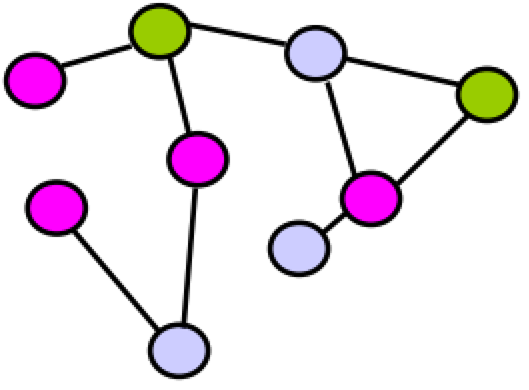
Network Algorithms

Vertex Coloring

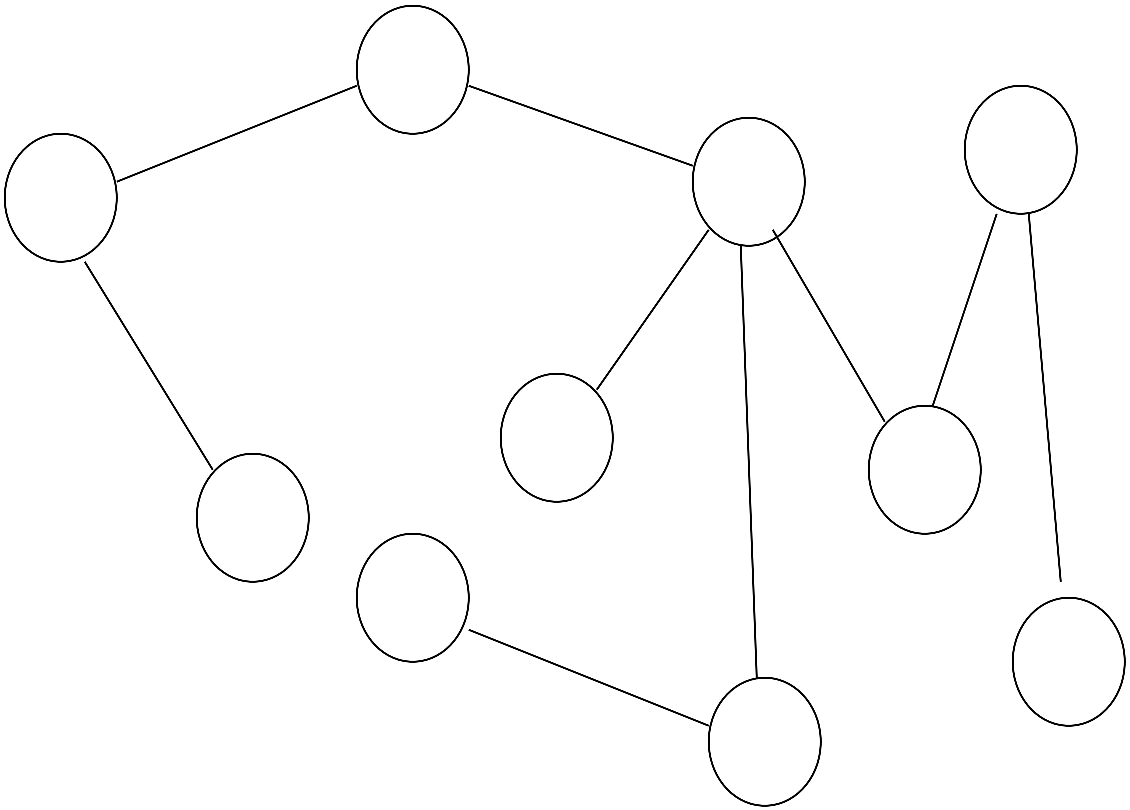
Graph Coloring

Vertex Coloring

Nodes should color themselves such that no adjacent nodes have same color – but minimize # colors!

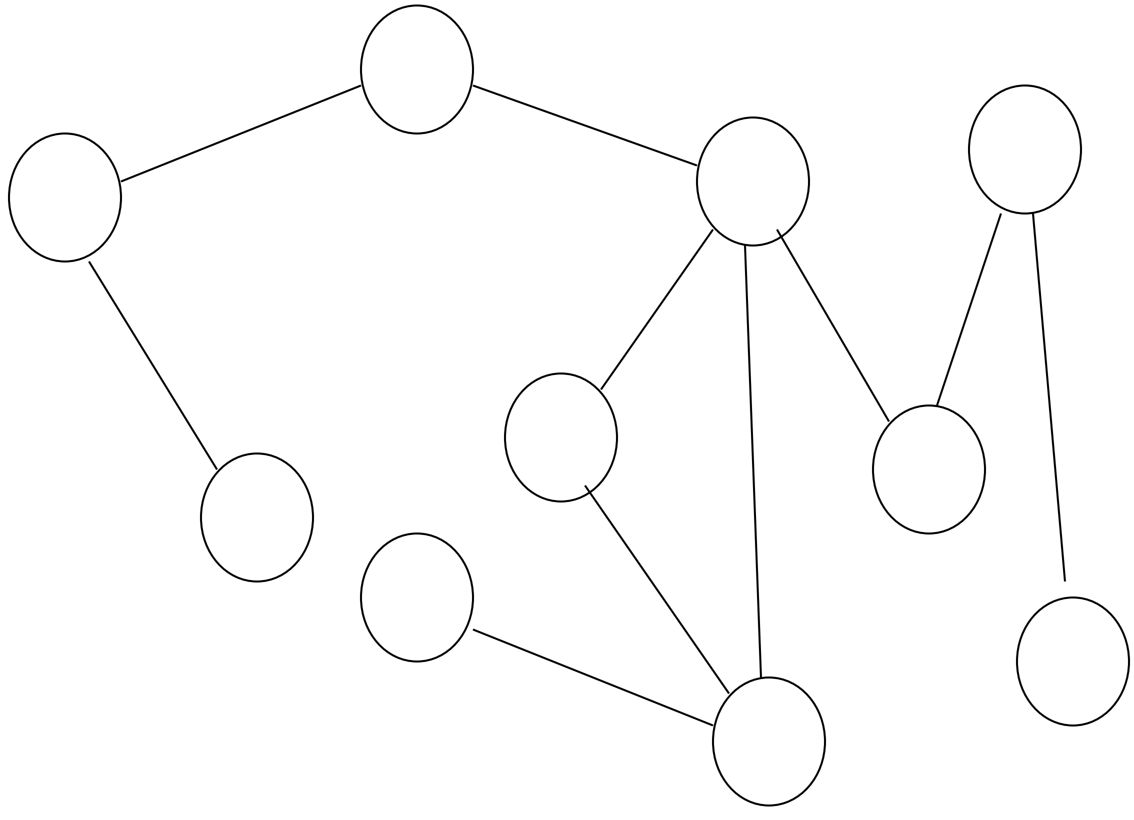


How to color? Chromatic number?



Tree! Two colors enough...

And now?



