# Distributed Coloring

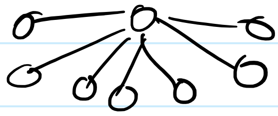
24 June 2019 12:56

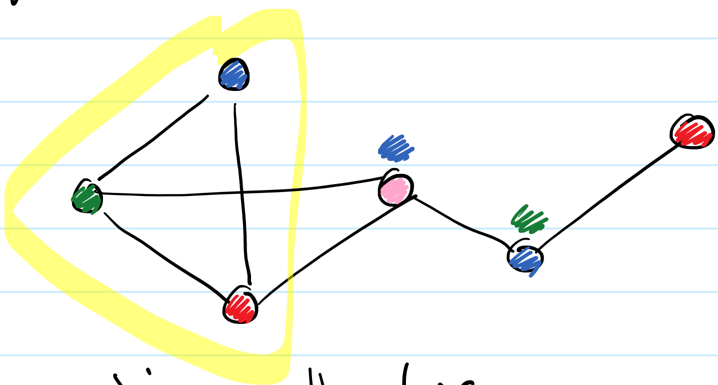
next 2 lectures: symmetry breaking in networks

## vertex coloring

Graph $G = (V, E)$

vertex coloring: coloring of nodes of $G$ s.t. no 2 neighbors get the same color





optimal coloring: minimum # colors

optimal #colors: chromatic number $\chi(G)$

finding $\chi(G)$ is NP-hard (even approximately)

## What about greedy coloring?

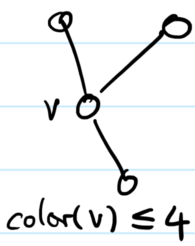
greedy coloring: iterate through nodes, always assign smallest possible color

$\Delta := \max_{v \in V} deg(v)$

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

**Proof:** $color(v) \leq deg(v) + 1$ ✓

$\leq \Delta + 1$



# Distributed Coloring?

24 June 2019 12:56

$\Rightarrow$ color the network graph $G = (V, E)$

Initially, the nodes of $G$ know their unique ID, but nothing else about $G$

At the end, each node $v \in V$ needs to know its own color

**Assumption on IDs**: $ID(v)$ is an $O(\log n)$-bit number

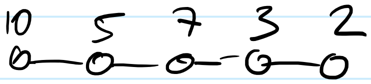
## Distributed greedy?

each node $v \in V$ does:

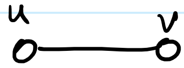
- wait for all neighbors $u$ with $ID(u) > ID(v)$ to pick a color
- color $v$ with smallest available color
- $v$ informs its neighbors about its choice

**Claim:** Alg. computes correct $(\Delta+1)$-coloring of $G$.

**Proof:**




correct coloring: no 2 neighbors choose color at the same time



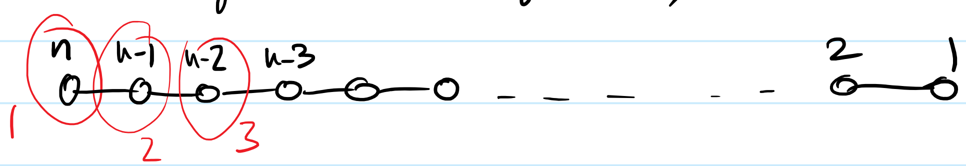

w.l.o.g.: $ID(u) < ID(v)$

$u$ waits for $v$ to be colored

**$\Delta+1$ colors:** same argument as for the seq. greedy alg.

**Thm:** Distr. Greedy computes a $(\Delta+1)$-coloring in $O(n)$ rounds.

**Proof:** $\geq 1$ node gets colored in each round (remaining node with largest ID)



24 June 2019 12:56

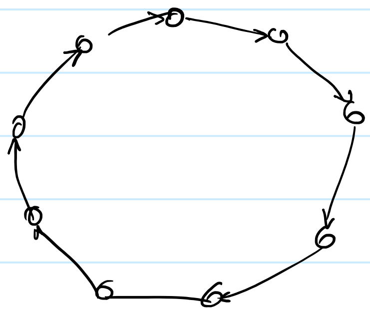
# Faster algorithms? Ideas?

maybe use randomization? e.g. with random IDs?

