Chapter 7
Distributed Coloring & MIS I

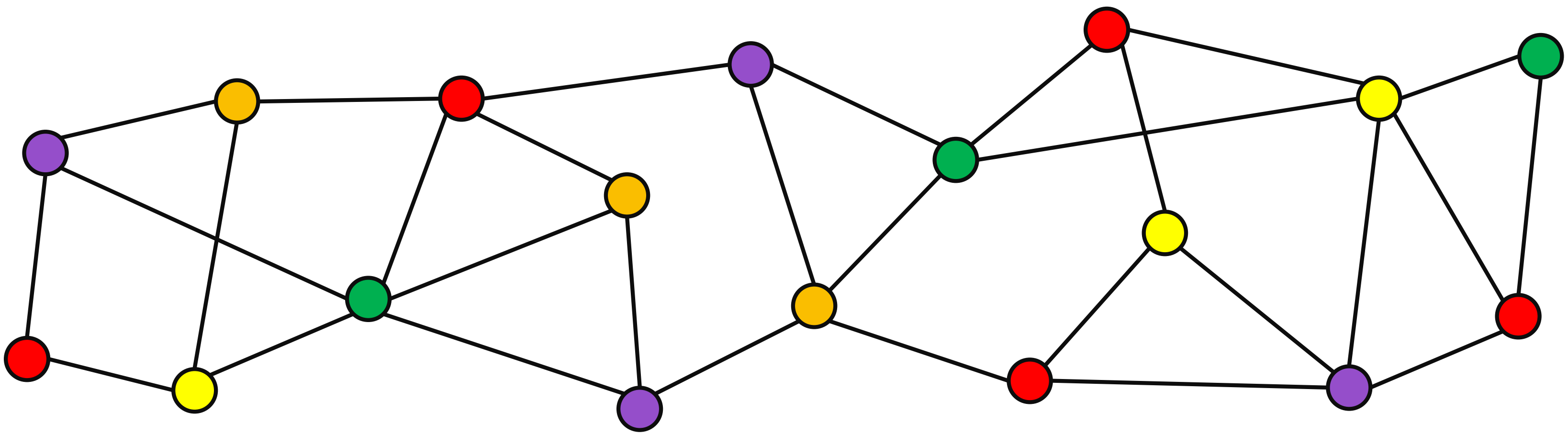
Distributed Systems

Summer Term 2020

Fabian Kuhn

Graph Coloring

Vertex Coloring

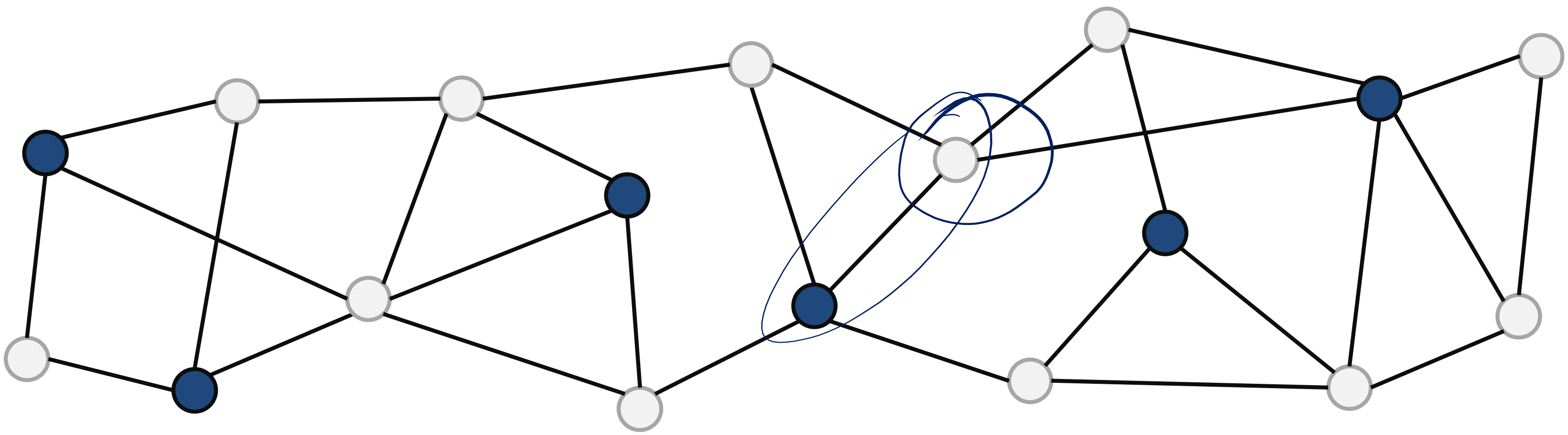


Objective: Assign a color to each node such that:

- If nodes u and v are neighbors, they get different colors.
- The total number of different colors is as small as possible.

Maximal Independent Set

Maximal Independent Set (MIS)

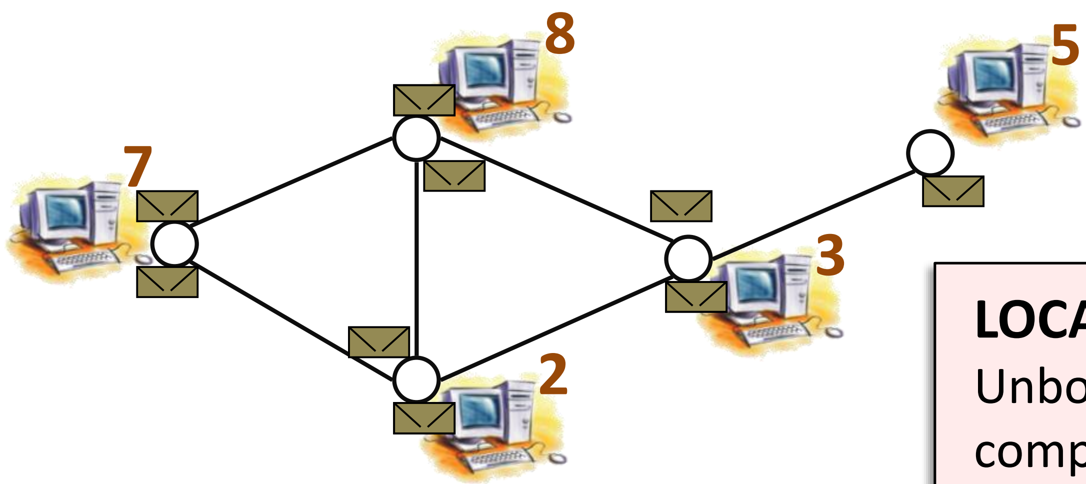


Objective: compute a maximal independent set (MIS)

- **Independent Set:** set of pairwise non-adjacent nodes
- **Maximal:** adding any additional node destroys independence (non-extendible set of pair-wise non-adjacent nodes)

