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 V_r and V'_r if V_r and V'_r can be the r-hop views of two adjacent nodes.

Lemma 8.6.

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

Lemma 8.6.

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

Road Map

Lemma 8.6.

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

How to find a good lower bound with this lemma?

We have to show that $\chi(N_r(G))$ is small only for a larger r ...

Road Map

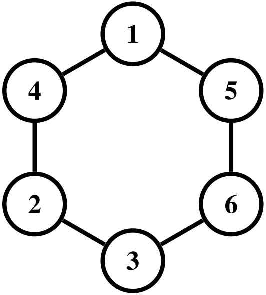
Lemma 8.6.

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

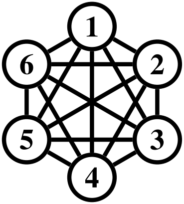
How to find a good lower bound with this lemma?

We have to show that $\chi(N_r(G))$ is small only for a larger r . . .

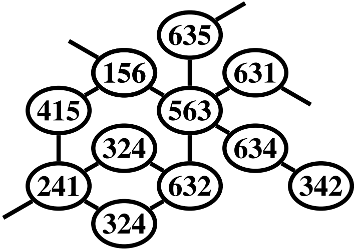
So how does $\chi(N_r(G))$ of a ring look like?
For example for our ring graph?



r-hop neighborhood graph for ring family (n=6 known)



$$\chi(N_0(G)) = ?$$



$$\chi(N_1(G)) = ?$$

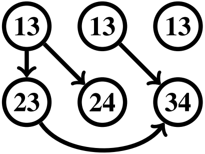
The Neighborhood Graph

Instead of directly defining the neighborhood graph for directed rings, we define directed graphs B_k that are closely related to the neighborhood graph. The node set of B_k **contains all k-tuples of increasing node labels** $([n] = \{1, \ldots, n\})$:

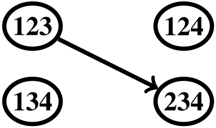
$$V[B_k] = \{(\alpha_1, \ldots, \alpha_k) : \alpha_i \in [n], i < j \rightarrow \alpha_i < \alpha_j\}$$

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

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



$B_2\ (n = 4)$



$B_3\ (n = 4)$

The Neighborhood Graph

Instead of directly defining the neighborhood graph for directed rings, we define directed graphs B_k that are closely related to the neighborhood graph. The node set of B_k **contains all k-tuples of increasing node labels** ($[n] = \{1, \ldots, n\}$):

$$V[B_k] = \{(\alpha_1, \ldots, \alpha_k) : \alpha_i \in [n], i < j \to \alpha_i < \alpha_j\}$$

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

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

Lemma 8.7.

Viewed as an undirected graph, the graph B_{2r+1} is a **subgraph** of the r-neighborhood graph of directed n-node tings with node lables from $[n]$.

The Neighborhood Graph

Instead of directly defining the neighborhood graph for directed rings, we define directed graphs B_k that are closely related to the neighborhood graph. The node set of B_k **contains all k-tuples of increasing node labels** ($[n] = \{1, \ldots, n\}$):

$$V[B_k] = \{(\alpha_1, \ldots, \alpha_k) : \alpha_i \in [n], i < j \to \alpha_i < \alpha_j\}$$

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

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

Lemma 8.7.

Viewed as an undirected graph, the graph B_{2r+1} is a **subgraph** of the r-neighborhood graph of directed n-node tings with node lables from $[n]$.

Coloring a subgraph is not harder!

Diline Graph

Definition 8.8 (Diline Graph).

The directed line graph (diline graph) $DL(G)$ of a directed graph $G = (V, E)$ is defined as follows. The node set of $DL(G)$ is $V[DL(G)] = E$. There is a directed edge $((w,x),(y,z))$ 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.

Diline Graph

Definition 8.8 (Diline Graph).

The directed line graph (diline graph) $DL(G)$ of a directed graph $G = (V, E)$ is defined as follows. The node set of $DL(G)$ is $V[DL(G)] = E$. There is a directed edge $((w,x),(y,z))$ 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 8.9.

if $n > k$, the graph B_{k+1} can be defined recursively as follows:

$$B_{k+1} = DL(B_k).$$

Definition 8.8 (Diline Graph).

The directed line graph (diline graph) $DL(G)$ of a directed graph $G=(V,E)$ is defined as follows. The node set of $DL(G)$ is $V[DL(G)]=E$. There is a directed edge $((w,x),(y,z))$ 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 8.9.