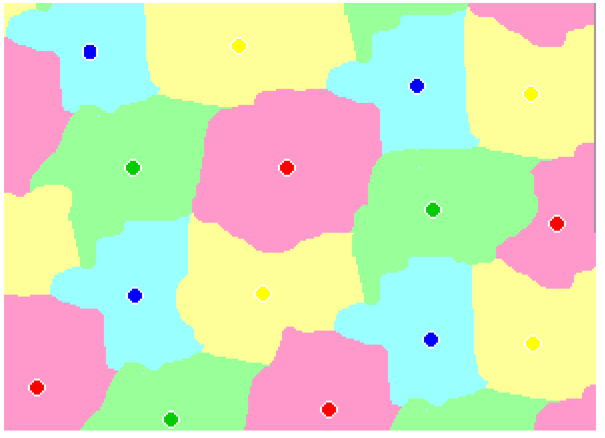
Three colors enough...

Graph Coloring

Why color a network?

Graph Coloring

Medium access: reuse frequencies in wireless networks at certain spatial distance such that there is „no“ interference.



Break symmetries: more generally...

Note: gives independent sets... How?

Science: „Human coloring“!

Human interaction as local algorithm?
How good are „we“?

REPORTS

An Experimental Study of the Coloring Problem on Human Subject Networks

Michael Kearns,* Siddharth Suri, Nick Montfort

Theoretical work suggests that structural properties of naturally occurring networks are important in shaping behavior and dynamics. However, the relationships between structure and behavior are difficult to establish through empirical studies, because the networks in such studies are typically fixed. We studied networks of human subjects attempting to solve the graph or network coloring problem, which models settings in which it is desirable to distinguish one's behavior from that of one's network neighbors. Networks generated by preferential attachment made solving the coloring problem more difficult than did networks based on cyclical structures, and "small worlds" networks were easier still. We also showed that providing more information can have opposite effects on performance, depending on network structure.

It is often thought that structural properties of naturally occurring networks are influential in shaping individual and collective behavior and dynamics. Examples include the popular notion that "hubs" or "connectors" are inordinately important in the routing of information in social and organizational networks (*1*, *2*). A long history of research has established the frequent empirical appearance of certain structural properties in networks from many domains, including sociology (*1*, *3*–*5*), biology (*6*, *7*), and technology (*8*). These properties include small diameter (the "six degrees of separation" phenomenon), local clustering of connectivity (*9*), and heavy-tailed distributions of connectivity (*10*). Theoretical models have sought to explain how some of these may interact with network dynamics (*11*).

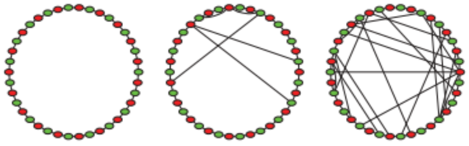
The relationships between structure and behavior are difficult to establish in empirical field studies of existing networks. In such studies, the network structure is fixed and given, thus preventing the investigation of alternatives. A different approach is to conduct controlled laboratory studies in which network structure is deliberately varied.

We have been performing human subject experiments in distributed problem-solving from local information on a variety of simple and complex networks. Subjects each simultaneously control a single vertex in a network of 38 vertices and attempt to solve the challenging graph coloring problem (*12*) on the network. In this problem, the collective goal is for every player to select a color for their vertex that

The graph coloring problem is a natural abstraction of many human and organizational problems in which it is desirable or necessary to distinguish one's behavior from that of neighboring parties. As a specific scenario, consider the problem faced by faculty members scheduling departmental events—recurring classes, one-time seminars, exams, and so on—in a limited number of available rooms. We can view the events to be scheduled as the vertices in a network, with an edge connecting any pair of events that temporally overlap, even partially. Clearly, two such events must be assigned to different rooms or "colors," thus yielding a natural graph coloring problem. Furthermore, even when there is a centralized first-come, first-serve sign-up sheet for rooms, this mechanism is simply the starting point for the negotiation of a solution, and the problem is still solved in a largely distributed fashion by the participants: Faculty members routinely query the current holder of a room whether they might be able to switch to a different room, whether their event will really require their entire time slot, and the like. Other coloring-like problems arise in a variety of social activities (such as selecting a cell phone ringtone that differs from those of family members, friends, and colleagues); technological coordination [selecting a channel unused by nearby parties in a wireless communication network (*13*, *14*)]; and individual differentiation within an organization (developing an expertise not duplicated by others nearby). Graph coloring also generalizes many traditional problems in logistics and operations research (*12*).

The coloring problem was chosen for both its simplicity of description and its contrast to other distributed network optimization problems. Unlike the well-studied studied navigation or shortest-paths problem, optimal coloring is notoriously intractable from the viewpoint of even centralized computation (*12*, *15*). In fact, even weak approximations (in which many more colors than the chromatic number are permitted) are known to be equally difficult (*16*, *17*).

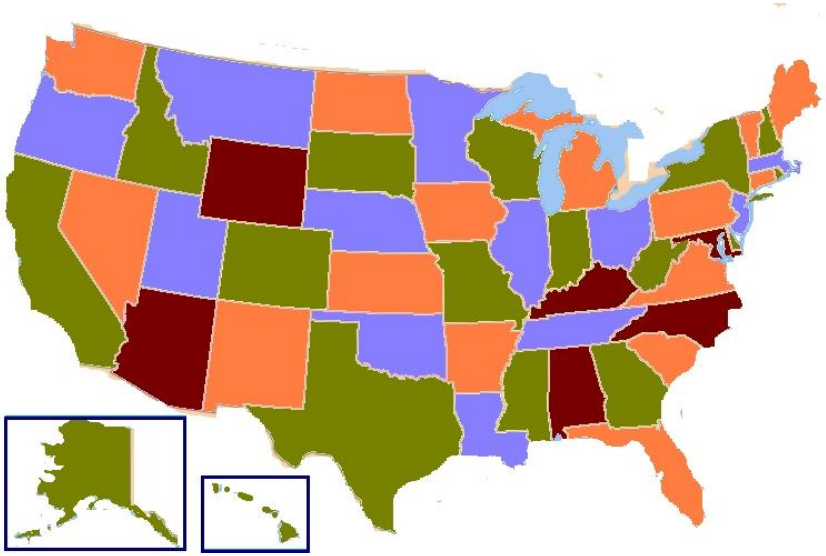
We report here on the findings from two extensive experimental sessions held in January 2006 with 55 University of Pennsylvania undergraduate students (*18*). Subjects were given a series of coloring experiments in which the network had one of six topologies, each chosen according to recently proposed models of network formation (Fig. 1 and Table 1). Three of these six begin with a simple cycle and then add a varying number of randomly chosen chords while preserving a chromatic number of two. These "small worlds" networks (*9*, *19*) are intended to model the mixture of local connectivity (as induced by geography) with long-distance connectivity (as induced by travel or chance meetings) often found in social and other networks. The fourth cycle-based network adopted a more engineered or hierarchical structure, with two distinguished individuals having inordinately high connectivity. The fifth and sixth networks were generated according to the well-studied preferential attachment

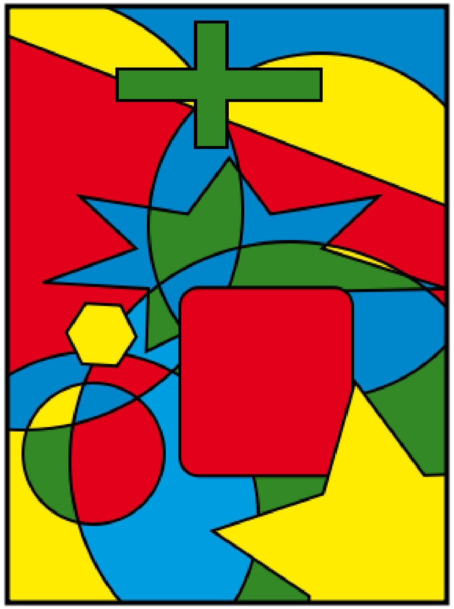


Also good to know...