---

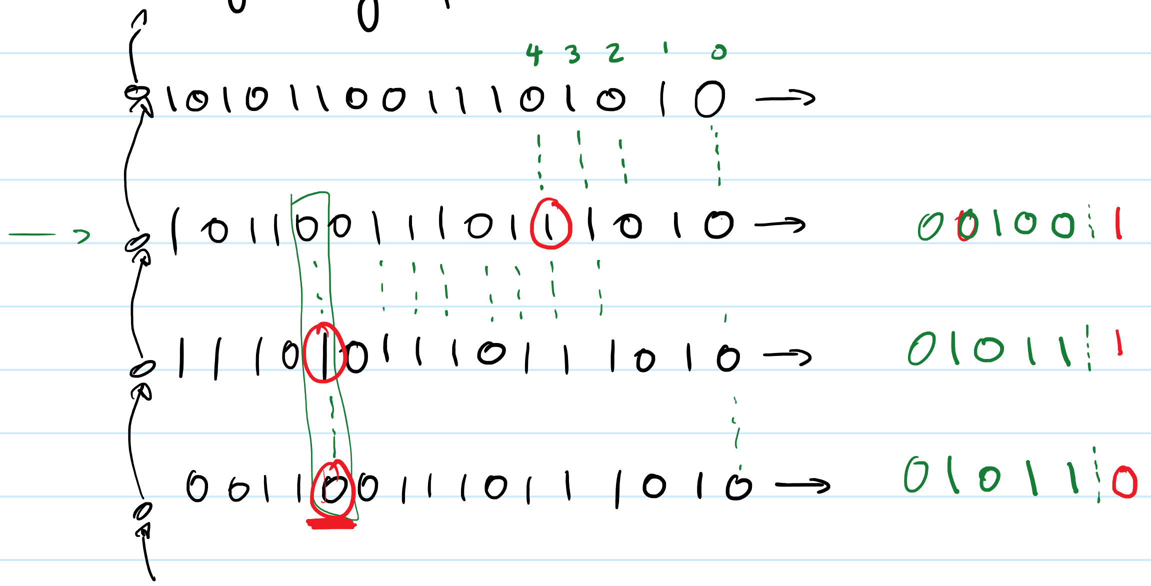
**today:** simple case: coloring rings (with orientation)

↑ distinguish clockwise & direction



every node has 1 incoming / 1 outgoing neighbor

## Fast color reduction by using representation




**new color of v**

old color $x_v$, out-neighbor of v is u

$pos(x_v, x_u)$ : position of first bit where $x_u, x_v$ differ (LSB is bit 0)

$bit(x_v, x_u)$ : bit of $x_v$ at position $pos(x_v, x_u)$

$$x_v' = \underbrace{pos(x_v, x_u) \mid bit(x_v, x_u)}_{\text{in binary repr.}}$$

# How fast is this algorithm?

24 June 2019 12:56

assume IDs are from $0, \dots, N-1$

initial colors : IDs

old color : $x_v$ , new color $x'_v$

$$\boxed{x'_v \leq 2 \lfloor \log_2 x_v \rfloor + 1}$$

$x_v = 10110111011011101111$
$\uparrow$
$\#\text{bits} = \lfloor \log_2 x_v \rfloor + 1$

$$x'_v \lessapprox \log x_v$$

$$x''_v \lessapprox \log (x'_v) \lessapprox \log \log (x_v)$$

If we start with N colors

$\Rightarrow$ after $i$ rounds (iteration)
largest color $\approx \log^{(i)}(N)$

$$\log^{(i)}(x) := \begin{cases} x & \text{if } i = 0 \\ \log(\log^{(i-1)}(x)) & \text{if } i > 0 \end{cases}$$

$$\log^* x := \underset{i}{\arg\min} \{ \log^{(i)}(x) \leq 2 \}$$

goes to $\infty$ extremely slowly

Thm: Above algorithm computes a $\underline{\underline{6}}$-coloring in $O(\log^* n)$ rounds.

- $O(\log^* n)$ rounds : after $i$ iterations, largest color is $\approx \log^{(i)}(N)$

When does the algorithm stop improving #colors?

24 June 2019 12:56

$$x_v' \leq 2\lfloor \log_2 x_v \rfloor + 1$$

for which $x_v$ is

$$2\lfloor \log_2 x_v \rfloor + 1 < x_v$$

$$\lfloor \log_2 x_v \rfloor < \frac{x_v - 1}{2}$$

$$\log_2 x_v < \frac{x_v - 1}{2} \quad \text{for } x_v \geq 7$$

for $x_v = 6$:

$$2\lfloor \log_2 x_v \rfloor + 1 = 5 < 6$$

When colors are between 0 and 5, alg. stops improving

---

Can we improve the number of colors?

1) all nodes with color 5 in parallel pick color $\in \{0,1,2\}$
2) " " " " 4 " " " " "
3) " " " " 3 " " " " "

Can we improve the time complexity?

No: $\Omega(\log^* n)$ rounds are necessary

[Linial '87]