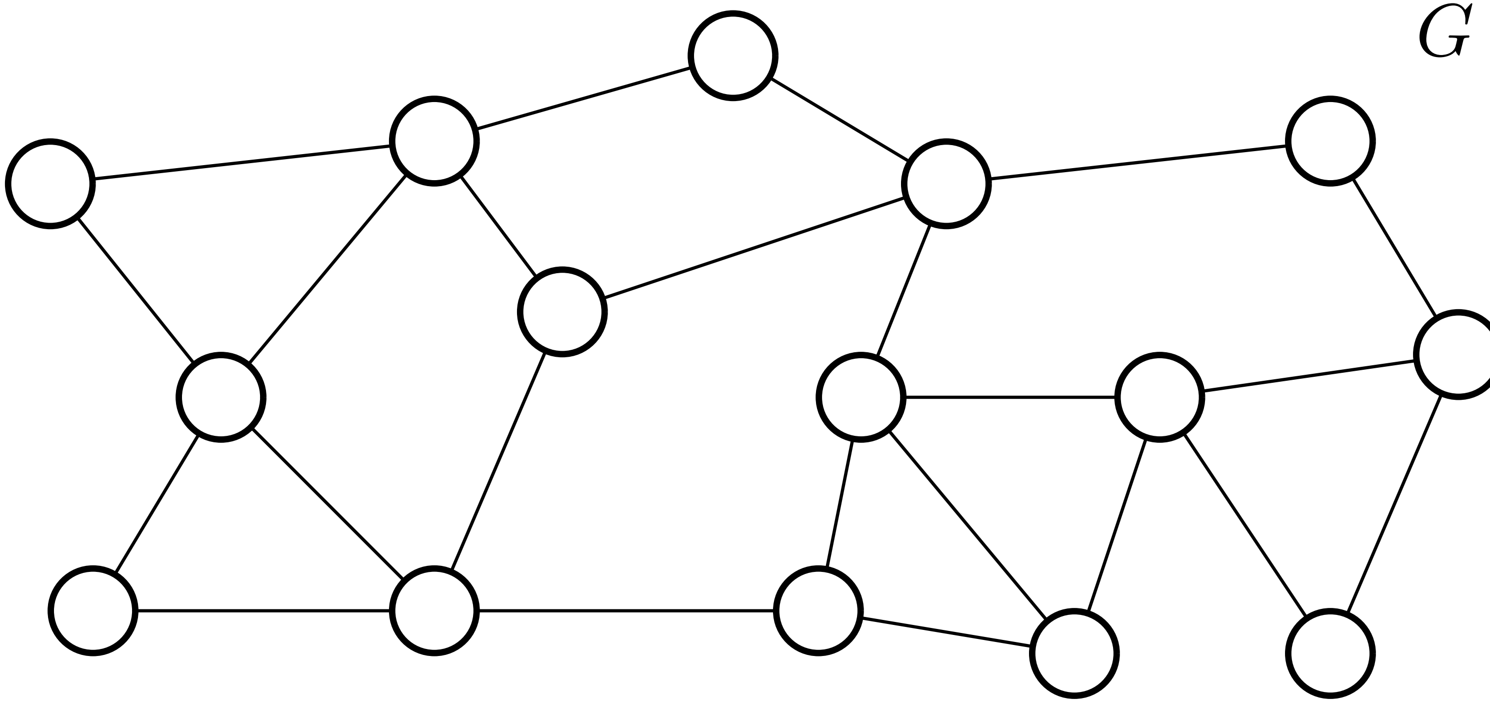
Distributed Vertex Coloring

Vertex Colorings

Vertex Coloring: Each vertex is assigned a color.

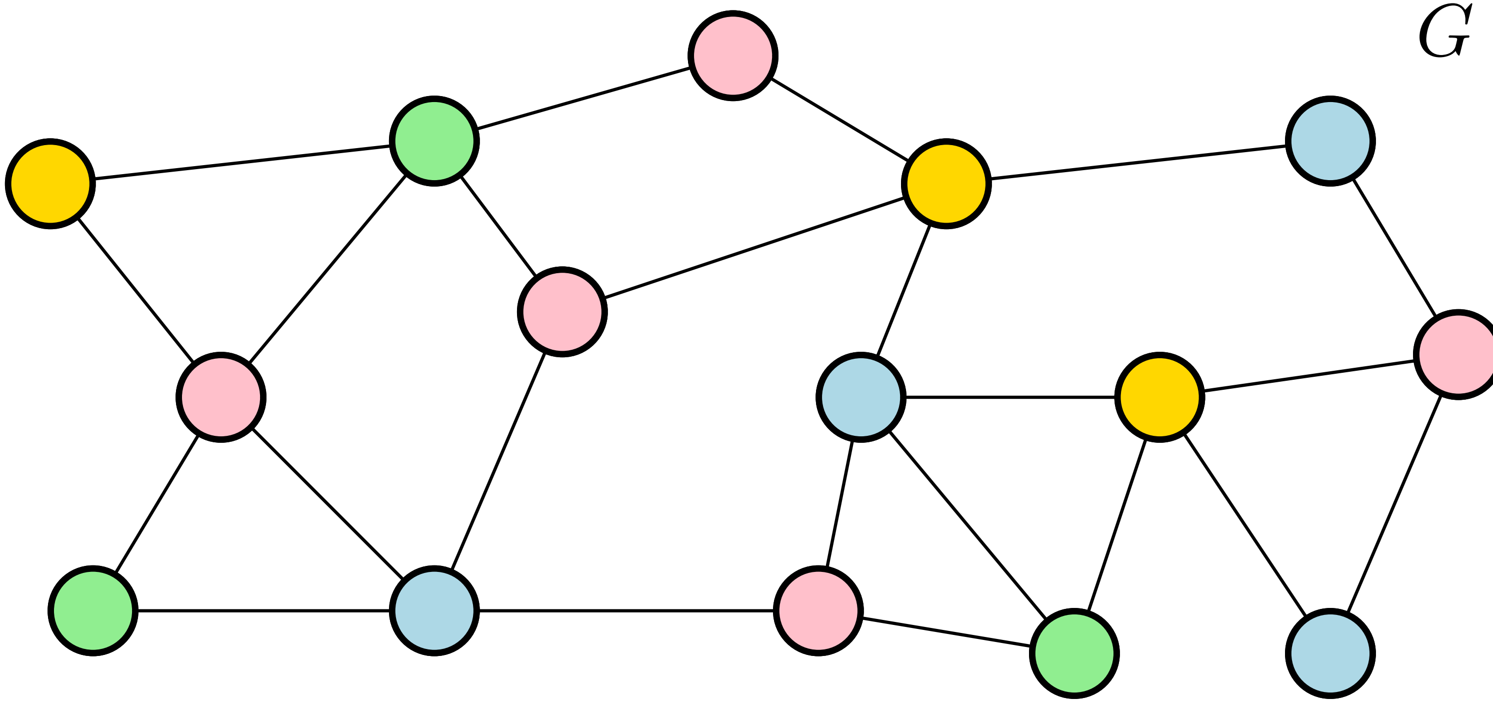
$G = (V, E)$



Vertex Colorings

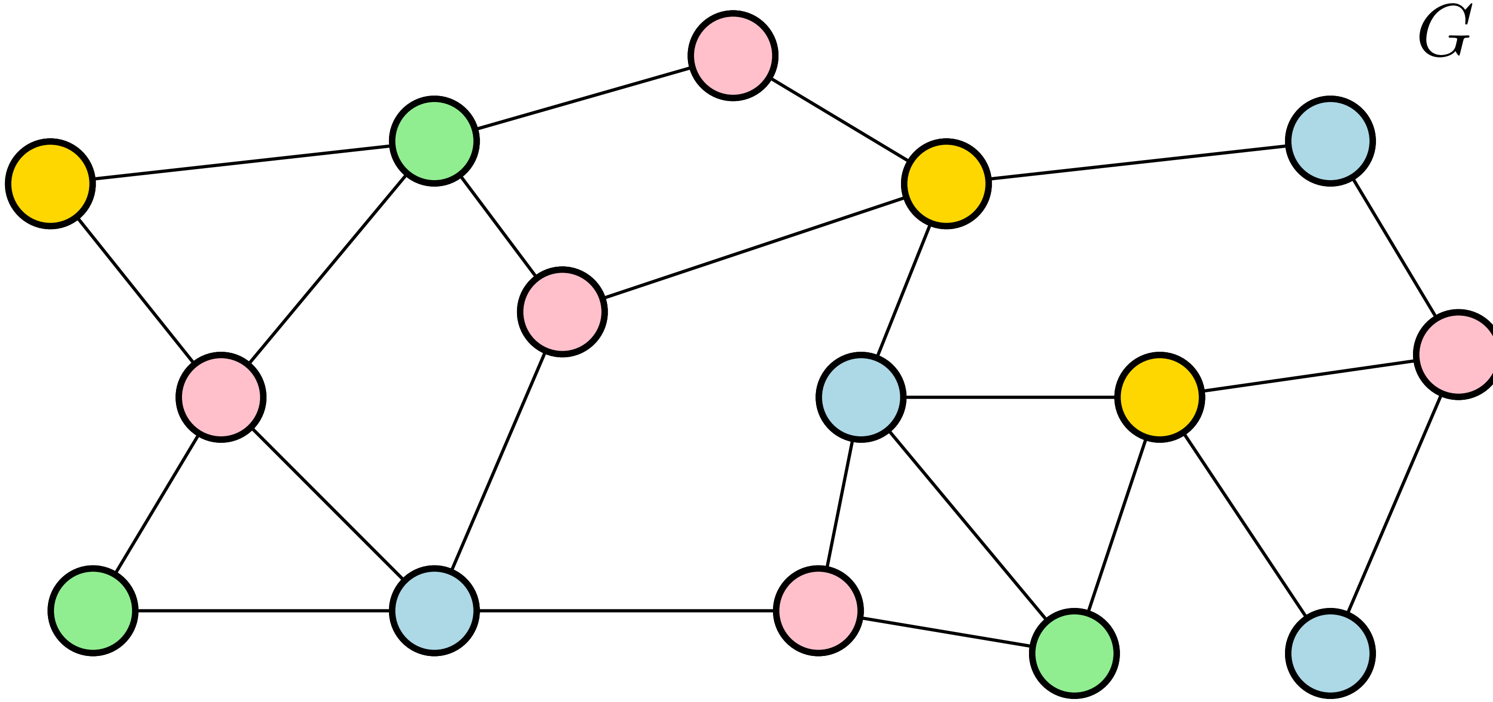
Vertex Coloring: Each vertex is assigned a color.

$G = (V, E)$



Vertex Colorings

Vertex Coloring: Each vertex is assigned a color.

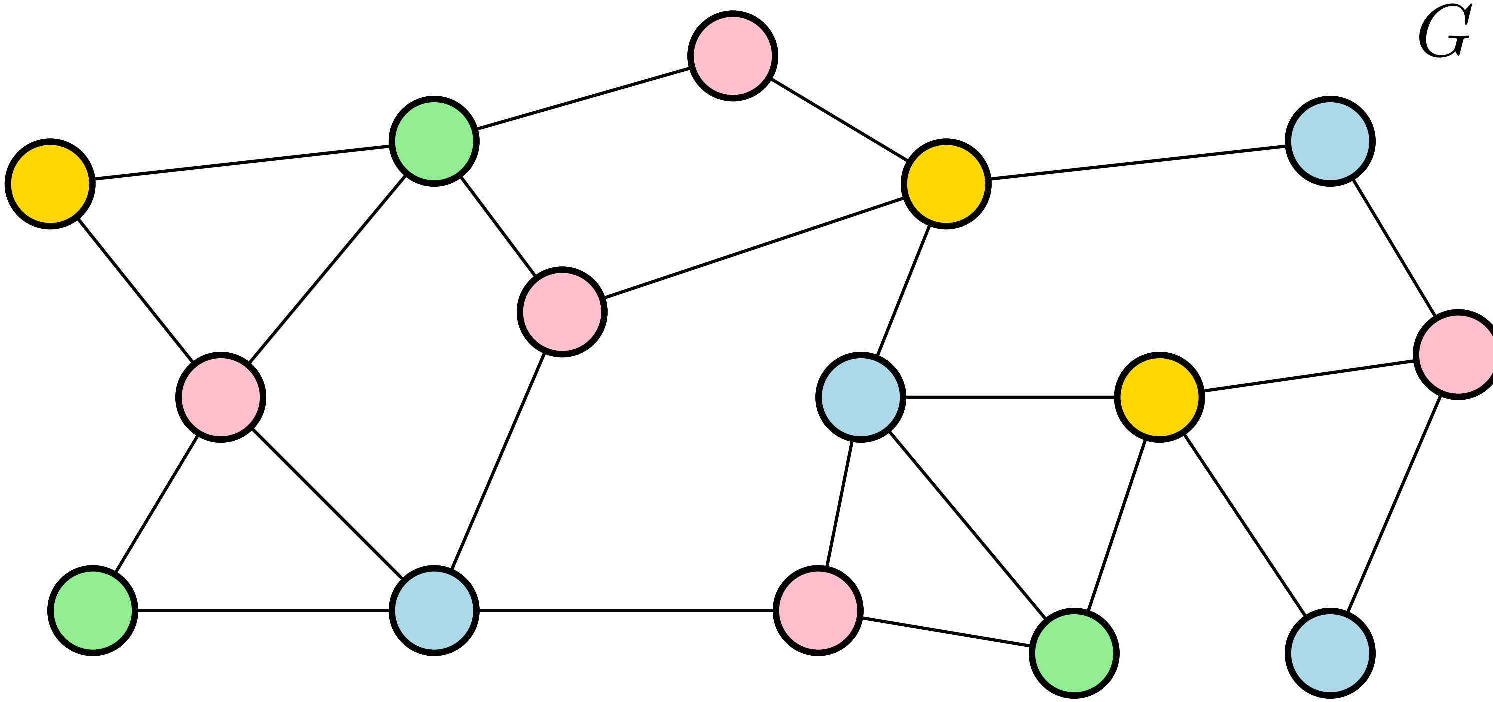


$G = (V, E)$

Valid Vertex Coloring: A vertex coloring is **valid** if no two adjacent nodes have the same color.

Vertex Colorings

Vertex Coloring: Each vertex is assigned a color.



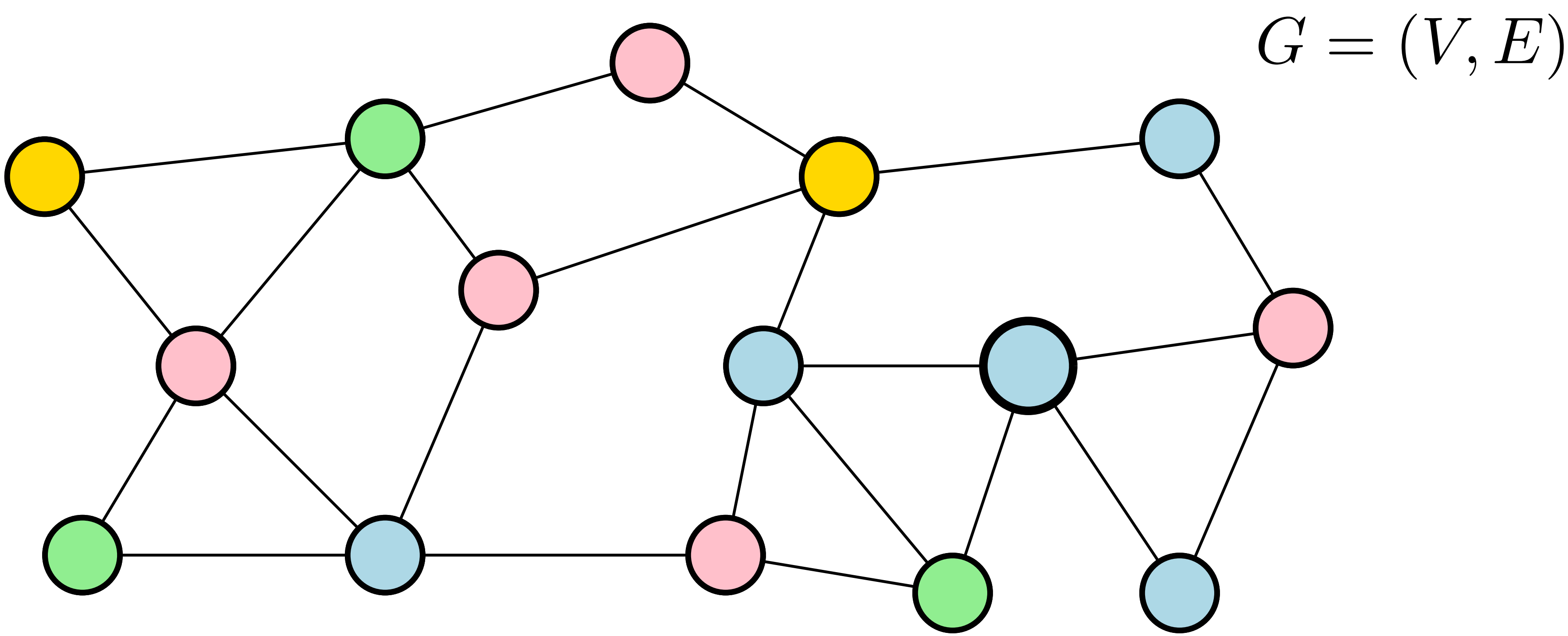
$G = (V, E)$

VALID

Valid Vertex Coloring: A vertex coloring is **valid** if no two adjacent nodes have the same color.