Distributed Graph Algorithms

Network is modeled as a graph



Graph properties
- n nodes
- unique IDs

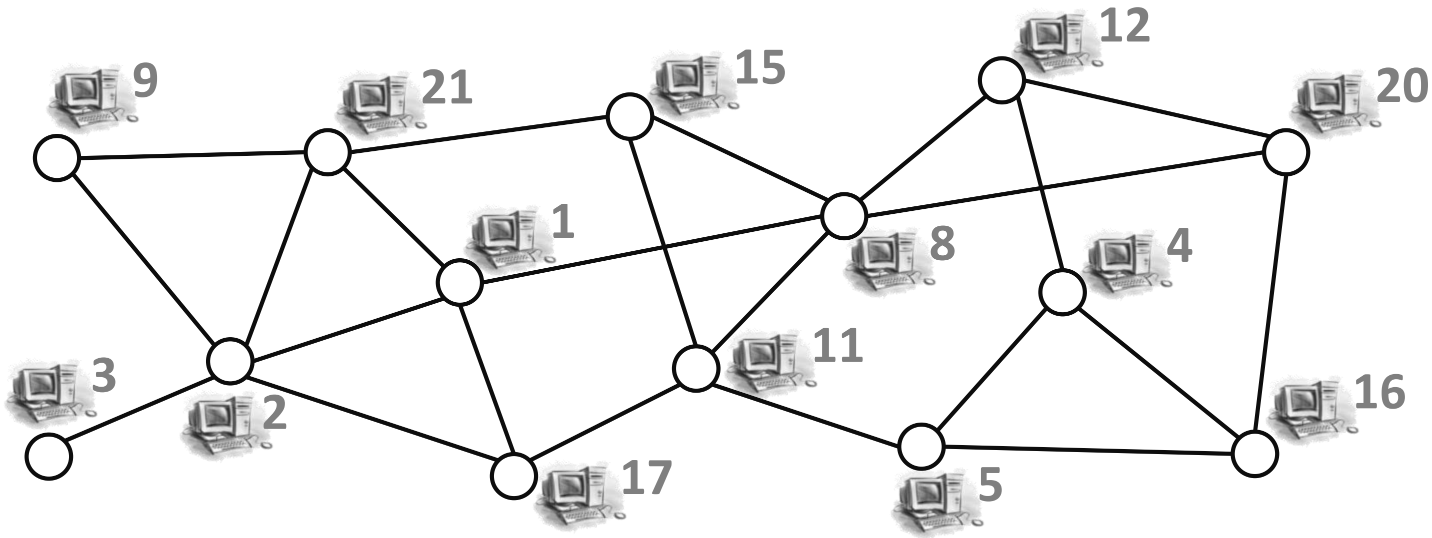
LOCAL Model [Linial; FOCS '87]
Unbounded internal computation & message size

Synchronous rounds
1. Each node/computer does some internal computation
2. Send a message to each neighbor
3. Receive message from each neighbor

time complexity = number of rounds

Distributed Graph Algorithms

Objective: solve some graph problem on the network graph



At the start: Each node knows its own ID and nothing else about the topology

At the end: Each node knows its part of the output
- In our case, its color or if it belongs to the MIS

Applications of Coloring and MIS in Networks

Wireless Networks

- If we have different communication channels (frequencies, time slots, etc.), we might want to assign a channel to each node.
- If we need to avoid conflicts, we essentially have to solve coloring.

- MIS can be used to compute some basic clustering in wireless networks
  - An MIS allows to select non-adjacent centers, such that every node is adjacent to at least one of the centers.

Generally

- Coloring and MIS are important "symmetry breaking" problems.
- They appear as subroutines in other algorithms.
- Techniques developed for MIS/coloring might be interesting for solving other problems.

Sequential Greedy Algorithms

Let's start with MIS:

```
S := Ø
for all v ∈ V do          // go through nodes v in an arbitrary order
  if v has no neighbor in S, add v to S
```

- At the end S clearly is an independent set
- Each node $u \notin S$ has a neighbor $v \in S$ (i.e., S is a maximal indep. set)

Greedy vertex coloring (use colors $1, 2, 3, \dots$):

```
all nodes uncolored
for all v ∈ V do          // go through nodes v in an arbitrary order
  v gets smallest color not used by a neighbor of v
```

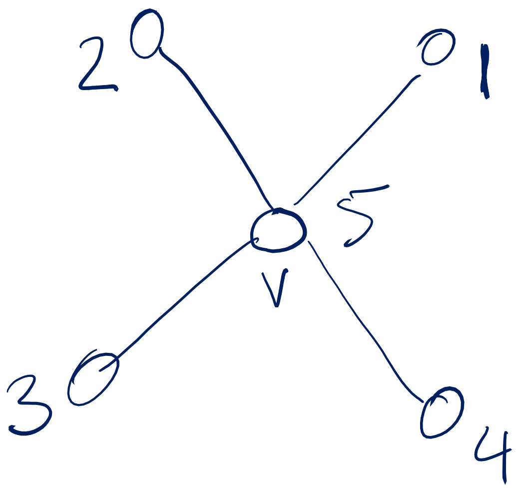
- Clearly computes a valid (a.k.a. proper) coloring
- What is the number of colors?

Greedy Vertex Coloring

Greedy Algorithm: Go through the nodes in an arbitrary order and always assign the smallest available color in $\{1, 2, 3, \dots\}$

How many colors do we need?

colors 1, 2, ...



v always gets one of the first 5 colors

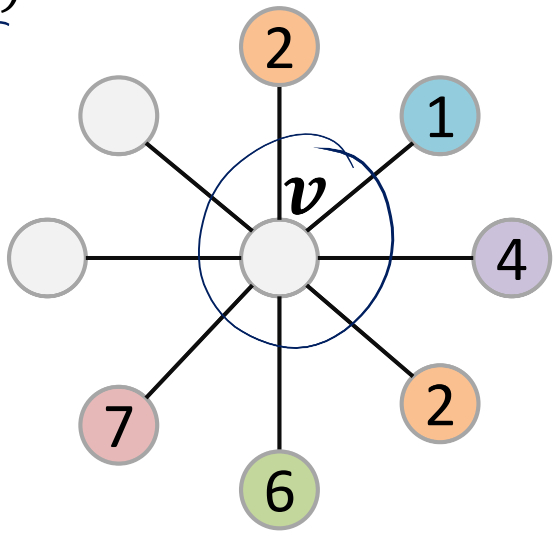
Greedy Vertex Coloring

Greedy Algorithm: Go through the nodes in an arbitrary order and always assign the smallest available color in $\{1, 2, 3, \dots\}$

Assumption: Graph $G = (V, E)$, Δ : largest node degree

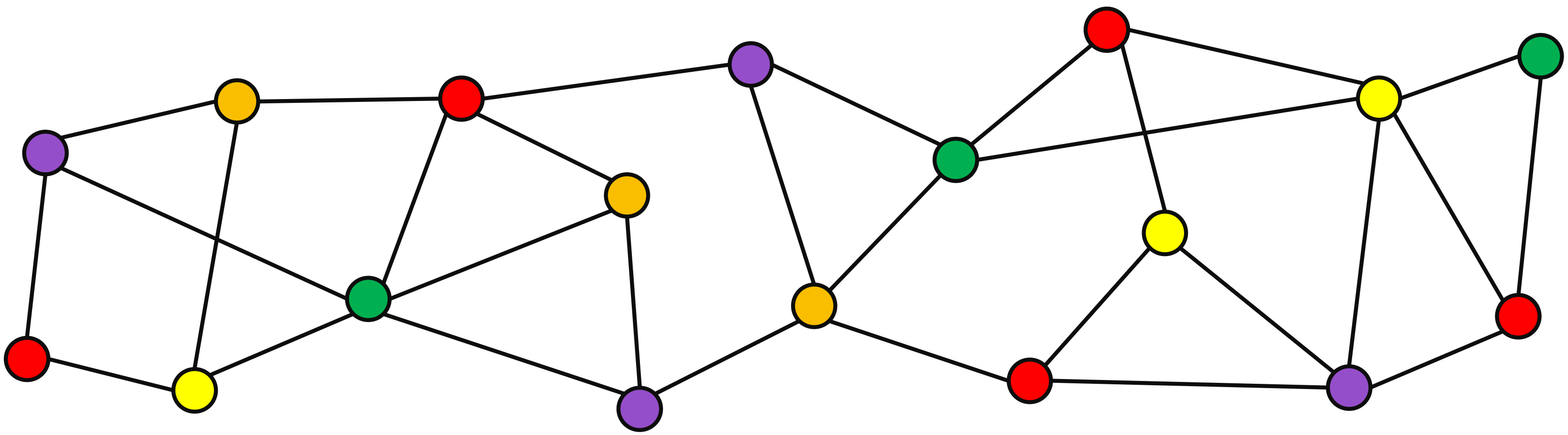
Theorem: The greedy vertex coloring algorithm requires $\leq \Delta + 1$ colors.

