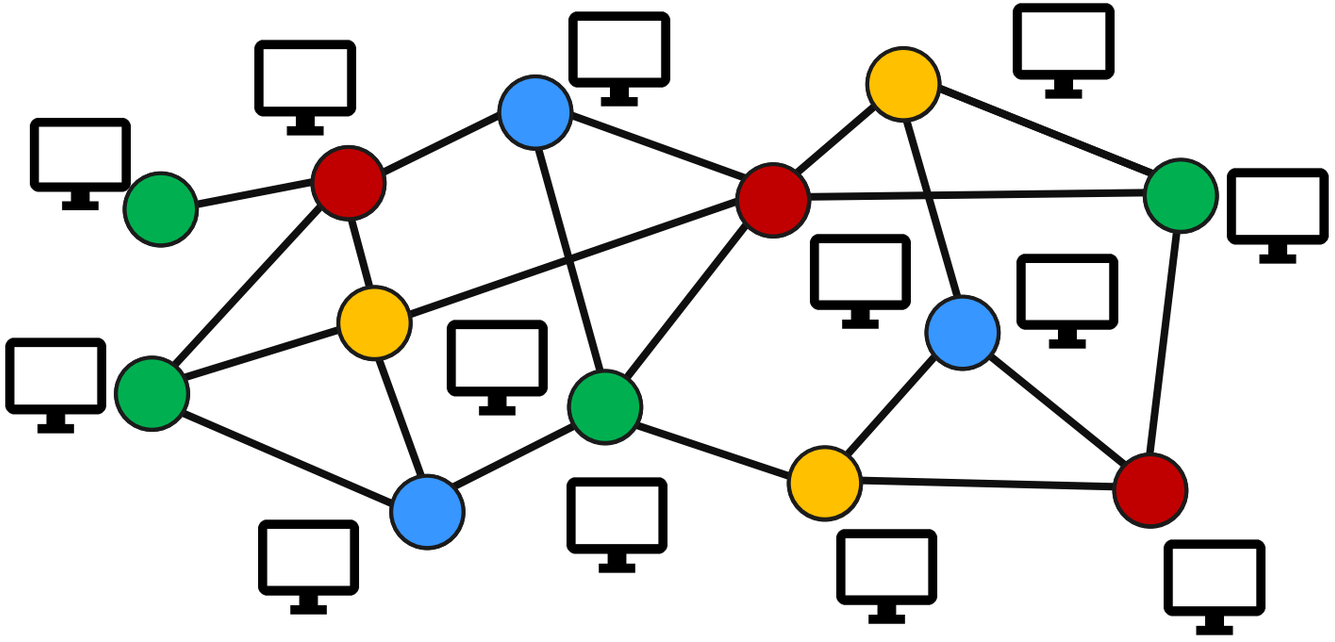


Distributed Graph Algorithms

Yannic Maus

Computers vs. Computer Networks

Theory of Computing

- *Which tasks can be solved efficiently with a **(single) computer**?*

Theory of Computing

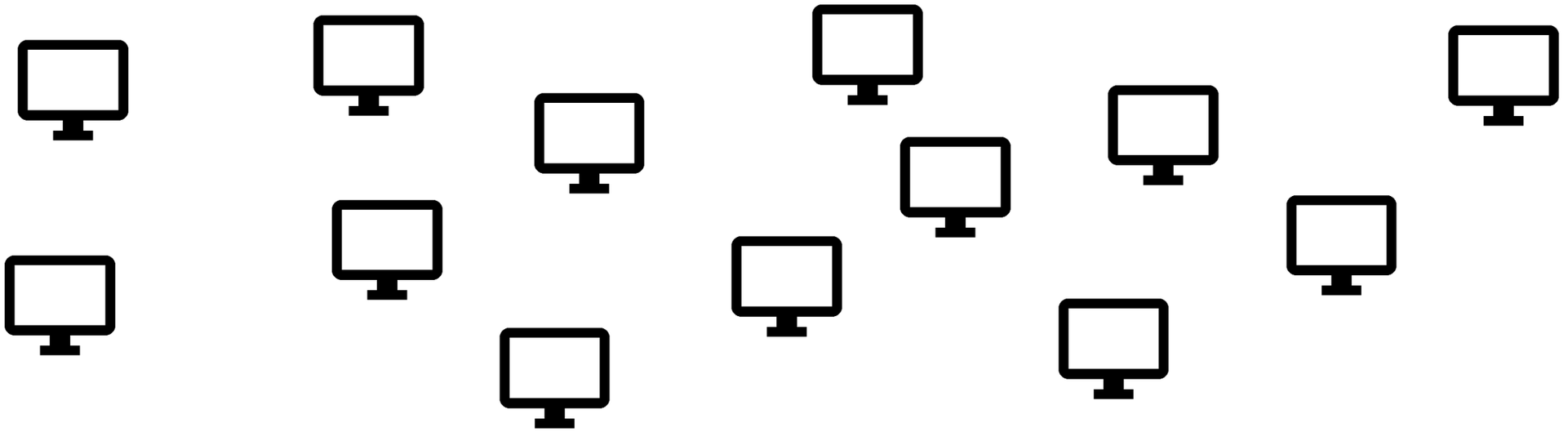
- *Which tasks can be solved efficiently with a **(single) computer**?*

Courses at TU Graz

- A.1 Fundamentals of Computer Science (Turing machines, finite automata,
- **C.2 Data Structures and Algorithms (divide & conquer, randomization, recursion, …)**
- D.1 Theoretical Computer Science (Turing machines, …)
- **D.3 Algorithm Design (graph algorithms, shortest paths, …)**

Theory of Computing

- *Which tasks can be solved efficiently with a **(single) computer**?*



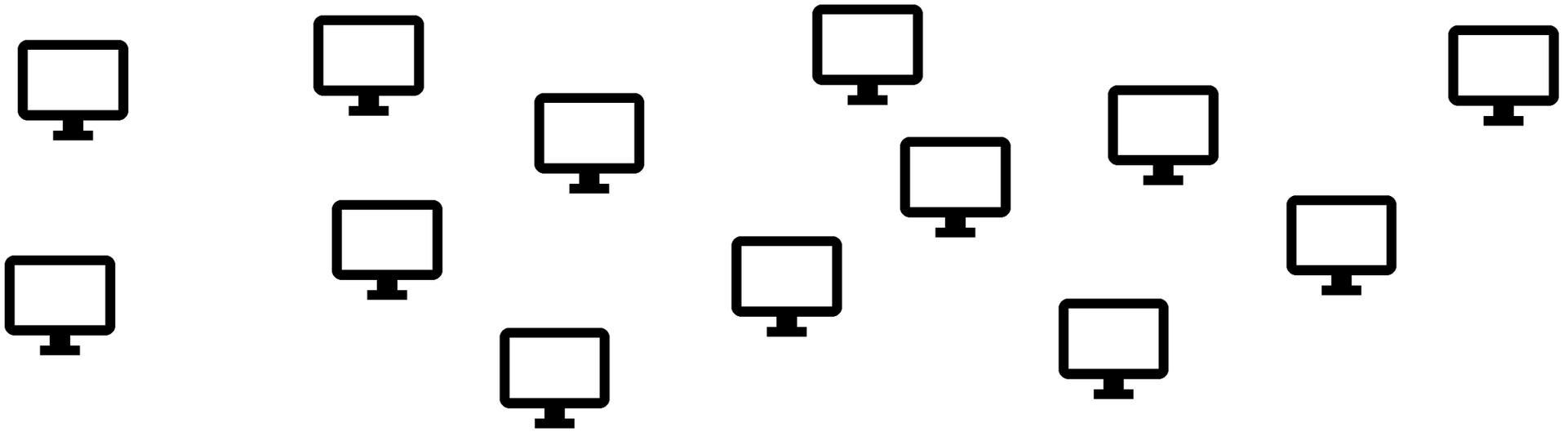
Computers vs. Computer Networks

Theory of Computing

- *Which tasks can be solved efficiently with a **(single) computer**?*

Theory of Distributed Computing (one aspect)

- *Which tasks can be solved efficiently in **large computer networks**?*



The Internet

Sensor Networks

Sensor Networks

Important problems in computer networks

routing problems

distributed scheduling

decentralized data processing

communication eff. algorithms

spreading of rumors

spreading of diseases

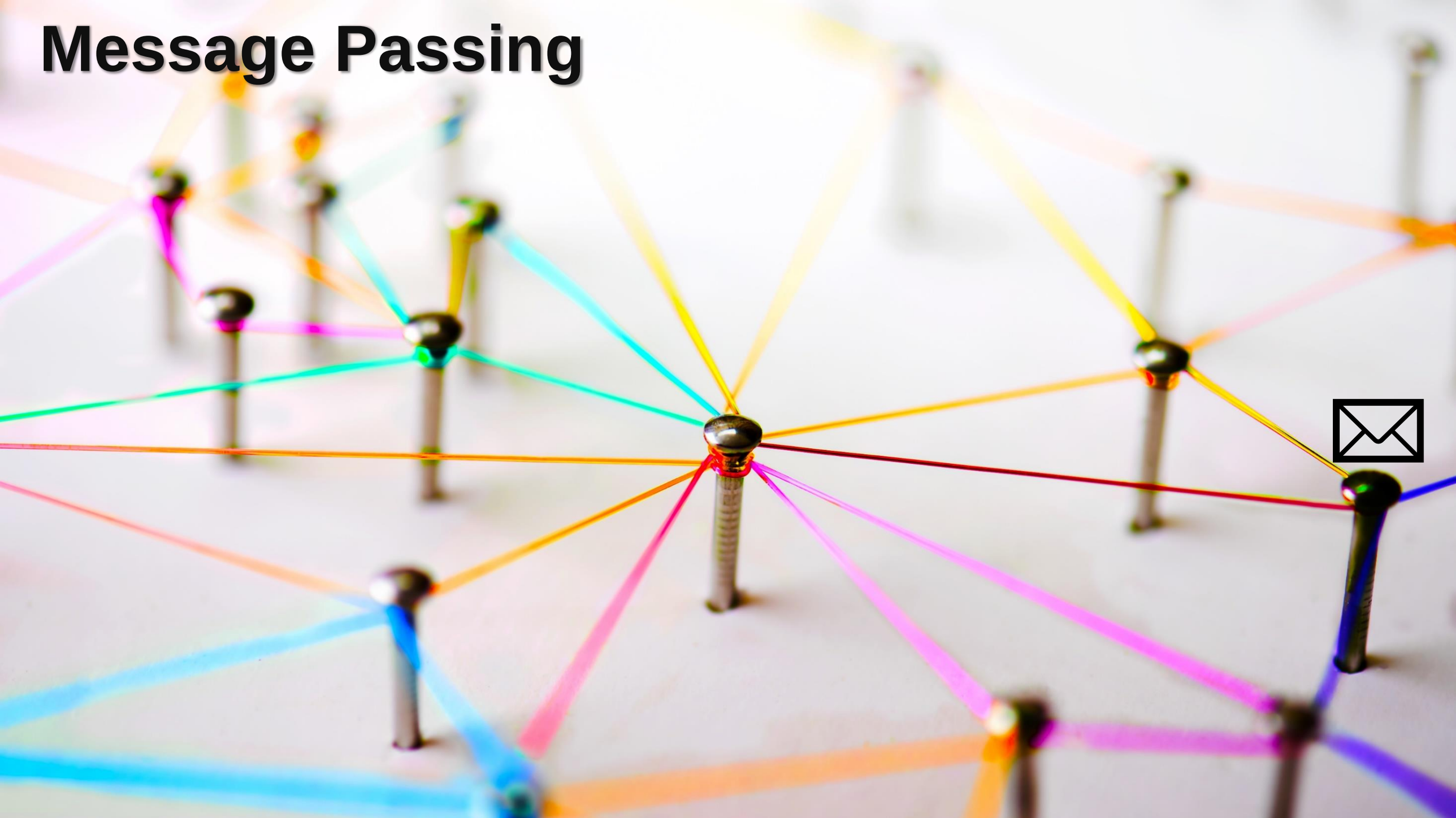
biological algorithms

dynamic graphs

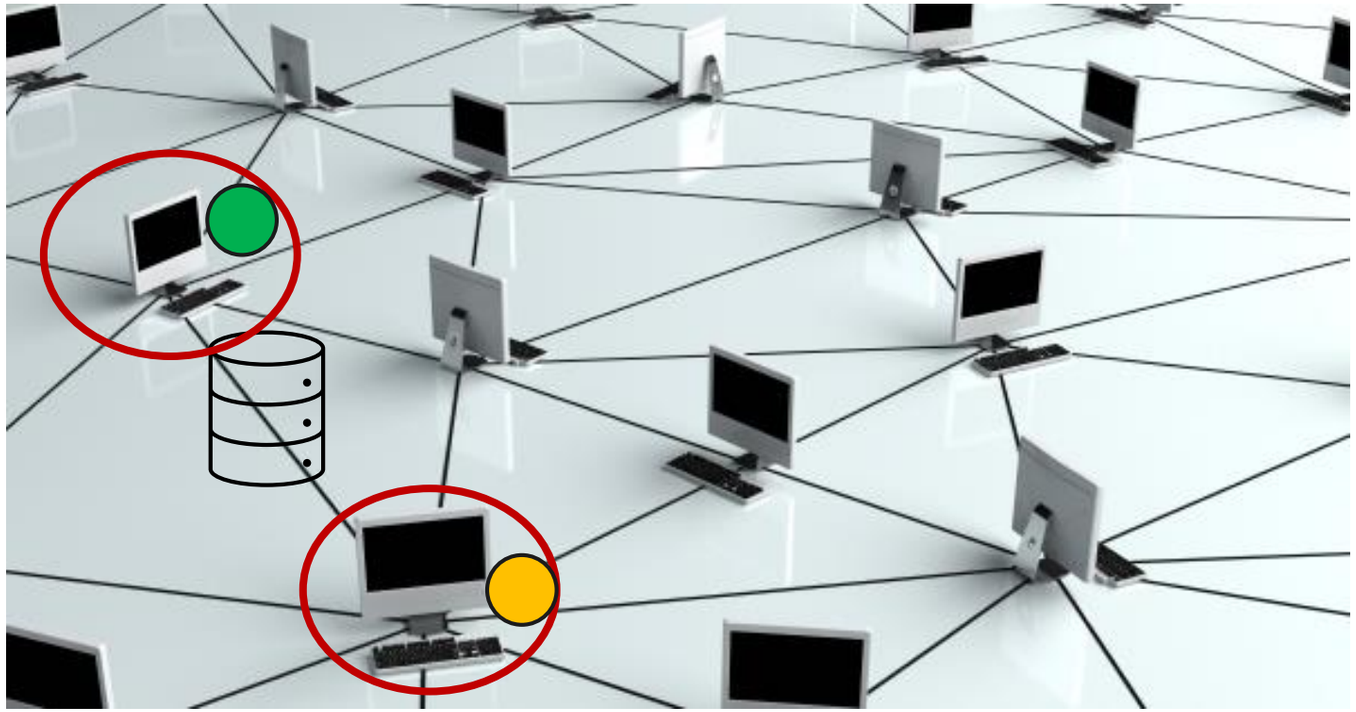
Today: Message Passing Algorithms
in Static Communication Networks

Message Passing

Message Passing



Message Passing on a Graph



Symmetry breaking

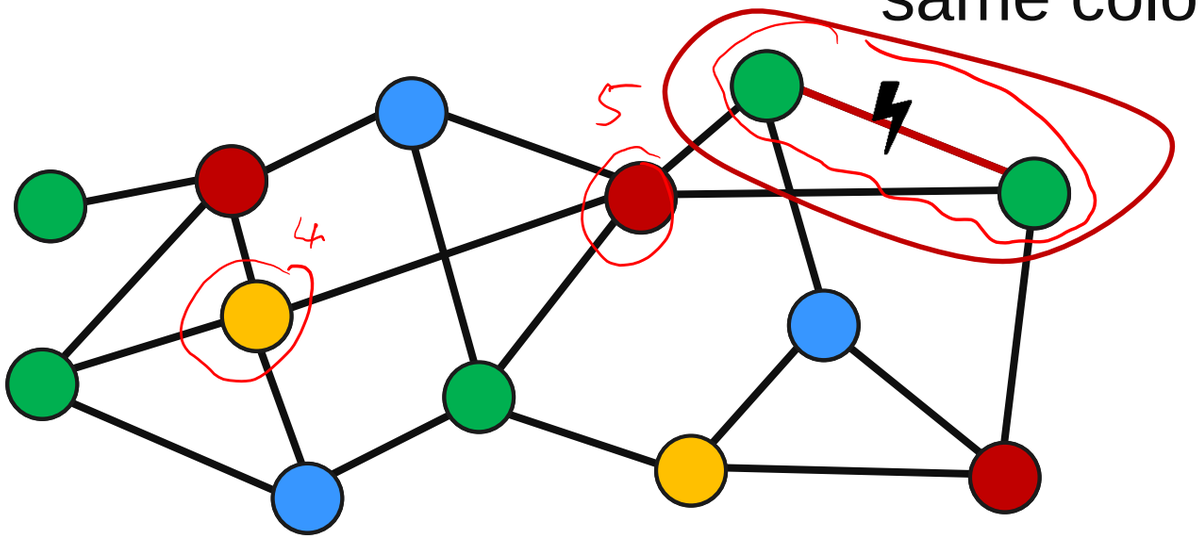
fundamental primitive

e.g., two computers need mutual exclusive access to a shared resource

need to assign time slots (= colors)

Symmetry Breaking: Vertex Coloring

same color not allowed!