4-Color Theorem

Can color each map using 4 colors only:
no two adjacent countries have same color.



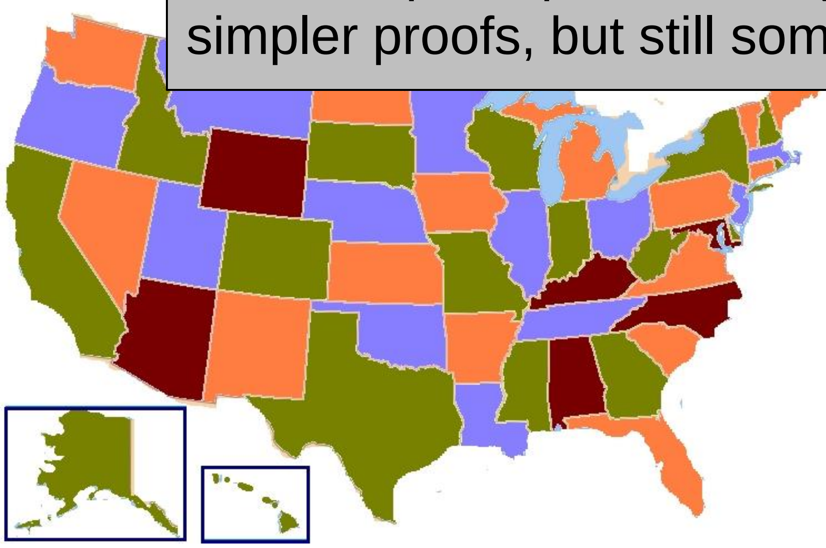


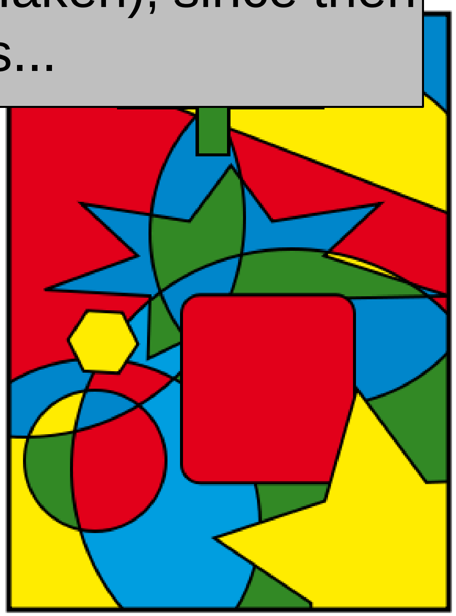
Also good to know...

4-Color Theorem

Can color each map using 4 colors only:
no two adjacent countries have same color.

First conjecture 1852, first proof with 5 colors 1890.
First computer proof 1976 (Appel+Haken), since then simpler proofs, but still some doubts...





Simple Coloring Algorithm? (Not distributed!)

Greedy Sequential

while (uncolored vertices v left):
    color v with minimal color that does not conflict with neighbors

Analysis?
 # rounds/steps?
 # colors?

Simple Coloring Algorithm? (Not distributed!)

Greedy Sequential

while (uncolored vertices v left):
    color v with minimal color that does not conflict with neighbors

 # steps

At most n steps: walk through all nodes...

 # colors

$\Delta+1$, where Δ is max degree.
Because: there is always a color free in $\{1, ..., \Delta+1\}$

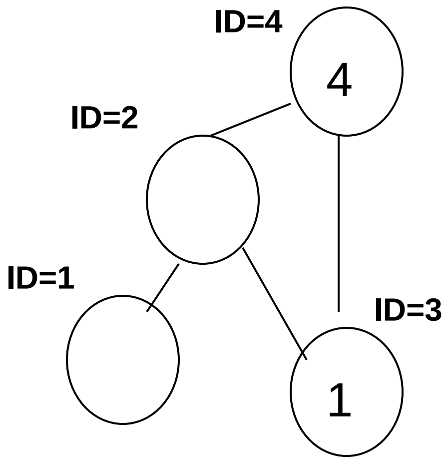
Note: many graphs can be colored with less colors! Examples?

How to do it in a distributed manner?

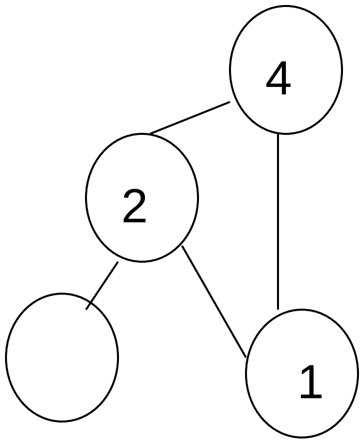
Now distributed!

First Free

Assume initial coloring (e.g., unique ID=color)
1. Each node uses smallest available color in neighborhood



Assume: two neighbors never choose color at the same time...