- Consider an arbitrary node v of degree $\deg(v)$
- When v gets colors, its neighbors already have $\leq \deg(v)$ different colors.
- Therefore, one of the first $\deg(v) + 1$ colors is still free for v.
- $\text{color}(v) \leq \deg(v) + 1 \leq \Delta + 1$



Distributed Coloring Problem

$(\boldsymbol{\Delta + 1})$-Vertex Coloring

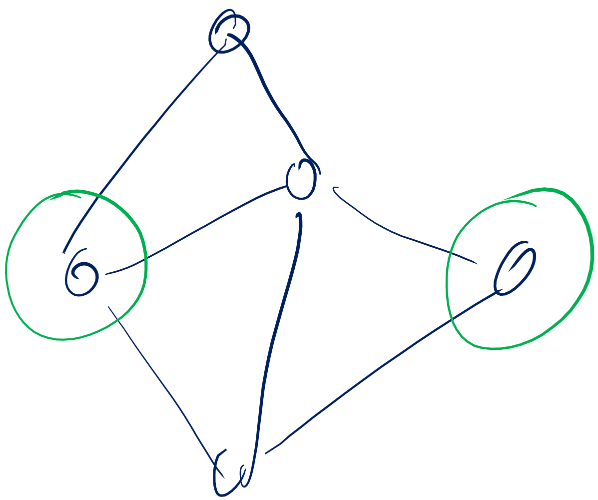


Objective: properly color the nodes with $\leq \Delta + 1$ colors

- Δ : maximum degree
- $\Delta + 1$ colors: what a simple sequential greedy algorithm achieves

Distributed Coloring Algorithm?

- How can we color in a distributed way?
- Each node picks smallest available color
  - available = color not picked by any neighbor
  - But how can we avoid conflicts between neighbors?
  - Neighbors should not choose a color at the same time.



Distributed Greedy Coloring Algorithm

Distributed Greedy Vertex Coloring for node v

1. wait until all neighbors of v with smaller IDs have a color
2. v chooses smallest available color
3. v informs its neighbors

- No two neighbors choose a color at the same time
  $\Longrightarrow$ algorithm computes a correct coloring with $\leq \Delta + 1$ colors.
- Computes the same coloring as the greedy algorithm when going through the nodes in order defined by IDs

- The same algorithm also works for MIS:

Distributed Greedy MIS Algorithm

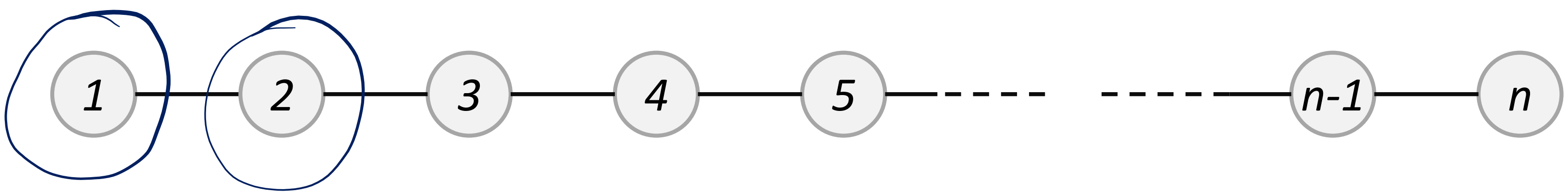
with smaller ID

1. wait until all neighbors of v are decided
2. v joins MIS if no neighbor of v is already in MIS
3. v informs its neighbors

Distributed Greedy : Time Complexity

Theorem: The distributed greedy algorithms for $(\Delta + 1)$-coloring and MIS terminate after at most $O(n)$ rounds.

- In each round, at least one new node is processed
  - unprocessed node with smallest ID
- $O(n)$ rounds is very slow, but unfortunately it is tight



- Can we be faster?
  - How can we make sure to color / process many nodes in parallel?
- **First:** we can be faster if we are already given some coloring
  - Say, we are given a proper coloring with C colors.

From C-Coloring to $(\Delta + 1)$-Coloring & MIS

Assumption:

- We are given a proper C-coloring of the nodes
  - a proper coloring with colors $1, 2, \dots, C$

In both algorithm, we can replace IDs by these colors:

Algorithm runs in phases $1, 2, \dots, C$

In phase i:

- Nodes with initial color i are processed
  - For coloring, pick smallest available color
  - For MIS, join MIS iff no neighbor is already in the MIS
- At the end of phase, newly processed nodes inform neighbors

- Algorithm works because nodes processed in parallel are non-adjacent
- **Time complexity of algorithm: C rounds**

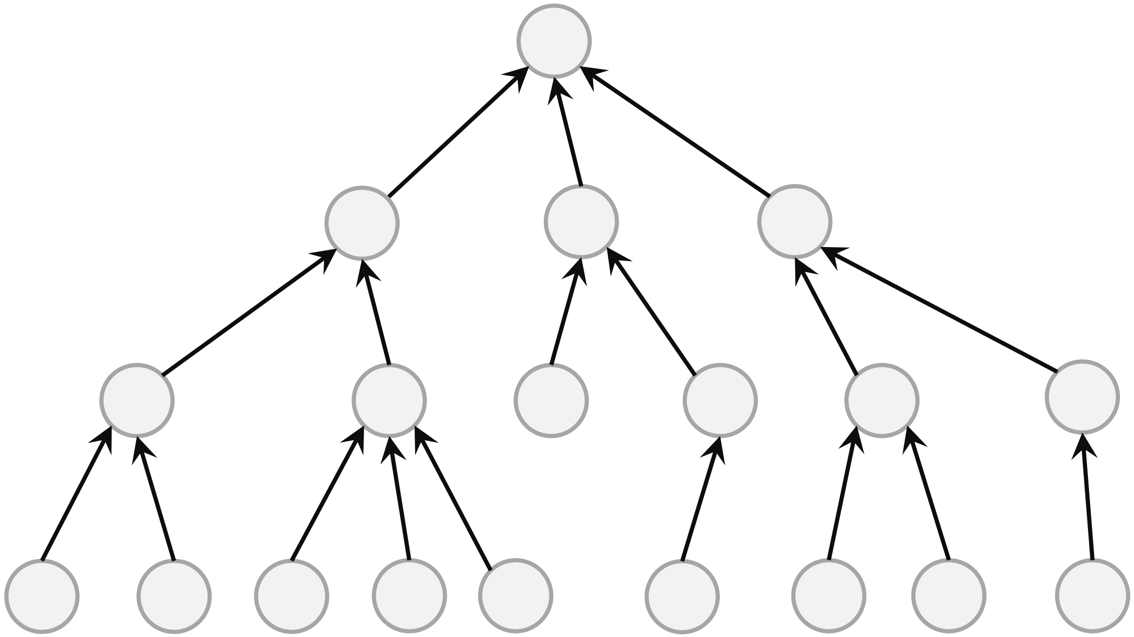
- Can we do better? What if we don't have a coloring to start?