$\{1, \ldots, \Delta+1\}$

#neighbors $\leq \Delta$

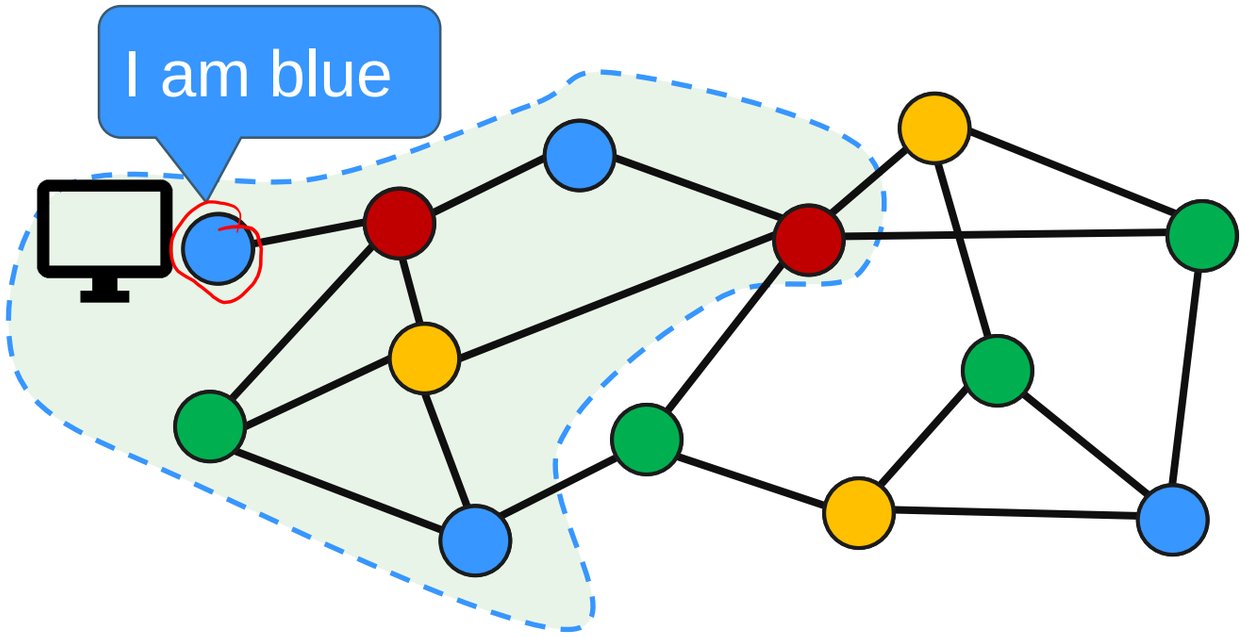
Goal: Color the graph

- adjacent nodes **different colors**
- prime symmetry breaking problem
- NP-complete

Use $\Delta + 1$ colors!

- Δ is the graph's maximum degree
- sequential greedy algorithm ✅

A computer in a big big world



Start

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

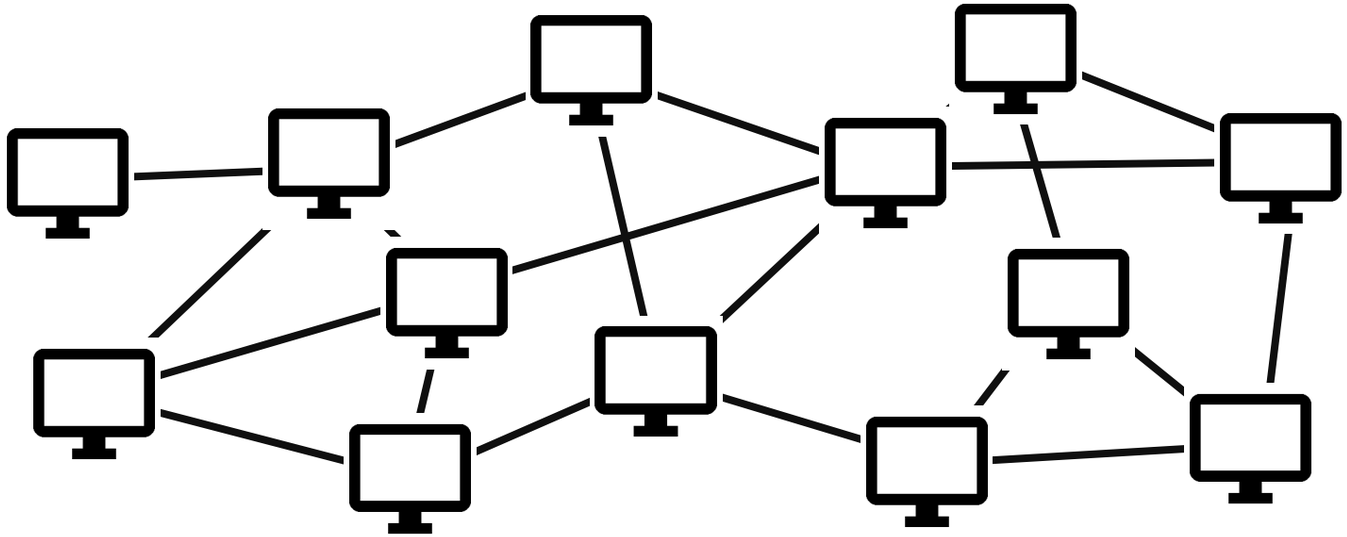
End

Each node knows its part of the output (e.g., its color)

inside view

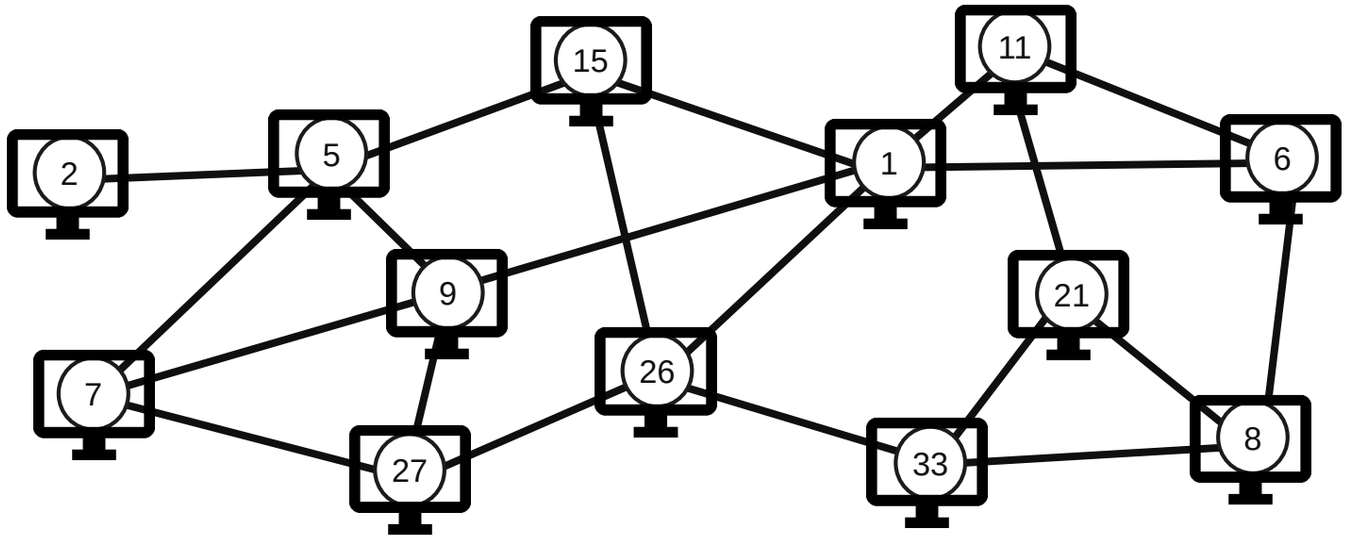
LOCAL Model [Linial; FOCS '87]

Communication Network:



LOCAL Model [Linial; FOCS '87]

Communication Network: $G = (V, E), n = |V|$

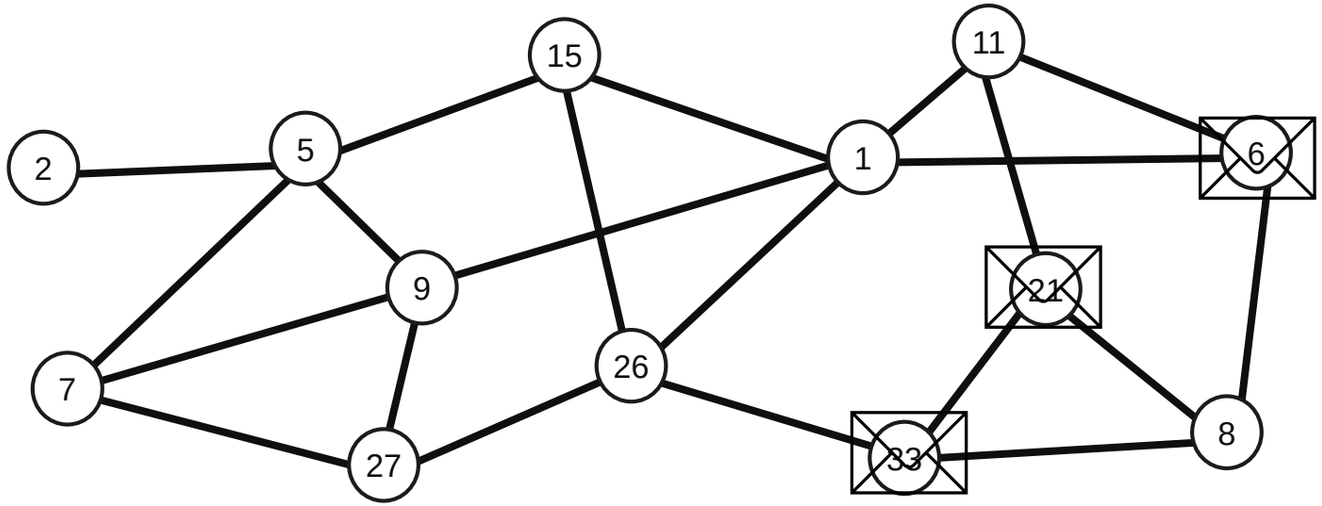


unique IDs,
e.g., IP-addresse

LOCAL Model [Linial; FOCS '87]

Communication Network: $G = (V, E), n = |V|$

unique IDs,
e.g., IP-addresse



Discrete synchronous rounds:

- local computations
- exchange messages with <u>all</u> neighbors

(computations are unbounded)

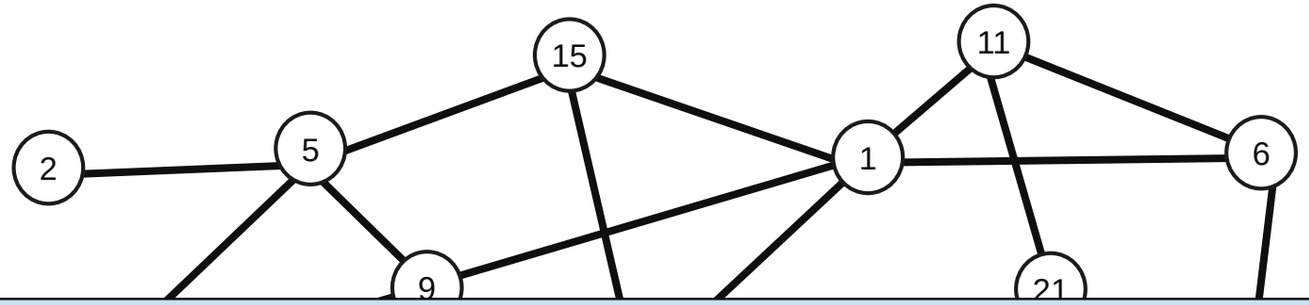
time complexity = number of rounds

abstracts away:

- asynchrony
- failures
- local computations

LOCAL Model [Linial; FOCS '87]

Communication Network: $G = (V, E), n = |V|$



unique IDs,
e.g., IP-addresse

! All nodes act simultaneously !
(all node start in the same round 0)

- local computations
- exchange messages with all neighbors

(computations are unbounded)

time complexity = number of rounds

abstracts away:

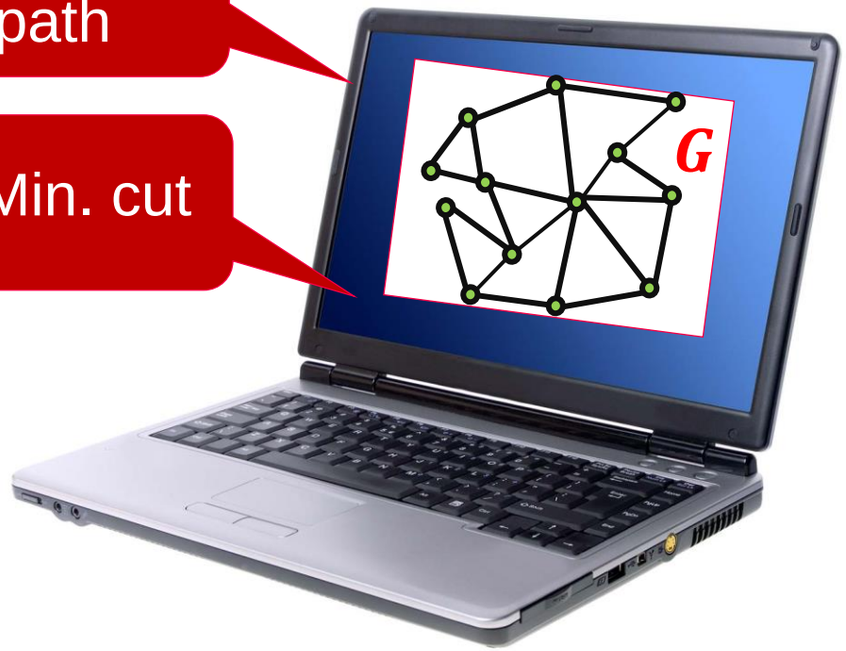
- asynchrony
- failures
- local computations