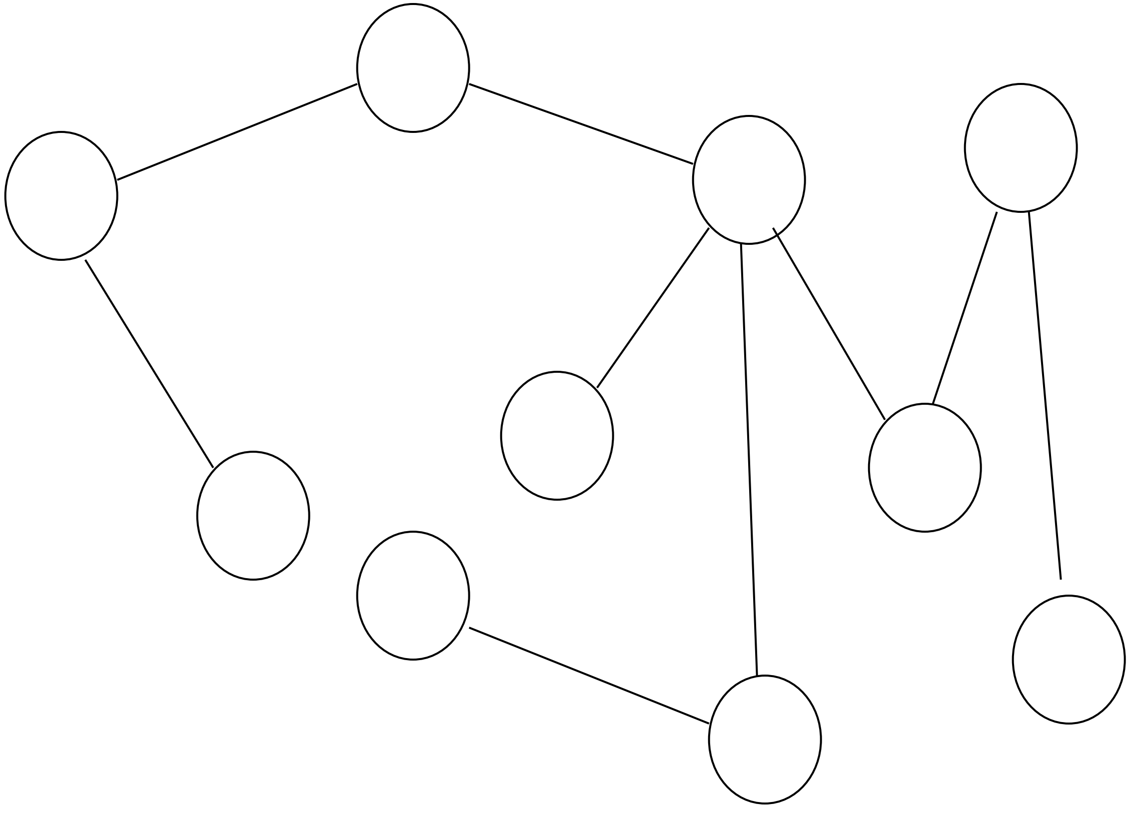


Reduce

Initial coloring = IDs
Each node v:
1. v sends ID to neighbors (idea: sort neighbors!)
2. while (v has uncolored neighbor with higher ID)
    1. v sends „undecided“ to neighbors
3. v chooses free color using **First Free**
4. v sends decision to neighbors

Analysis? Not parallel!

Let us focus on trees now....
Chromatic number?
Algo?



Slow Tree

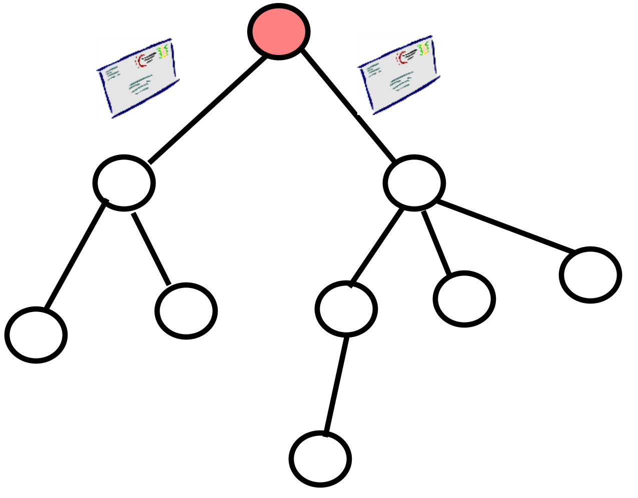
1. Color root 0, send to kids

Each node v does the following:

- Receive message x from parent
- Choose color y=1-x
- Send y to kids

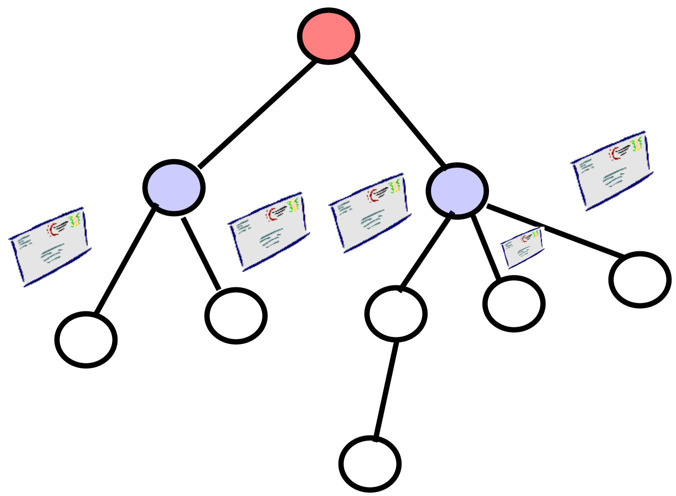
Slow Tree

Two colors suffice: root sends binary message down...



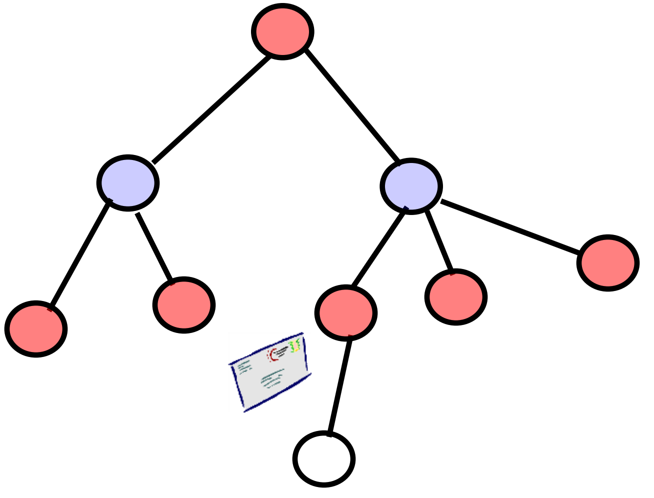
Slow Tree

Two colors suffice: root sends binary message down...



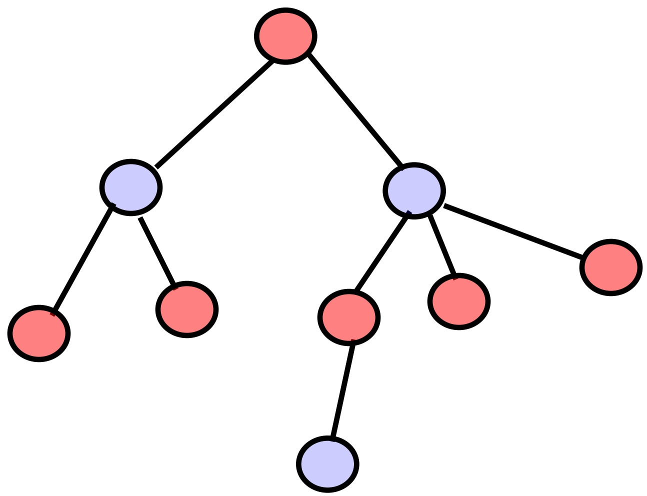
Slow Tree

Two colors suffice: root sends binary message down...



Slow Tree

Two colors suffice: root sends binary message down...



Time complexity?

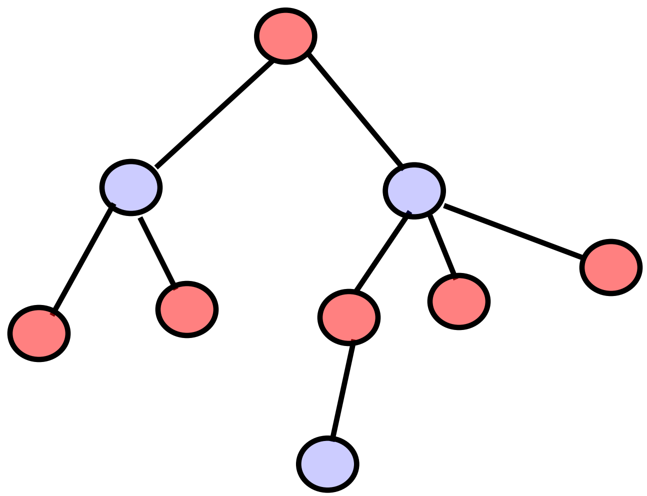
Message complexity?