Vertex Colorings

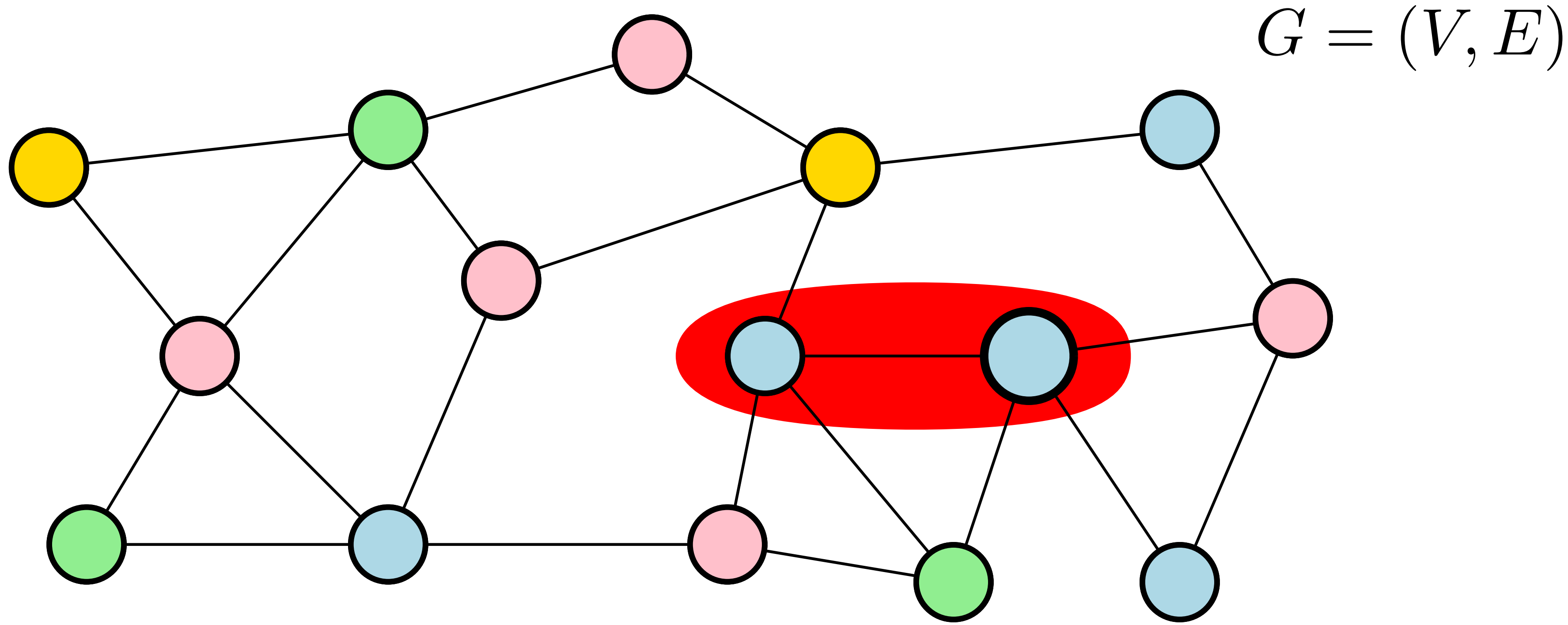
Vertex Coloring: Each vertex is assigned a color.



Valid Vertex Coloring: A vertex coloring is **valid** if no two adjacent nodes have the same color.

Vertex Colorings

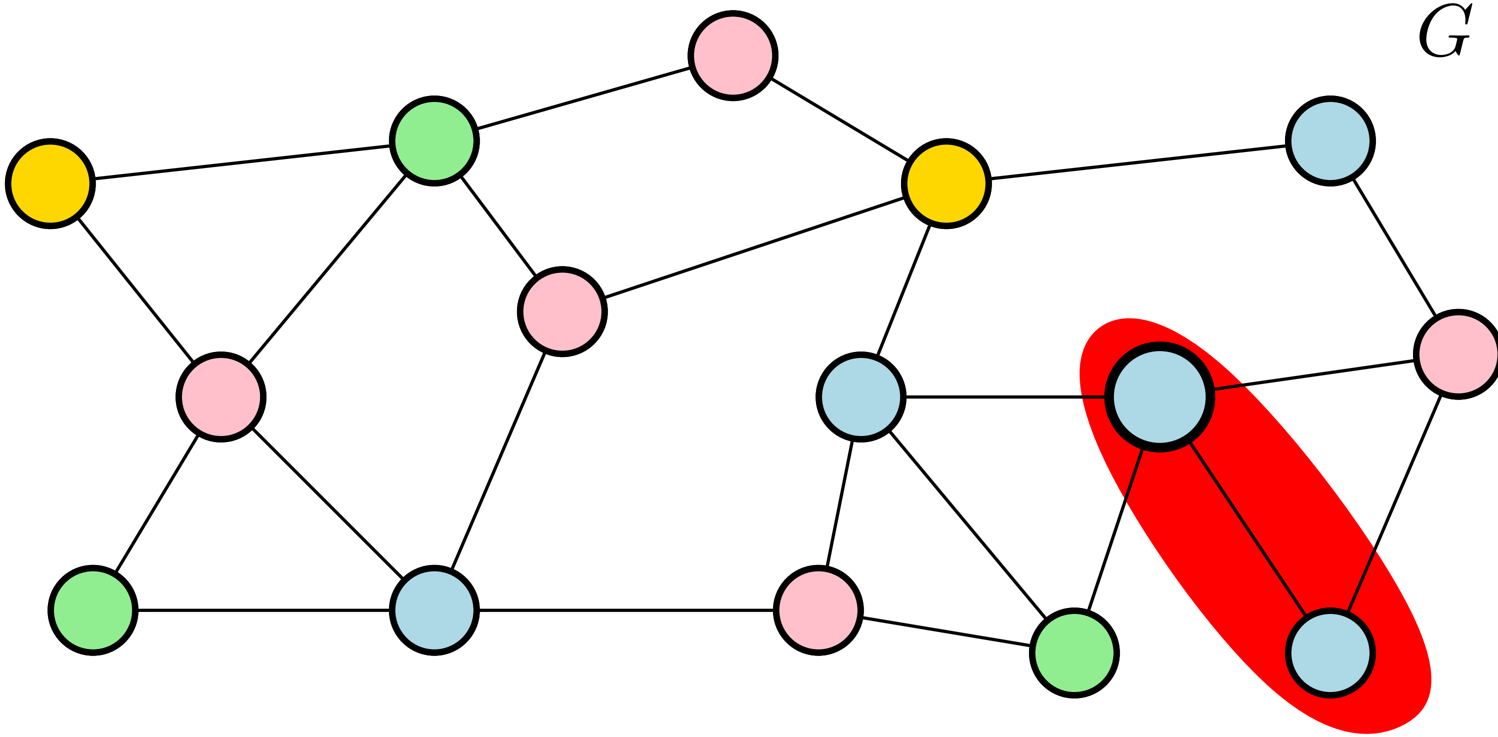
Vertex Coloring: Each vertex is assigned a color.



Valid Vertex Coloring: A vertex coloring is **valid** if no two adjacent nodes have the same color.

Vertex Colorings

Vertex Coloring: Each vertex is assigned a color.

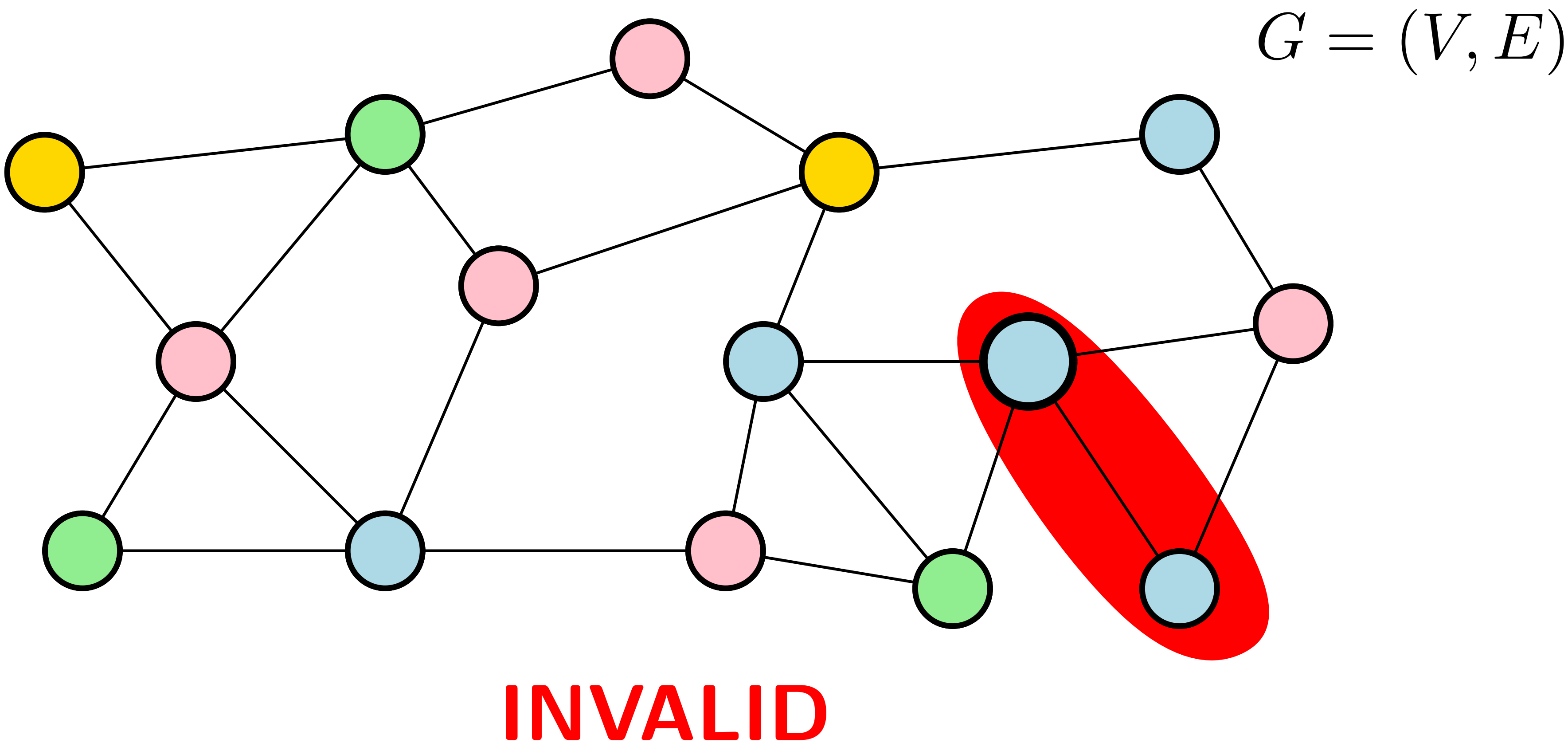


$G = (V, E)$

Valid Vertex Coloring: A vertex coloring is **valid** if no two adjacent nodes have the same color.

Vertex Colorings

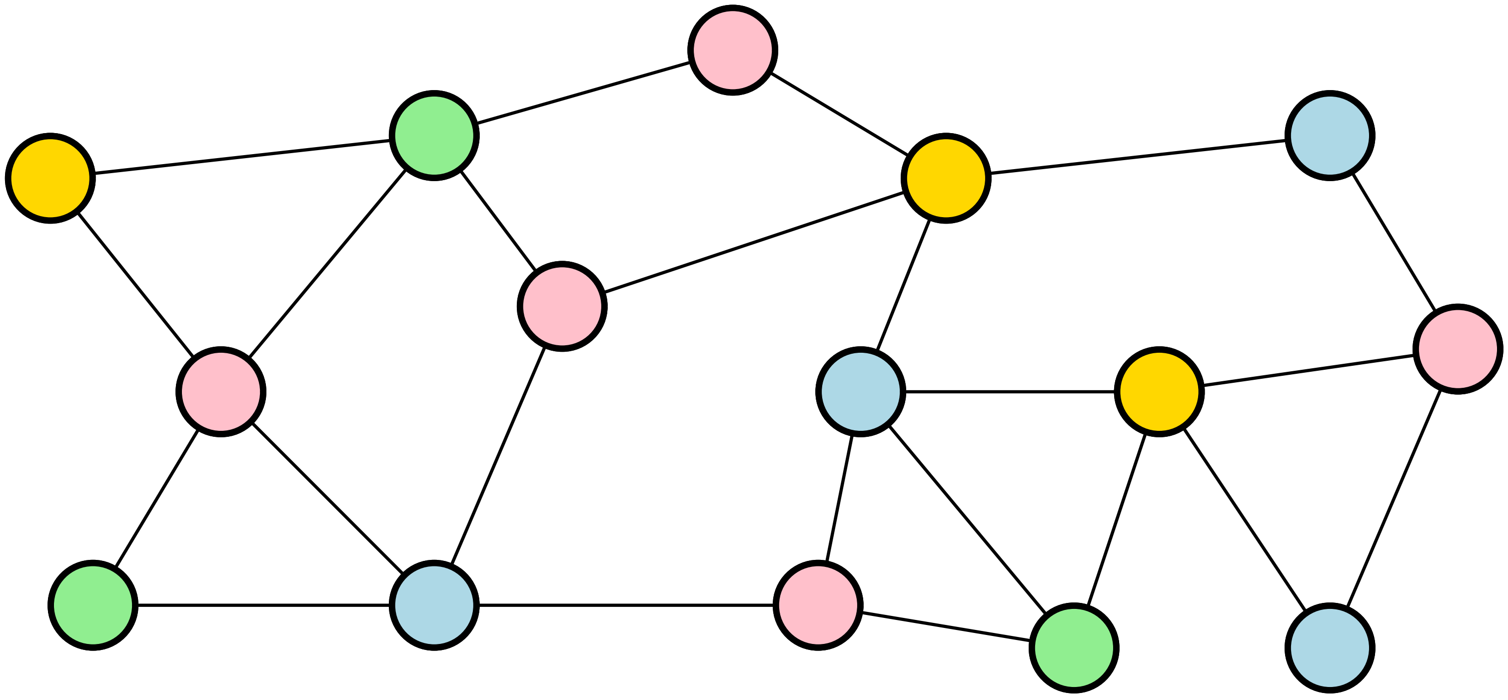
Vertex Coloring: Each vertex is assigned a color.



Valid Vertex Coloring: A vertex coloring is **valid** if no two adjacent nodes have the same color.

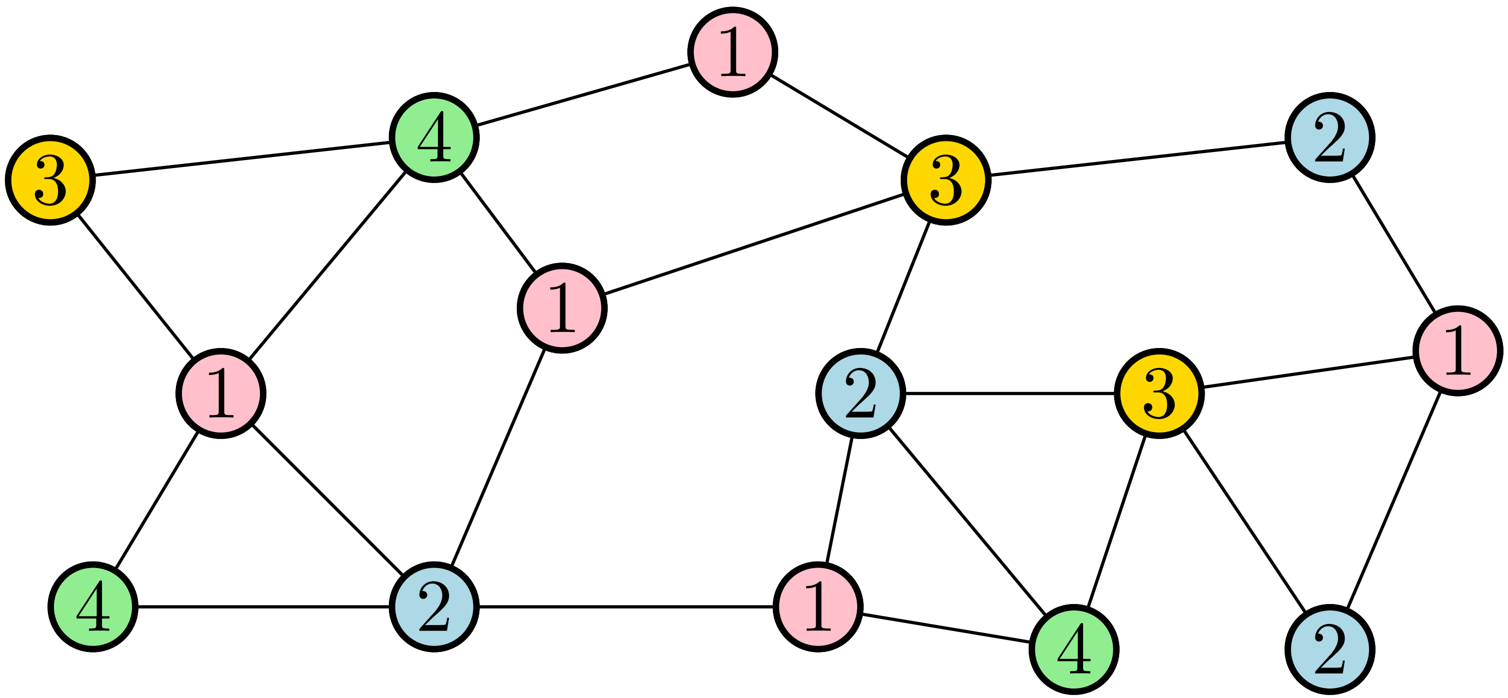
Vertex Colorings

In algorithms each color can be represented with an integer.