Coloring Special Graph Classes

- It's not clear how to easily improve this
  - Let's therefore first look at special classes of graphs

Rooted Trees

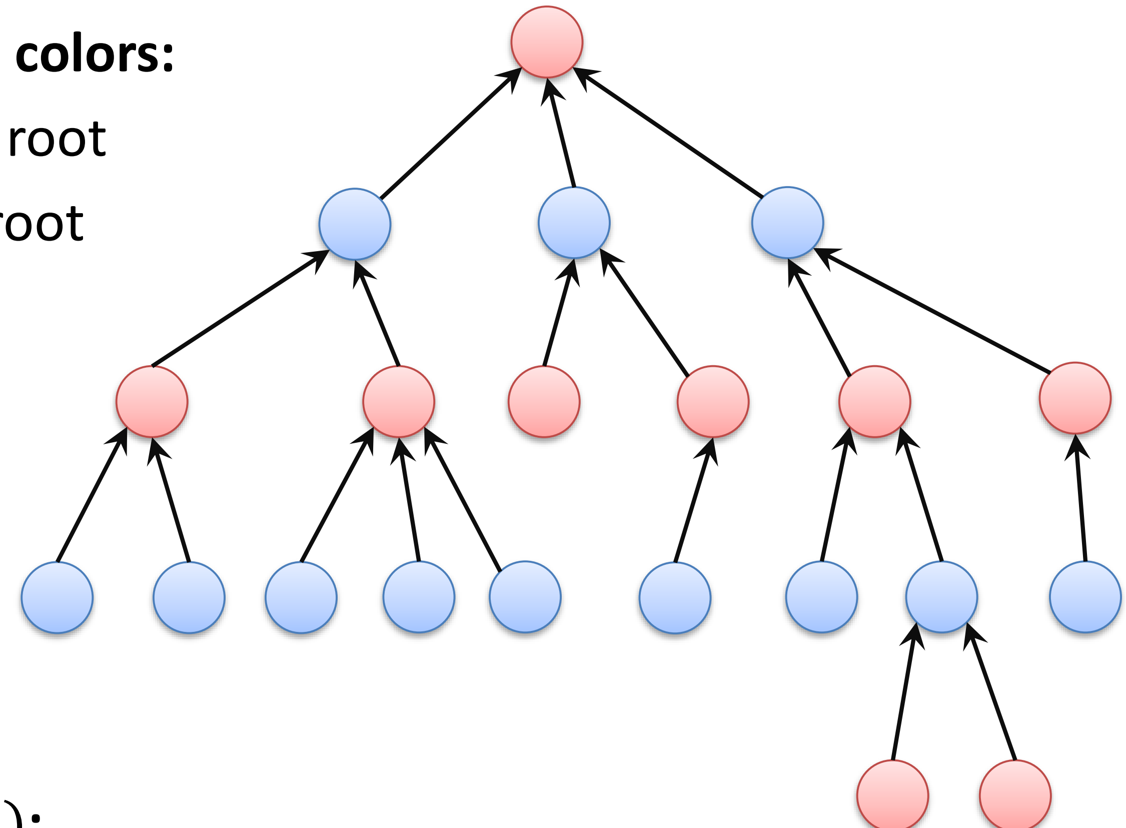
- Graph is a tree, each node knows which neighbor is its parent
  - and the root knows it is the root



Coloring Rooted Trees

Trees can be colored with 2 colors:

- Color 1: even distance to root
- Color 2: odd distance to root

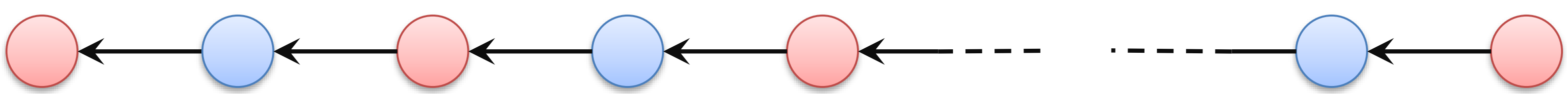


Distributed Algorithm:

- Color level by level, starting at the root

Time complexity: $\boldsymbol{O(D)}$

This is tight and can be $\boldsymbol{\Theta(n)}$:



- Nodes need to know parity of their distance to the root
  - You will see a formal argument in a later lecture.

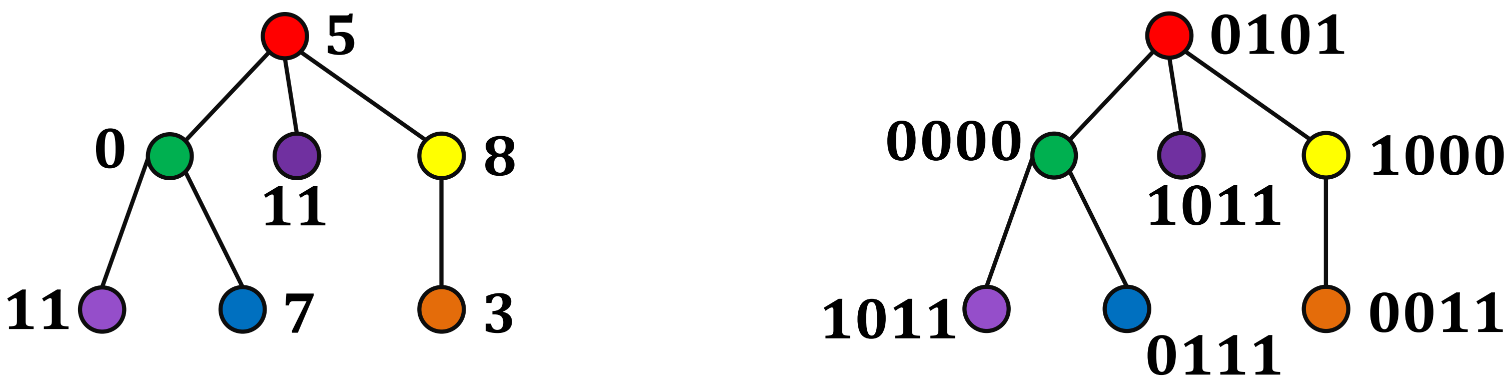
Coloring Rooted Trees with More Colors

Color Reduction:

- Assume, we are given a proper coloring with C colors
  - Initially, if we have unique IDs from an ID space of size N, we have $C = N$
- Can we reduce the number of colors?
  - And what happens if we reduce them iteratively?

Specific Assumptions:

- Initial coloring with colors $\in \{0, \dots, C-1\}$ for some $C \in \mathbb{N}$
- Interpret color as bit string of length $\lceil \log_2 C \rceil$
- Example (for $C = 12$)



Cole-Vishkin Color Reduction Scheme

Fast color reduction by using the bit representation:

- Consider node u and its parent v (x_u and x_v are initial colors of u and v)
  - The root node just imagines a parent with a different color
- Define

Least significant bit is bit 0

$$i_u := \{\text{first bit, where } x_u \text{ and } x_v \text{ differ}\}$$

in binary representation

- New color

bit at position i_u in color x_u

$$x'_u := i_u \circ x_u[i_u]$$

conc.

v

u

Example:

11 10 9 8 7 6 5 4 3 2 1 0

x: 1 0 1 0 1 1 0 0 1 1 1 0 1 0 1 0

w: 1 0 1 1 0 0 1 1 1 0 1 1 1 0 1 0

v: 1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0

u: 0 0 1 1 0 0 1 1 1 0 1 1 1 0 1 0

$i_w = 4 \to x'_w = \underbrace{100}_{i_w} : 1$

$i_v = 11 \to x'_v = 1011 : 1$

$x'_u = 1011 : 0$

Cole-Vishkin Color Reduction Scheme

- Define

Least significant bit is bit 0

$$i_u := \{\text{first bit, where } x_u \text{ and } x_v \text{ differ}\}$$

- New color

in binary representation

bit at position i_u in color x_u

$$x'_u := i_u \circ x_u[i_u]$$

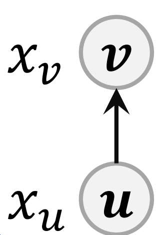
Theorem: For any two neighbors, if $x_u \neq x_v$, then we also have $x'_u \neq x'_v$.