Local compuations?

Synchronous or
asynchronous?

Slow Tree

Two colors suffice: root sends binary message down...



Time complexity? depth $\leq$ n

Message complexity? n-1

Local compuations? laughable...

Synchronous or asynchronous? both!

Discussion

Time complexity? depth $\leq$ n

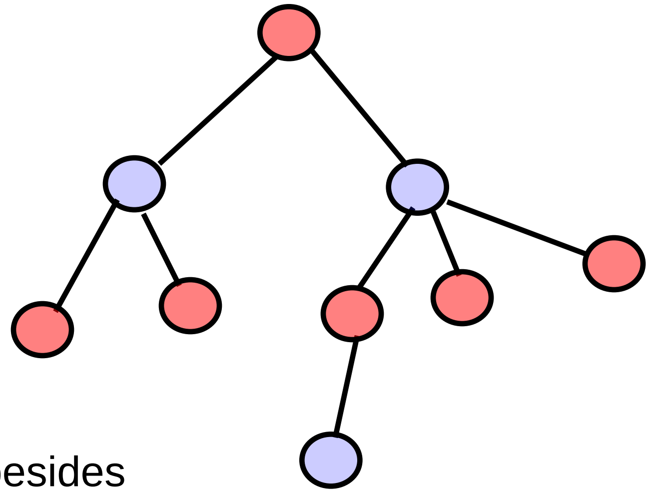
Message complexity? n-1

Local compuations? laughable...

Synchronous or asynchronous? both!

Can we do better?

Local Vertex Coloring for Tree?



Can we do faster than diameter of tree?!

Yes! With **constant** number of colors in

log*(n) time

One of the fastest non-constant time algos that exist! (... besides inverse Ackermann function or so)

(log = divide by two, loglog = ?, log* = ?)

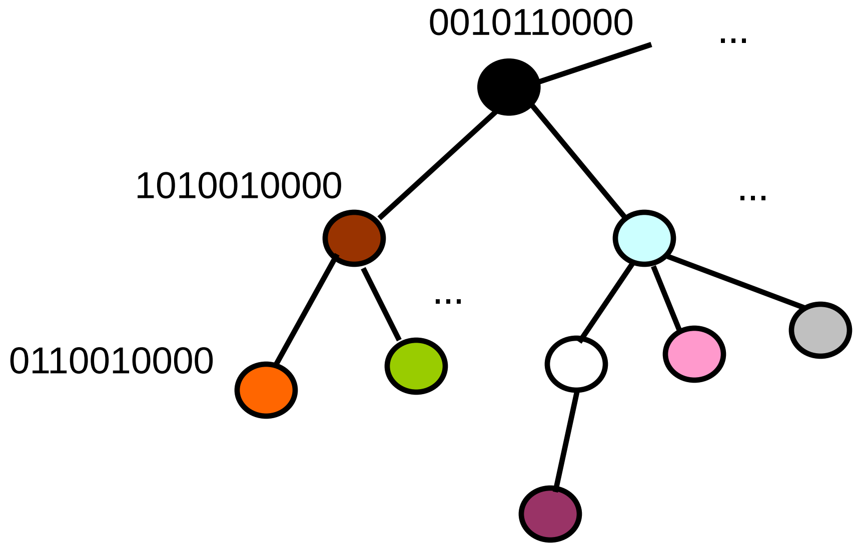
log* (# atoms in universe) $\approx$ 5

Why is this good? If something happens (dynamic network), back to good state in a sec!

There is a lower bound of log-star too, so that‘s optimal!

How does it work?

Initially: each node has unique log(n)-bit ID = legal coloring
(interpret ID as color => n colors)



Idea:

root should have label 0 (fixed)
in each step: send ID to c_v to all children;
receive c_p from parent and interpret as little-endian bit string: $c_p=c(k)...c(0)$
let i be smallest index where c_v and c_p differ
set new c_v = i (as bit string) || $c_v(i)$
until $c_v \in \{0,1,2,...,5\}$ (at most 6 colors)

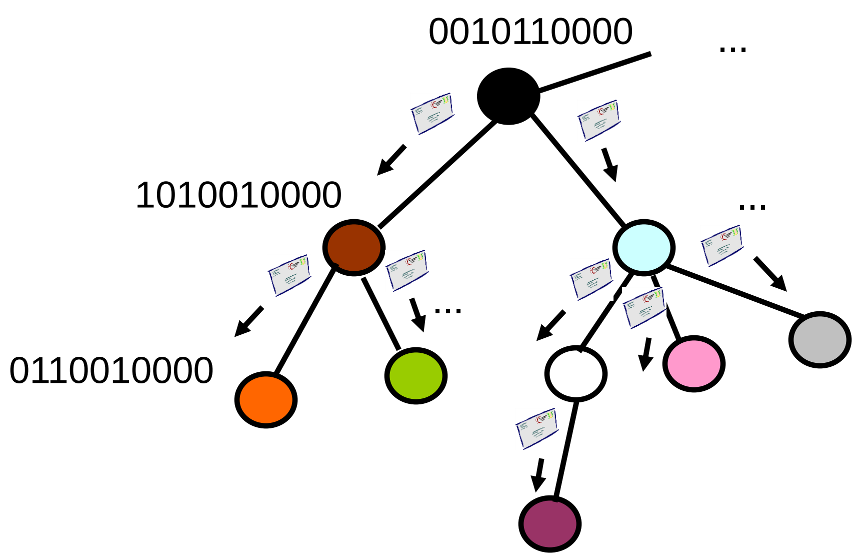
6-Colors

Assume legal initial coloring
Root sets itself color 0
Each other node v does (in parallel):

1. Send c_v to kids
2. Repeat (until $c_w \in \{0,...,5\}$ for all w):
   1. Receive c_p from parent
   2. Interpret c_v/c_p as little-endian bitstrings c(k)...c(1)c(0)
   3. Let i be smallest index where c_v and c_p differ
   4. New label is: **$i||c_v(i)$**
   5. Send c_v to kids

How does it work?