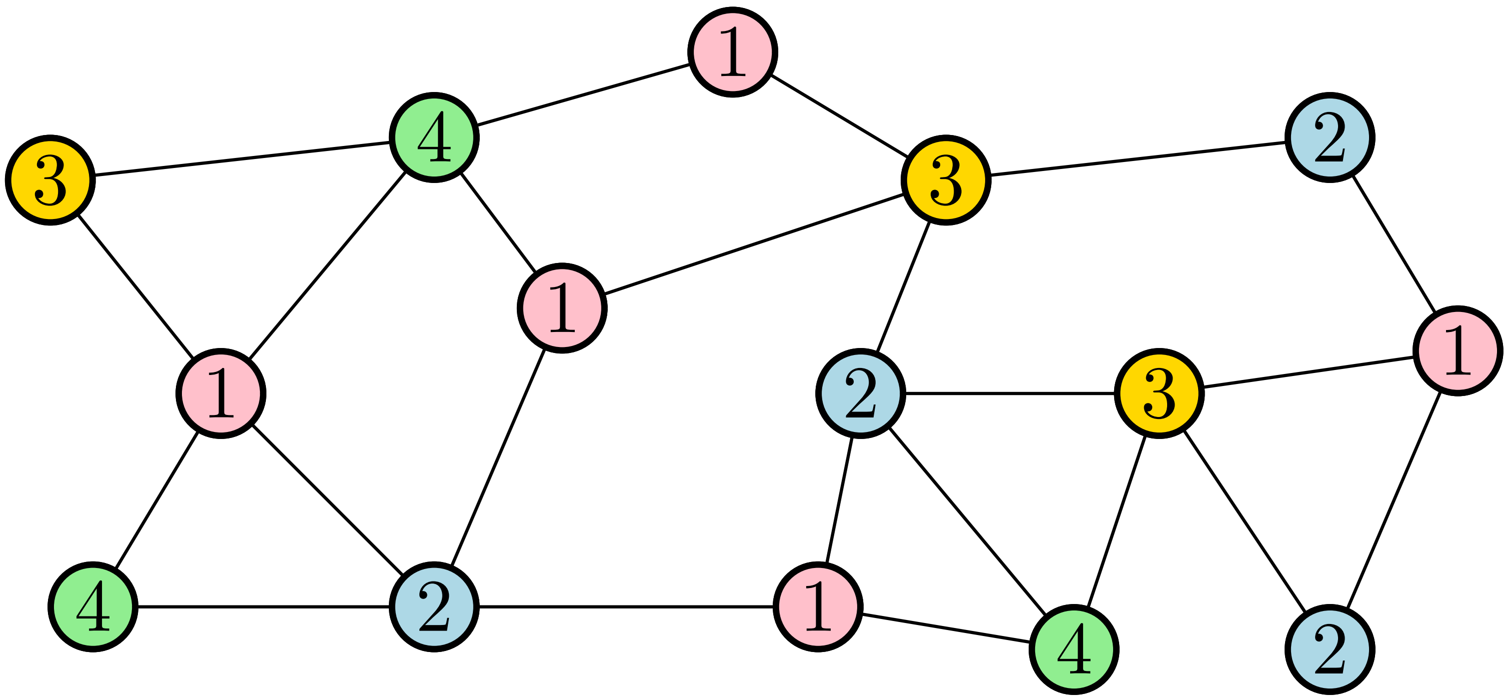
Vertex Colorings

In algorithms each color can be represented with an integer.



Vertex Colorings

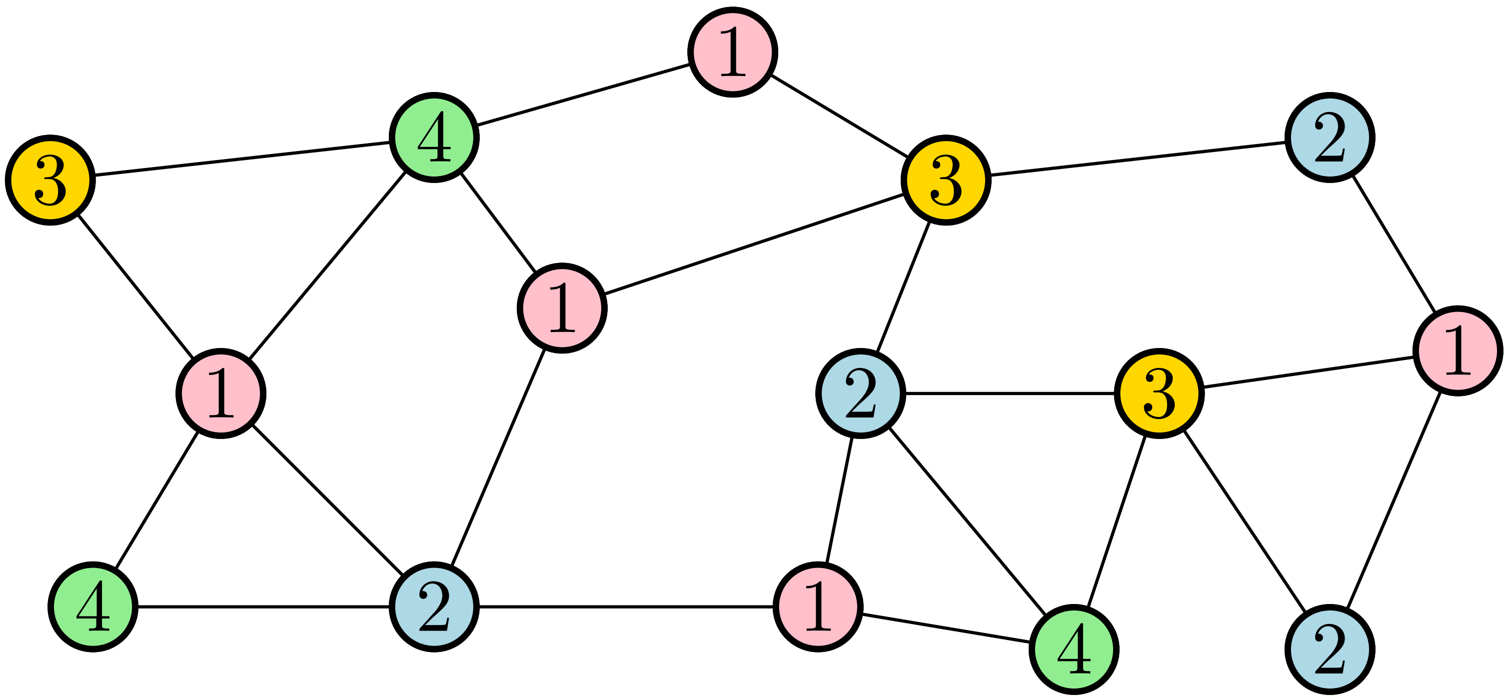
In algorithms each color can be represented with an integer.



Definition: A k-**coloring** is a coloring with k colors.

Vertex Colorings

In algorithms each color can be represented with an integer.

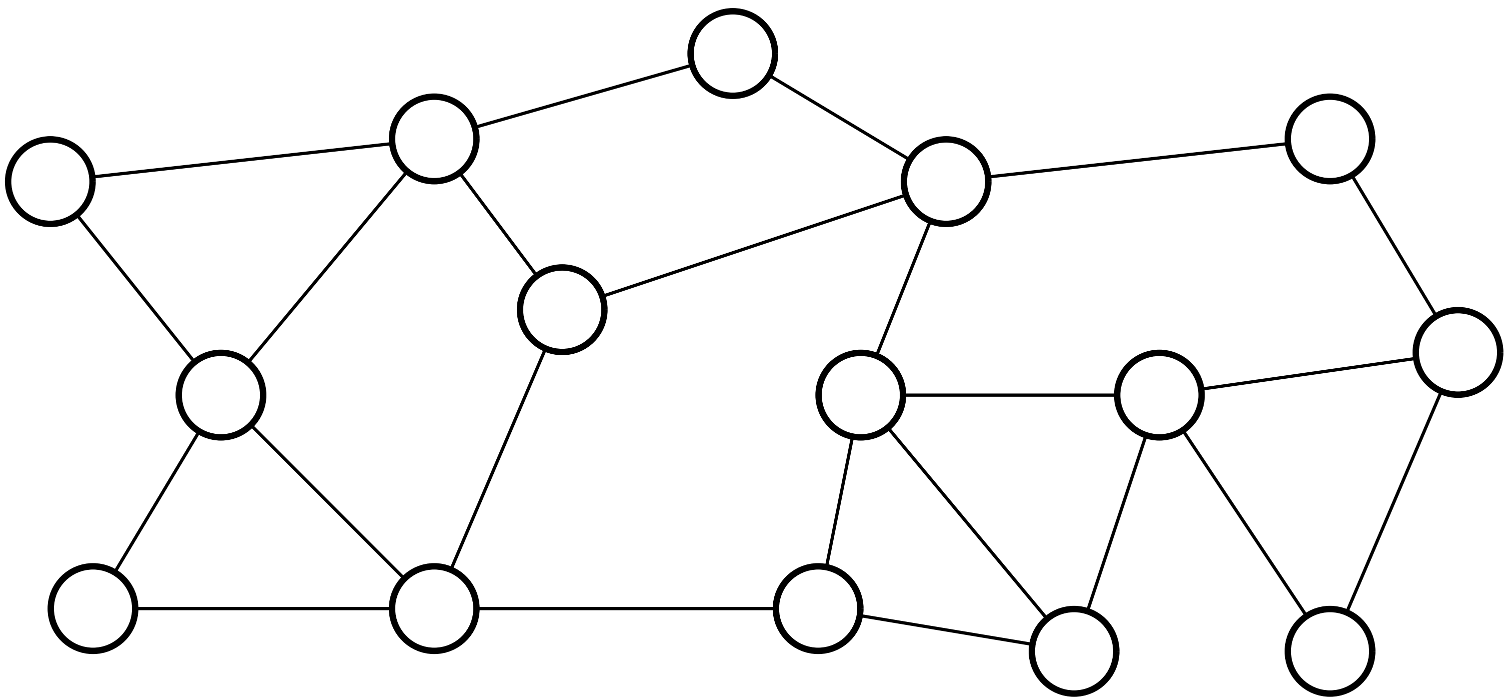


4-coloring

Definition: A k-**coloring** is a coloring with k colors.

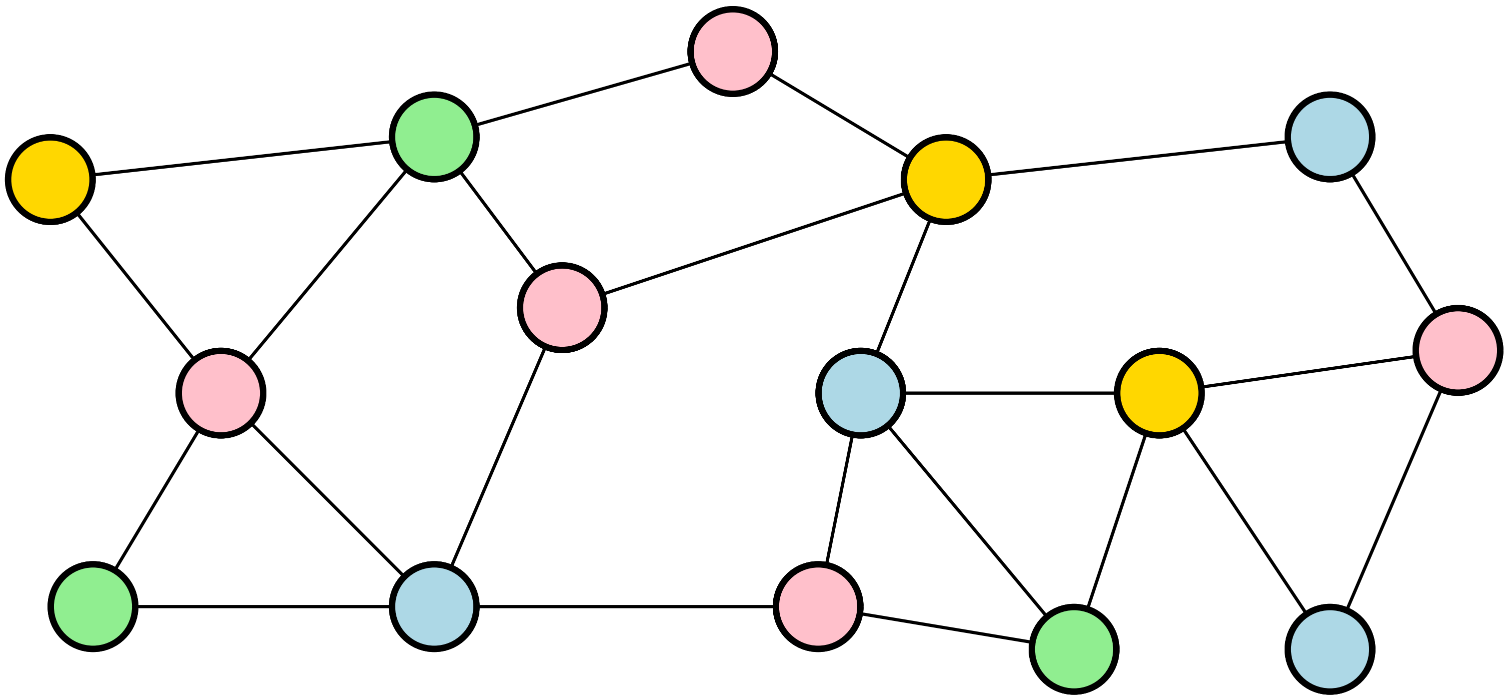
Chromatic Number

Definition: Given a graph G, the **chromatic number** $\varphi(G)$ of G is the smallest integer k such that there exists a valid k-coloring of G.



Chromatic Number

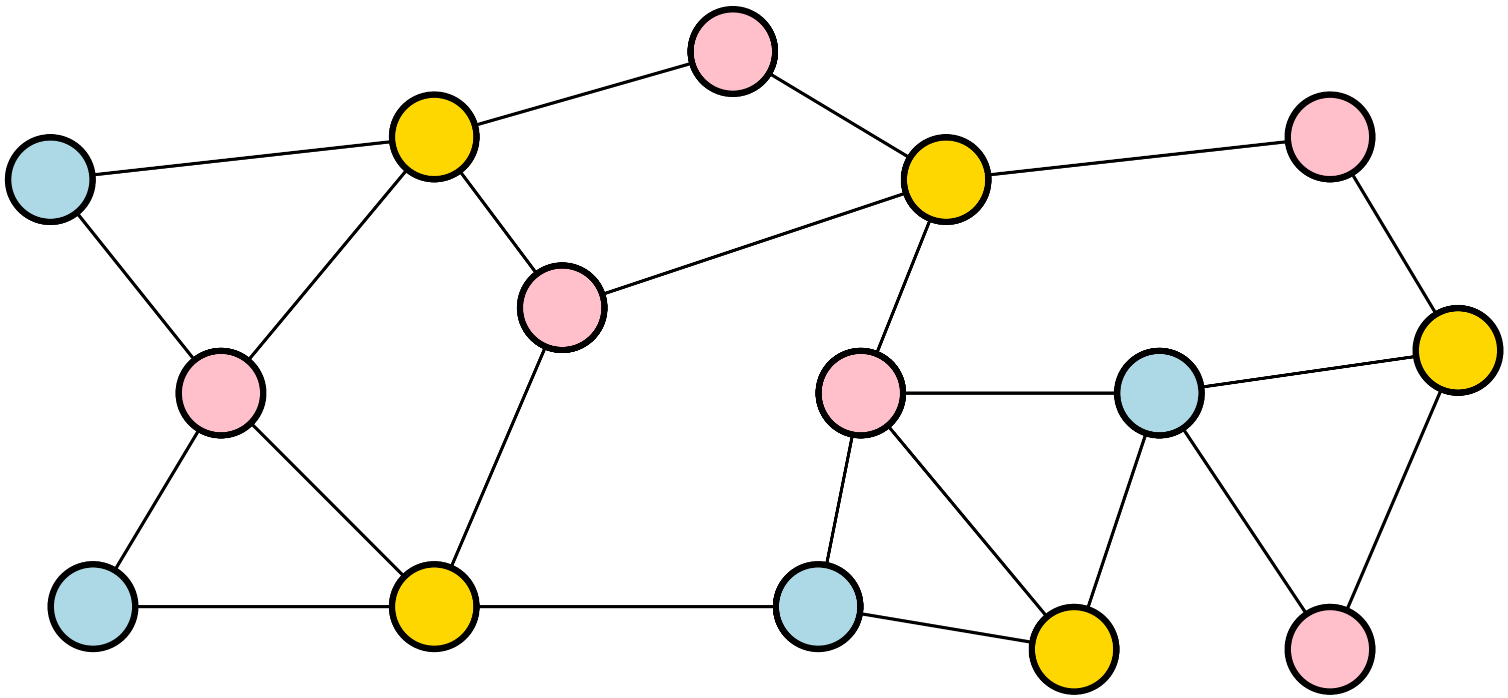
Definition: Given a graph G, the **chromatic number** $\varphi(G)$ of G is the smallest integer k such that there exists a valid k-coloring of G.