Proof:

- We have $x'_u = i_u \circ x_u[i_u]$ and $x'_v = i_v \circ x_v[i_v]$.
- We have $x'_u \neq x'_v$ if and only if $i_u \neq i_v$ or $x_u[i_u] \neq x_v[i_v]$
- W.l.o.g., assume that v is the parent of u

either $i_u \neq i_v$ ✓

or $i_u = i_v \rightarrow x_u[i_u] \neq x_v[i_v]$



$$i_u := \{\text{first bit, where } x_u \text{ and } x_v \text{ differ}\} \Longrightarrow x_u[i_u] \neq x_v[i_u]$$

Cole-Vishkin Color Reduction Scheme

How much do we reduce the colors?

- Each node u has an initial color $x_u \in \{0, \dots, C-1\}$.
- Initial color can therefore be written as a $\lceil \log_2 C \rceil$-bit number
- Therefore:

$$i_u \in \{0, \dots, \lceil \log_2 C \rceil - 1\}$$

- An thus:

$$x'_u = i_u \circ x_u[i_u] \leq 2 \cdot i_u + 1 \leq 2\lceil \log_2 C \rceil - 1$$

Theorem: In one color reduction step, the number of colors is reduced from C to at most $2\lceil \log_2 C \rceil$.

Theorem: If the color reduction step is applied iteratively, the alg. eventually computes a coloring with the six colors $\{0, \dots, 5\}$.

Proof: $C > 2\lceil \log_2 C \rceil$ for all $C > 6$.

Rooted Tree Coloring : Time Complexity

The Log-Star Function

- For a real number $x > 1$ and an integer $i \geq 1$, we define

$$\log_2^{(i)} n \coloneqq \log_2\left(\log_2^{(i-1)} n\right), \qquad \log_2^{(1)} n \coloneqq \log_2 n$$

- For an integer $n \geq 2$, the function $\log^* n$ is defined as

$$\boldsymbol{\log^* n \coloneqq \min\left\{i : \log_2^{(i)} n \leq 1\right\}}$$

$\log^* n$: # of times one has to apply the $\log_2$ function to get a number ≤ 1

$\log^* 2 = 1$, $\log^* 4 = 2$, $\log^* 16 = 3$, $\log^* 2^{16} = 4$

$\log^* 2^{2^{16}} = 5$

huge number " > " # atoms in universe

Rooted Tree Coloring : Time Complexity

The Log-Star Function

- For a real number $x > 1$ and an integer $i \geq 1$, we define

$$\log_2^{(i)} n := \log_2\left(\log_2^{(i-1)} n\right), \qquad \log_2^{(1)} n := \log_2 n$$

- For an integer $n \geq 2$, the function $\log^* n$ is defined as

$$\boldsymbol{\log^* n := \min\left\{i : \log_2^{(i)} n \leq 1\right\}}$$

Theorem: When starting with colors in $\{0, \dots, N-1\}$, the Cole-Vishkin color reduction algorithm computes a **6-coloring of a rooted tree** in $\boldsymbol{O(\log^* n)}$ **rounds.**

Proof Sketch: Color are reduced as follows:

$$\underline{N} \Longrightarrow \underbrace{O(\log N)}_{2\lceil \log_2 N \rceil} \Longrightarrow O(\log\log N) \Longrightarrow O(\log\log\log N) \Longrightarrow \cdots$$

From Six to Three Colors

Coloring Rooted Trees

- We have seen that computing a 2-coloring requires $\Omega(D)$ time
- We have seen how to compute a 6-coloring in $O(\log^* n)$ rounds.
- What about 3, 4, or 5 colors?

6 colors : colors 0, ..., 5

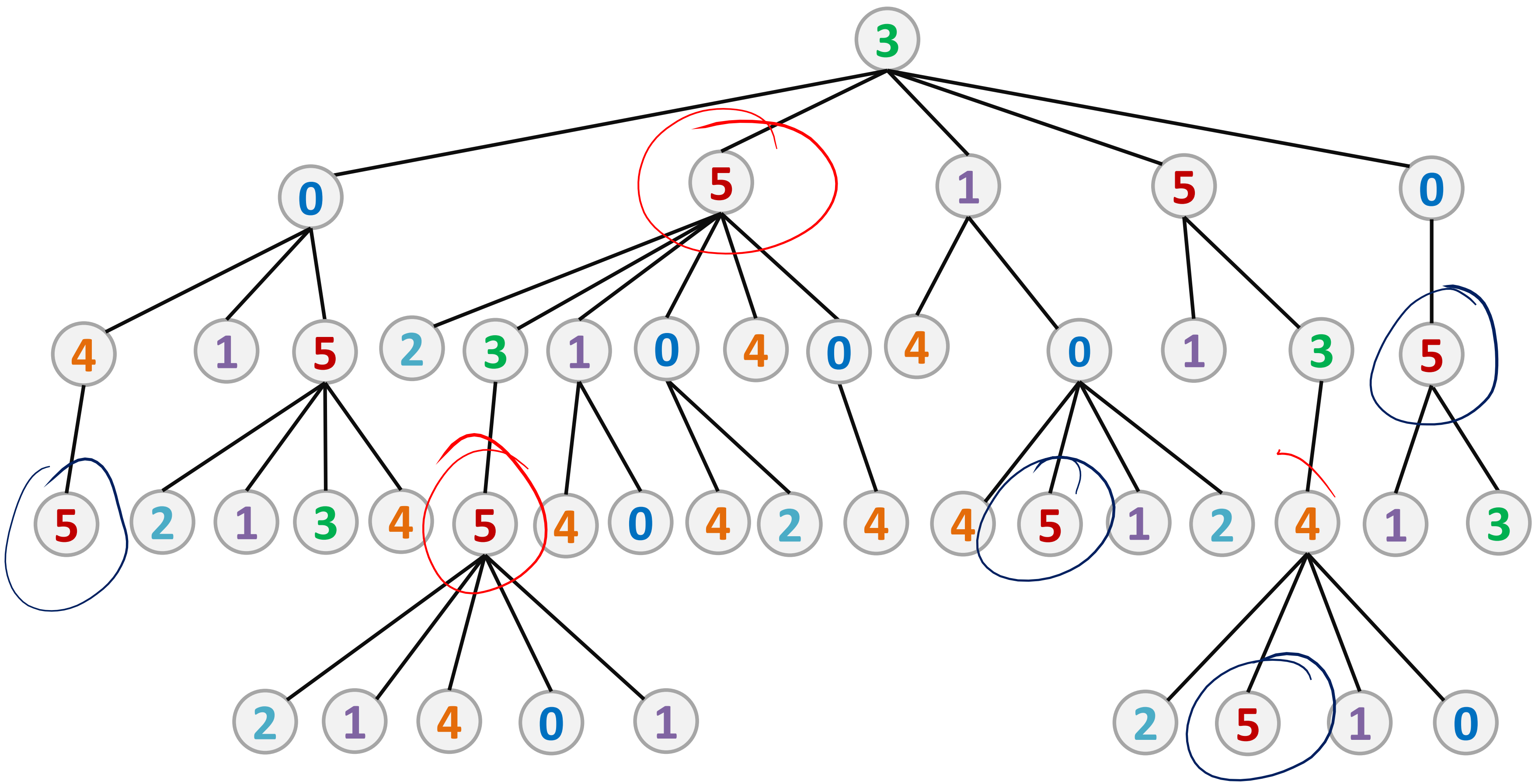
Reducing to 5 colors?