if $n>k$, the graph B_{k+1} can be defined recursively as follows:

$$B_{k+1}=DL(B_k).$$

Proof. The edges of B_k are pairs of k-tuples $\underline{\alpha}=(\alpha_1,\ldots,\alpha_k)$ and $\underline{\beta}=(\beta_1,\ldots,\beta_k)$ that satisfy Conditions (8.1) and (8.2). Because the last $k-1$ labels in $\underline{\alpha}$ are equal to the first $k-1$ labels in $\underline{\beta}$, **the pair $(\underline{\alpha},\underline{\beta})$ can be represented by a (k+1)-tuple $\underline{\gamma}=(\gamma_1,\ldots,\gamma_{k+1})$** with $\gamma_1=\alpha_1$, $\gamma_i=\beta_{i-1}=\alpha_i$ for $2\le i\le k$, and $\gamma_{k+1}=\beta_k$. Because the labels in $\underline{\alpha}$ and the labels in $\underline{\beta}$ are increasing, the labels in $\underline{\gamma}$ are increasing as well. The two graphs B_{k+1} and $DL(B_k)$ therefor habe the same node sets. There is an edge between two nodes $(\underline{\alpha}_1,\underline{\beta}_1)$ and $(\underline{\alpha}_2,\underline{\beta}_2)$ of $DL(B_k)$ if $\underline{\beta}_1=\underline{\alpha}_2$. This is equivalent to requiring that the two corresponding $(k+1)$-tuples $\underline{\gamma}_1$ and $\underline{\gamma}_2$ are neighbors in B_{k+1},*i.e.*, that the last k labels of $\underline{\gamma}_1$ are equal to the first k labels of $\underline{\gamma}_2$.

Lemma 8.10.

For the chromatic numbers $\chi(G)$ and $\chi(DL(G))$ of a directed grapg G and its diline graph, it holds that

$$\chi(DL(G)) \geq \log_2(\chi(G)).$$

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

Assume that we are given a c-coloring of $DL(G)$. A c-coloring of the diline graph $DL(G)$ 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 nodes such that G contains a directed edge (u,v) from u to v and let c 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.

Theorem 8.12

Lemma 8.11.

For all $n \geq 1$, $\chi(B_1) = n$. Further, for $n \geq k \geq 2$, $\chi(B_k) \geq \log^{(k-1)} n$.

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

Theorem 8.12

Lemma 8.11.

For all $n \geq 1$, $\chi(B_1) = n$. Further, for $n \geq k \geq 2$, $\chi(B_k) \geq \log^{(k-1)} n$.

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

Proof. For $k = 1$, B_k is the complete graph on n nodes with a directed edge from node i to node j *iff* $i < j$. Therefore, $\chi(B_1) = n$.
For $k > 2$, the claim follows by induction and Lemmas 8.9 and 8.10.

Theorem 8.12

Lemma 8.11.

For all $n \geq 1$, $\chi(B_1) = n$. Further, for $n \geq k \geq 2$, $\chi(B_k) \geq \log^{(k-1)} n$.

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

Proof. For $k = 1$, B_k is the complete graph on n nodes with a directed edge from node i to node j *iff* $i < j$. Therefore, $\chi(B_1) = n$.
For $k > 2$, the claim follows by induction and Lemmas 8.9 and 8.10.

Theorem 8.12.

Every determinstic, distributed algorithms to color a directed ring with 3 or less colors needs at least $(\log^* n)/2 - 1$ rounds.

Lemma 8.11.

For all $n \geq 1$, $\chi(B_1) = n$. Further, for $n \geq k \geq 2$, $\chi(B_k) \geq \log^{(k-1)} n$.

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

Proof. For $k = 1$, B_k is the complete graph on n nodes with a directed edge from node i to node j *iff* $i < j$. Therefore, $\chi(B_1) = n$.
For $k > 2$, the claim follows by induction and Lemmas 8.9 and 8.10.

Theorem 8.12.

Every determinstic, distributed algorithms to color a directed ring with 3 or less colors needs at least $(\log^* n)/2 - 1$ rounds.

We need to show that $\chi(B_{2r+1,n}) > 3$ for all $r < (\log^* n)/2 - 1$.
We know that $\chi(B_{2r+1,n}) \geq \log^{(2r)} n$.
And $B_{2r+1,n}$ is subgraph of neighborhood graph we actually want!
The rest is simple maths...

Remarks

- 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 $(1+o(1))\Delta^2$ in n colors, and that every one-round algorithm needs at least $\Omega(\Delta^2/\log^2\Delta + \log\log n)$ colors.
- One may also extend the proof to other problems, for instance one may show that a constant approximation of the minimum dominating set problem on unit disk graphs costs at least log-star time.

References

- An alternative proof that omits the neighborhood graph construction:
  - Juhana Laurinharju and Jukka Suomela. Brief Announcement: Linials Lower Bound Made Easy. In PODC, 2014.
- The lower bound is also true for randomized algorithms:
  - Moni Naor. A Lower Bound on Probabilistic Algorithms for Distributive Ring Coloring. SIAM J. Discrete Math., 1991.
- More substantial lower bounds for a number of combinatorial problems:
  - Reuven Bar-Yehuda, Keren Censor-Hillel, and Gregory Schwartzman. A distributed $(2+\varepsilon)$-approximation for vertex cover in $o(\log\delta/\varepsilon \log\log\delta)$ rounds. In arXiv, 2016.
  - F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot Be Computed Locally! In PODC, 2004.
- This lower bound technique was adapted to other problems:
  - A. Czygrinow, M. Hanckowiak, and W. Wawrzyniak. Fast Distributed Approximations in Planar Graphs. In DISC, 2008.
  - Christoph Lenzen and Roger Wattenhofer. Leveraging Linials Locality Limit. In DISC, 2008.
- Surveys:
  - Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. Local Computation: Lower and Upper Bounds. In JACM, 2016.
  - Jukka Suomela. Survey of Local Algorithms. http://www.cs.helsinki.fi/local-survey/, 2012.