$\varphi(G) \leq 4$

Chromatic Number

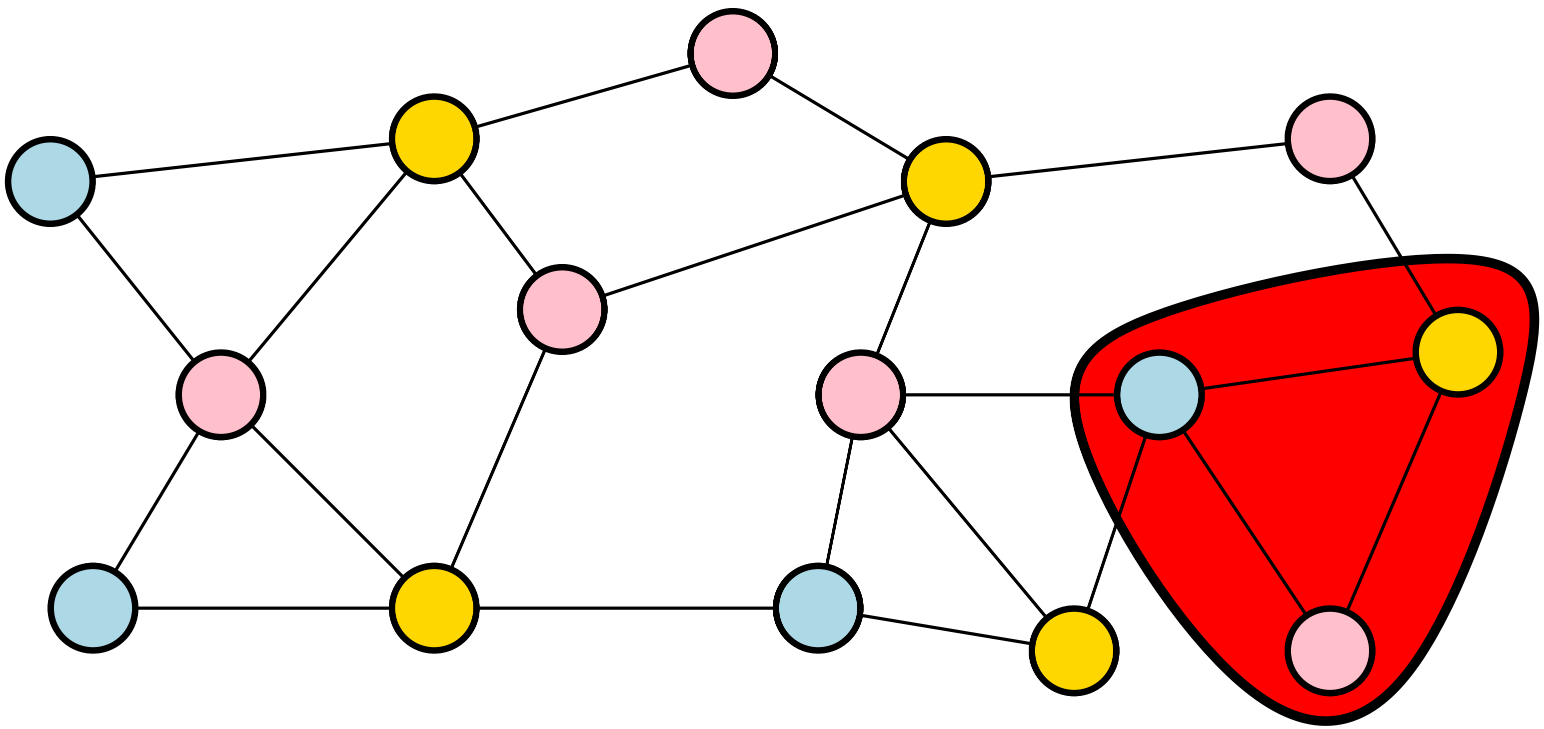
Definition: Given a graph G, the **chromatic number** $\varphi(G)$ of G is the smallest integer k such that there exists a valid k-coloring of G.



$\varphi(G) \leq 3$

Chromatic Number

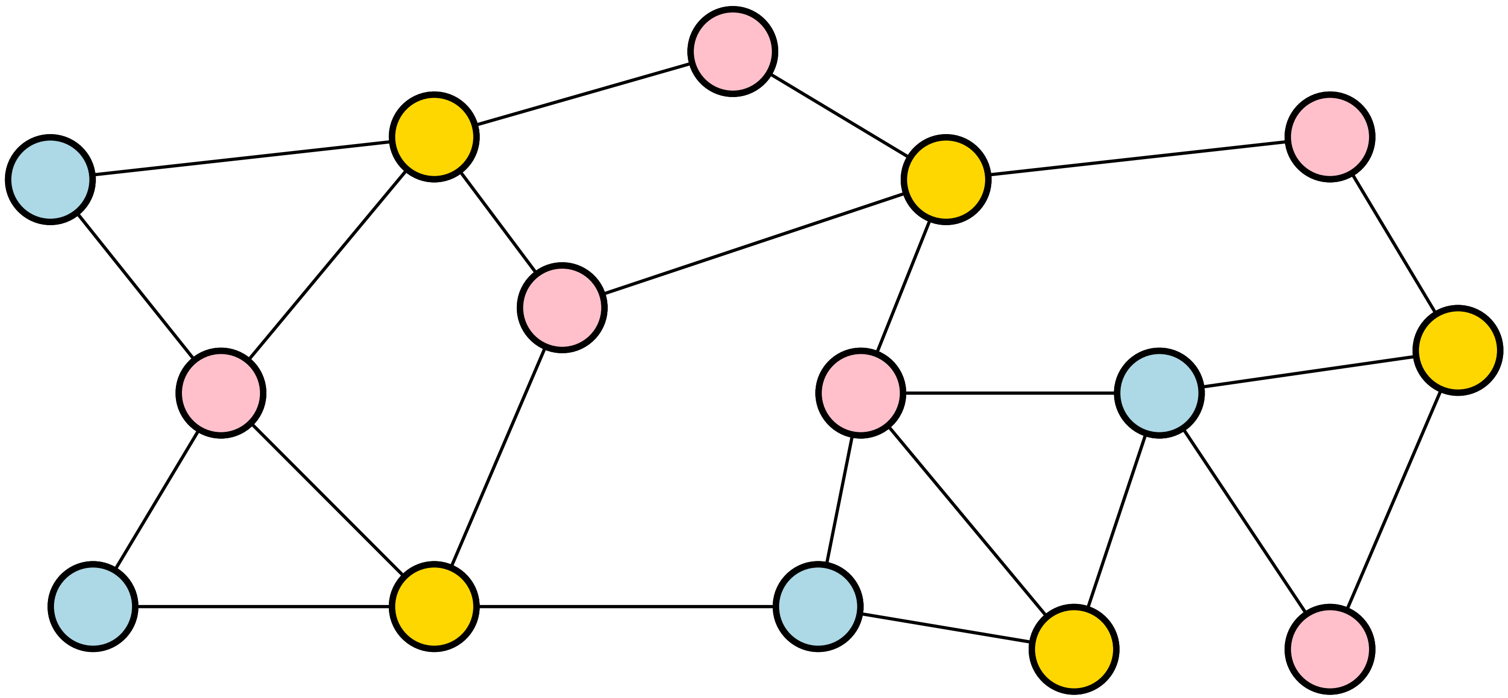
Definition: Given a graph G, the **chromatic number** $\varphi(G)$ of G is the smallest integer k such that there exists a valid k-coloring of G.



$\varphi(G) \leq 3$ $\varphi(G) \geq 3$

Chromatic Number

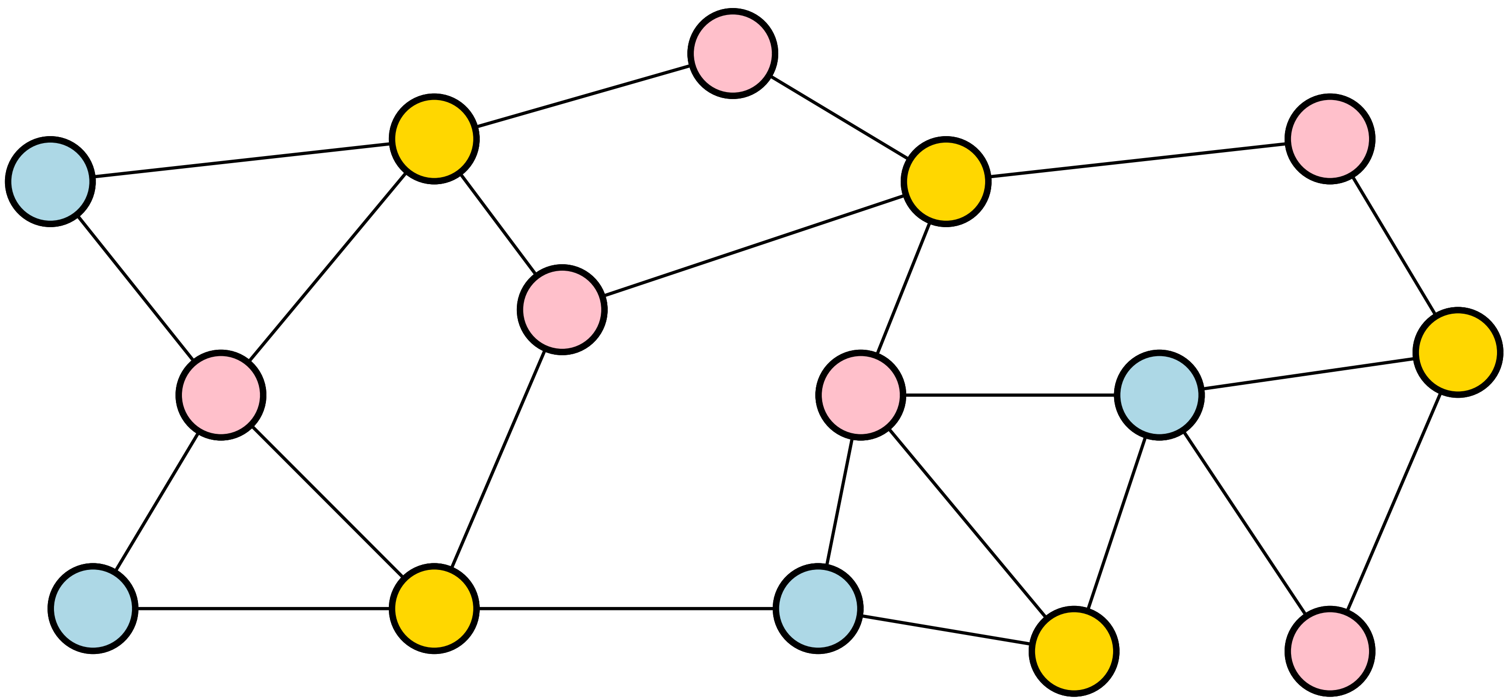
Definition: Given a graph G, the **chromatic number** $\varphi(G)$ of G is the smallest integer k such that there exists a valid k-coloring of G.



$$\varphi(G) = 3$$

Chromatic Number

Definition: Given a graph G, the **chromatic number** $\varphi(G)$ of G is the smallest integer k such that there exists a valid k-coloring of G.



If P $\neq$ NP, the chromatic number of a graph cannot be approximated within a factor of $n^{1-\epsilon}$, for any constant $\epsilon > 0$.

(Where $n = |V|$)

A $(\Delta + 1)$-coloring

Given $v \in V$, let $\delta(v)$ be the degree of v in G.

Let $\Delta = \max_{v \in V} \delta(v)$ be the **maximum degree** of G.

Claim: Any graph G admits a $(\Delta + 1)$-coloring, i.e.,

$$\varphi(G) \leq \Delta + 1$$

A $(\Delta + 1)$-coloring

Given $v \in V$, let $\delta(v)$ be the degree of v in G.

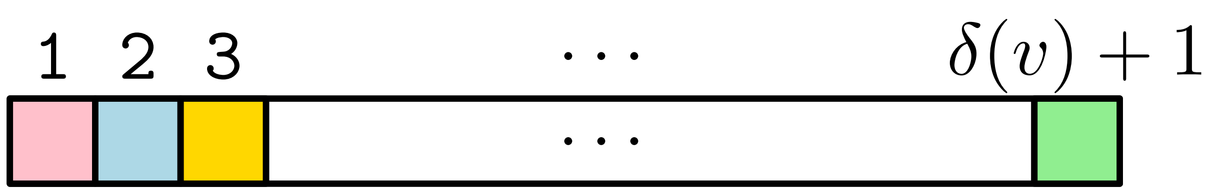
Let $\Delta = \max_{v \in V} \delta(v)$ be the **maximum degree** of G.

Claim: Any graph G admits a $(\Delta + 1)$-coloring, i.e.,

$$\varphi(G) \leq \Delta + 1$$

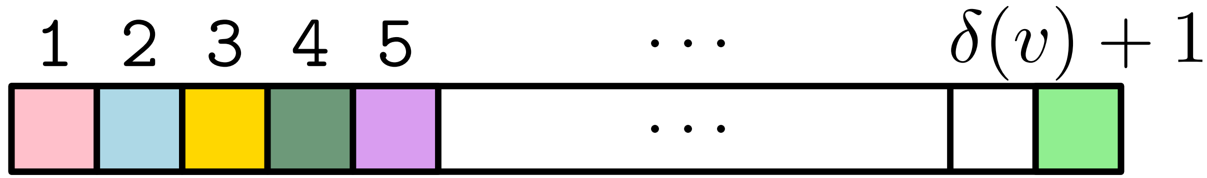
Proof: we will show an algorithm that computes a $\Delta + 1$ coloring.

Each node v keeps a *palette* of $\delta(v) + 1$ avaliable colors: $1, 2, \ldots, \delta(v) + 1$.



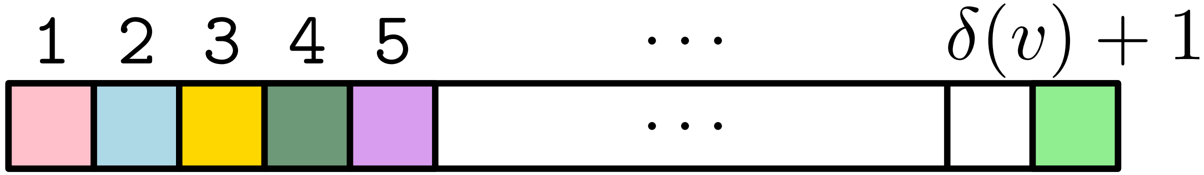
Computing a $(\Delta + 1)$-coloring

v's palette:

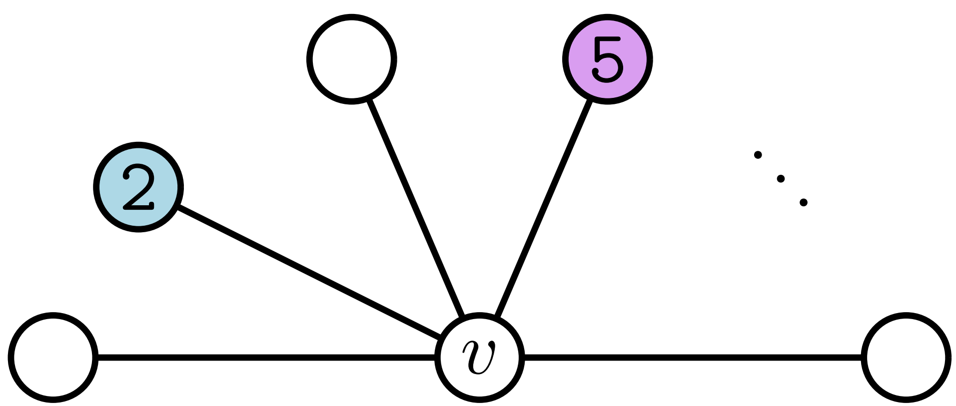


Computing a $(\Delta + 1)$-coloring

v's palette:

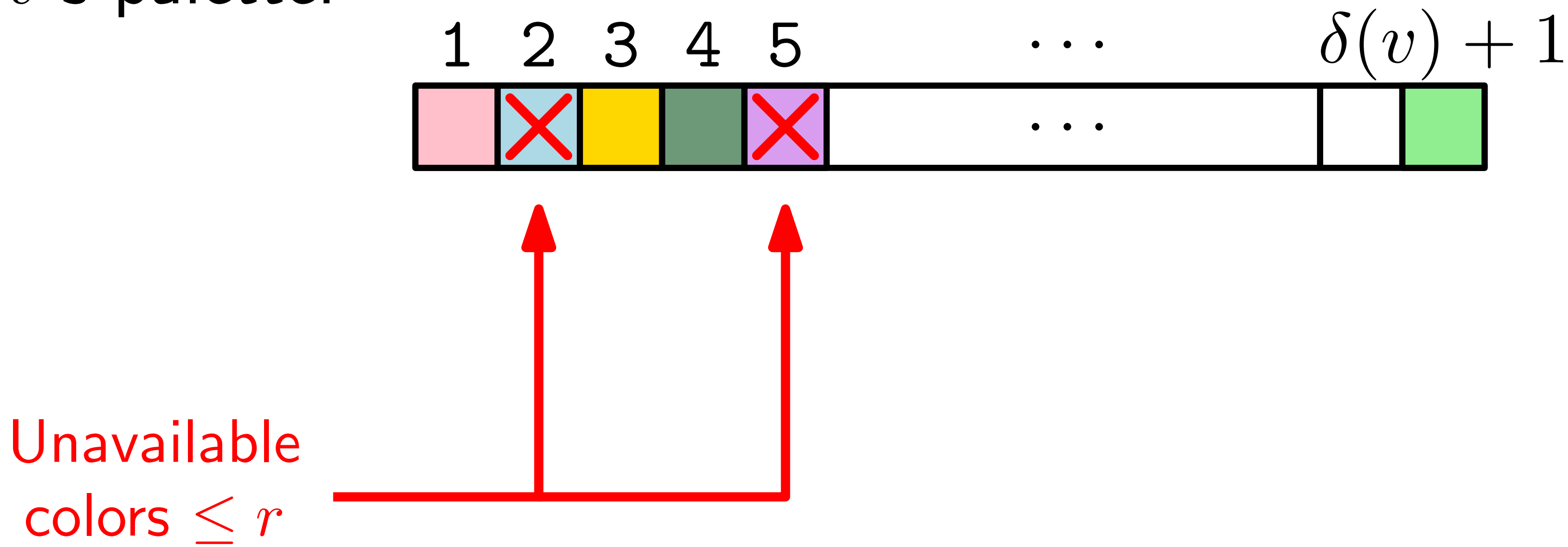


If r neighbors of v have already been colored:

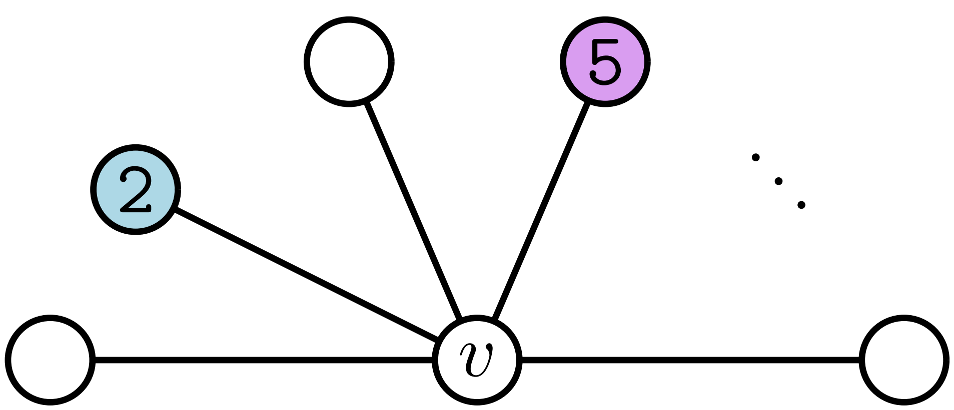


Computing a $(\Delta + 1)$-coloring

v's palette:

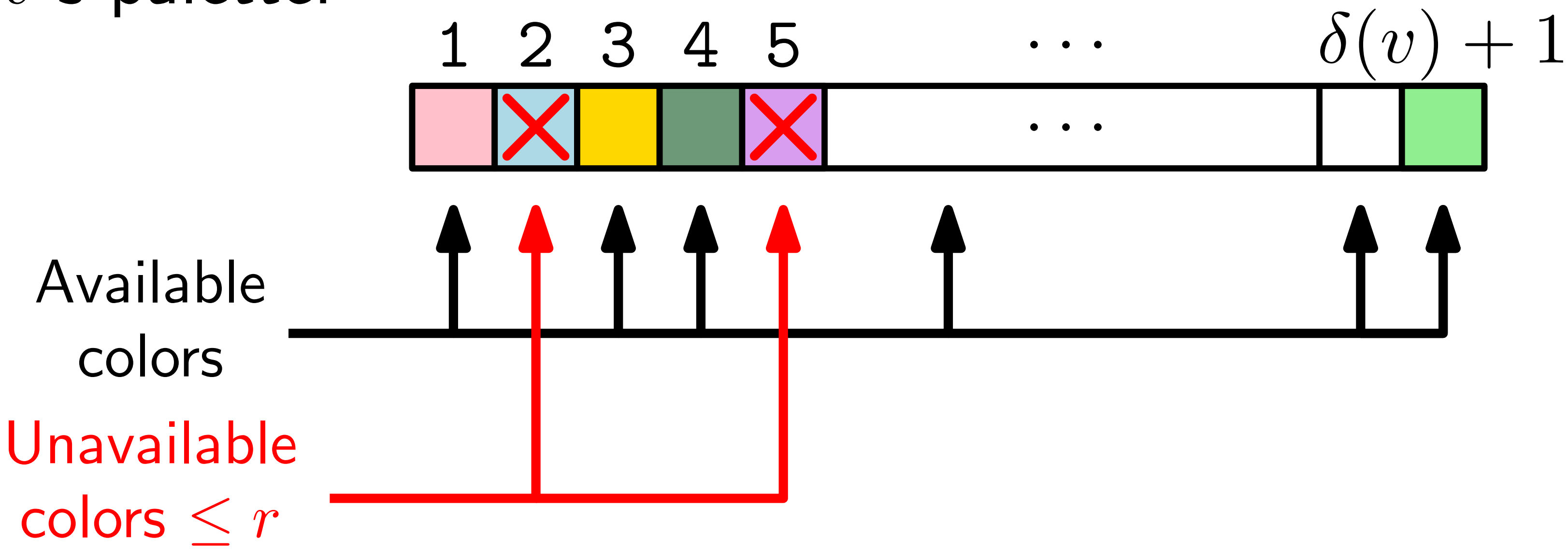


If r neighbors of v have already been colored:

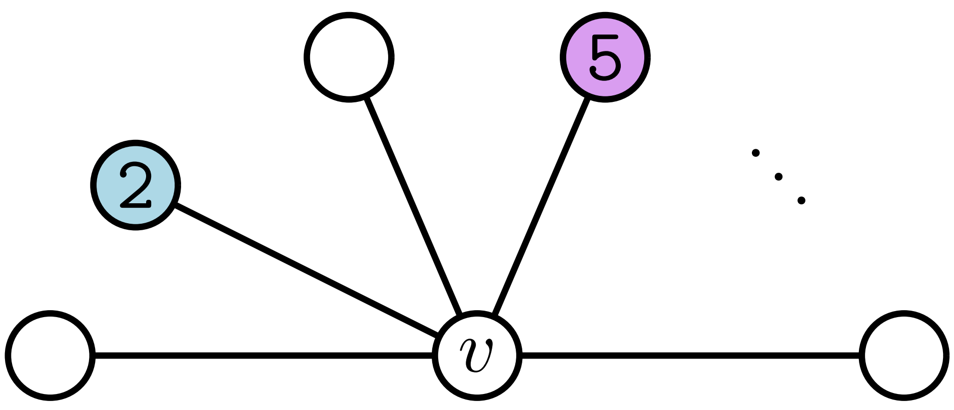


Computing a $(\Delta + 1)$-coloring

v's palette:

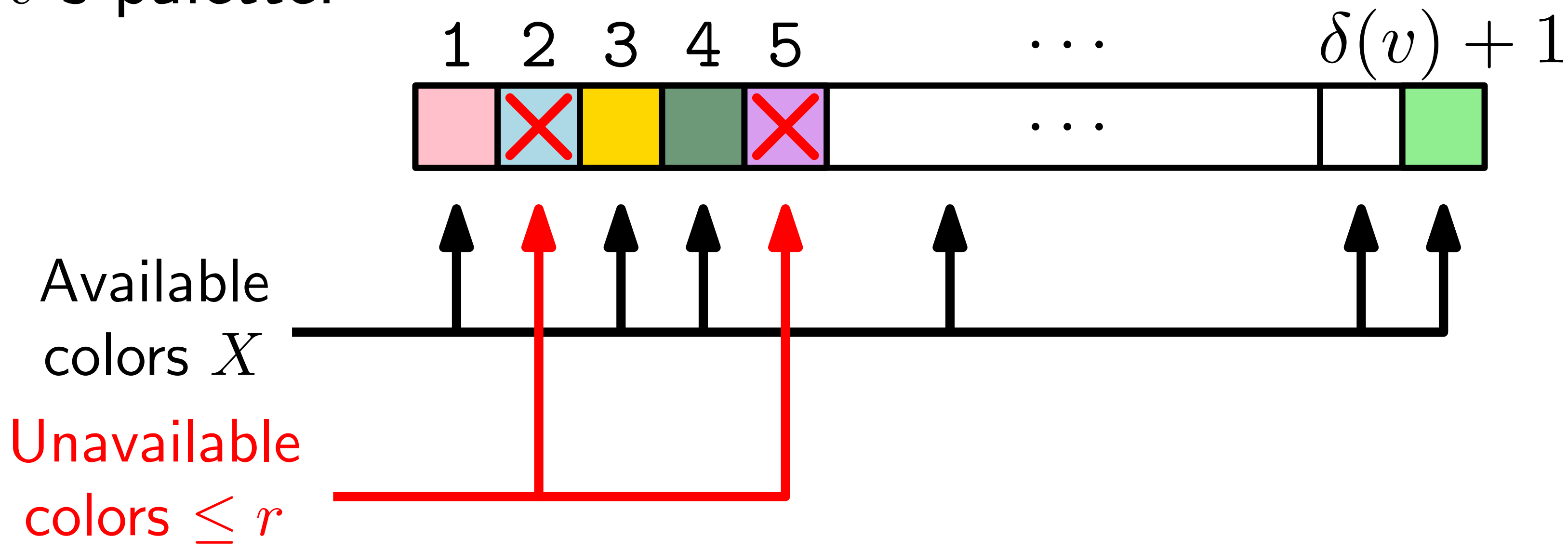


If r neighbors of v have already been colored:



Computing a $(\Delta + 1)$-coloring

v's palette:

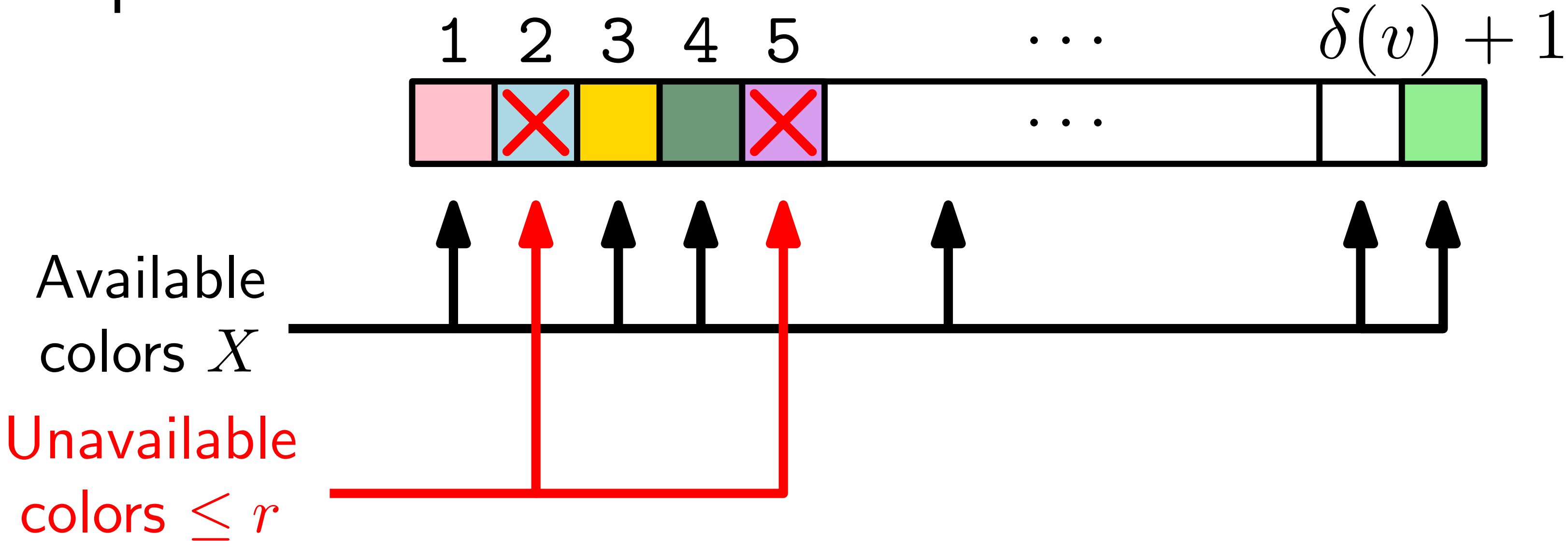


Number of available colors in v's palette:

$$X = \delta(v) + 1 - r$$

Computing a $(\Delta + 1)$-coloring

v's palette:

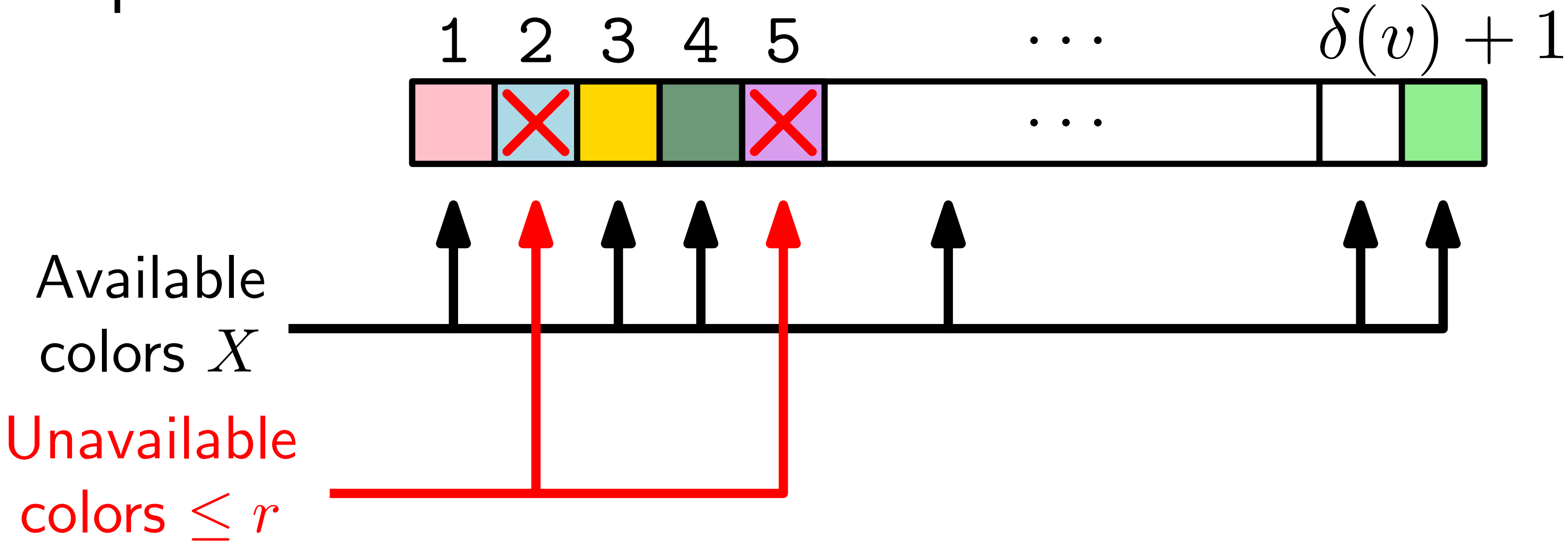


Number of available colors in v's palette:

$$X = \delta(v) + 1 - r \geq \delta(v) + 1 - \delta(v)$$

Computing a $(\Delta + 1)$-coloring

v's palette:



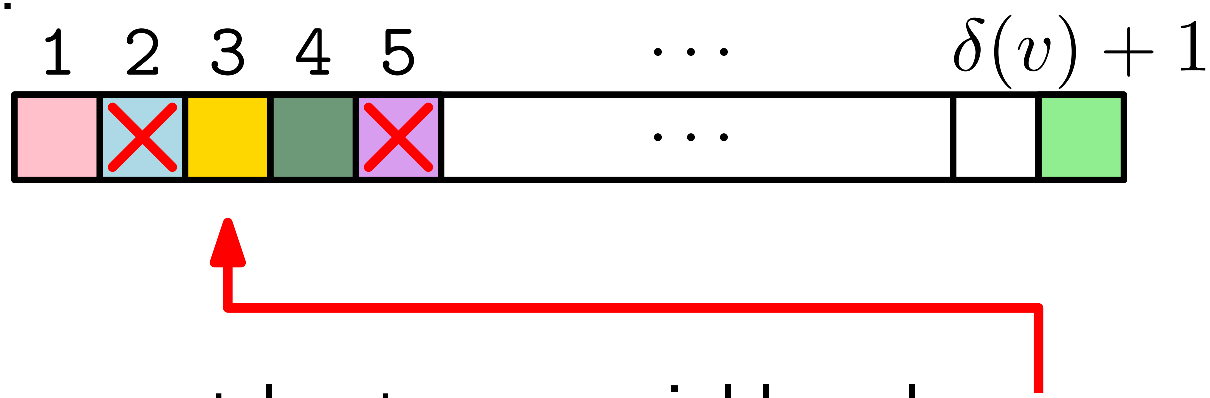
Number of available colors in v's palette:

$$X = \delta(v) + 1 - r \geq \delta(v) + 1 - \delta(v) = 1$$

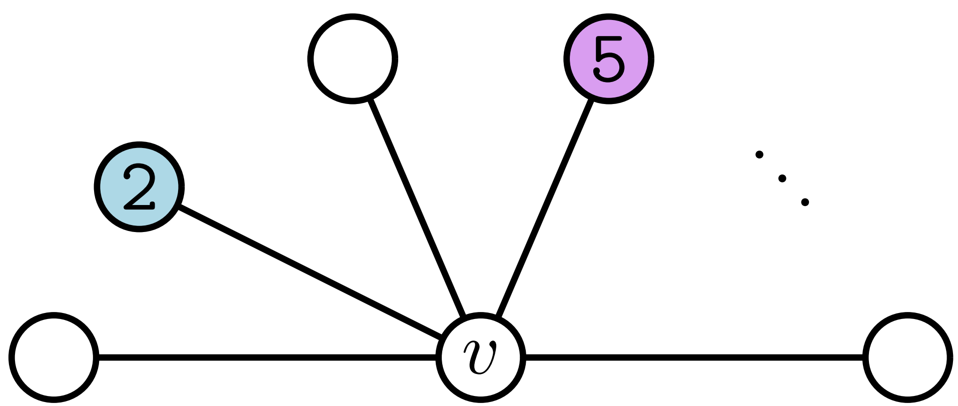
There is always at least one available color.

Computing a $(\Delta + 1)$-coloring

v's palette:

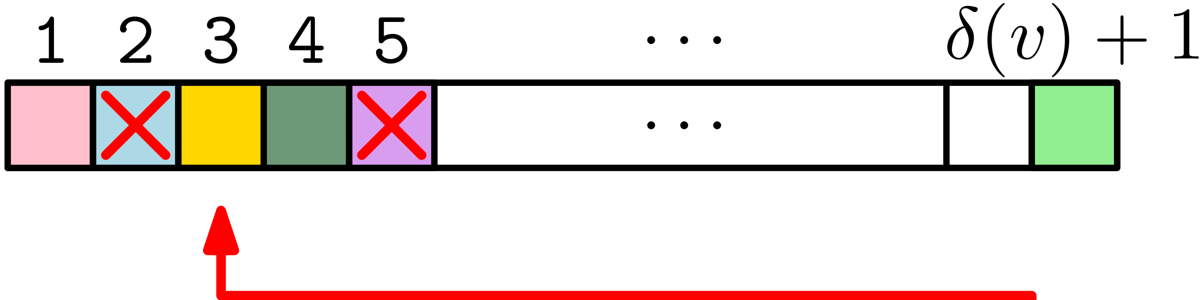


There is always at least one avaiable color c.

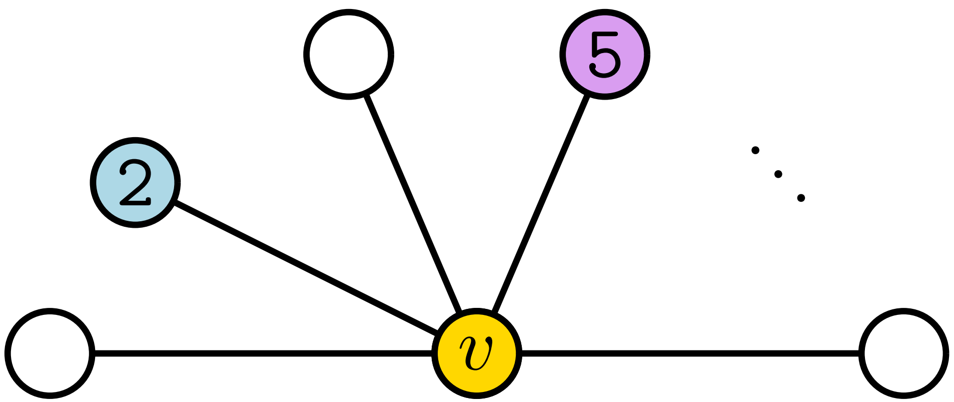


Computing a $(\Delta + 1)$-coloring

v's palette:



There is always at least one avaiable color c.