- Can we recolor all the nodes with color 5 to a smaller color?
- We could do for all those nodes in parallel in one round if $\Delta \leq 4$
  - Then one of the colors $0, \dots, 4$ is free for every node with color 5

- What can we do if $\Delta > 4$?

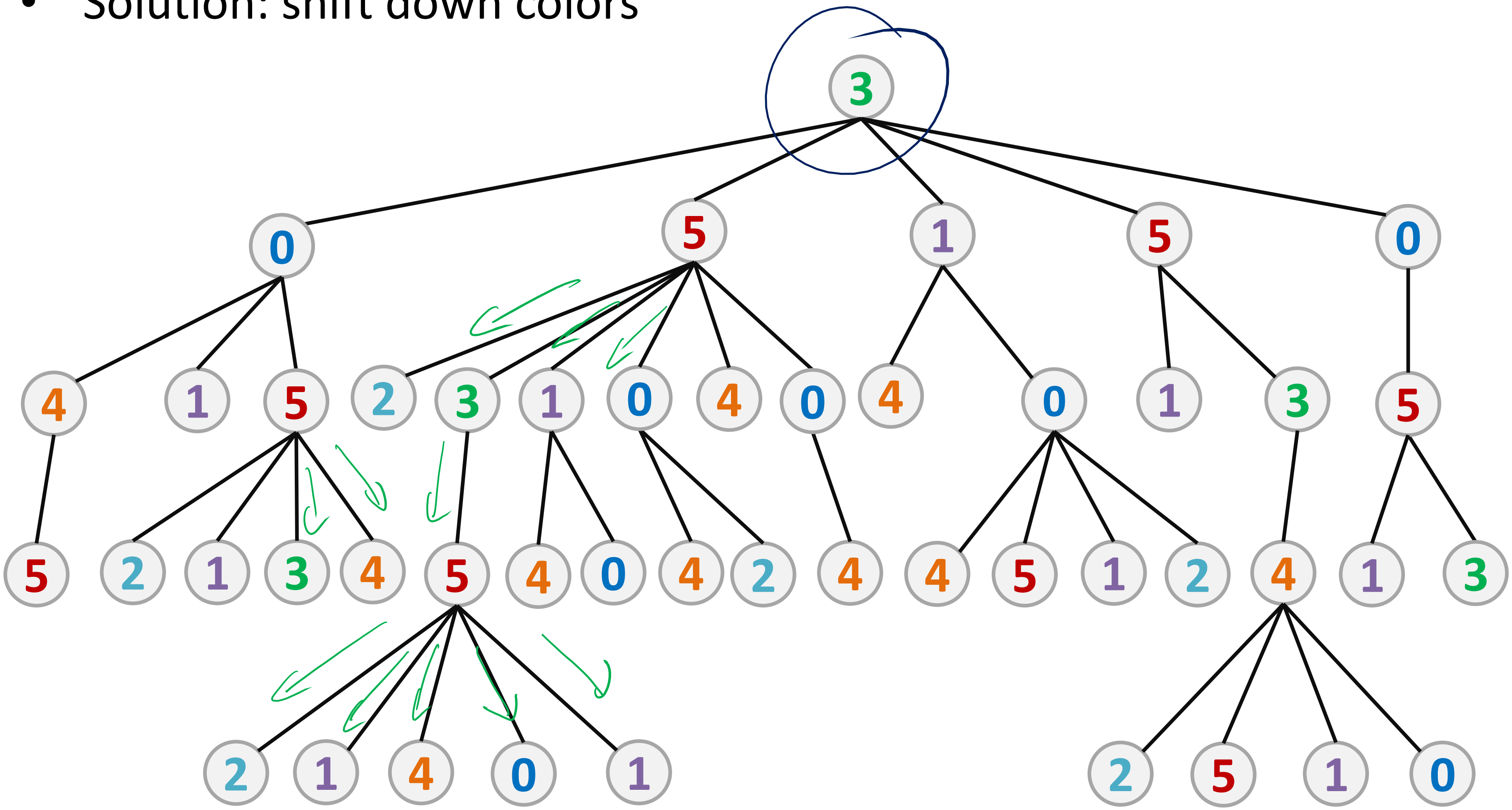
From Six to Five Colors

- Consider a rooted tree that is colored with colors $0, \ldots, 5$
- Can we get rid of color 5?