Distributed vs. Centralized Algorithms

Centralized Setting

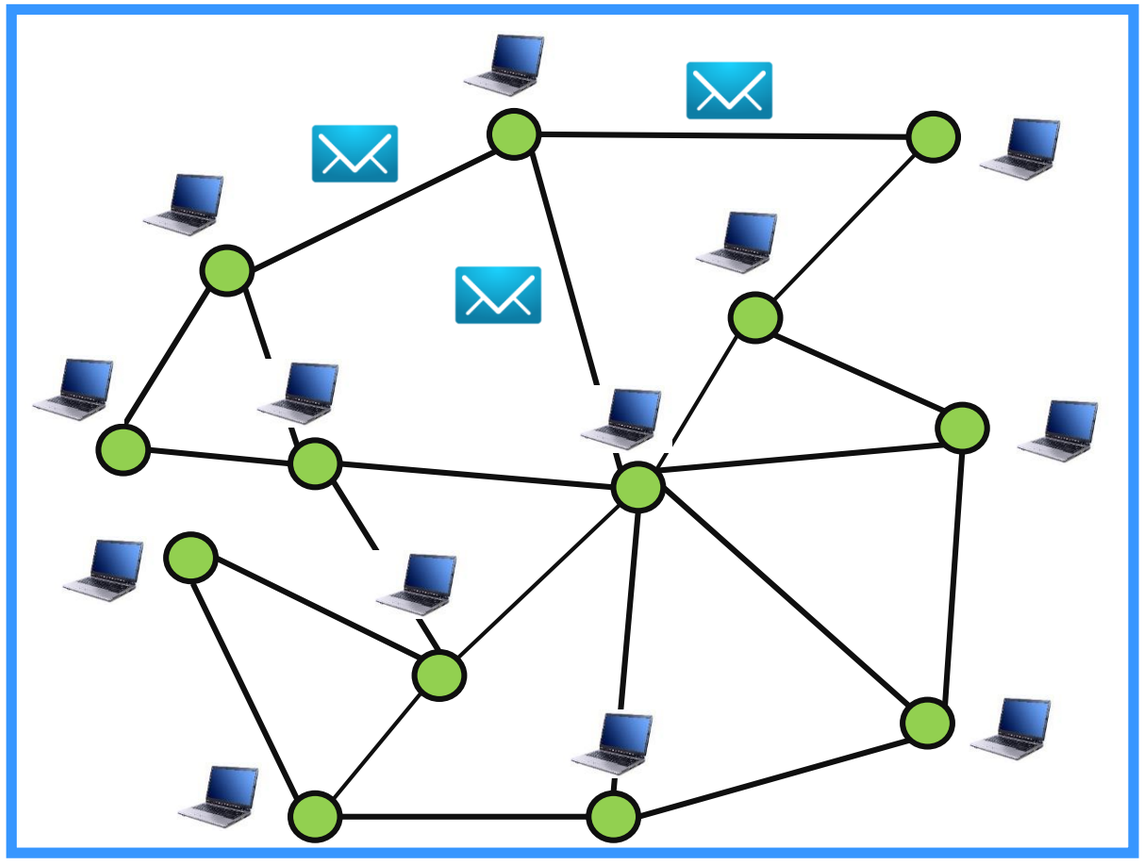
Shortest path

Min. cut



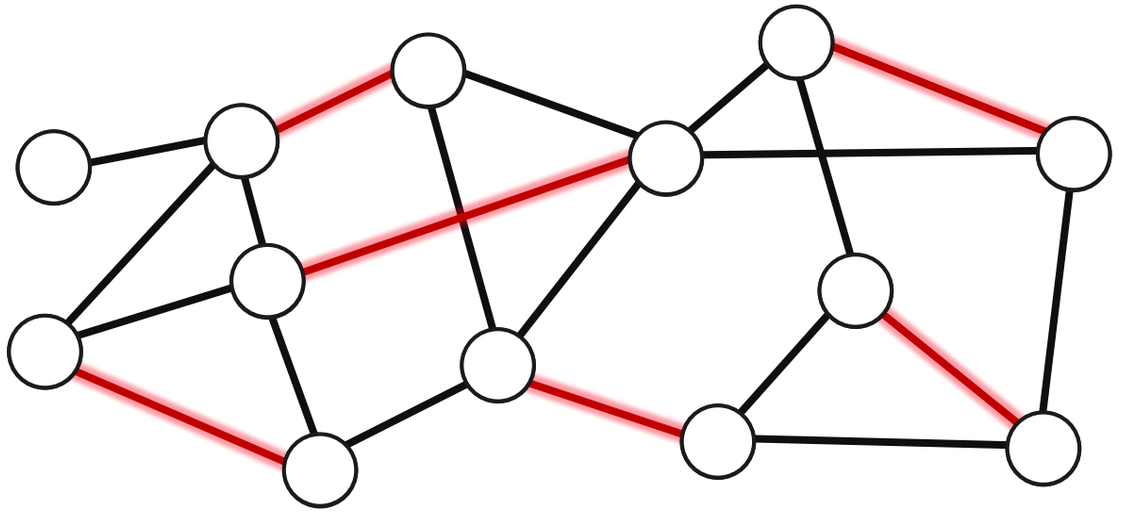
Computation time

Distributed Setting



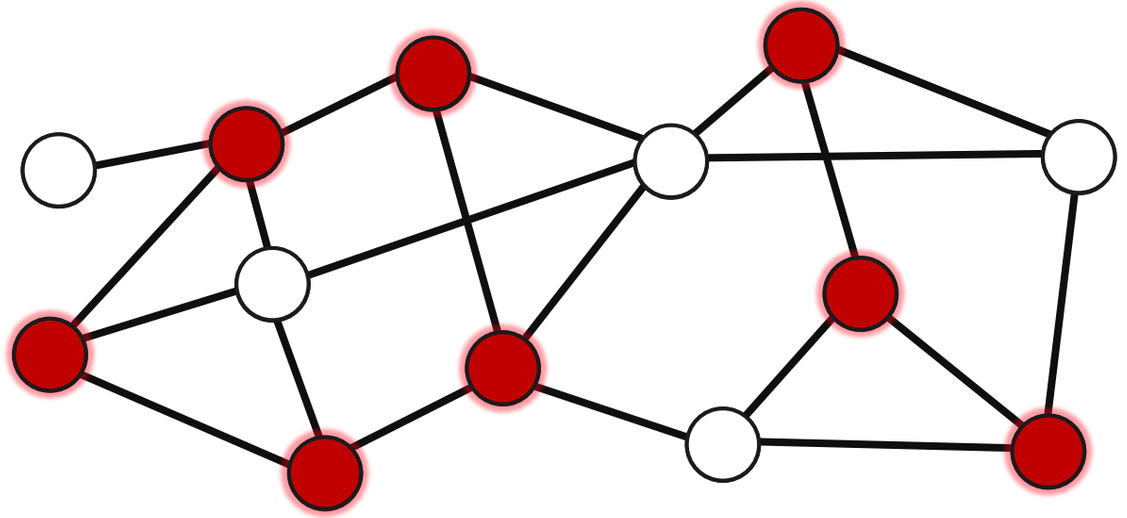
Communication rounds

You name it: Maximal Matching



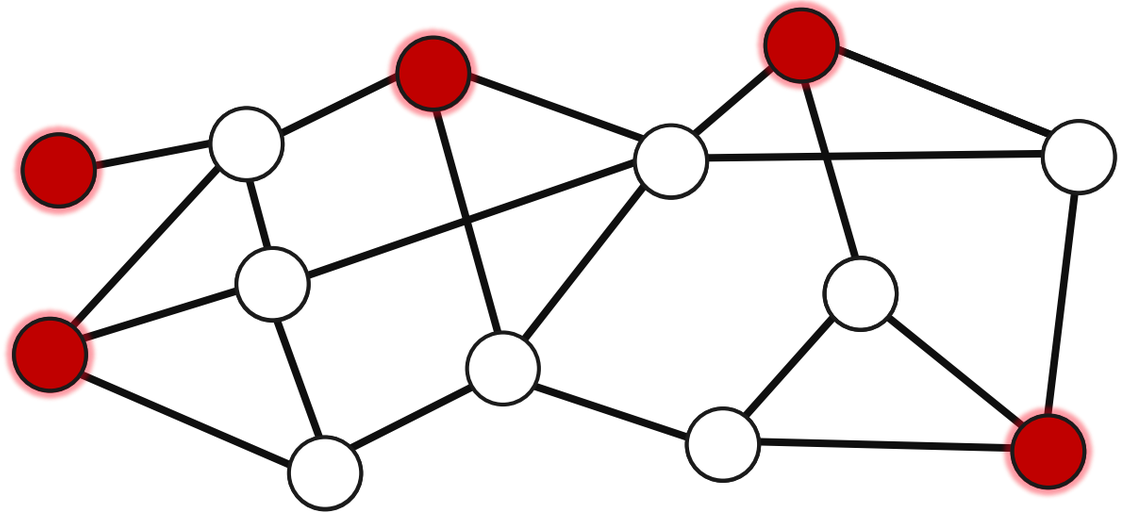
Output: Each node knows which of its incident edges are part of the matching.

You name it: Vertex Cover



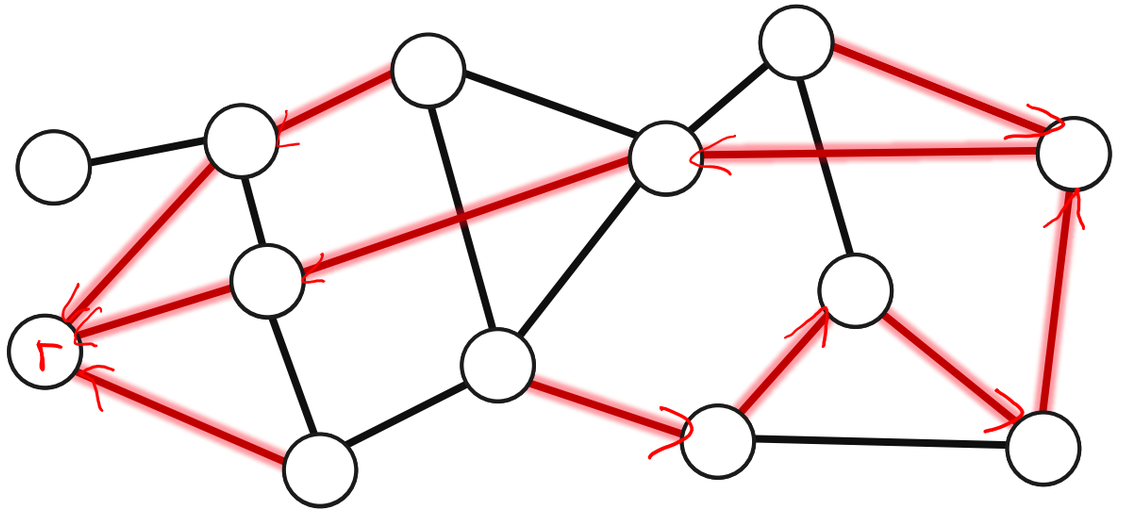
Output: Each node knows whether it is contained in the vertex cover.

You name it: Maximal Independent Set



Output: Each node knows whether it is contained in the independent set.

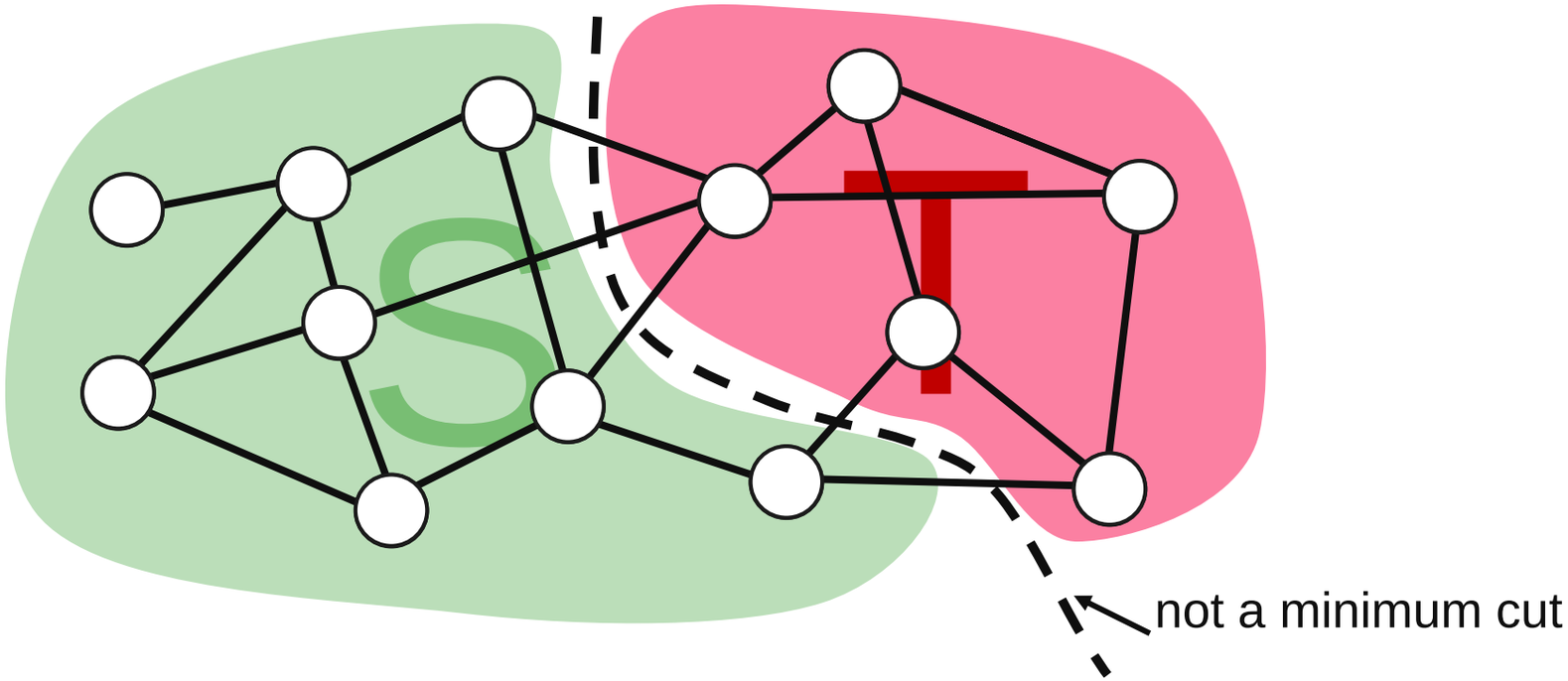
You name it: Spanning Tree



Output: Each node knows which incident edges are part of the spanning tree.

(extension: rooted spanning tree)

You name it: Minimum Cut



Output: Each node knows whether it is part of S or T.

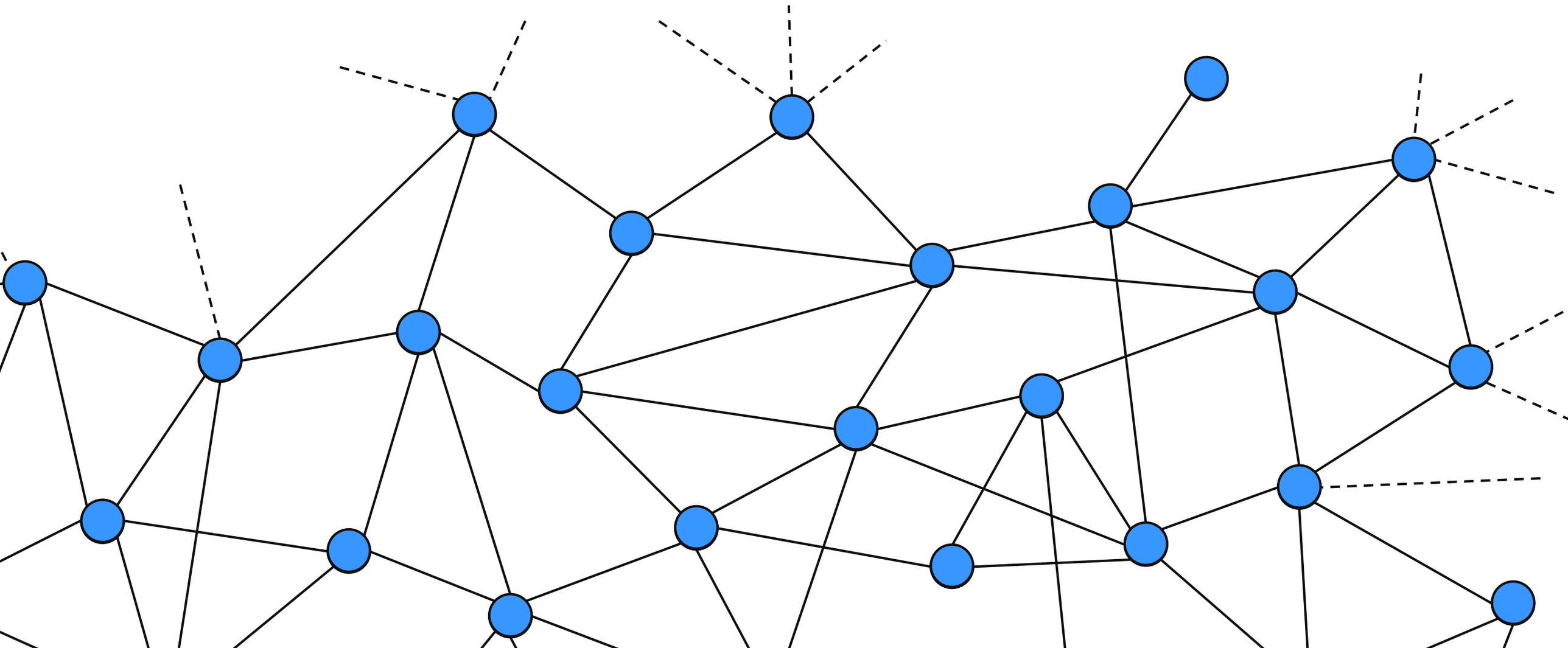
Many more, you name it!

Is distributed coloring difficult?

Try 1: Everyone picks the **smallest free color** at the same time

Is distributed coloring difficult?