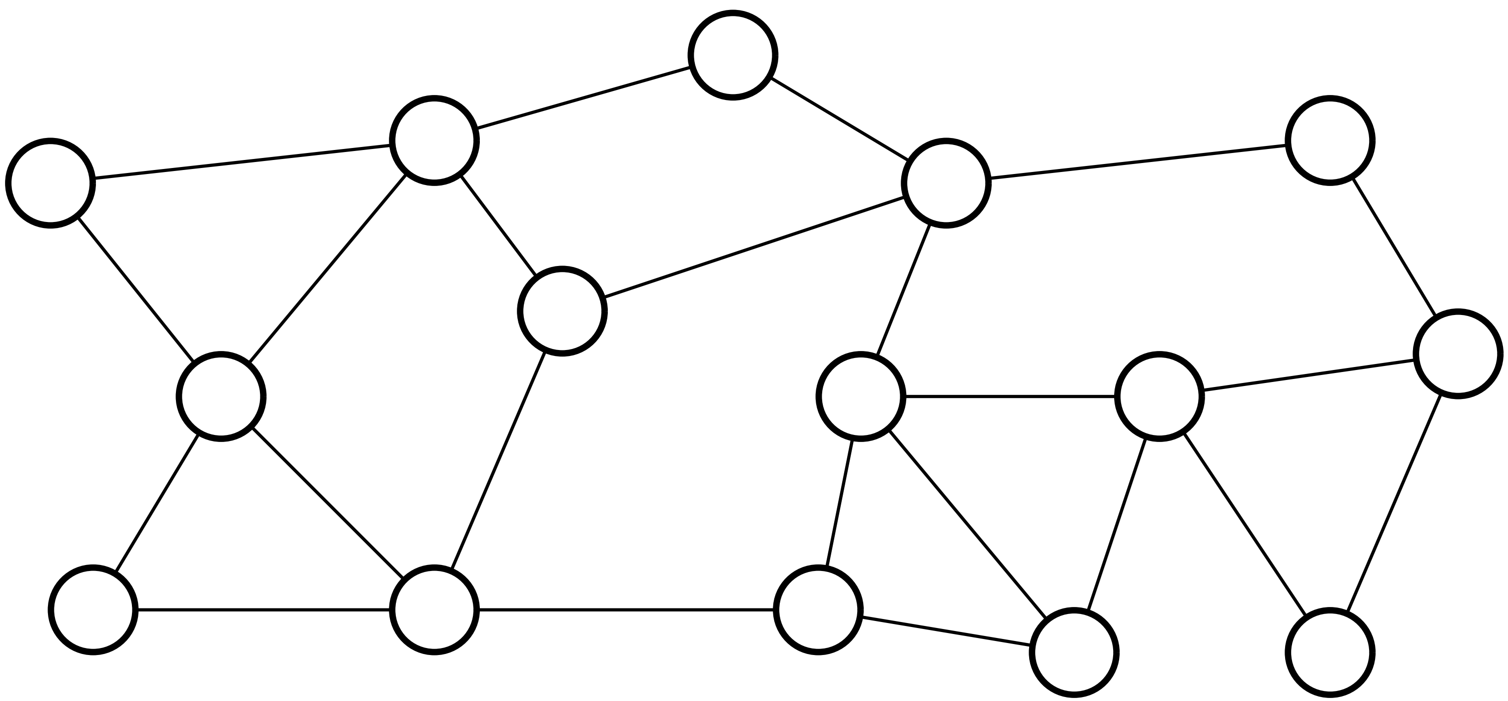
Vertex v can pick color c for itself!

Sequential $(\Delta + 1)$-coloring algorithm

- For each node v: create a palette of $\delta(v) + 1$ colors.
- Mark all colors (in all palettes) as available.
- While $\exists$ uncolored node v:
  - Let c be any available color from v's palette
    (recall that such a color always exists)
  - Color v with color c
  - For every neighbor u of v:
    - Mark c as unavailable in u's palette