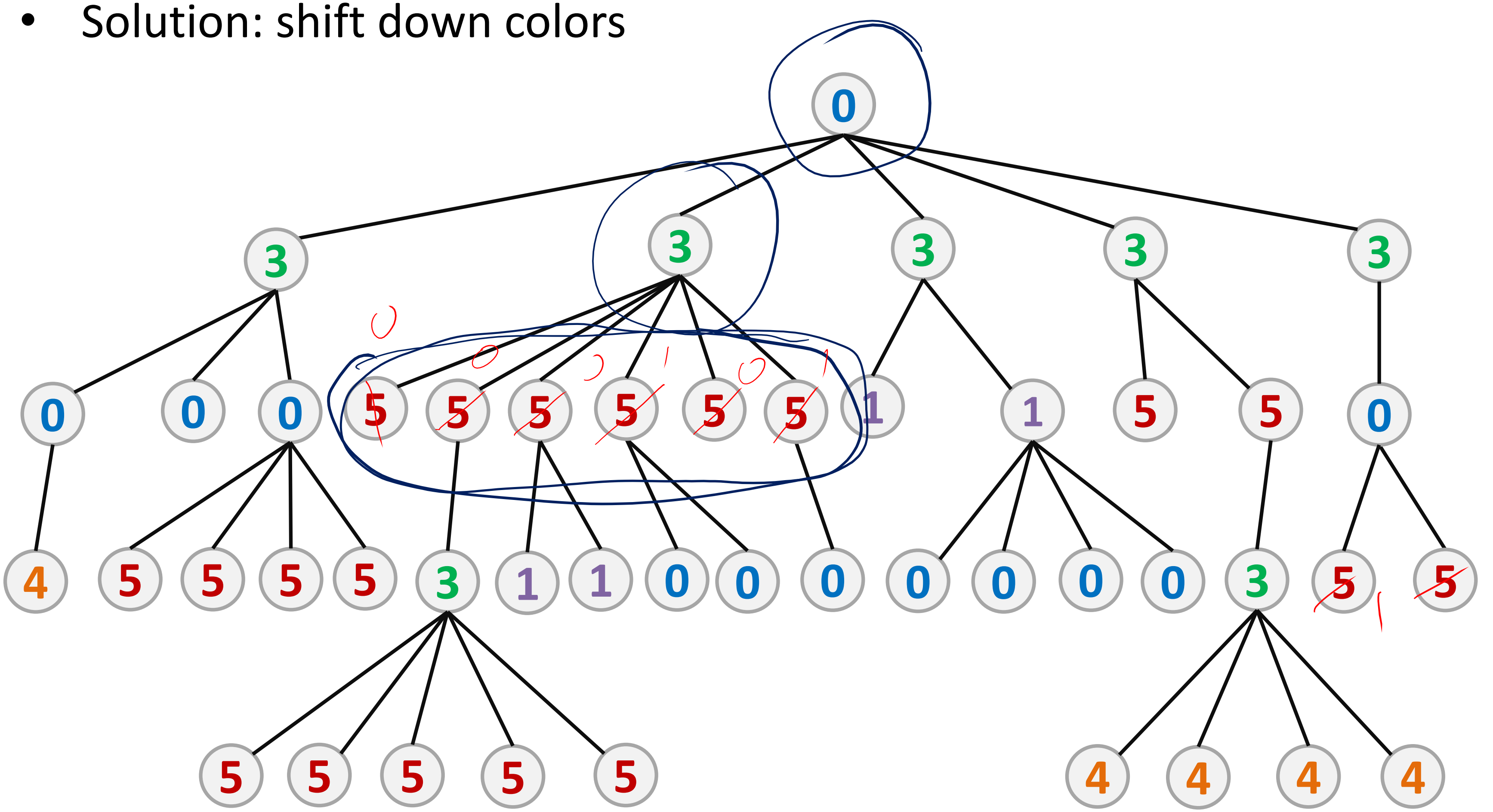
From Six to Five Colors

- Can we get rid of color 5?
- Solution: shift down colors



From Six to Five Colors

- Can we get rid of color 5?
- Solution: shift down colors



From Six to Three Colors

Color Reduction Phase For Rooted Trees

- Assume that we are given a coloring with colors $0, \ldots, C$ for $C > 2$
- Goal: compute a coloring with colors $0, \ldots, C-1$

1. **Shift-down step**
   - The root chooses a different color.
     (if the root was colored 0, it chooses color 1, otherwise it chooses color 0)
   - Every other node takes the color of its parent.
   - After this step, for every node v, all children of v have the same color.
2. **Color reduction step**
   - Each node of color C now picks the smallest color not chosen by a neighbor.
   - Each such node picks a color $\in \{0, 1, 2\}$ because after the shift-down step, the neighbors of each node are colored with only 2 different colors.

Theorem: As long as the number of colors is larger than 3 ($C > 2$), we can reduce the number of colors by 1 in 2 rounds.

Rooted Trees : Coloring and MIS

- Combining the Cole-Vishkin algorithm (to get 6 colors) and the color reduction algorithm, we get a fast 3-coloring algorithm

Theorem: When starting with colors in $\{0, \ldots, N-1\}$, there is a distributed algorithm to computes a **3-coloring of a rooted tree** in $\boldsymbol{O(\log^* N)}$ **rounds**.

- Unique IDs in $\{0, \ldots, N-1\}$ can be used as an initial coloring.

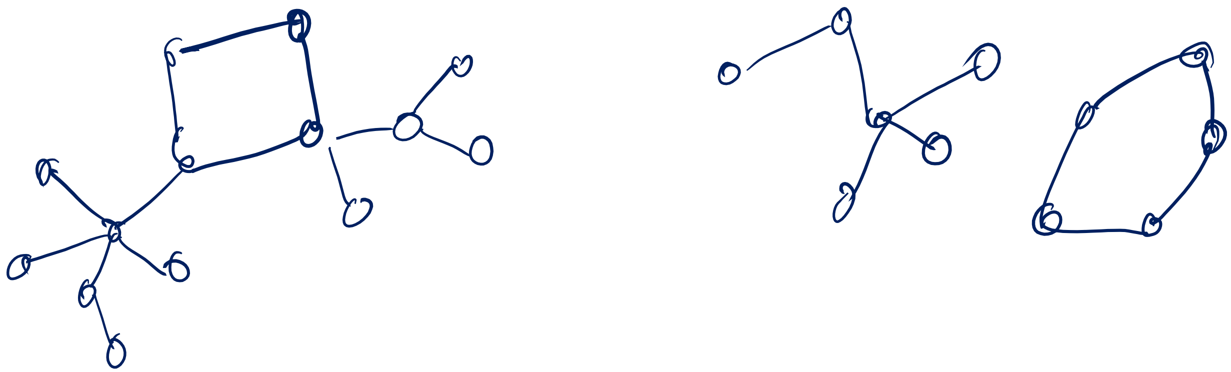
Theorem: When starting with colors in $\{0, \ldots, N-1\}$, there is a distributed algorithm to computes an **MIS** in $\boldsymbol{O(\log^* N)}$ **rounds**.

- One first computes a 6-coloring (or a 3-coloring)
- Then, an MIS can be computed in $O(1)$ rounds
  - We have seen before that from a C-coloring we get an MIS in C rounds.

Coloring Directed Pseudoforests

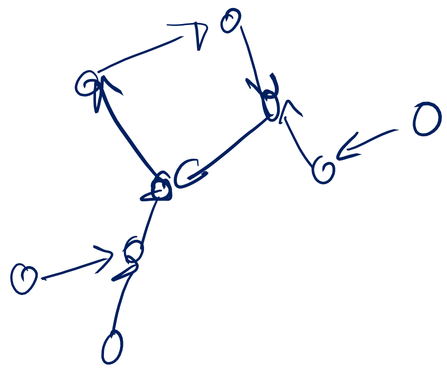
Pseudoforest

- A graph in which each node has at most one cycle



Directed Pseudoforest

- A directed graph, where the out-degree of every node is at most 1



Coloring Directed Pseudoforests

Directed Pseudoforest

- A directed graph, where the out-degree of every node is at most 1

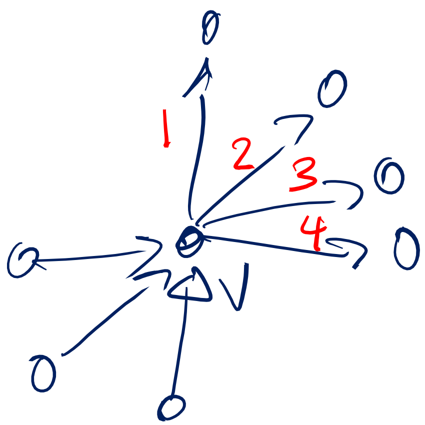
Claim: The 3-coloring algorithm for rooted trees can also be applied in a directed pseudoforest.

- The Cole-Vishkin algorithm works as before
  - Nodes with out-degree 1 treat their out-neighbor as parent
  - Other nodes behave like the root and imagine an out-neighbor with some color
- The color reduction algorithm also works in the same way
  - "Shift-down": Every node with out-degree 1 picks color of out-neighbor, every other node just picks a new color (either 0 or 1)
  - All in-neighbors of a node then have the same color and each node therefore only sees 2 different colors among its neighbors

Coloring Graphs with Maximum Degree Δ

1. We first orient each edge of the graph arbitrarily.
   - An edge $\{u, v\}$ can for example be oriented from u to v iff $\mathrm{ID}(u) < \mathrm{ID}(v)$.