Try 1: Everyone picks the **smallest free color** at the same time



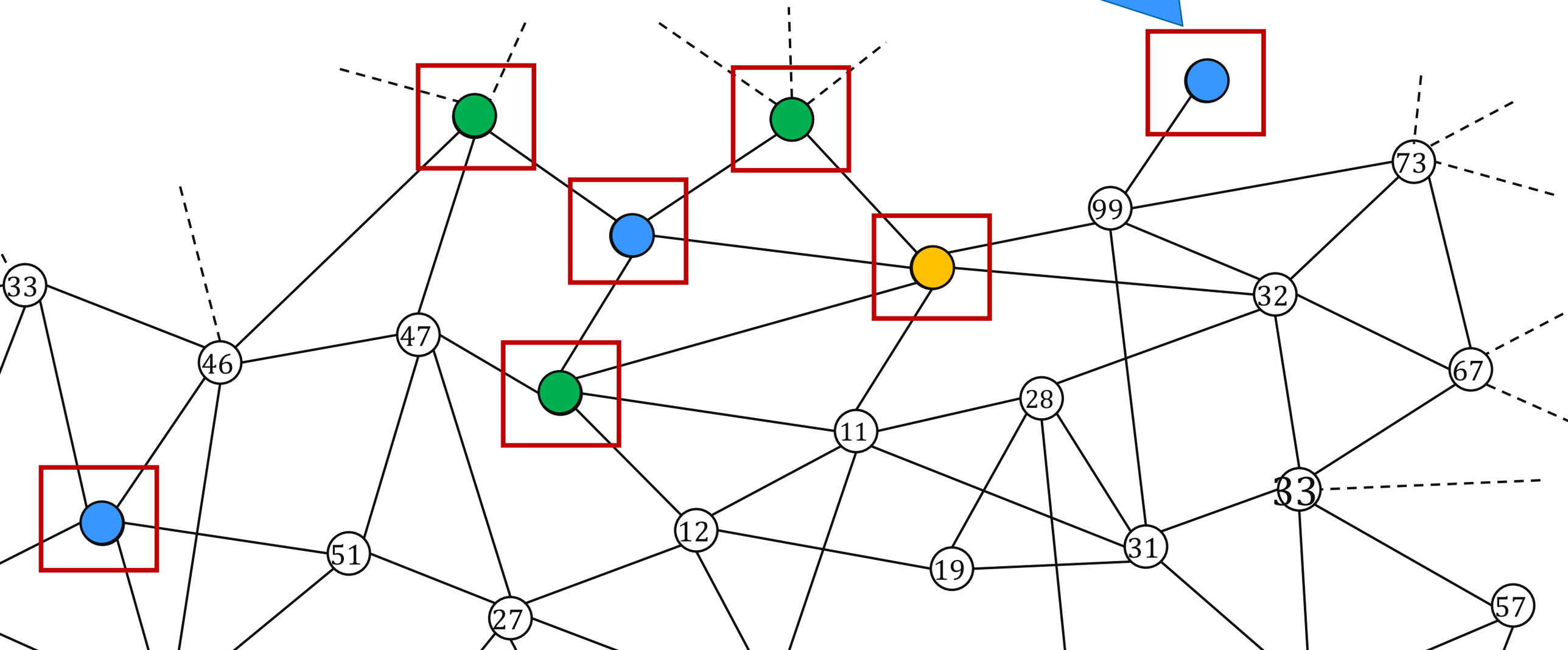
Try 2: Use your IP-address

Alg: Color local minima (among uncolored nodes)

Try 2: Use your IP-address

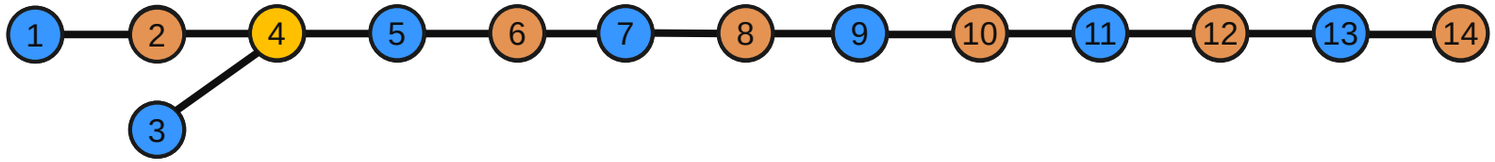
Alg: Color local minima (among uncolored node

I color myself because I have no uncolored neighbors with a **smaller IP-address**



Try 2 is very slow

Alg: Color local minima (among uncolored nodes)



Greedy performs **badly:** takes $\Omega(n)$ rounds

Far from **efficient = $\mathbf{poly\log n \ll n}$** rounds.

Can we color $O(1)$ rounds?

Linial's Seminal Results

Linial's LB: Coloring rings with $\boldsymbol{O(1)}$ **colors** requires $\Omega(\log^* n)$ rounds

[Linial; FOCS '87]

- taught in every distributed graph algorithms course (1 hour)

---

extremely small

$\log^* n$: Iterated logarithm

$\log^*($ #atoms universe $) = 5$

$\log^* n = \min\{i \mid \log^{(i)} n \leq 2\}$

$\log^{(1)} n = \log n$

$\log^{(i+1)} n = \log(\log^{(i)} n)$

We will see this algorithm later

Linial's algorithm: $O(\log^* n)$ rounds for $\boldsymbol{O(\Delta^2)}$**-coloring**.

[Linial; FOCS '87]

Pro: Tight in terms of the LB

Downside: Many colors: ($\Delta^2 \gg \Delta + 1$),

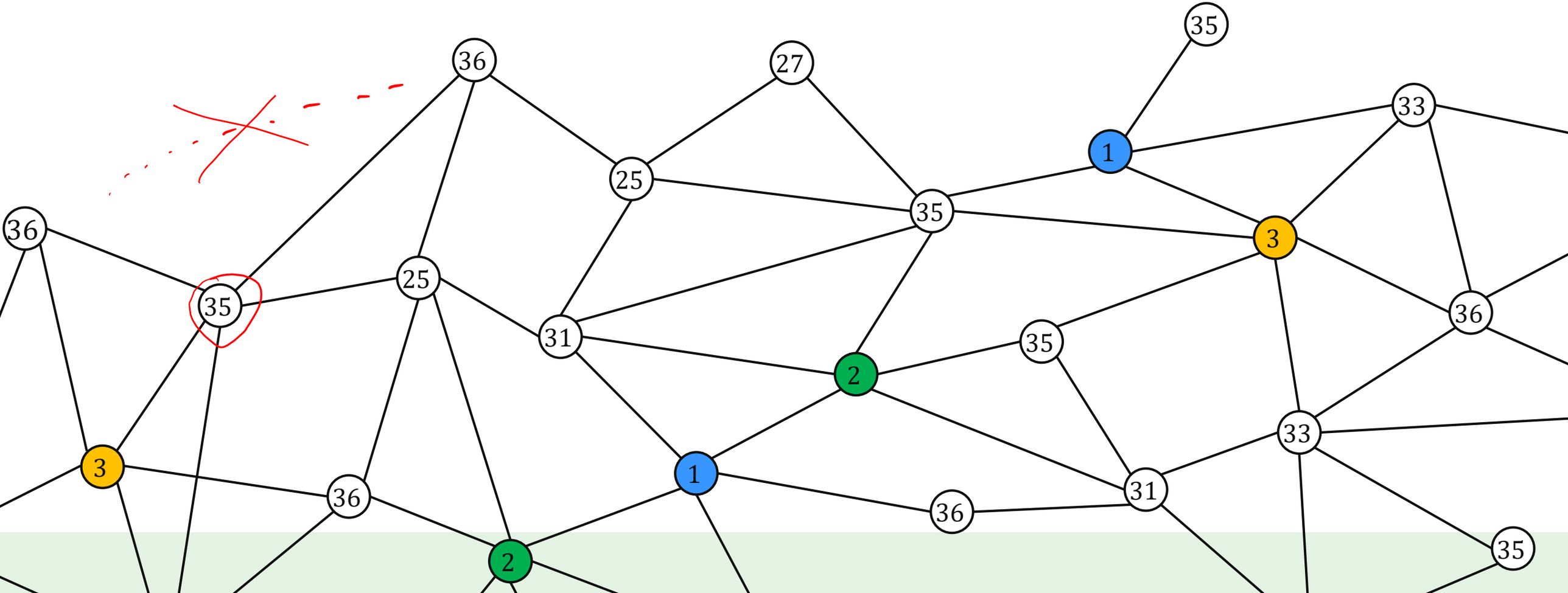
Goal (next few slides):
Color reduction $O(\Delta^2) \to \Delta + 1$

Color Reduction: Naïve Algorithm

Input: 36-coloring

Goal to use colors: 1 2 3 4 5 6 7 ($\Delta = 6$)

Algorithm: reduce one color per round

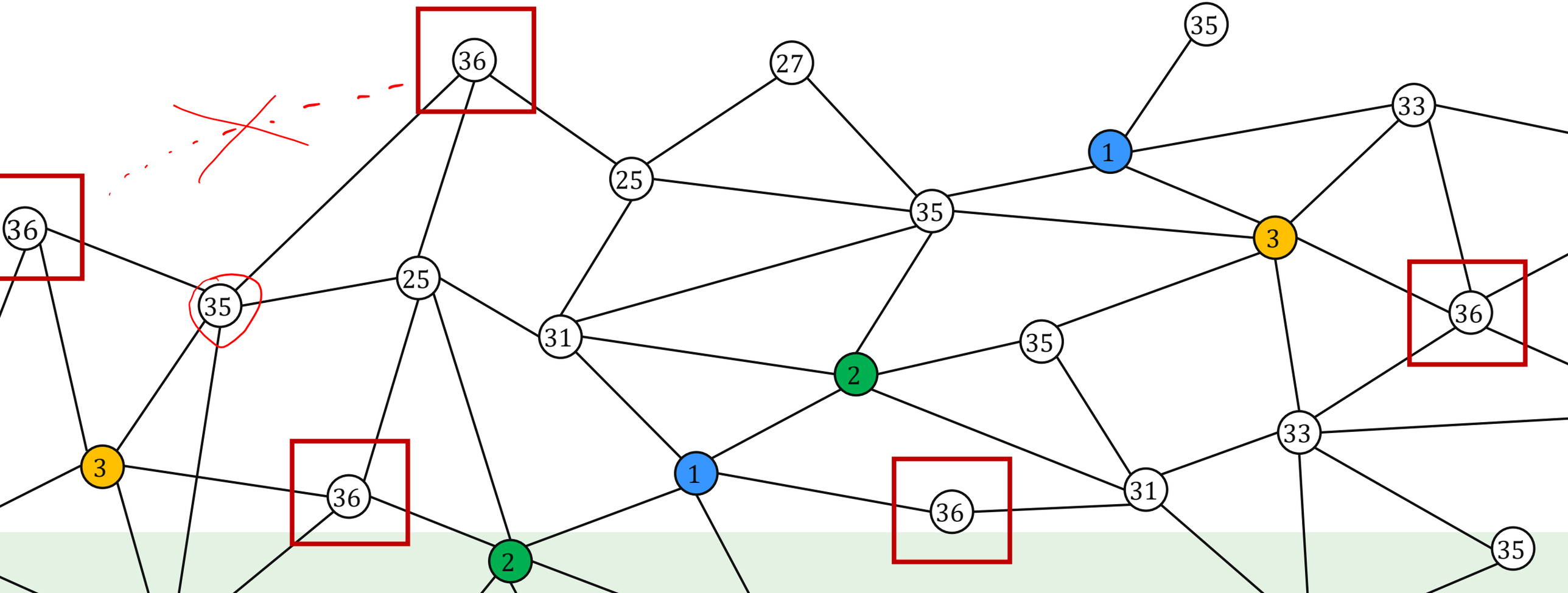


Color Reduction: Naïve Algorithm

Input: 36-coloring

Goal to use colors: 1 2 3 4 5 6 7 ($\Delta = 6$)

Algorithm: reduce one color per round

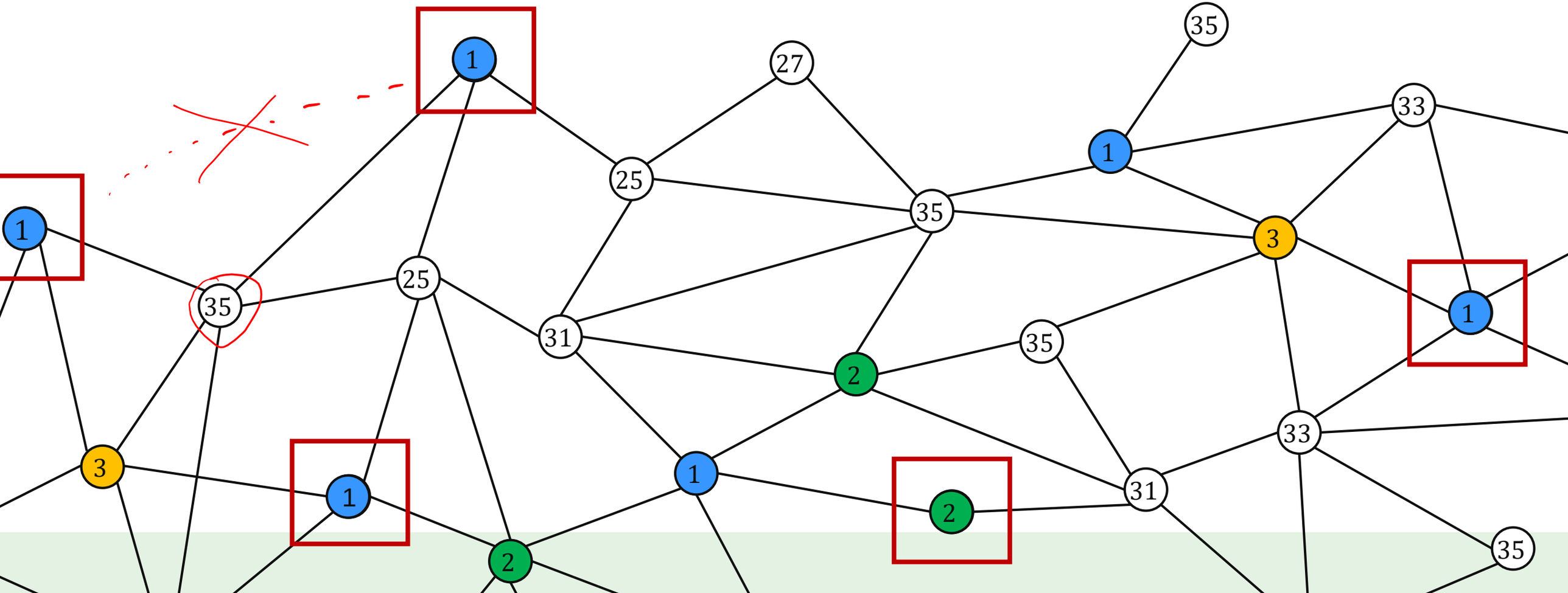


Color Reduction: Naïve Algorithm

Input: 36-coloring

Goal to use colors: 1 2 3 4 5 6 7 ($\Delta = 6$)

Algorithm: reduce one color per round

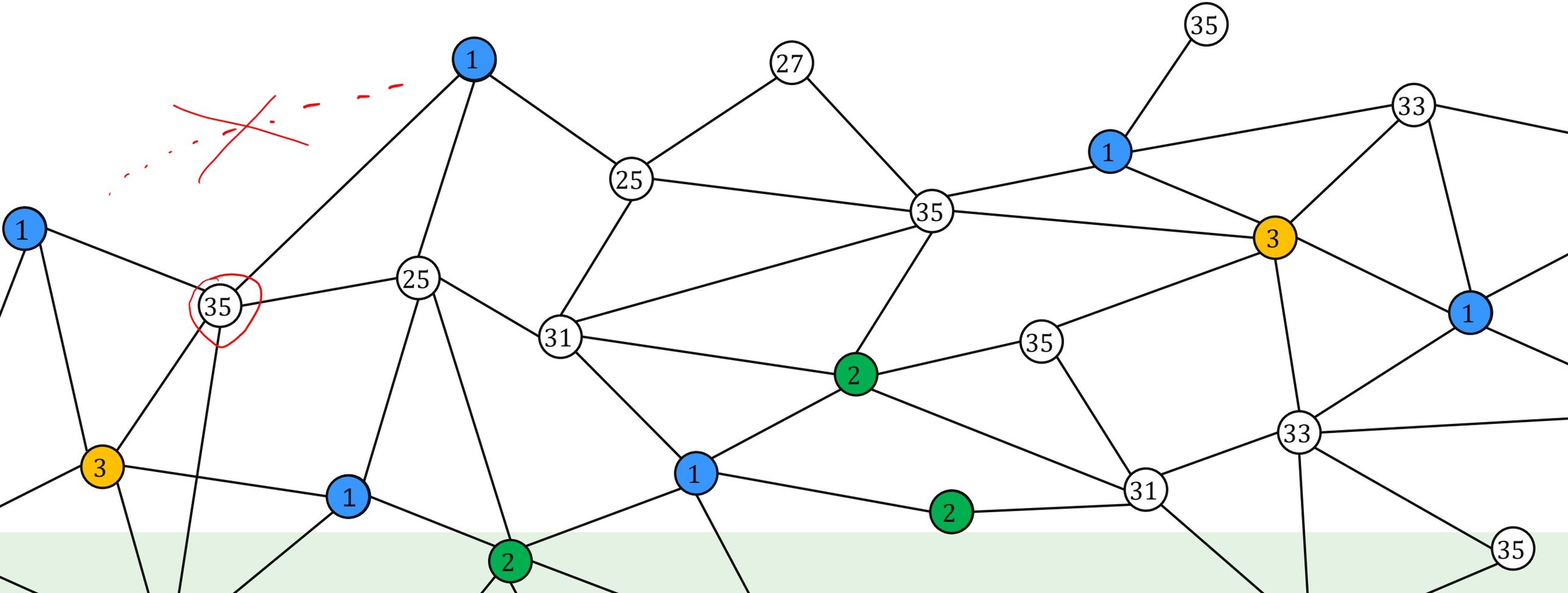


Color Reduction: Naïve Algorithm

Input: 36-coloring

Goal to use colors: 1 2 3 4 5 6 7 ($\Delta = 6$)