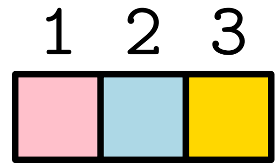
$\square$

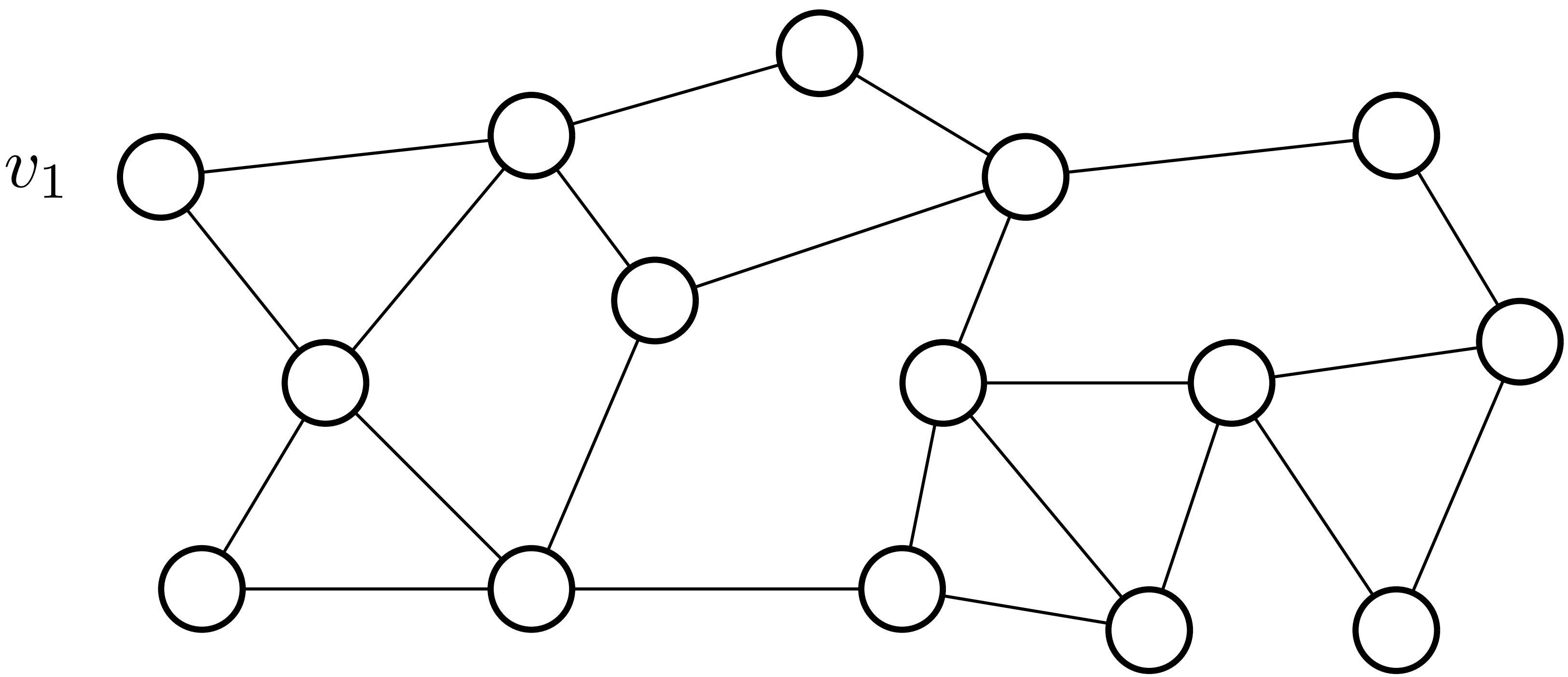
A sample execution



A sample execution

v_1's palette

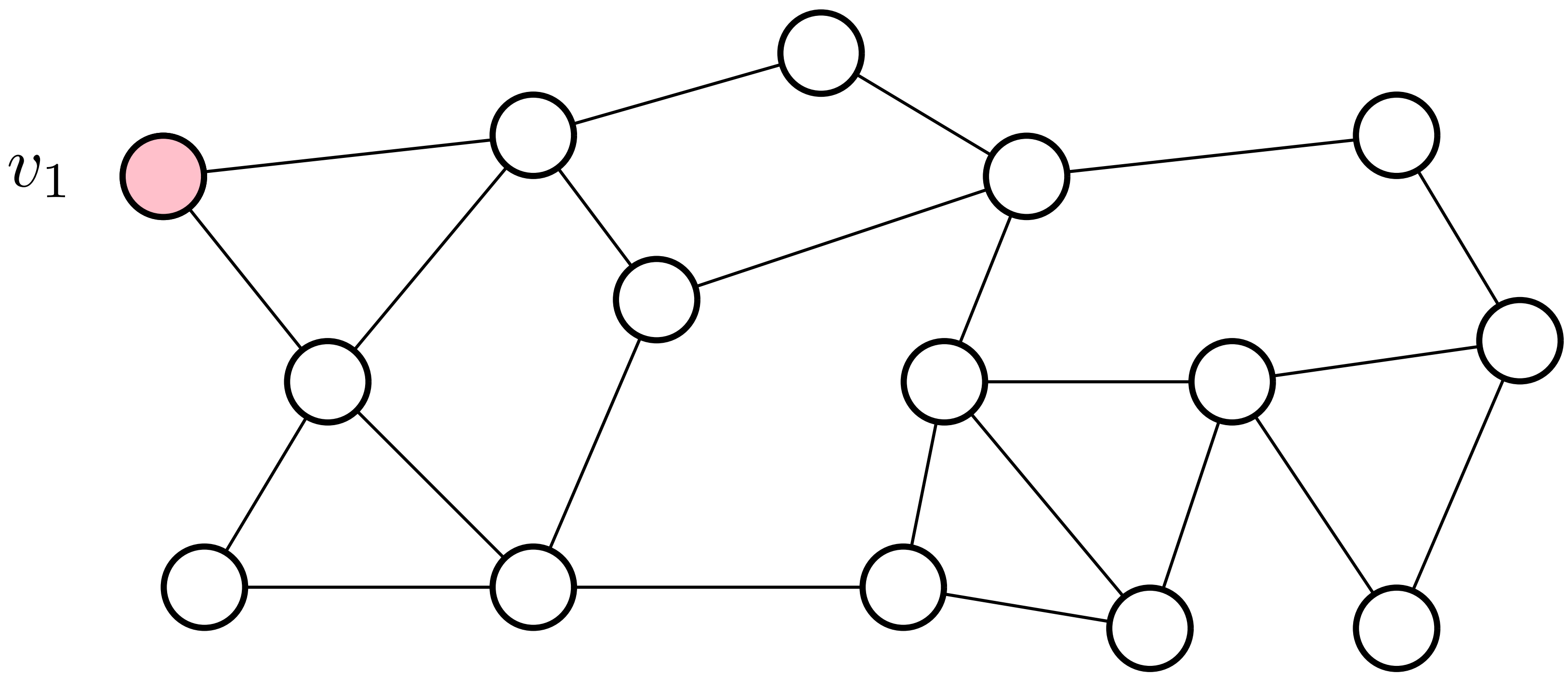




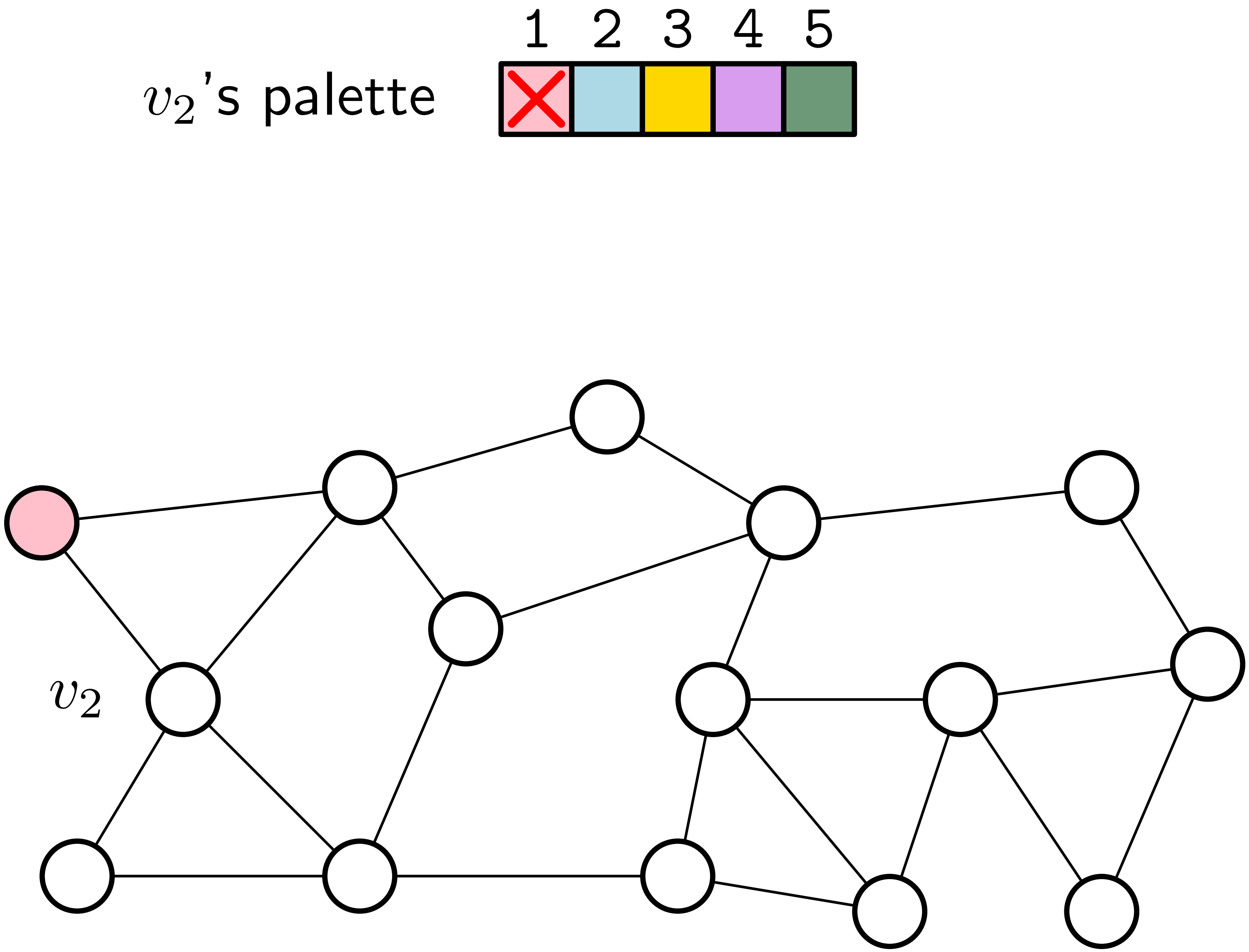
A sample execution

v_1's palette

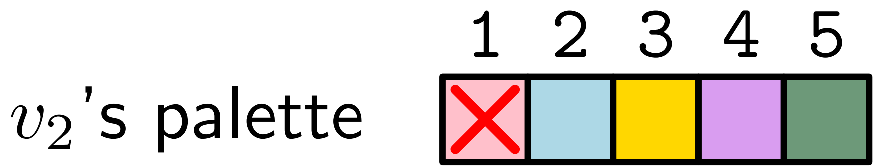
1	2	3

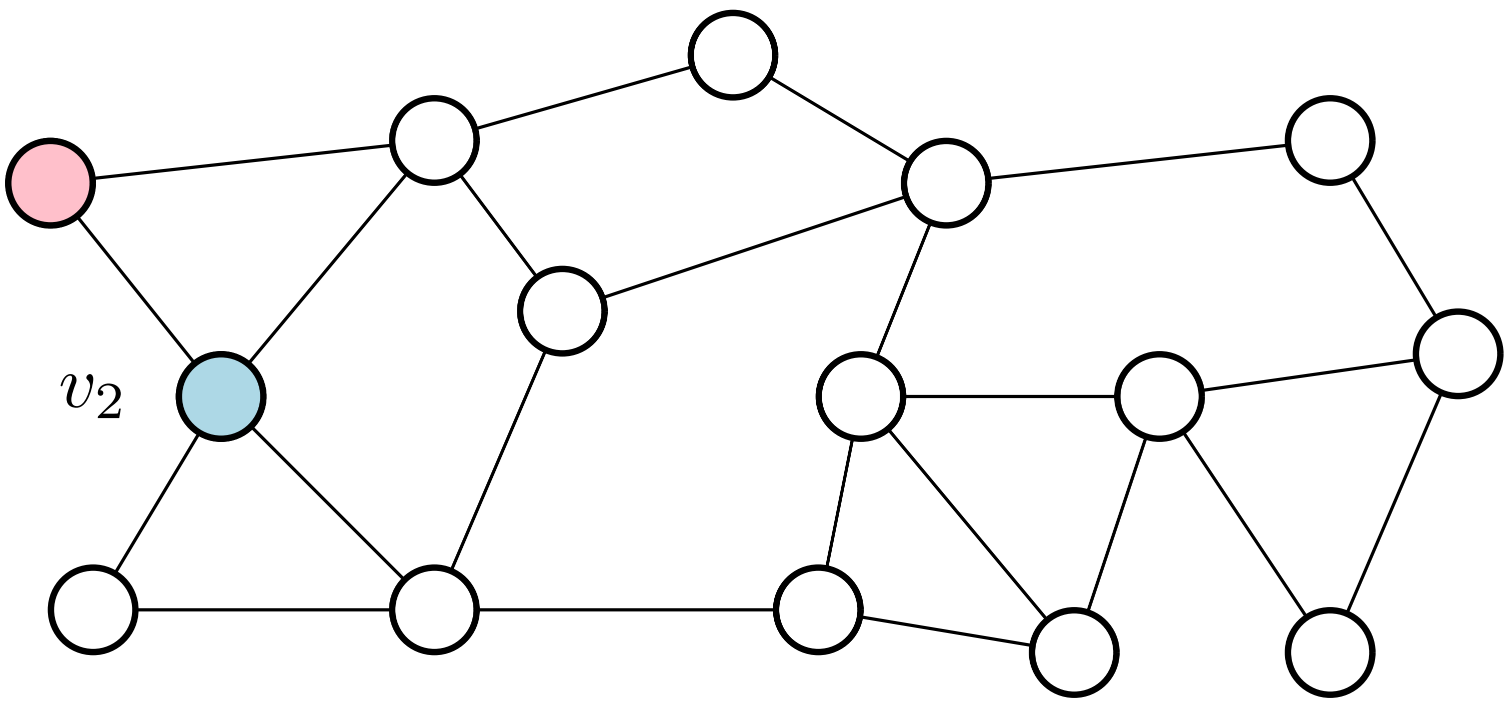


A sample execution



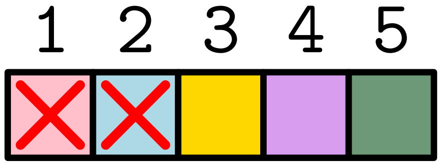
A sample execution

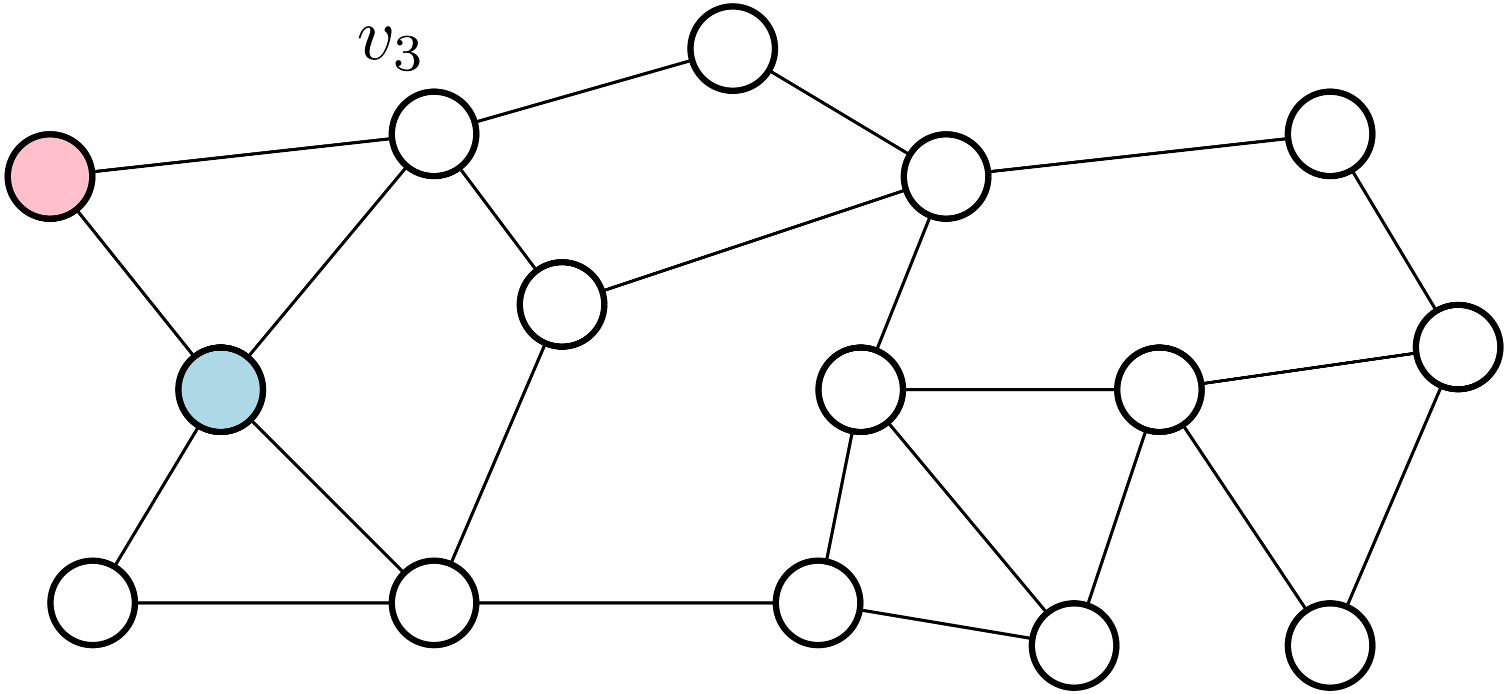




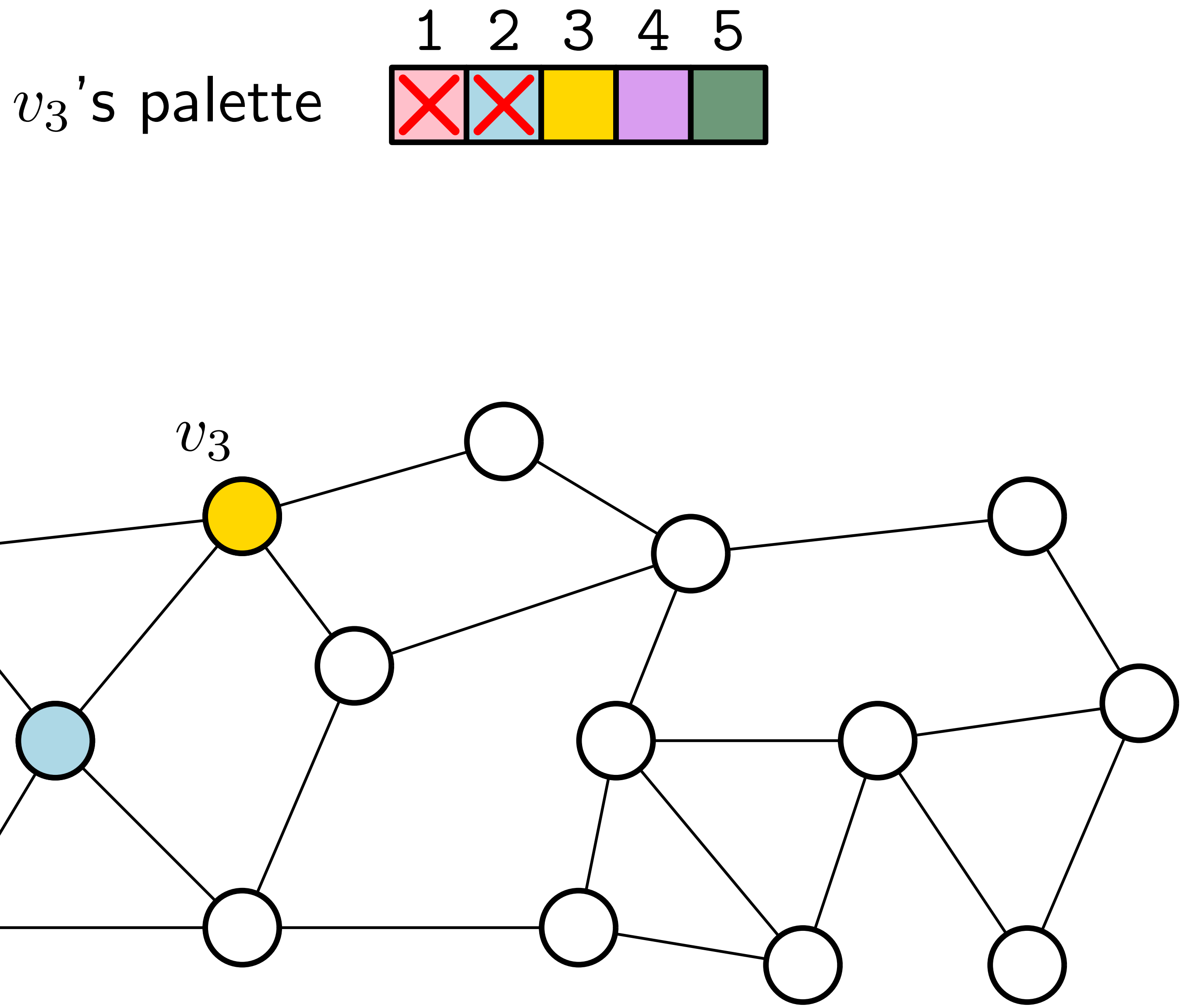
A sample execution

v_3's palette

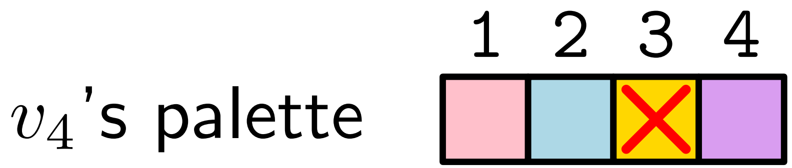


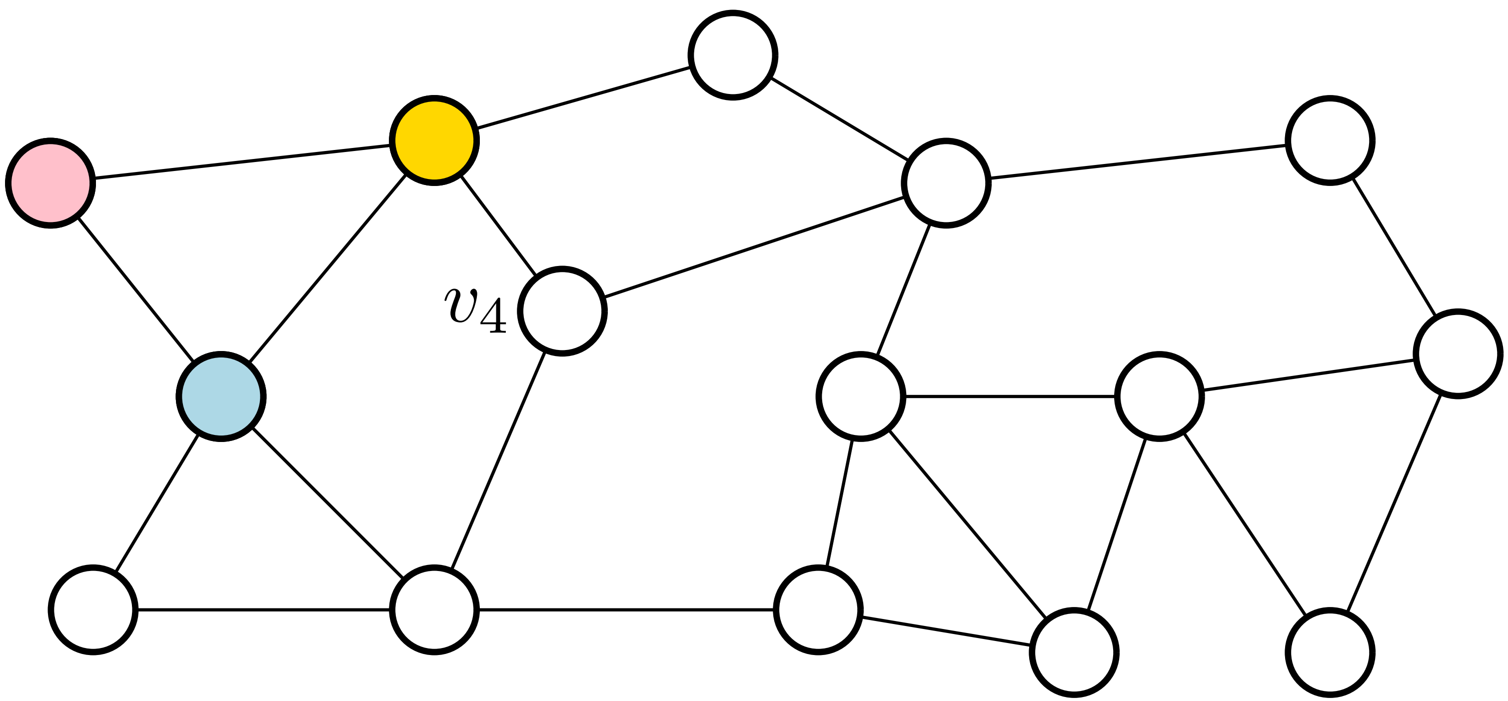


A sample execution



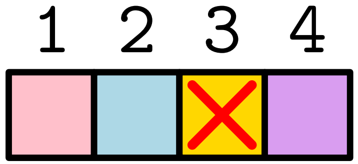
A sample execution

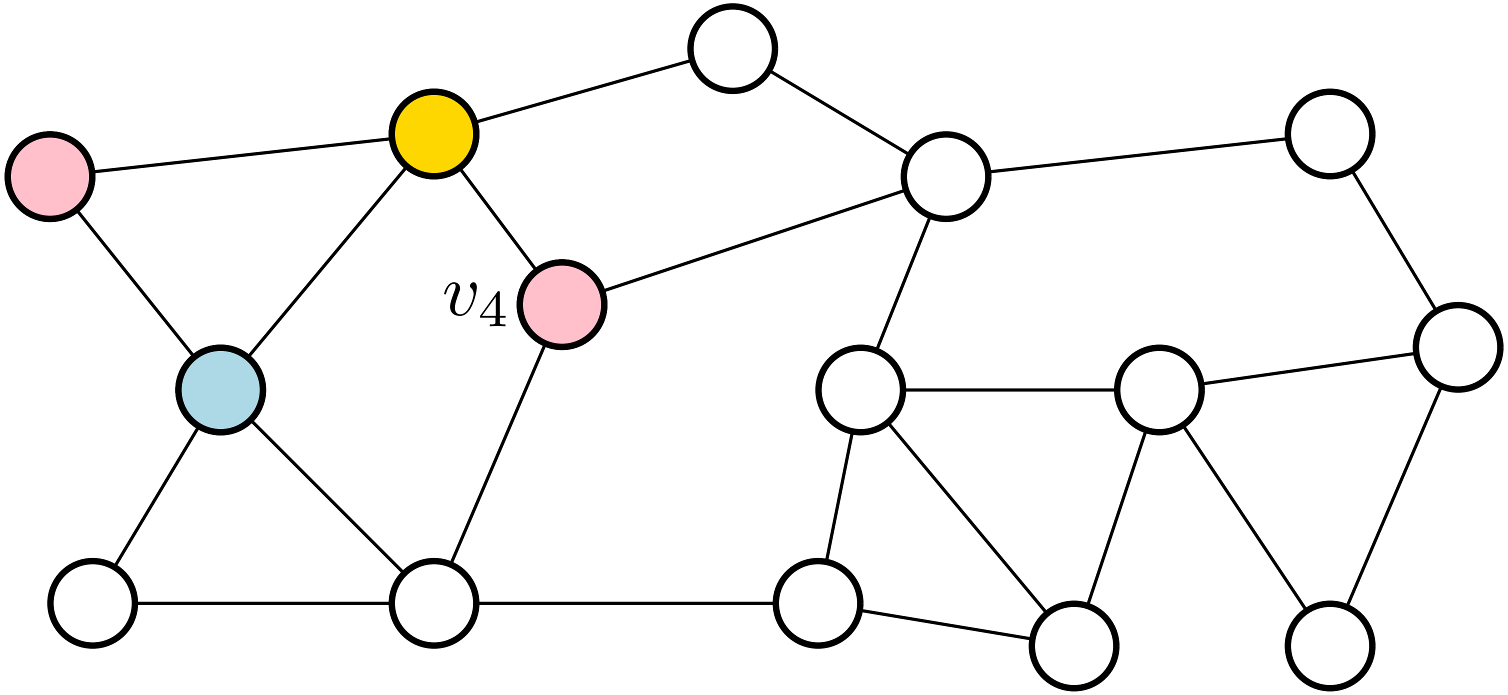




A sample execution

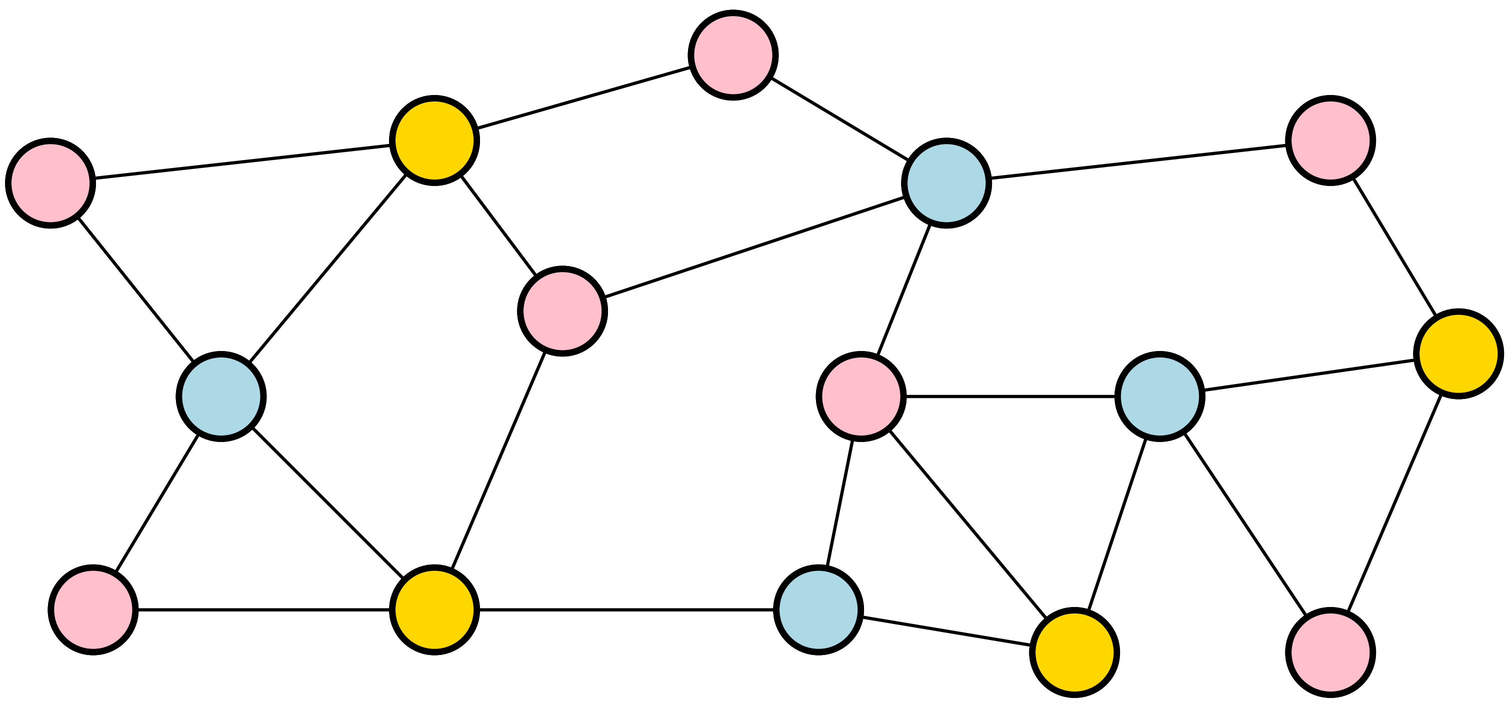
v_4's palette





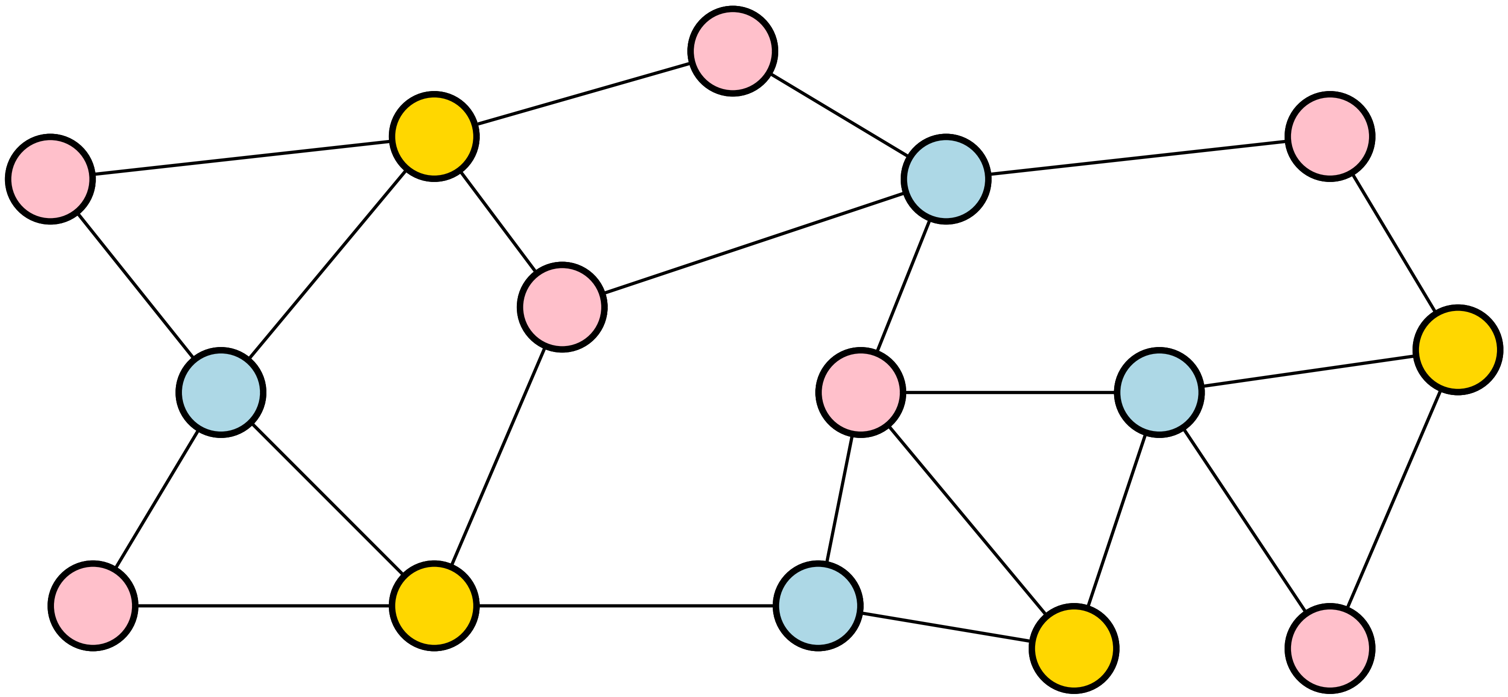
A sample execution

Termination