2. Assume that a node v has d_v out-going edges.
   Node v labels these edges from 1 to d_v

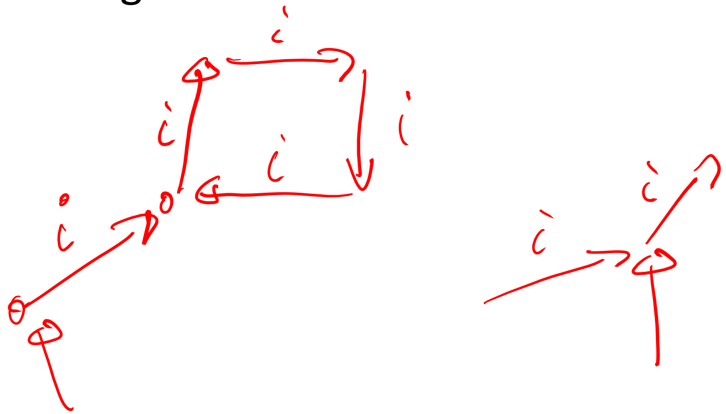


Coloring Graphs with Maximum Degree Δ

1. We first orient each edge of the graph arbitrarily.
   - An edge $\{u, v\}$ can for example be oriented from u to v iff $\mathrm{ID}(u) < \mathrm{ID}(v)$.

2. Assume that a node v has d_v out-going edges.
   Node v labels these edges from 1 to d_v

3. Every edge now has a label between 1 and Δ and every node has at most one out-going edge for each label.
   - For each label $i \in \{1, \dots, \Delta\}$, in the subgraph G_i induced by label i therefore every node has out-degree at most 1 $\Longrightarrow$ each label induces a directed pseudoforest



Coloring Graphs with Maximum Degree Δ

1. We first orient each edge of the graph arbitrarily.
   - An edge $\{u, v\}$ can for example be oriented from u to v iff $\mathrm{ID}(u) < \mathrm{ID}(v)$.
2. Assume that a node v has d_v out-going edges. Node v labels these edges from 1 to d_v ← $d_v \leq \Delta$
3. Every edge now has a label between 1 and Δ and every node has at most one out-going edge for each label.
   - For each label $i \in \{1, \dots, \Delta\}$, in the subgraph G_i induced by label i therefore every node has out-degree at most $1 \Longrightarrow$ each label induces a directed pseudoforest
4. For all $i \in \{1, \dots, \Delta\}$, compute a 3-coloring of G_i in $O(\log^* n)$ rounds.
5. Every node $v \in V$ then gets a vector $\boldsymbol{x}_v \in \{0, 1, 2\}^\Delta$ of colors, where $x_{v,i}$ is the color of v in graph G_i
6. For every two neighbors u and v, we have $\boldsymbol{x}_u \neq \boldsymbol{x}_v$
   - If the edge $\{u, v\}$ has label i, we have $x_{u,i} \neq x_{v,i}$

Coloring Bounded-Degree Graphs

Theorem: For a graph with maximum degree Δ, there is a distributed algorithm to compute a **3^{Δ}-coloring** in **$O(\log^* n)$ rounds**.

- Assumes that the graph initially has unique IDs between 0 and n^c
  - Or actually just between 0 and $2^{2^{\cdot^{\cdot^{2^n}}}}$, where the power tower is of size $O(\log^* n)$.
  - We will from now on just assume this.
- Use the algorithm from before.
- There are 3^{Δ} different vectors in $\{0,1,2\}^{\Delta}$.

Theorem: For a graph with maximum degree $\Delta = O(1)$, there are distributed algorithms to compute a **$(\Delta+1)$-coloring** and an MIS in **$O(\log^* n)$ rounds**.

- We saw that if a C-coloring is given, we can compute a $(\Delta+1)$-coloring and an MIS in C rounds.

Coloring Unrooted Trees

- How can we color a tree if it is not rooted?
- Electing a root and orienting towards the root costs $\Theta(D)$ rounds!
- A rooted tree provides an orientation of the edges of a tree such that the out-degree of each node is at most 1.
- With the algorithm from before, an orientation, where the out-degree is at most c (for some constant c) would also be useful.
  - Label the edges with c labels such that each label induces a directed pseudoforest
- We can then compute a 3^c-coloring in time $O(\log^* n)$.
  - For each label, we compute a 3-coloring
  - Each node then gets a vector in $\{0, 1, 2\}^c$
- How can we compute such an orientation for a small c?
  - Let's try $c = 2$.
  - This would give a 9-coloring…

Computing an Orientation With Out-Degree 2

- Computing an orientation with out-degree ≤ 2 is trivial for nodes of degree ≤ 2

Observation: In an n-node tree, at least $n/2$ nodes have degree ≤ 2.

$n/3$

$\#\text{edges} = n-1$

$$\sum_{v\in V} \deg(v) = 2n-2 < 2n$$

Assume that k nodes have deg. ≥ 3

$$\Rightarrow \sum_{v\in V} \deg(v) \geq 3\cdot k + n-k = \underbrace{n+2k}_{<2n}$$

$$\hookrightarrow k < \frac{n}{2}$$

Computing an Orientation With Out-Degree 2

Observation: In an n-node tree, at least $n/2$ nodes have degree ≤ 2.

$< \frac{n}{2}$ nodes

remaining nodes
(average remaining degree still < 2)

recursively comp. out-degr.-2 or.

nodes of degree ≤ 2

does not matter how we orient

Computing an Orientation With Out-Degree 2

Claim: In an (unrooted) n-node tree $T = (V, E)$, an edge orientation with out-degree ≤ 2 can be computed in time $O(\log n)$

1. Define

$$V_0 \coloneqq \{v \in V : \deg_T(v) \leq 2\}$$

$$E_0 \coloneqq \{e \in E : e \cap V_0 \neq \emptyset\}$$

2. Orient edges $\{u, v\} \in E_0$ as follows
   - If $|\{u, v\} \cap V_0| = 1$, orient edge from the node in V_0 to the node in $V \setminus V_0$
   - If $|\{u, v\} \cap V_0| = 1$, orient edge arbitrarily

3. Recursively compute an out-degree ≤ 2 orientation of $T[V \setminus V_0]$

9-Coloring Unrooted Trees

Theorem: In an unrooted n-node tree, there is a distributed algorithm to compute a **9-coloring in $O(\log n)$ rounds**.

- We saw that an orientation with out-degree ≤ 2 can be computed in time $O(\log n)$.
- This allows to decompose the tree into two directed pseudoforests
  - Because it is a tree, actually into two forests, where each tree is rooted
- Each forest can be colored with 3 colors in time $O(\log^* n)$.
- Every node v then has two colors, $x_{v,1}$ for forest 1 and $x_{v,2}$ for forest 2
  - The number of possible color combinations for a node is 9.
- For every edge $\{u, v\}$, we have $x_{u,1} \neq x_{v,1}$ or $x_{u,2} \neq x_{v,2}$

Remark: Algorithm also works for (undirected) pseudoforests.

Summary

Coloring Trees

- Trees can be colored with 2 colors, this however requires time $\Omega(D)$.
- Rooted trees can be 3-colored in time $O(\log^* n)$.
- Unrooted trees can be 3-colored in time $O(\log n)$.

Coloring General Graphs

- 3^Δ-coloring of graphs with max. degree Δ in time $O(\log^* n)$
- $(\Delta + 1)$-coloring of graphs with max. degree Δ in time $O\left(3^\Delta + \log^* n\right)$
  - If $\Delta = O(1)$, this is $O(\log^* n)$.
  - This algorithm can be improved significantly.

$O(\sqrt{\Delta} + \log^* n)$

Outlook

- So far, we looked at deterministic algorithms, next week, we will see randomized for $(\Delta + 1)$-coloring and MIS in general graphs.
- We will later also see that for deterministic algorithms, the bounds from today are essentially tight.