Algorithm: reduce one color per round

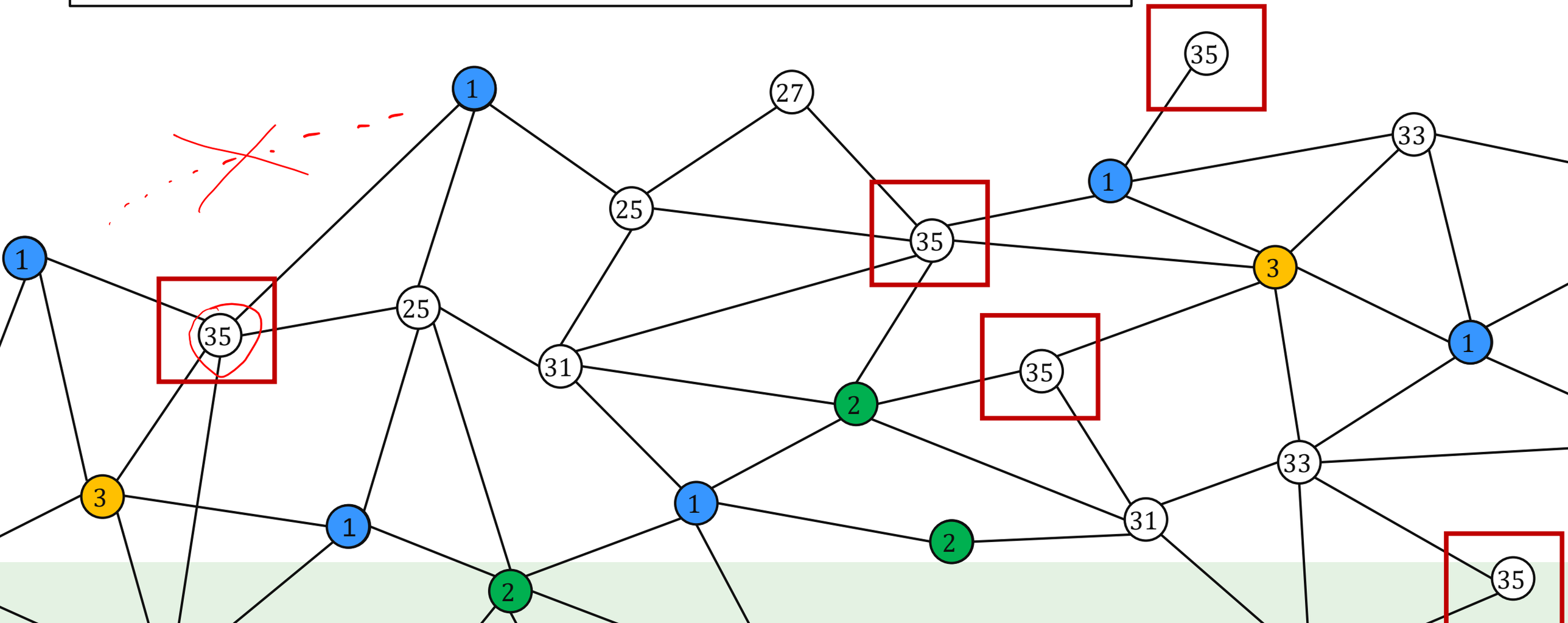


Color Reduction: Naïve Algorithm

Input: 36-coloring

Goal to use colors: 1 2 3 4 5 6 7 ($\Delta = 6$)

Algorithm: reduce one color per round

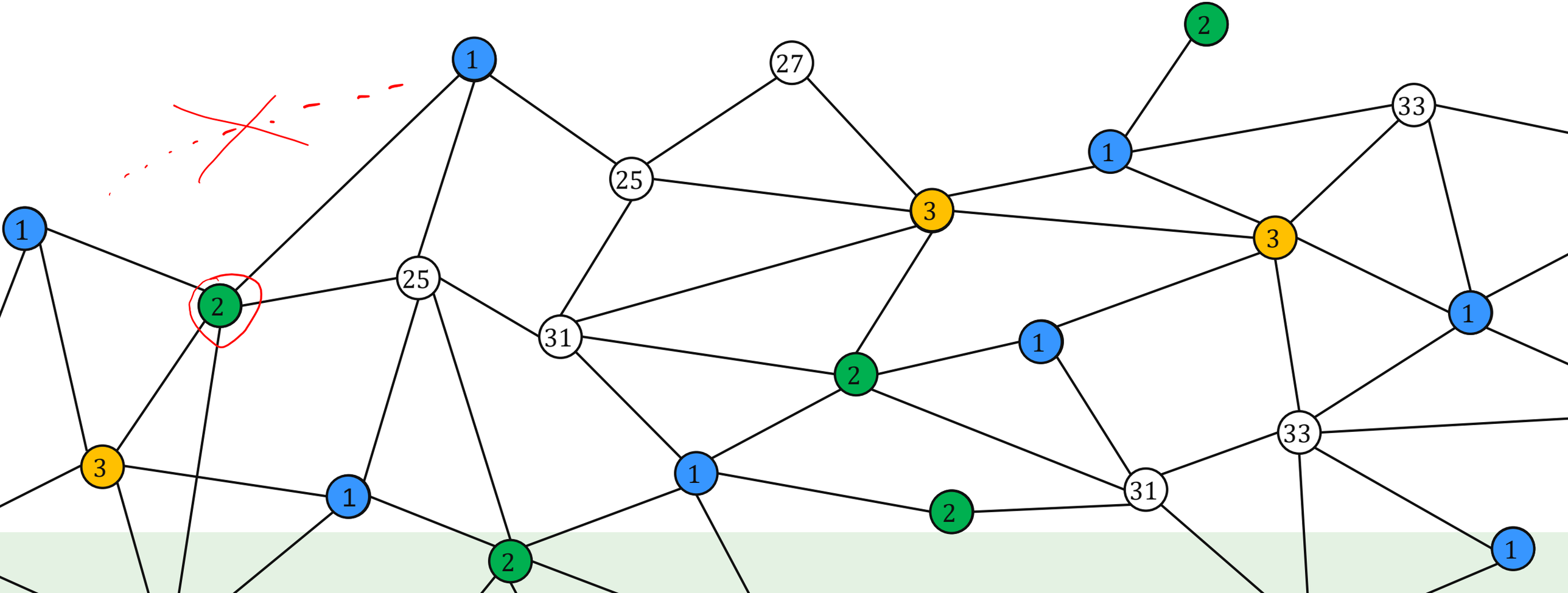


Color Reduction: Naïve Algorithm

Input: 36-coloring

Goal to use colors: 1 2 3 4 5 6 7 ($\Delta = 6$)

Algorithm: reduce one color per round

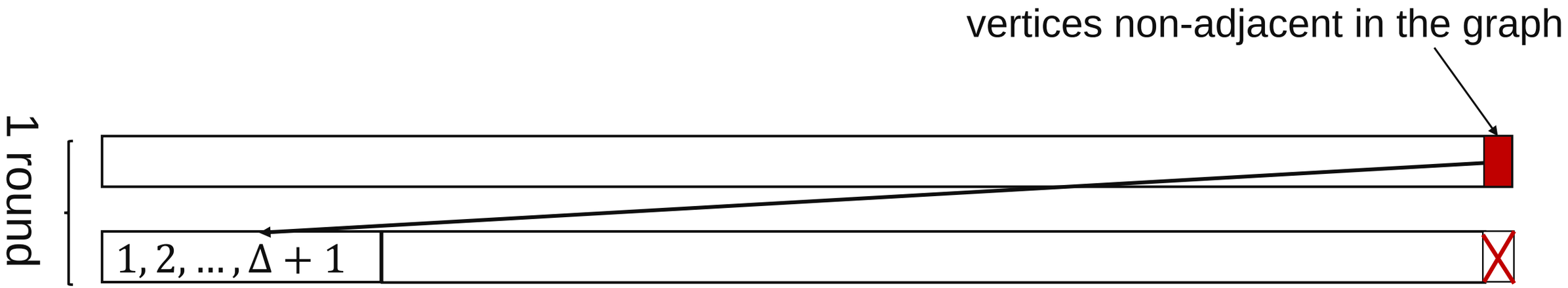


Simple Color Reduction

$(\Delta+1)$-coloring: $O(\Delta^2)+\log^* n$ rounds.

[Linial; FOCS '87]

$$\text{Linial} \to \Delta^2 \to (\Delta^2-1) \to (\Delta^2-2) \to \cdots \to (\Delta+2) \to (\Delta+1)$$



Linial's Algorithm

$O(\Delta^2)$-coloring in $O(\log^* n)$ rounds

Linial's Color Reduction

Theorem (Linial '87):

Given an m-coloring, we can compute a $\left(\mathbf{100\Delta^2 \cdot \log m}\right)$**-coloring** in 1 round.

ID-assignment is a coloring |ID-SPACE|=N colors

$$N\text{-coloring} \to O(\Delta^2 \log N) \to O(\Delta^2(\log\Delta + \log\log N) \to \cdots \to O(\Delta^2 \log\Delta) \to^* O(\Delta^2)$$

$\Delta^3 \to^* O(\Delta^2)$

$O(\log^* N)$ iterations/rounds

(won't be discussed in detail)

Linial's Color Reduction

Theorem (Linial '87):
Given a Δ^3-coloring, we can compute a $(\mathbf{100\Delta^2})$**-coloring** in 1 round.

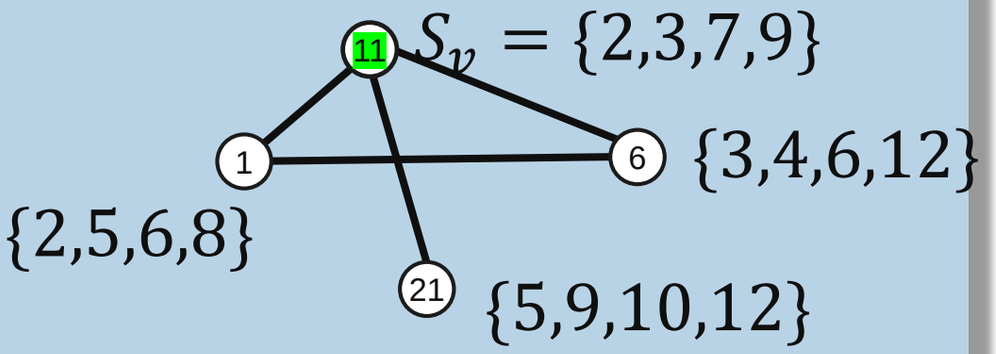
Linial's Color Reduction

Theorem (Linial '87):
Given a Δ^3-coloring, we can compute a $(\mathbf{100\Delta^2})$**-coloring** in 1 round.

Algorithm (1 round):

- node v selects a **candidate color set** $S_v \subseteq [\mathbf{100\Delta^2}]$
- v **picks** color in

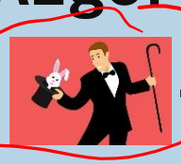
$$\boldsymbol{S_v \setminus \bigcup_{w \in N(v)} S_w}$$



Linial's Color Reduction

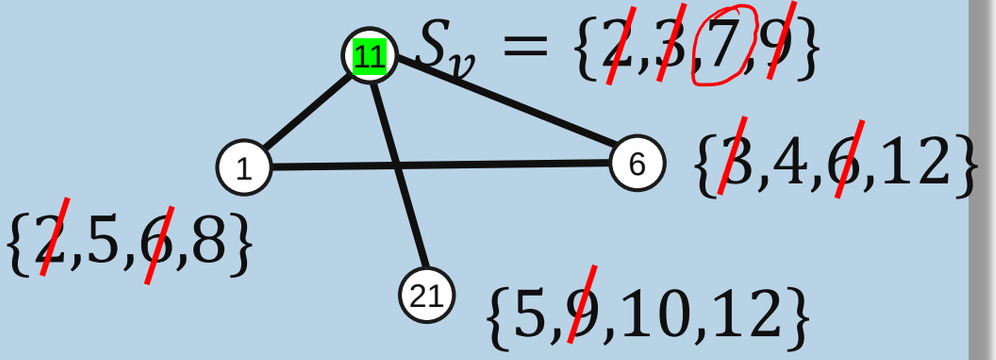
Theorem (Linial '87):
Given a Δ^3-coloring, we can compute a $(\mathbf{100\Delta^2})$**-coloring** in 1 round.

Algorithm (1 round):



- node v selects a **candidate color set** $S_v \subseteq [\mathbf{100\Delta^2}]$
- v **picks** color in

$$\boldsymbol{S_v} \setminus \bigcup_{w \in N(v)} \boldsymbol{S_w} \neq \emptyset$$



Low intersecting sets via Polynomials

Bertrands Postulate $(x, 2x]$

Fix prime $q \in [10 \cdot \Delta, 20\Delta]$. Polynomials $p_0, p_1, p_2, \ldots$ of degree 3 over $\mathbb{F}_q$

(we assume just $q = 10\Delta$)

$$p\colon \mathbb{F}_q \to \mathbb{F}_q, \qquad x \mapsto p(x)$$

$\mathbb{F}_7 = \{0,1,2,3,4,5,6\}$ $(+,\cdot)$ modulo 7 $q=7$

Node v with **input color** $\boldsymbol{i} \in \boldsymbol{\Delta^3}$ picks polynomial p_i (#polynomials=$q^{d+1} \geq \Delta^3$)

$(10\Delta)^4 \geq \Delta^3$

$p(x) = 5x^3 + 2x^2 + 3x + 1$

$p(x) = ax^3 + bx^2 + cx + d$

$a, b, c, d \in \{0, \ldots, q-1\}$

q^4

Low intersecting sets

$$S_v = S_i = \{(x, p_i(x)) \mid x \in \mathbb{F}_q\} \subseteq \mathbb{F}_q \times \mathbb{F}_q$$

Example: Low Intersecting Sets

Example:

prime $q = 7$.

$p_v(x) = 6x^3 + 4x^2 + 3 \in \mathbb{F}_q$

Low intersecting sets

$$S_v = S_i = \{(x, p_i(x)) \mid x \in \mathbb{F}_q\} \subseteq \mathbb{F}_q \times \mathbb{F}_q$$

$$S_v = \{(0, p_v(0)), (1, p_v(1)), (2, p_v(2)), (3, p_v(3)), (4, p_v(4)), (5, p_v(5)), (6, p_v(6))\}$$
$$S_v = \{(0,3), (1,13), (2,67), (3,201), (4,451), (5,853), (6,1443)\}$$
$$S_v = \{(0,3), (1,6), (2,4), (3,5), (4,3), (5,6), (6,1)\}$$

mod q = mod 7

- Interpret each tuple (x, y) as an output color
- $|S_v| = q = 7$

output colors $|\mathbb{F}_q| \times |\mathbb{F}_q| = q^2$

Sets have small intersections

Lemma: $|S_i \cap S_j| \leq 3$ for $i \neq j$.

Proof:

For each element $(x, y) \in S_i \cap S_j$ we have

$$(x, y) = \big(x, p_i(x)\big) = (x, p_j(x))$$

So, the polynomials p_i and p_j over the prime field $\mathbb{F}_q$ intersect at x.

we have $d = 3$

Two polynomials of degree d over a prime field can intersect in at most d points.

(we won't prove this, see your favorite Algebra class for a proof)

Low intersecting sets via Polynomials

Fix prime $q \in [10 \cdot \Delta, 20\Delta]$. Polynomials $p_0, p_1, p_2, \ldots$ of degree 3 over $\mathbb{F}_q$

(we assume just $q = 10\Delta$)

$$p\colon \mathbb{F}_q \to \mathbb{F}_q, \qquad x \mapsto p(x)$$

Node v with **input color $i \in \Delta^3$** picks polynomial p_i (#polynomials=$q^{d+1} \geq \Delta^3$)

Low intersecting sets

$$S_v = S_i = \{(x, p_i(x)) \mid x \in \mathbb{F}_q\} \subseteq \mathbb{F}_q \times \mathbb{F}_q$$

- $|S_i \cap S_j| \leq 3$ for $i \neq j$
- $|S_i| = |\mathbb{F}_q| = 10 \cdot \Delta$

Output color space: $[q^2] = 100\Delta^2$

Can always choose a color

$$\left|S_v \setminus \bigcup S_w\right| \geq |S_v| - 3 \cdot \Delta > 0$$

$10\Delta - 3\Delta$

Δ = #neighbors

Input coloring provides enough symmetry breaking to assign distinct polynomials to neighboring nodes

Low intersecting sets via Polynomials

Fix prime $q \in [10 \cdot \Delta, 20\Delta]$. Polynomials $p_0, p_1, p_2, \ldots$ of degree 3 over $\mathbb{F}_q$

(we assume just $q = 10\Delta$)

$$p\colon \mathbb{F}_q \to \mathbb{F}_q, \qquad x \mapsto p(x)$$

Node v with **input color $i \in \Delta^3$** picks polynomial p_i (#polynomials=$q^{d+1} \geq \Delta^3$)

Low intersecting sets

$$S_v = S_i = \{(x, p_i(x)) \mid x \in \mathbb{F}_q\} \subseteq \mathbb{F}_q \times \mathbb{F}_q$$

- $|S_i \cap S_j| \leq 3$ for $i \neq j$
- $|S_i| = |\mathbb{F}_q| = 10 \cdot \Delta$

Output color space: $[q^2] = 100\Delta^2$

Can always choose a color

$$\left| S_v \setminus \bigcup S_w \right| \geq |S_v| - 3 \cdot \Delta > 0$$

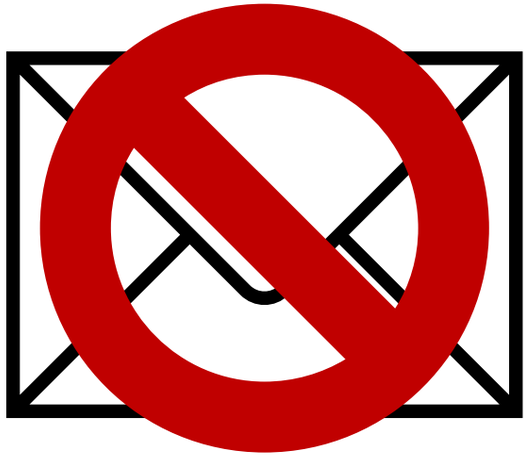
Input coloring provides enough symmetry breaking to assign distinct polynomials to neighboring nodes

The magic: Low intersecting sets w. no comm.