Initially: each node has unique log(n)-bit ID = legal coloring
(interpret ID as color => n colors)



Round 1

Idea:

root should have label 0 (fixed)

in each step: send ID to c_v to all children;

receive c_p from parent and interpret as little-endian bit string: $c_p=c(k)...c(0)$

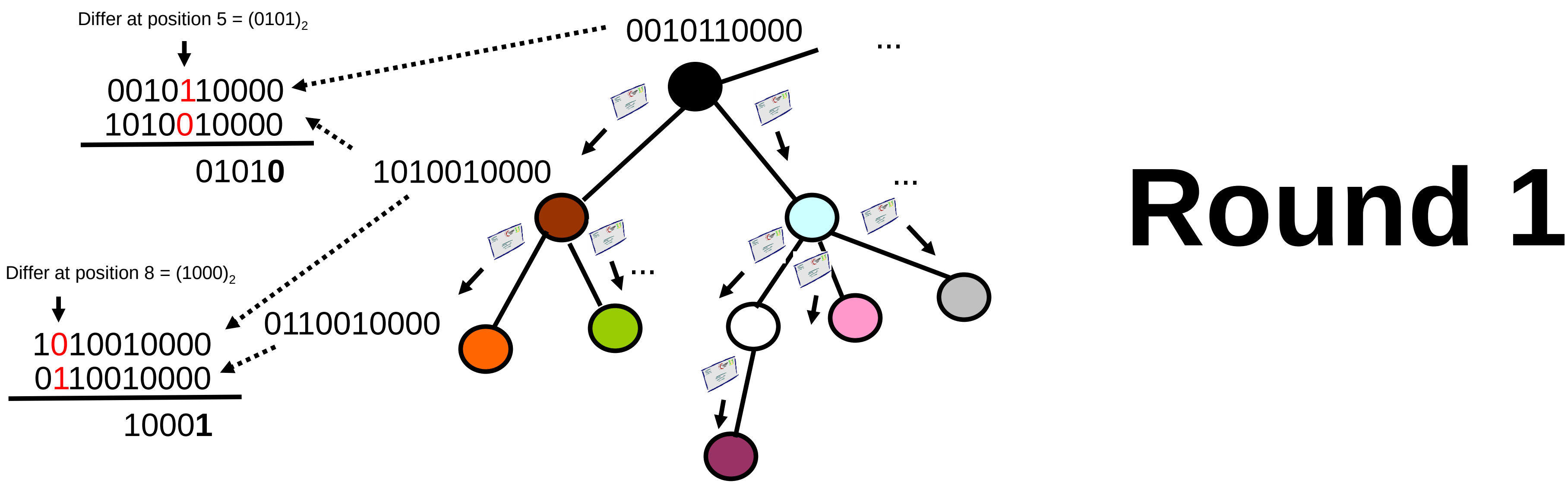
let i be smallest index where c_v and c_p differ

set new c_v = i (as bit string) || $c_v(i)$

until $c_v \in \{0,1,2,...,5\}$ (at most 6 colors)

How does it work?

Initially: each node has unique log(n)-bit ID = legal coloring
(interpret ID as color => n colors)



Idea:

root should have label 0 (fixed)

in each step: send ID to c_v to all children;

receive c_p from parent and interpret as little-endian bit string: $c_p=c(k)...c(0)$

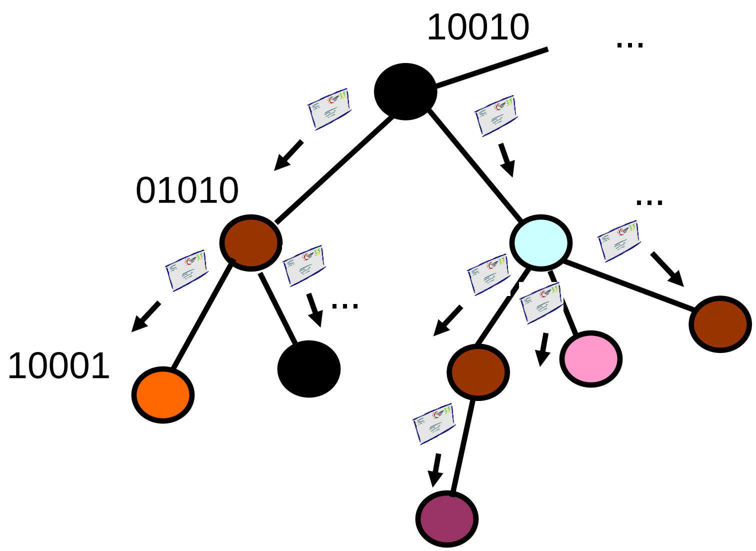
let i be smallest index where c_v and c_p differ

set new c_v = i (as bit string) || $c_v(i)$

until $c_v \in \{0,1,2,...,5\}$ (at most 6 colors)

How does it work?

Initially: each node has unique log(n)-bit ID = legal coloring
(interpret ID as color => n colors)

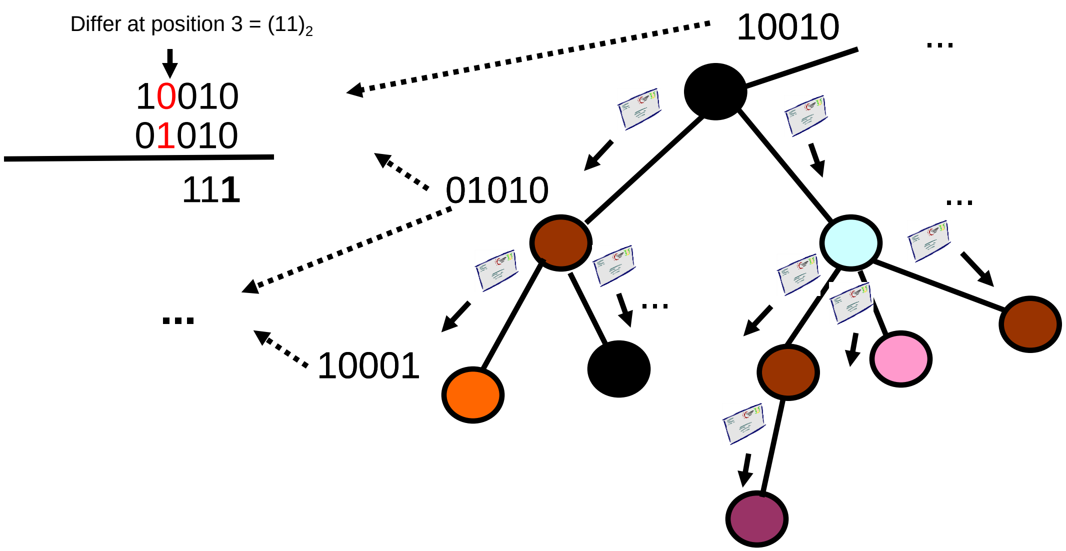


Round 2

Idea:
- root should have label 0 (fixed)
- in each step: send ID to c_v to all children;
- receive c_p from parent and interpret as little-endian bit string: $c_p=c(k)...c(0)$
- let i be smallest index where c_v and c_p differ
- set new c_v = i (as bit string) || $c_v(i)$
- until $c_v \in \{0,1,2,...,5\}$ (at most 6 colors)

How does it work?