Input coloring provides enough symmetry breaking to assign distinct polynomials to neighboring nodes

Input color $i \mapsto$ polynomial $p_i \mapsto$ set S_i

(adjacent nodes do not receive the same set)



No communication!

requires global knowledge of m, Δ

Additional work (exponential probing, + list coloring extensions) is needed to get rid of these, not discussed today)

Recap: Linial's Color Reduction

Theorem (Linial '87):
Given an m-coloring, we can compute a $\left(\mathbf{100}\Delta^{\mathbf{2}} \cdot \mathbf{log}\, \boldsymbol{m}\right)$**-coloring** in 1 round.

(slightly different, not discussed, similar flavour)

ID-assignment is a coloring |ID-SPACE|=N colors

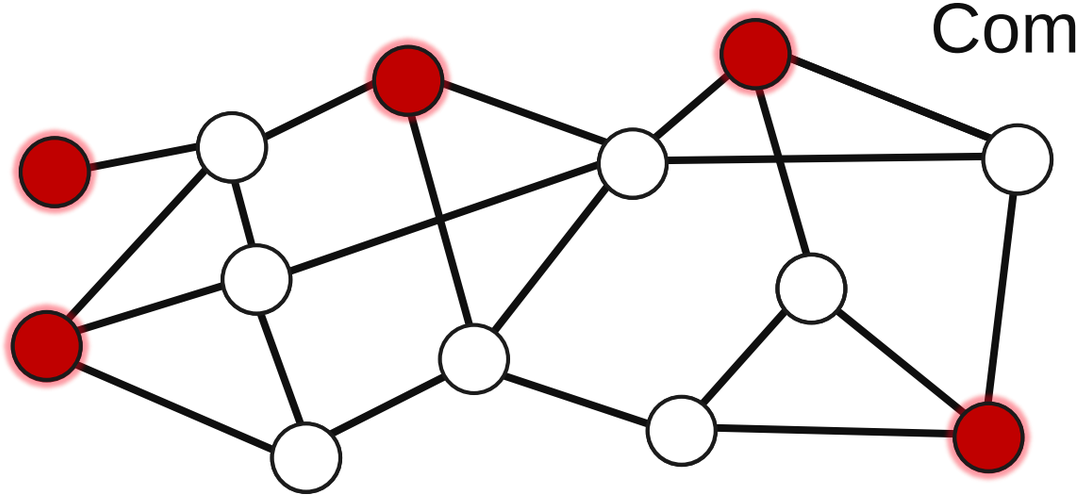
N-coloring$\rightarrow O(\Delta^2 \log N) \rightarrow O(\Delta^2(\log\Delta + \log\log N) \rightarrow \cdots \rightarrow O(\Delta^2 \log\Delta) \rightarrow^* O(\Delta^2)$

$\Delta^3 \rightarrow^* O(\Delta^2)$

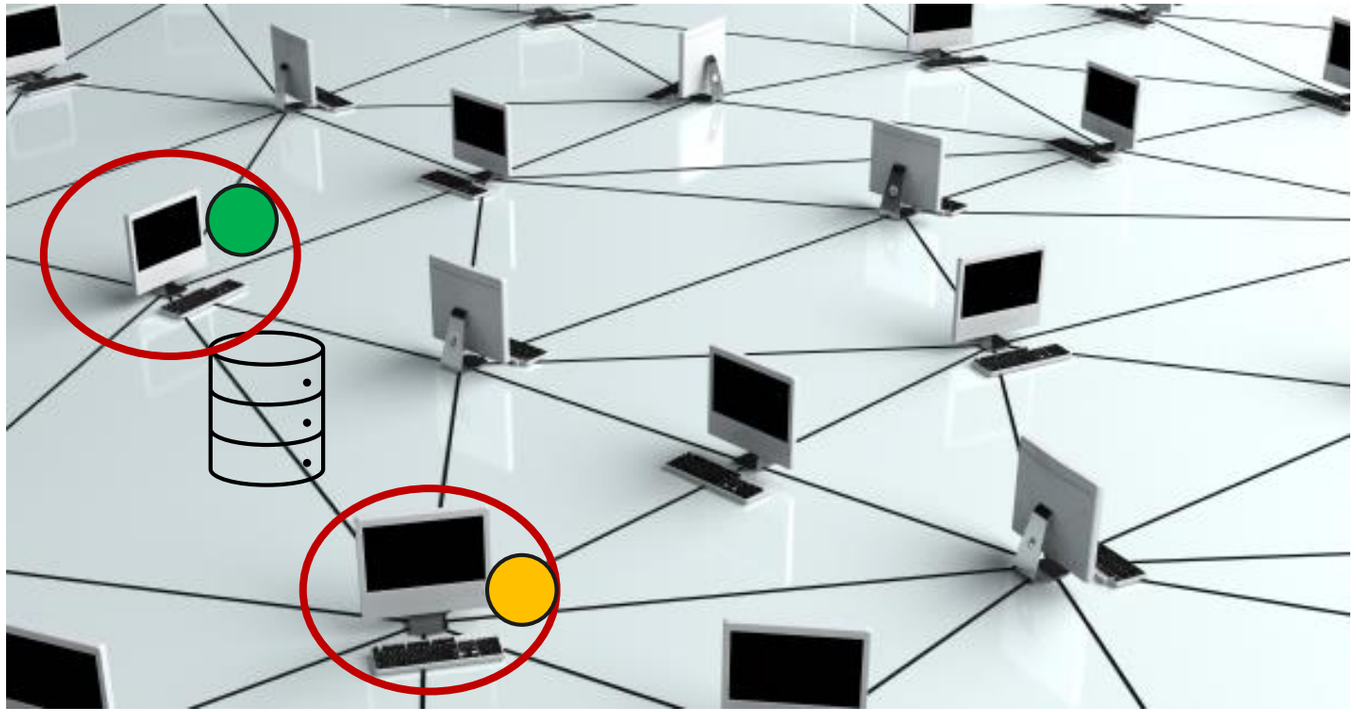
$O(\log^* N)$ iterations/rounds

Application:
Coloring as a Schedule

Computing a Maximal Independent Set (MIS)



Message Passing on a Graph



Symmetry breaking

fundamental primitive

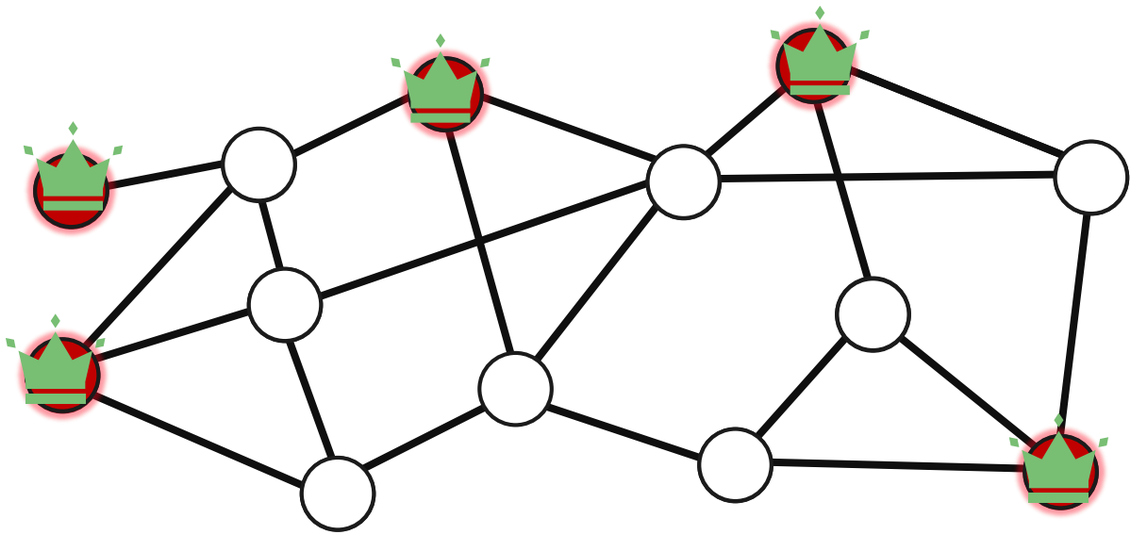
e.g., two computers need mutual exclusive access to a shared resource

need to assign time slots (= colors)

coloring = time slots = schedule

(we already used that in the “simple color reduction” re-coloring)

Maximal Independent Set (MIS)



*A **maximal independent set (MIS)** of a graph $G = (V, E)$ is a subset $S \subseteq V$ of the nodes such that*

- *S is an **independent set** in G (no two vertices in S are adjacent)*
- *S is **maximal**, that is, for any $v \in V \setminus S$, the set $S \cup \{v\}$ is **not** an independent set*

Output (LOCAL): Each node knows whether it is contained in the independent set.

MIS!=MaxIS

- Not to be confused with the **maximum independent set problem**, where the objective is to compute an independent set of **maximum size.**

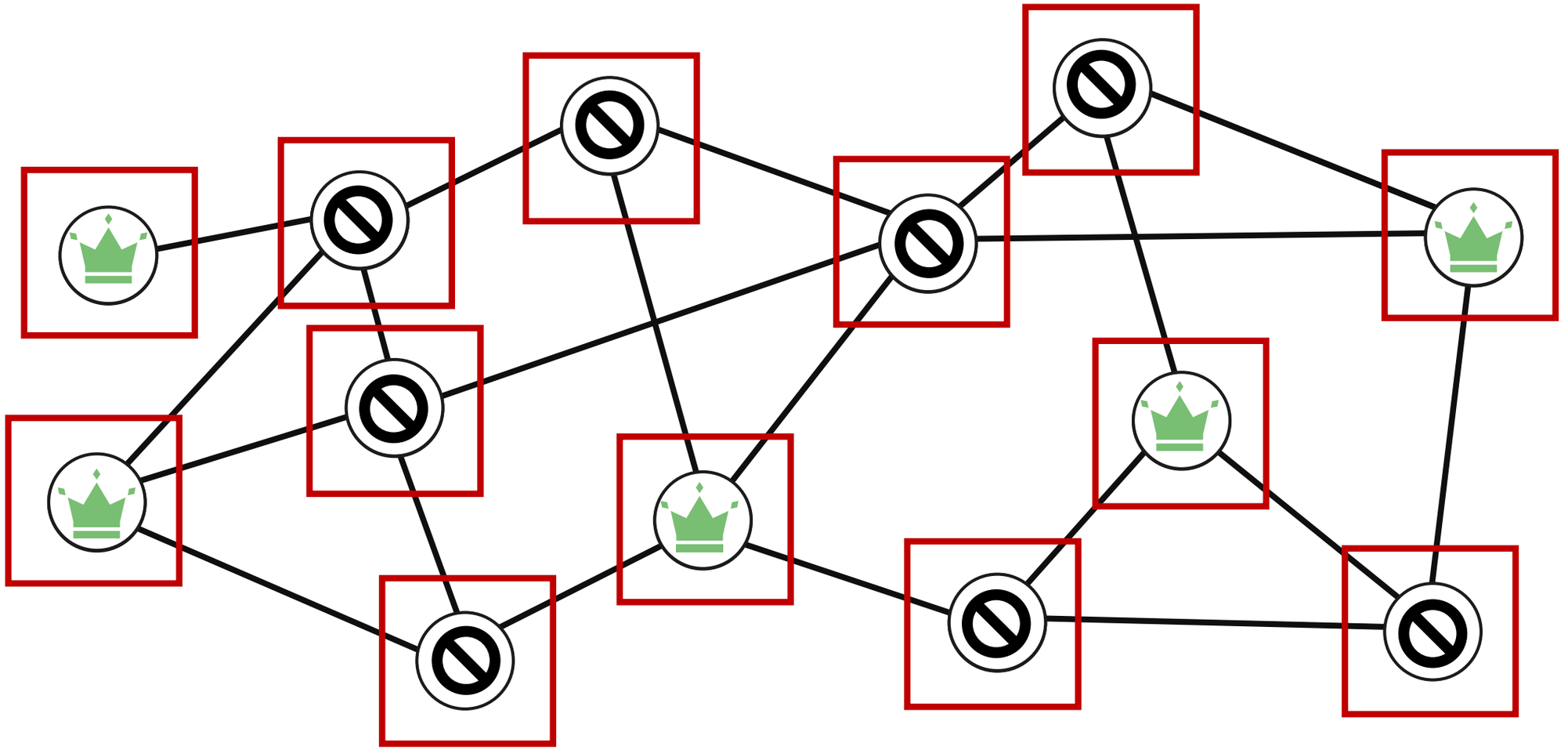
Exercise: Provide an example of a graph G and an MIS of G that is not a maximum independent set

Maximal Independent Set: Centralized

- Simple greedy algorithm solves MIS in the centralized setting

$S = \emptyset$.
iterate through the vertices in arbitrary order
    add a node to S, if none of its neighbors is already in S.

MIS via centralized greedy



The computed MIS depends on the order of vertices!

Maximal Independent Set: Facts

- **Simple greedy** algorithm solves MIS in the **centralized setting**
- In the distributed setting (LOCAL) it is one of the most challenging problems

Our goal for the next slides: Parallelize the greedy centralized greedy algorithm

Maximal Independent Set via given C-Coloring

Input: Graph $G = (V, E)$ with C-vertex coloring ϕ

Output: Maximal Independent Set S

Each node v executes the following code in parallel in synchronous rounds:

S=∅

For i=1 to C

    If $\phi(v) = i$ then

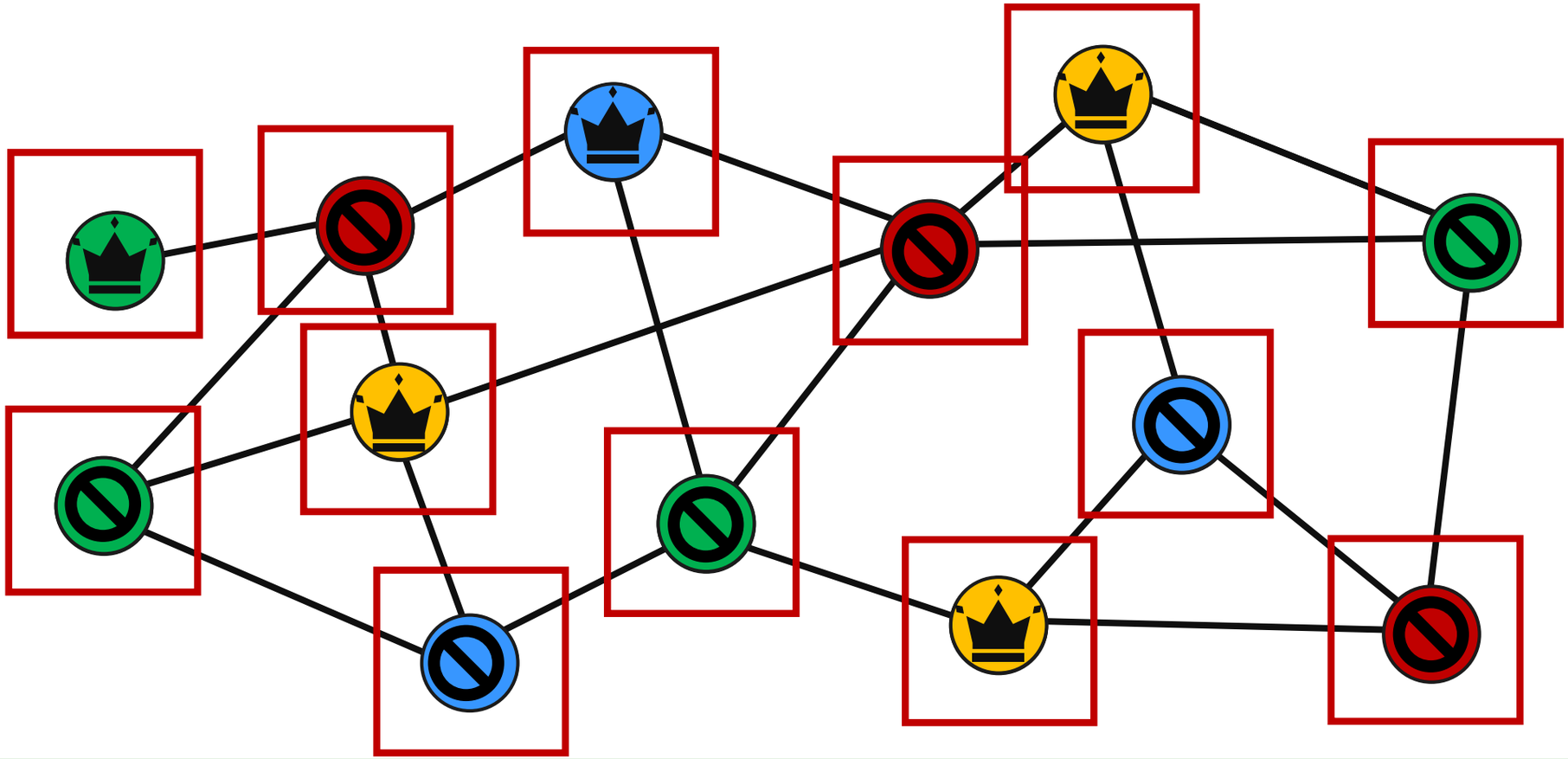
        if $N(v) \cap S = \emptyset$, join S and notify neighbors

Output whether $v \in S$.

coloring as a schedule

MIS via Coloring Schedule

```
For i=1 to C
    If 𝜙(𝑣) = 𝑖 then
        if 𝑁(𝑣) ∩ 𝑆 = ∅, join S and notify neighbors
```



Maximal Independent Set in LOCAL

Theorem:
There is a deterministic distributed algorithm to compute an MIS in $\boldsymbol{O(C)}$ rounds, given a C-vertex coloring.