Initially: each node has unique log(n)-bit ID = legal coloring
(interpret ID as color => n colors)



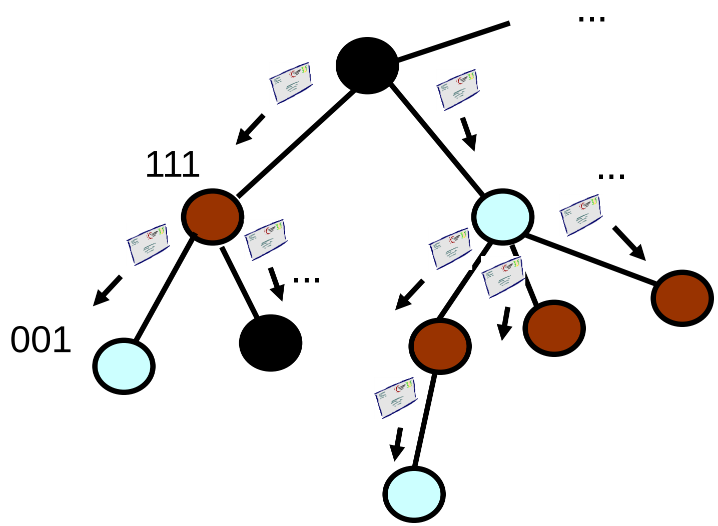
Round 2

Idea:

root should have label 0 (fixed)
in each step: send ID to c_v to all children;
receive c_p from parent and interpret as little-endian bit string: $c_p=c(k)...c(0)$
let i be smallest index where c_v and c_p differ
set new c_v = i (as bit string) || $c_v(i)$
until $c_v \in \{0,1,2,...,5\}$ (at most 6 colors)

How does it work?

Initially: each node has unique log(n)-bit ID = legal coloring
(interpret ID as color => n colors)



Round 3, etc.

Idea:

root should have label 0 (fixed)

in each step: send ID to c_v to all children;

receive c_p from parent and interpret as little-endian bit string: c_p=c(k)...c(0)

let i be smallest index where c_v and c_p differ

set new c_v = i (as bit string) || c_v(i)

until $c_v \in \{0,1,2,...,5\}$ (at most 6 colors)

Why does it work?

Why is this log* time?!

Idea: In each round, the size of the ID (and hence the number of colors) is reduced by a log factor:
To index the bit where two labels of size n bits differ, log(n) bits are needed!
Plus the one bit that is appended...

Why is this a valid vertex coloring?!

Idea: During the entire execution, adjacent nodes always have different colors (invariant!) because:
IDs always differ as new label is index of difference to parent plus own bit there (if parent would differ at same location as grand parent, at least the last bit would be different).

Why $c_w \in \{0,...,5\}$?! Why not more or less?

Idea: {0,1,2,3} does not work, as two bits are required to address index where they differ, plus adding the „difference-bit" gives more than two bits...

Idea: {0,1,2,...,7} works, as $7=(111)_2$ can be described with 3 bits, and to address index (0,1,2) requires two bits, plus one „difference-bit" gives three again.

Moreover: colors 110 (for color „6") and 111 (for color „7") are not needed, as we can do another round! (IDs of three bits can only differ at positions 00 (for „0"), 01 (for „1"), 10 (for „2")

Everything super?

When can I terminate?

Not a local algorithm like this! Node cannot know when *all* other nodes have colors in that range!

Kid should not stop before parent stops! Solution: wait until parent is finished?

No way, this takes linear time in tree depth!

Ideas?

If nodes know n, they can stop after the (deterministic) execution time...

Other ideas? Maybe an exercise...

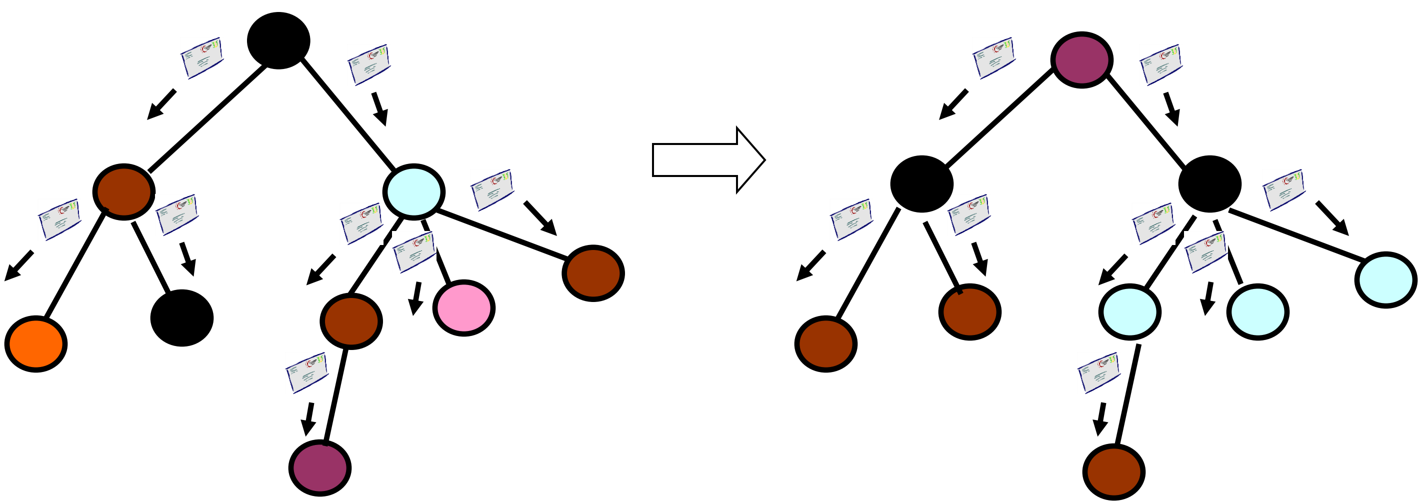
Six colors is good: but we know that tree can be colored with two only!

How can we improve coloring quickly?

Shift Down

Shift Down

Each node v concurrently does:
recolor v with color of parent



Property?
Preserves coloring legality!
Siblings become monochromatic!
(Make siblings „independent".)

6-to-3

Each other node v does (in parallel):

1. Run „**6-Colors**“ for log*(n) rounds
2. For x=5,4,3:
   1. Perform **Shift Down**
   2. If (c_v=x) choose new color $c_v \in \{0,1,2\}$ according „**first free**“ principle

Why still log*?

Rest is fast....

Why {3,4,5} recoloring not in same step?

Make sure coloring remains legal....

Cancel remaining colors one at a time

(nodes of same color independent, but not others: parent may also be in {3,4,5})!

Why does it work?

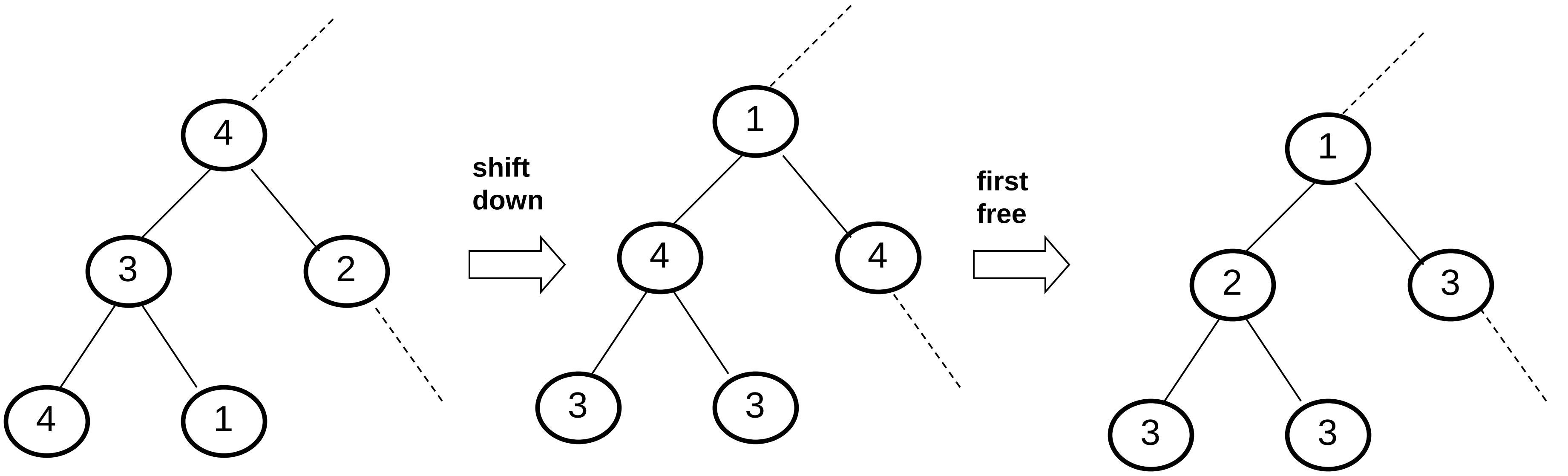
One of the three colors must be free!

(Need only two colors in tree, and due to shift down, one color is occupied by parent, one by children!)

We only recolor nodes simultaneously which are not adjacent.

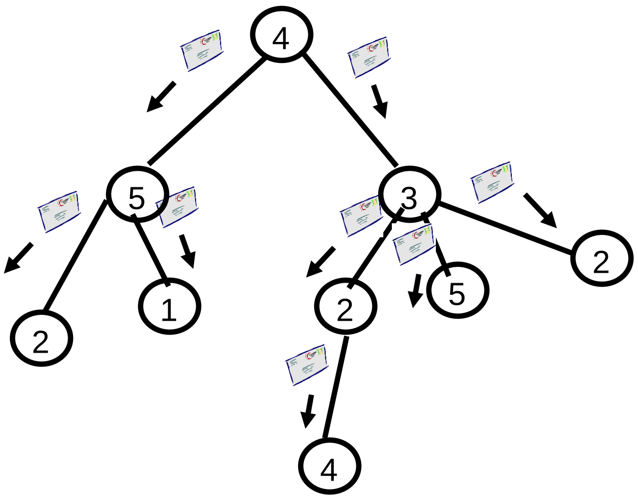
And afterwards no higher color is left...

Example: Shift Down + Drop Color 4

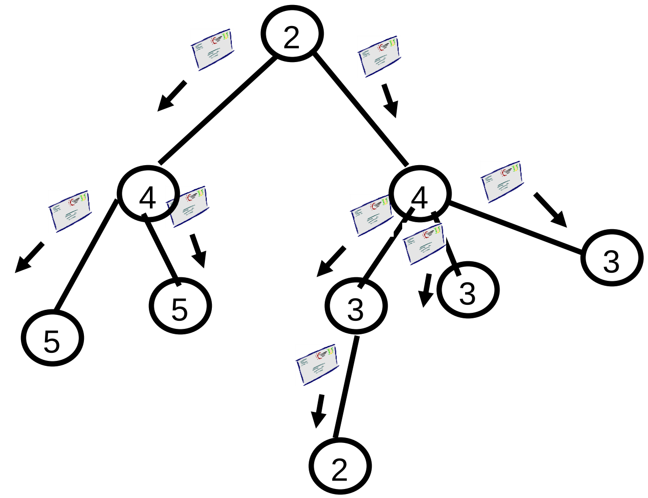


Siblings no longer have same color => must do shift down again first!

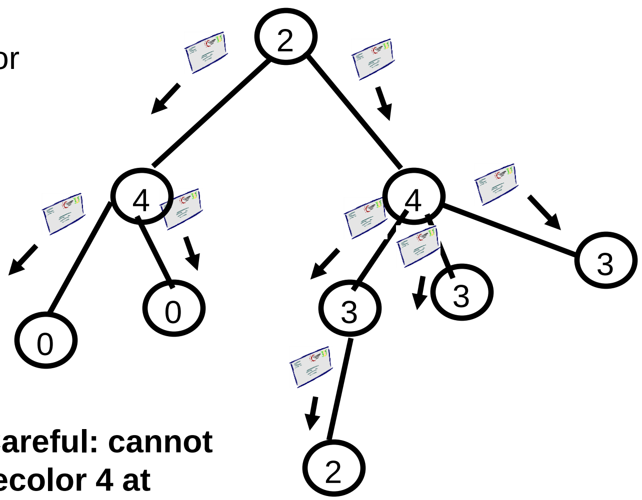
Example: 6-to-3



shift down

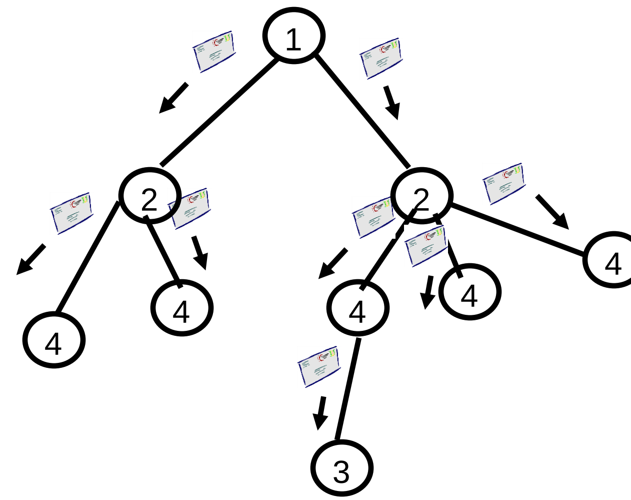


new color for
5: **first free**



Careful: cannot recolor 4 at same time!

shift
down



Discussion

Can we reduce to 2 colors?

Not without increasing runtime significantly!
(Linear time, more than exponentially worse!)

Other topologies?

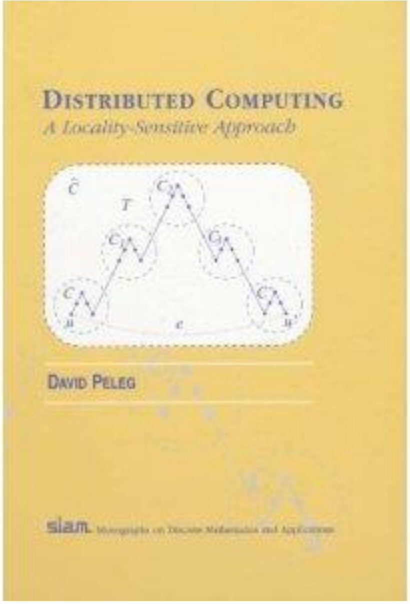
Yes, similar ideas to $O(\Delta)$-color general graphs
with constant degree Δ in log* time!
How?

Lower bounds?

Yes. ☺
In particular, runtime of our algorithm is asymptotically optimal.

Literature for further reading:

- Peleg‘s book:



End of lecture