Proof (correctness, independent set)
Proof (correctness, maximality)
Proof (runtime)

Maximal Independent Set in LOCAL

Theorem:
There is a deterministic distributed algorithm to compute an MIS in $\boldsymbol{O(C)}$ rounds, given a C-vertex coloring.

Proof (correctness, independent set):
Notation: $S_i, i = 0, \dots, C$, the set S after iteration i
Induction: S_i is an independent set.

Induction start: Initially $S_0 = \emptyset$ is an IS

Induction Step ($\boldsymbol{S_i \to S_{i+1}}$):

Assume for contradiction that there exists adjacent nodes $u, v \in S_{i+1}$:
As S_i is an IS, u or v needs to have input color $i + 1$. Wlog $\phi(v) = i + 1$.
Case 1 ($\boldsymbol{\phi(u) = i + 1}$): Contradiction, as u, v, are adjacent
Case 2 ($\boldsymbol{\phi(u) < i + 1}$): Contradiction, as then $u \in S_i$, and v would not have joined S_i, as the test $N(v) \cap S = \emptyset$ would have been negative.

Maximal Independent Set in LOCAL

Theorem:
There is a deterministic distributed algorithm to compute an MIS in $\boldsymbol{O(C)}$ rounds, given a C-vertex coloring.

Proof (correctness, maximality):

Assume for contradiction there exists some node $v \in V \setminus S$ such that $S \cup \{v\}$ is an IS.

As S only keeps growing, this implies that v had no neighbor in $S_{\phi(v)-1}$ when v was processed, and v would have joined $S_{\phi(v)} \subseteq S$, a contradiction.

Maximal Independent Set in LOCAL

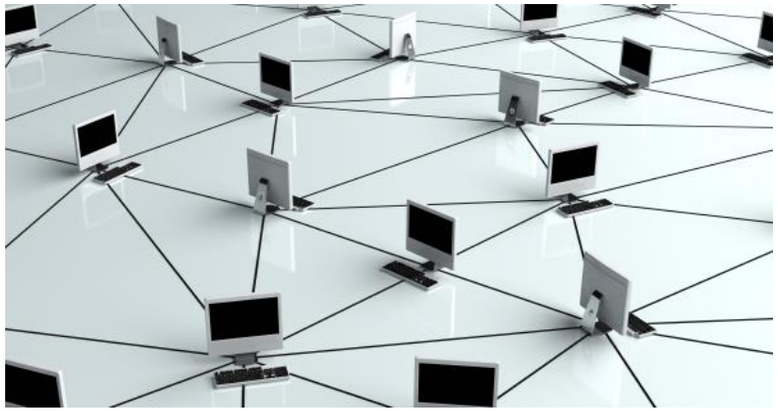
Theorem:
There is a deterministic distributed algorithm to compute an MIS in $O(C)$ rounds, given a C-vertex coloring.

Proof (runtime):
The previous algorithm requires one round of communication per color class to inform neighbors whether one has joined the MIS or not

Corollary:
There is a deterministic distributed algorithm to compute an MIS in $O(\Delta^2 + \log^* n)$ rounds.

Proof:
Use Linial's algorithm to compute an $O(\Delta^2)$-coloring in $O(\log^* n)$ rounds. Then use the theorem above.

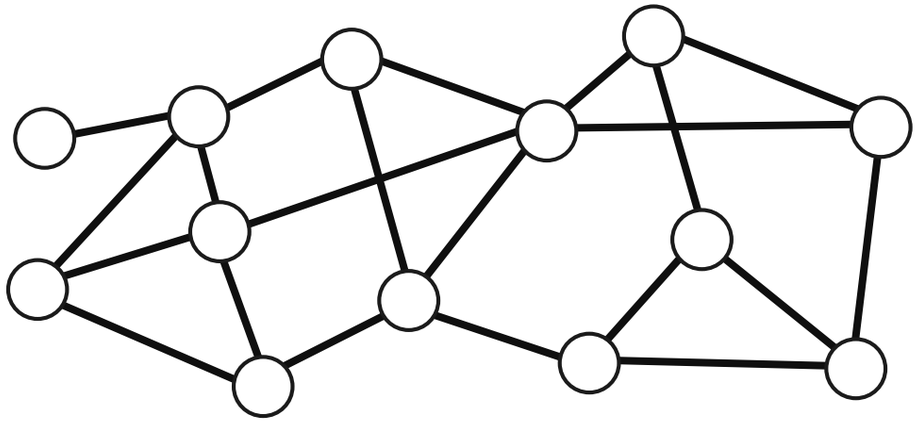
Outlook 1: Locality



Distributed Graph Algorithms

?

=

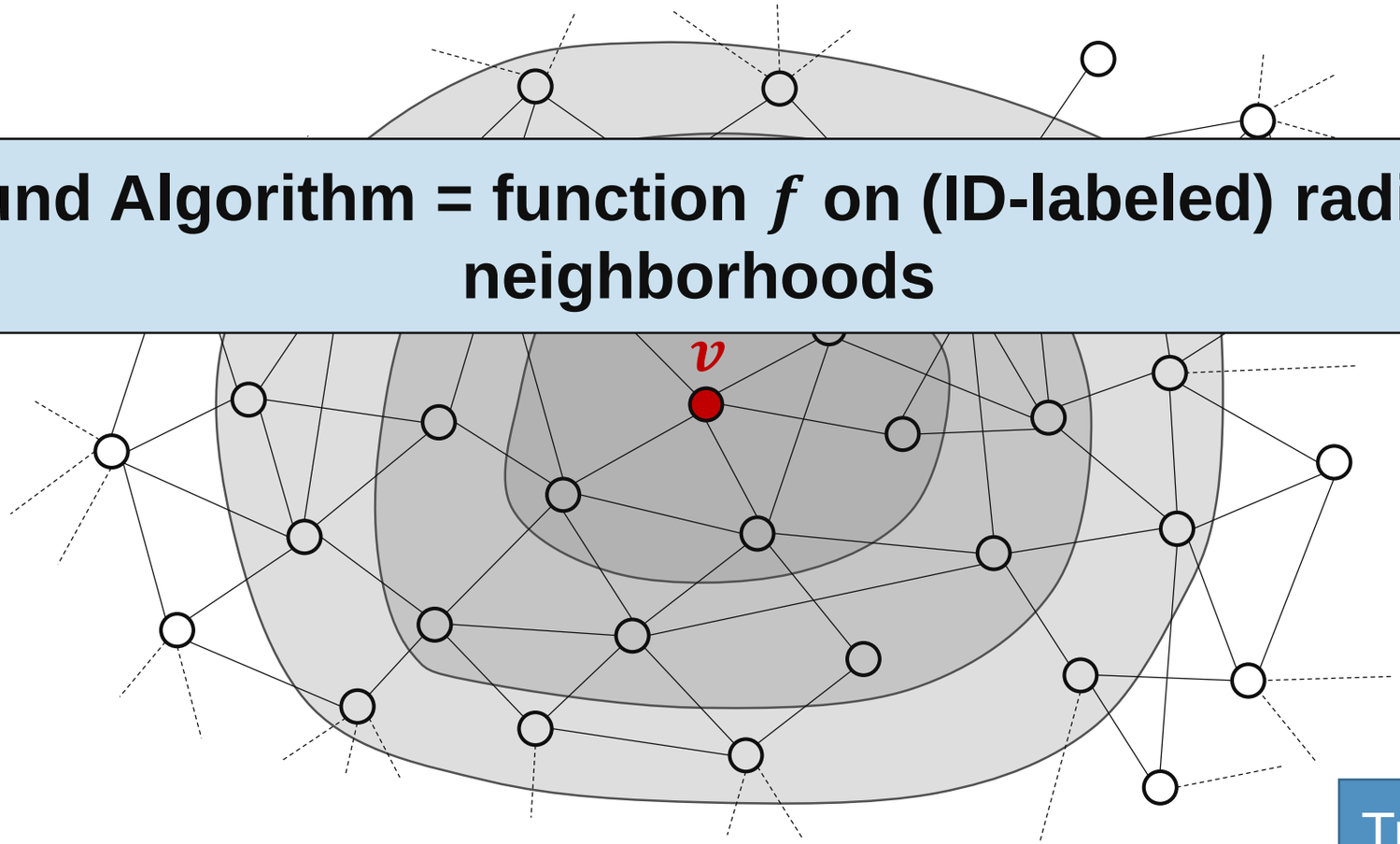


Graph Theory

Graph Theoretic Concept: Locality

LOCAL Model
= unbounded
Message size

r-round Algorithm = function f on (ID-labeled) radius-r neighborhoods



Trivial upper bound:
$O(Diam(G))$ rounds

r-round algorithm:

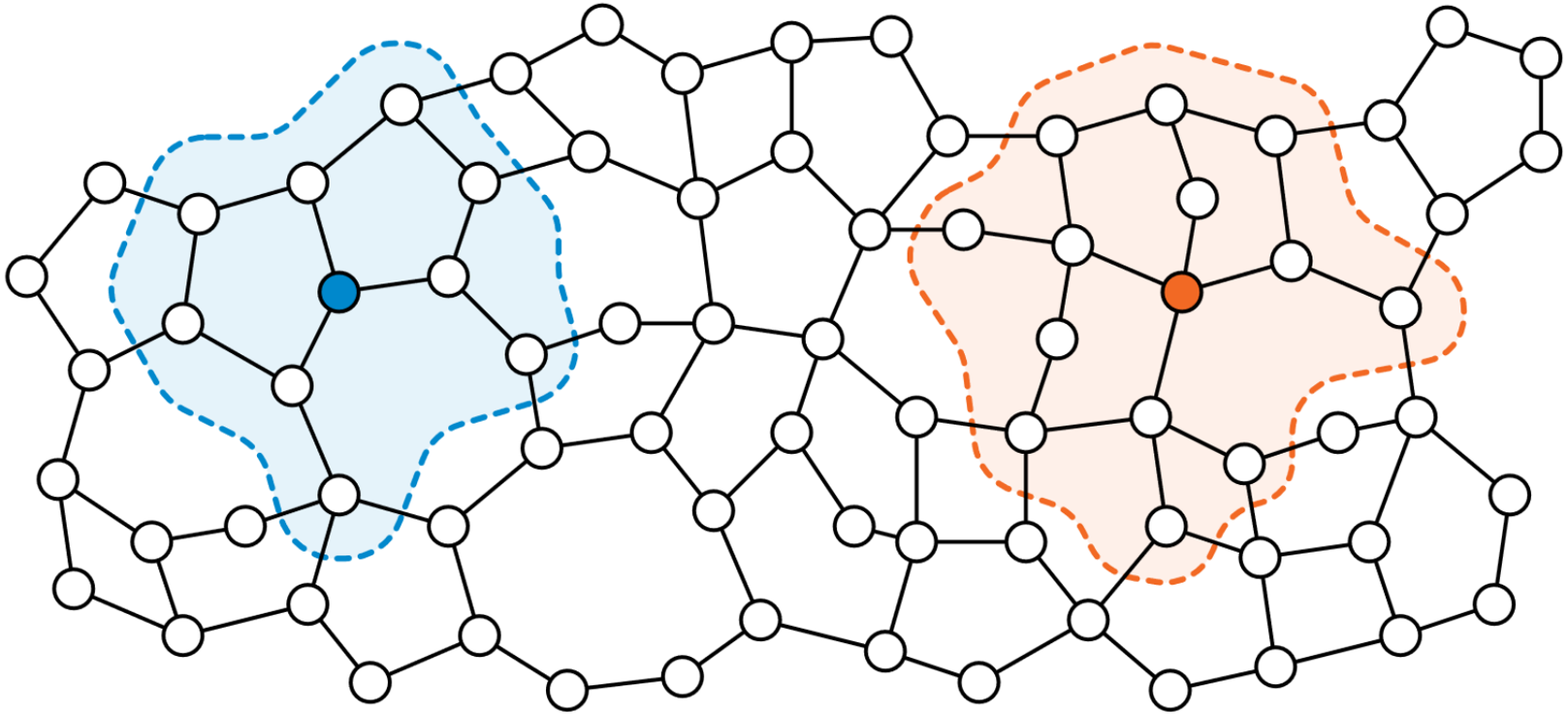
each node has to base its output on
the initial information in its r-neighborhood

Communication restriction: locality

locality

=

How far do I need to see to produce my own part of the solution?



Outlook 1b:
Message Size

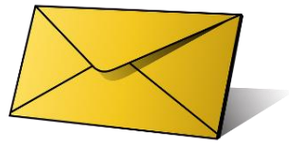
LOCAL

locality

graph theoretic algorithmic

CONGEST

locality + bandwidth

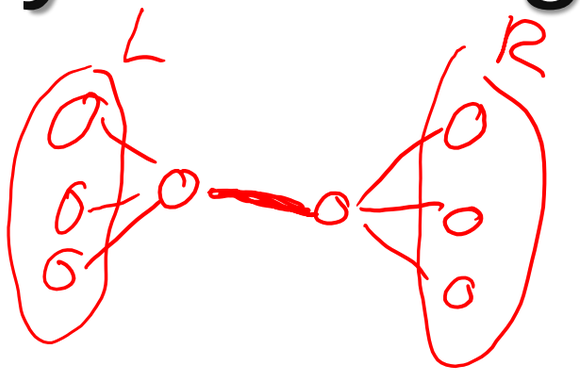


bandwidth restricted

Locality is not the only challenge

algorithmic

Congestion



Outlook 2:
Randomized Coloring

Randomized coloring is easy

candidate color

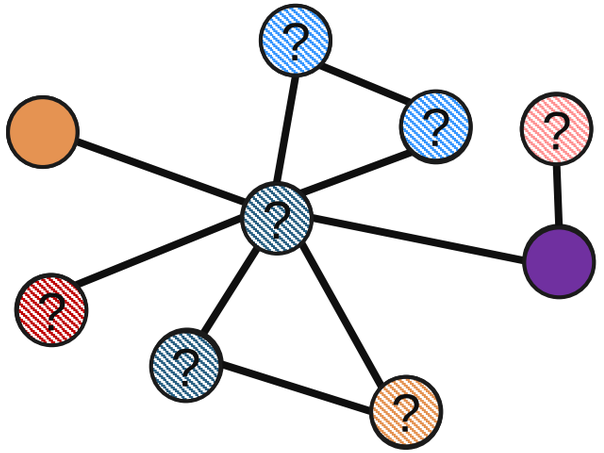
Repeat until colored
- pick a random **free color** c
- keep it, if no *conflict*

Randomized coloring is easy

candidate color

Repeat until colored
- pick a random **free color** c
- keep it, if no *conflict*

v's free colors:

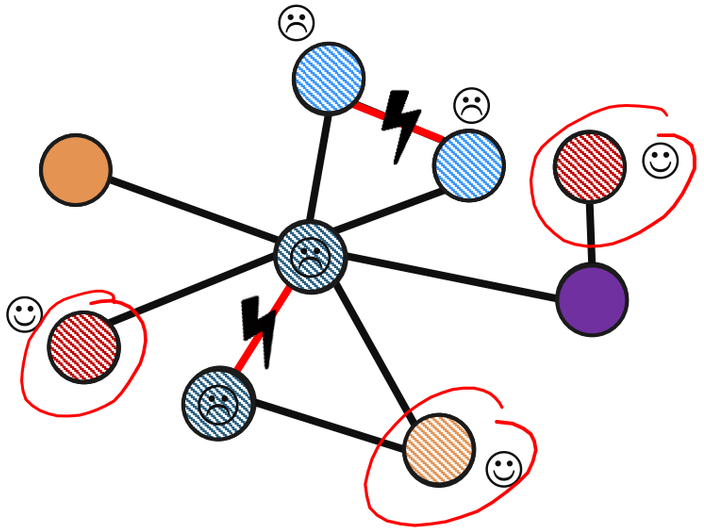


Randomized coloring is easy

candidate color

Repeat until colored
- pick a random **free color** c
- keep it, if no *conflict*

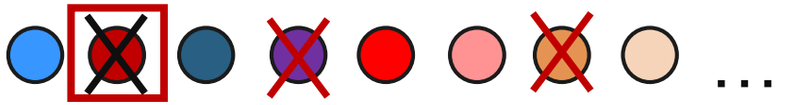
v's free colors:

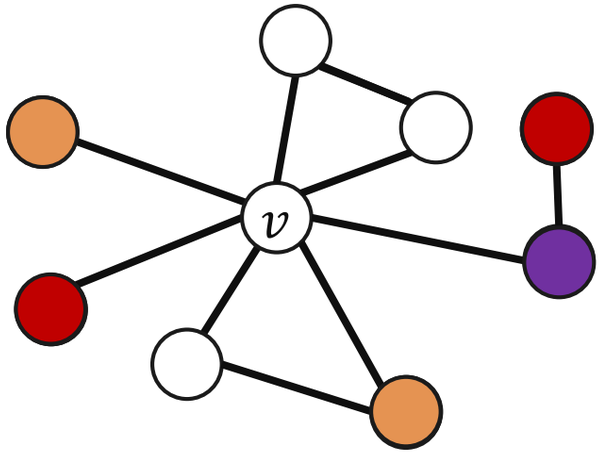


Randomized coloring is easy

Repeat until colored
- pick a random **free color** c
- keep it, if no *conflict*

v's free colors:

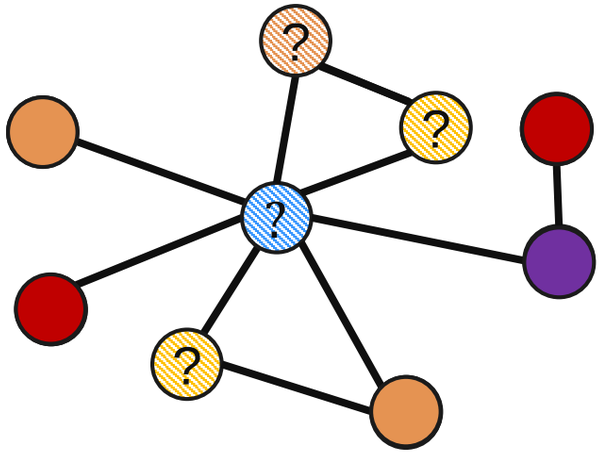




Randomized coloring is easy

Repeat until colored
- pick a random **free color** c
- keep it, if no *conflict*

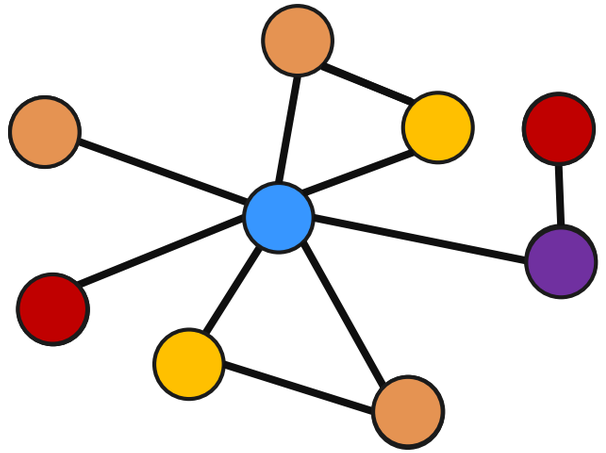
v's free colors:



Randomized coloring is easy

Repeat until colored
- pick a random **free color** c
- keep it, if no *conflict*

v's free colors:



With 2Δ colors:
a node gets colored with probability 0.5
→ in expectation half the vertices get colored

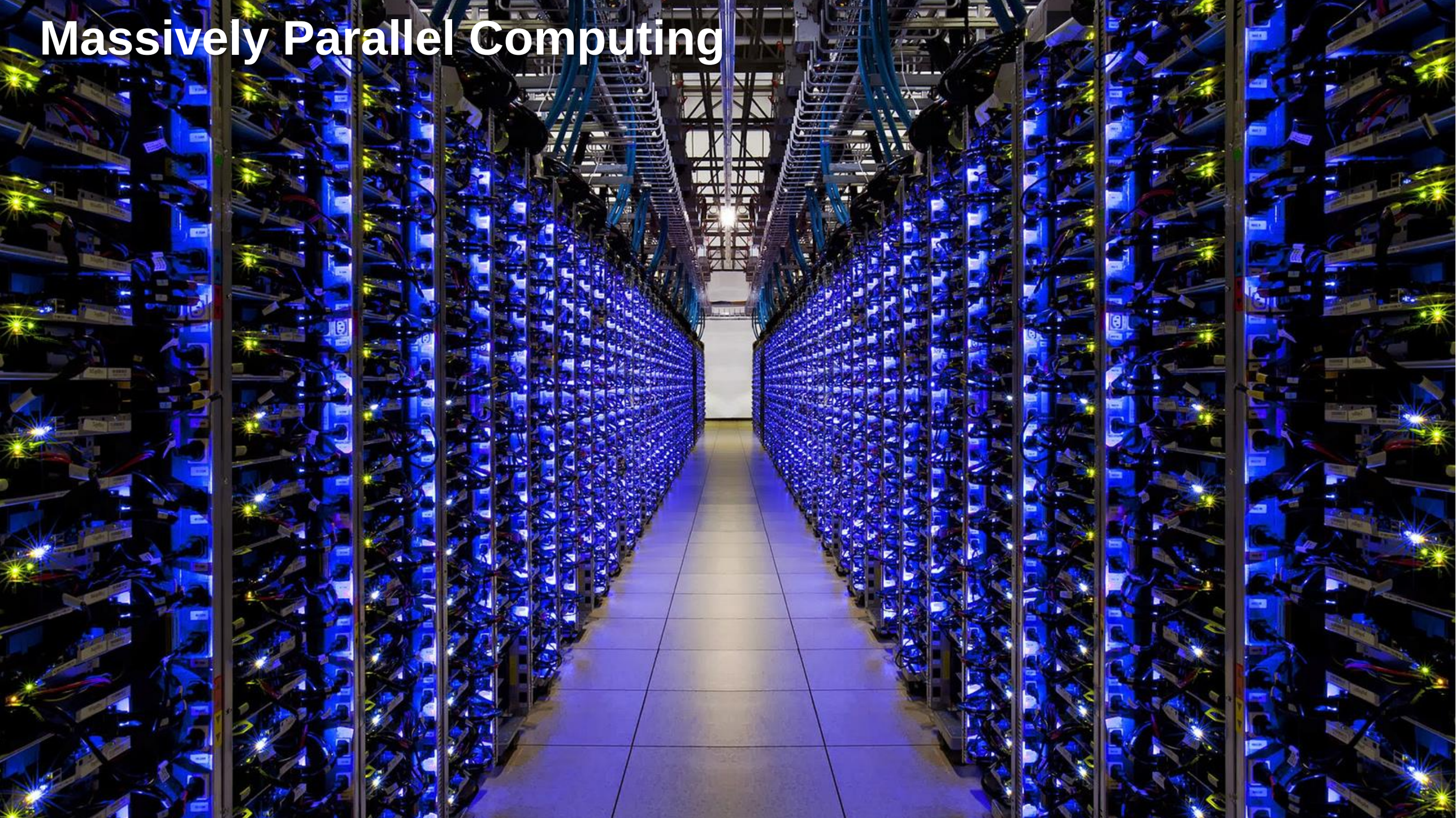
Randomized: $O(\log n)$ rounds for 2Δ-coloring, whp.

Randomized: $O(\log n)$ rounds for $(\Delta + 1)$-coloring, whp.

[Luby; Siam Journal '86], [Linial; FOCS '87], [Johansson; Inf. Proc. Letters '99]

Outlook 3:
Massively Parallel Computing

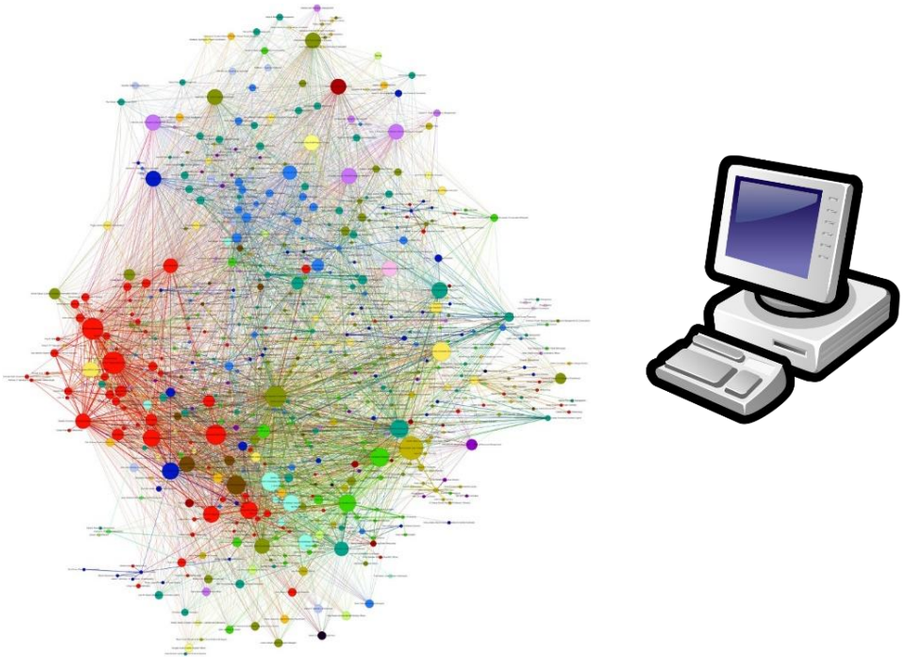
Massively Parallel Computing



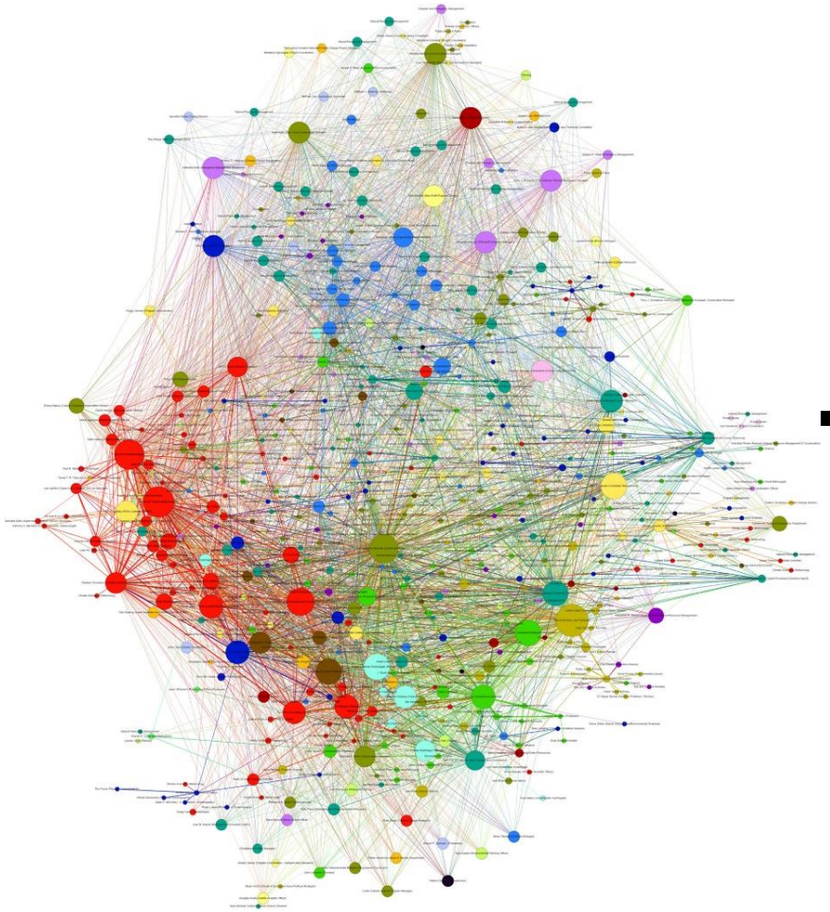
Massively Parallel Graph Processing

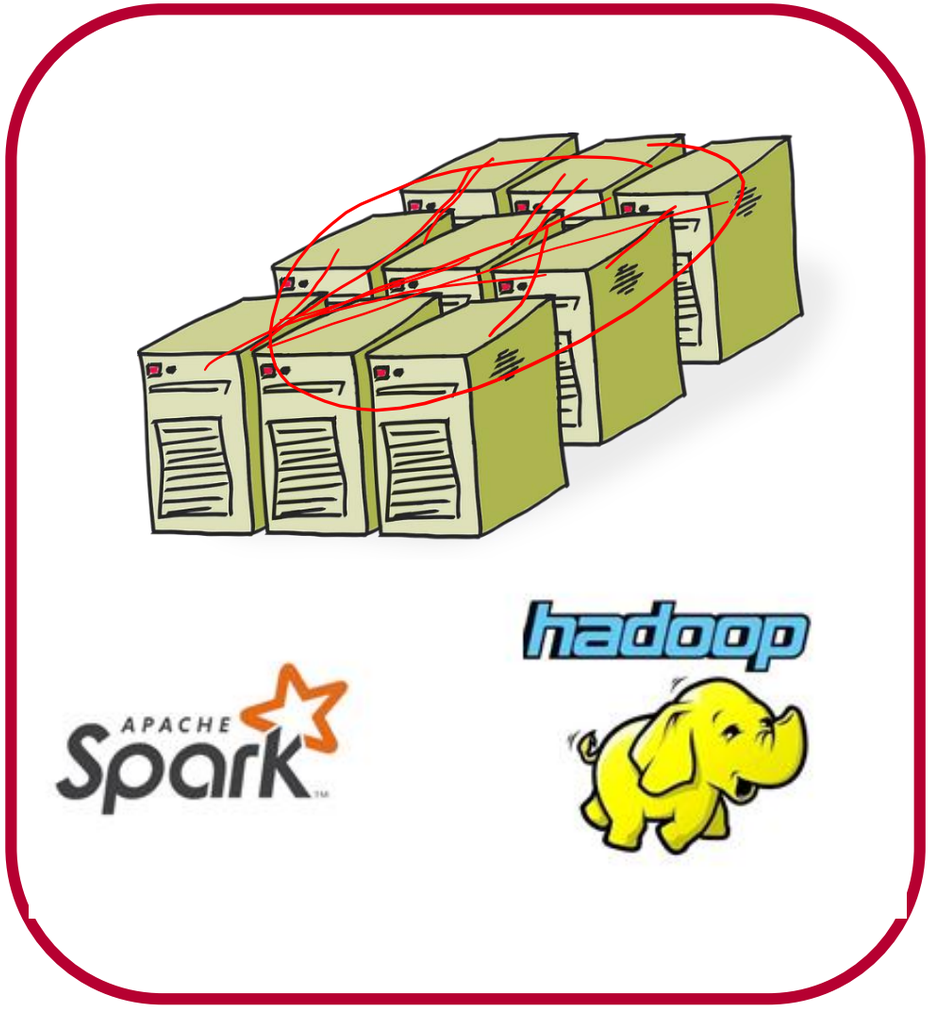
Data does not fit onto a single computer

“Distributed computing is more than message passing in networks”

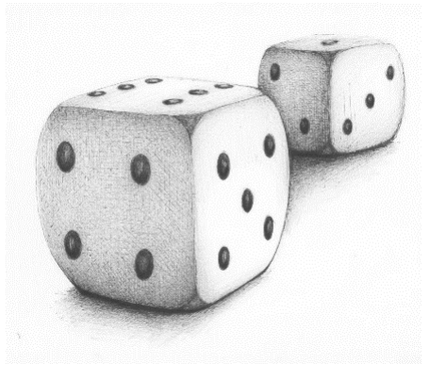


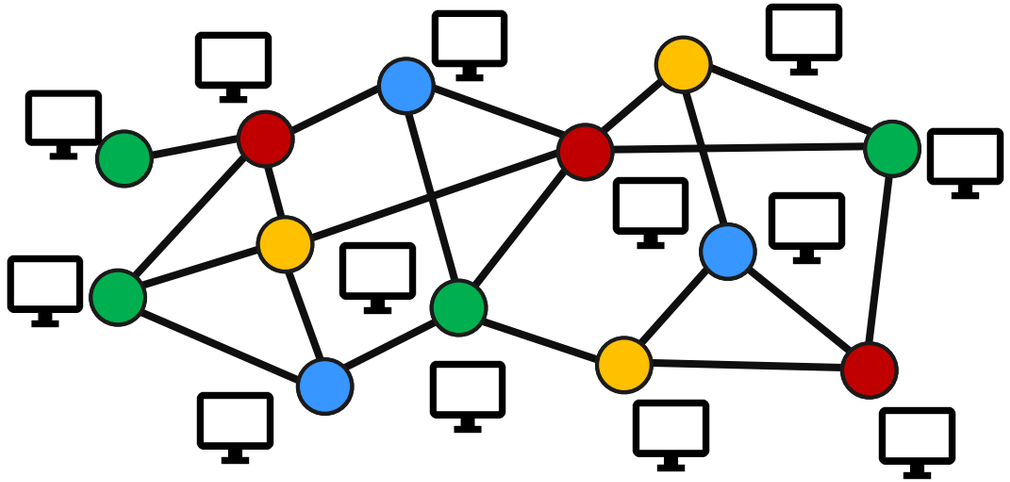
Massive Graph Processing

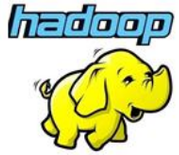




All-to-all communication: No locality, but bandwidth restrictions between the machines







Thank you very much