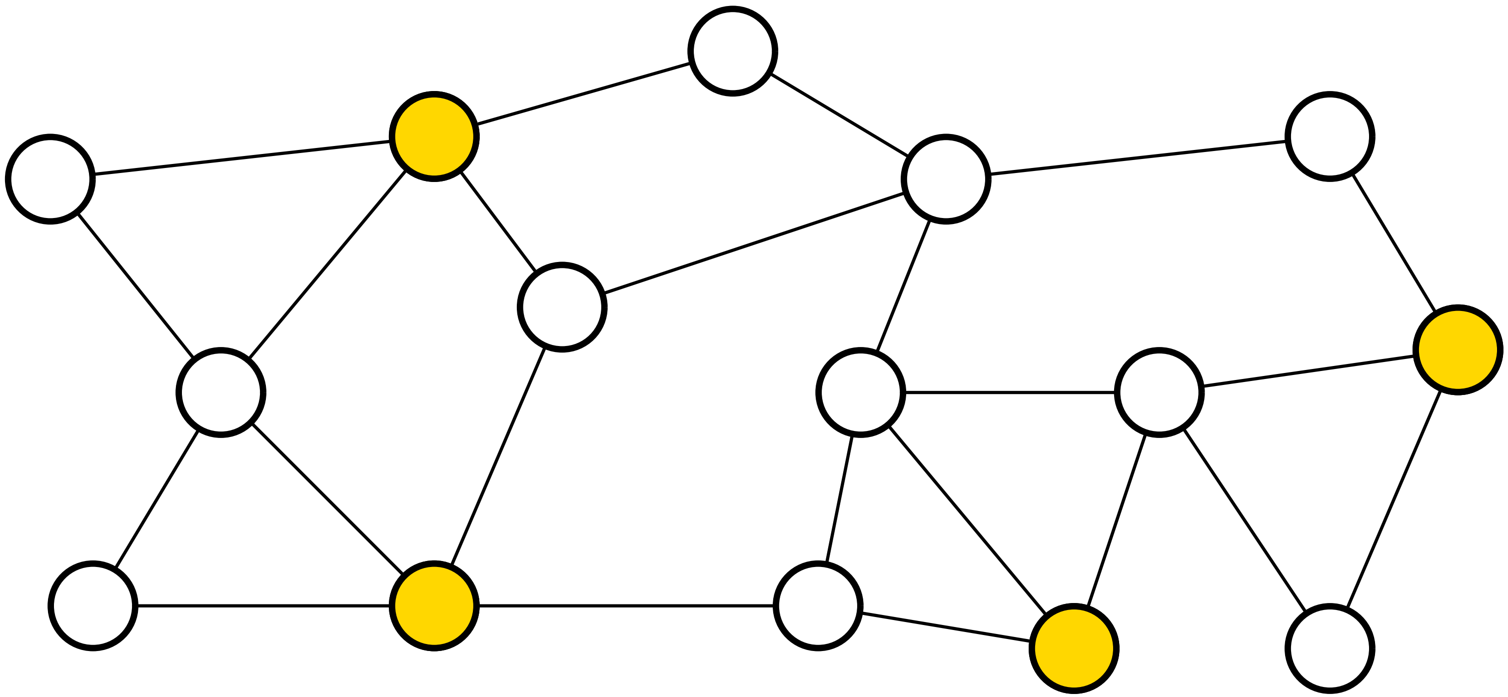
Coloring and Independent Sets

In any valid coloring, the nodes of the same color form an independent set.



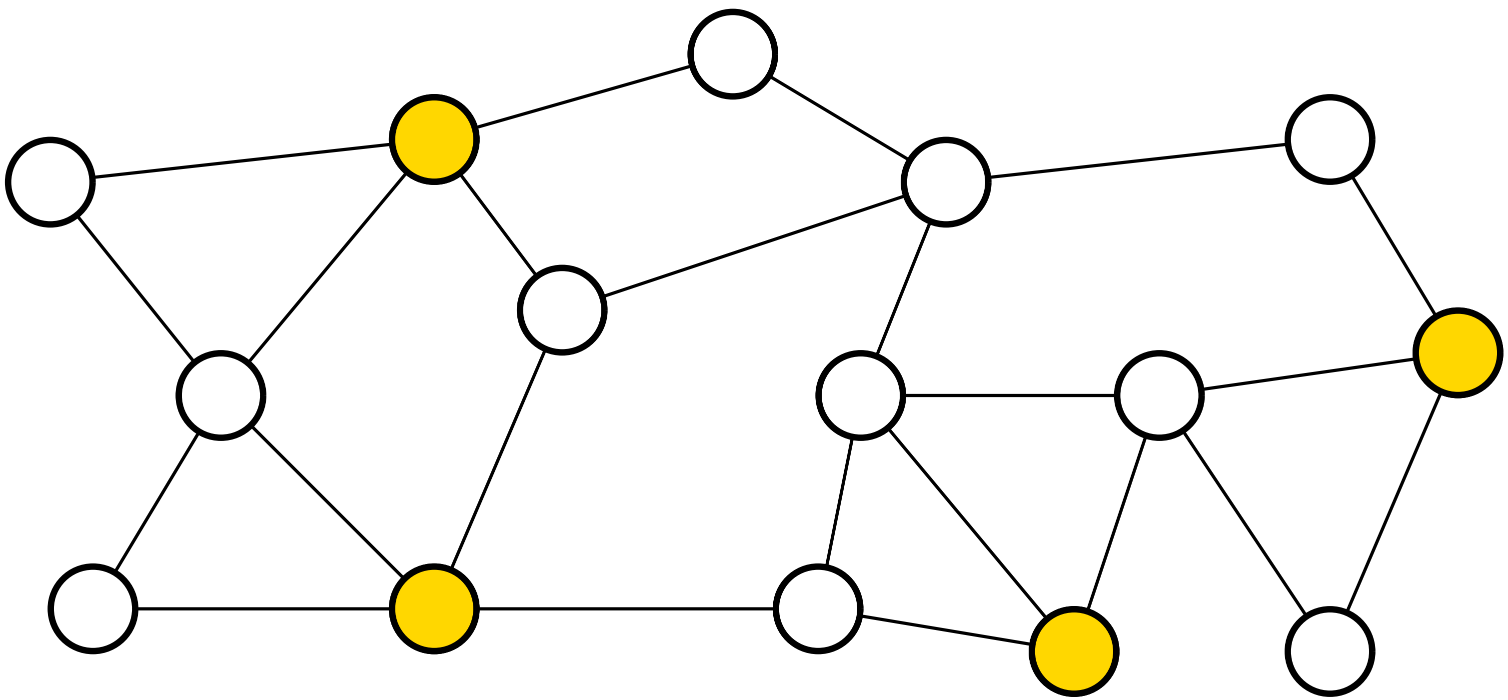
Coloring and Independent Sets

In any valid coloring, the nodes of the same color form an independent set.



Coloring and Independent Sets

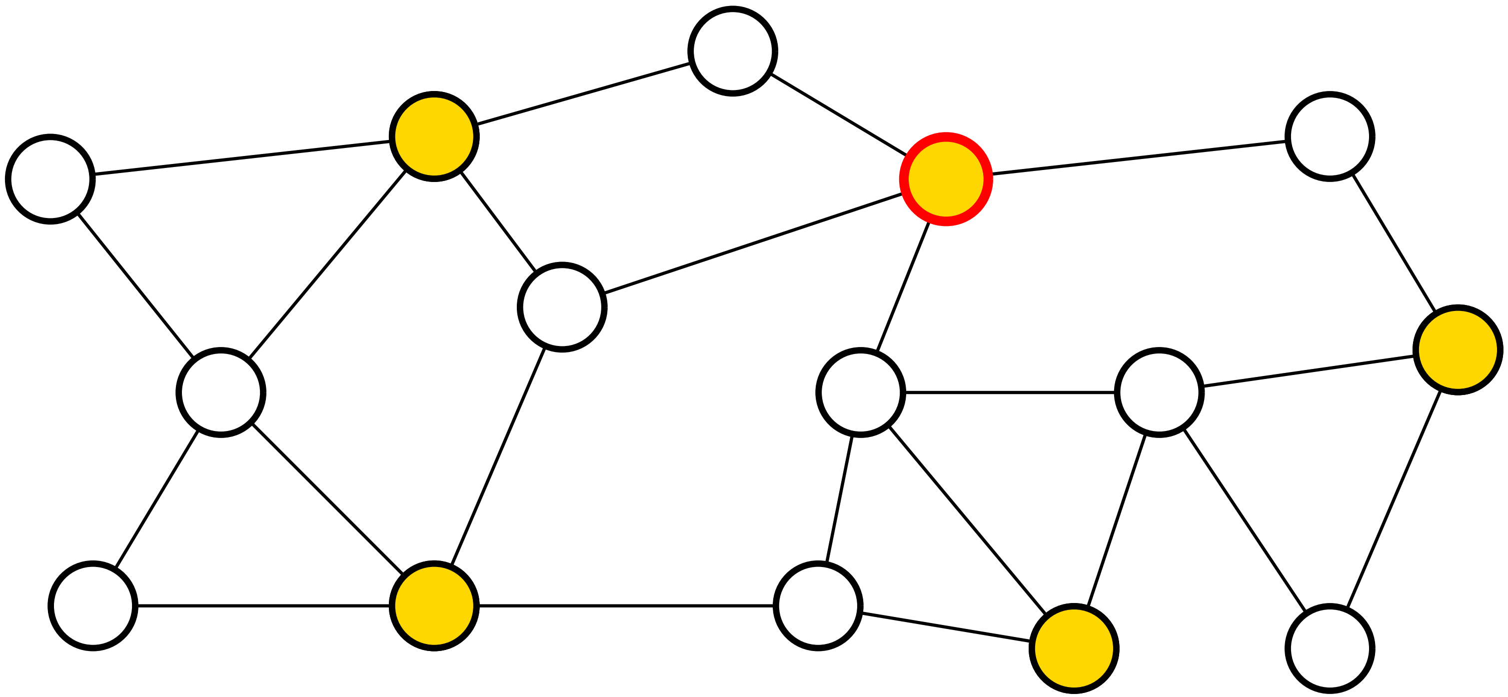
In any valid coloring, the nodes of the same color form an independent set.



Such an independent set is not necessarily maximal

Coloring and Independent Sets

In any valid coloring, the nodes of the same color form an independent set.



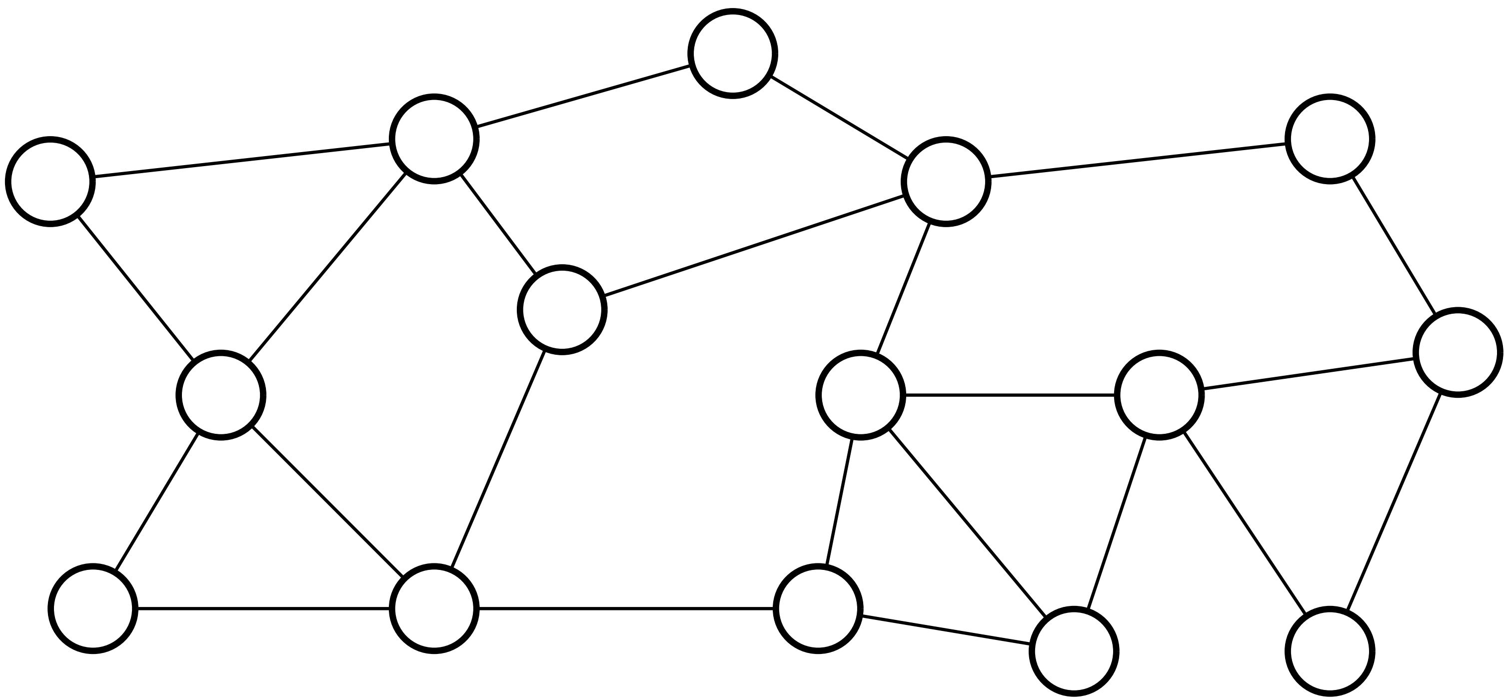
Such an independent set is not necessarily maximal

Coloring via Maximal Independent Sets

- $c \leftarrow 1$
- While $\exists$ uncolored nodes in G:
  - Find a MIS $\mathcal{I}$ of the subraph of G induced by the *uncolored* nodes.
  - Assign color c to all nodes in $\mathcal{I}$
  - $c \leftarrow c + 1$

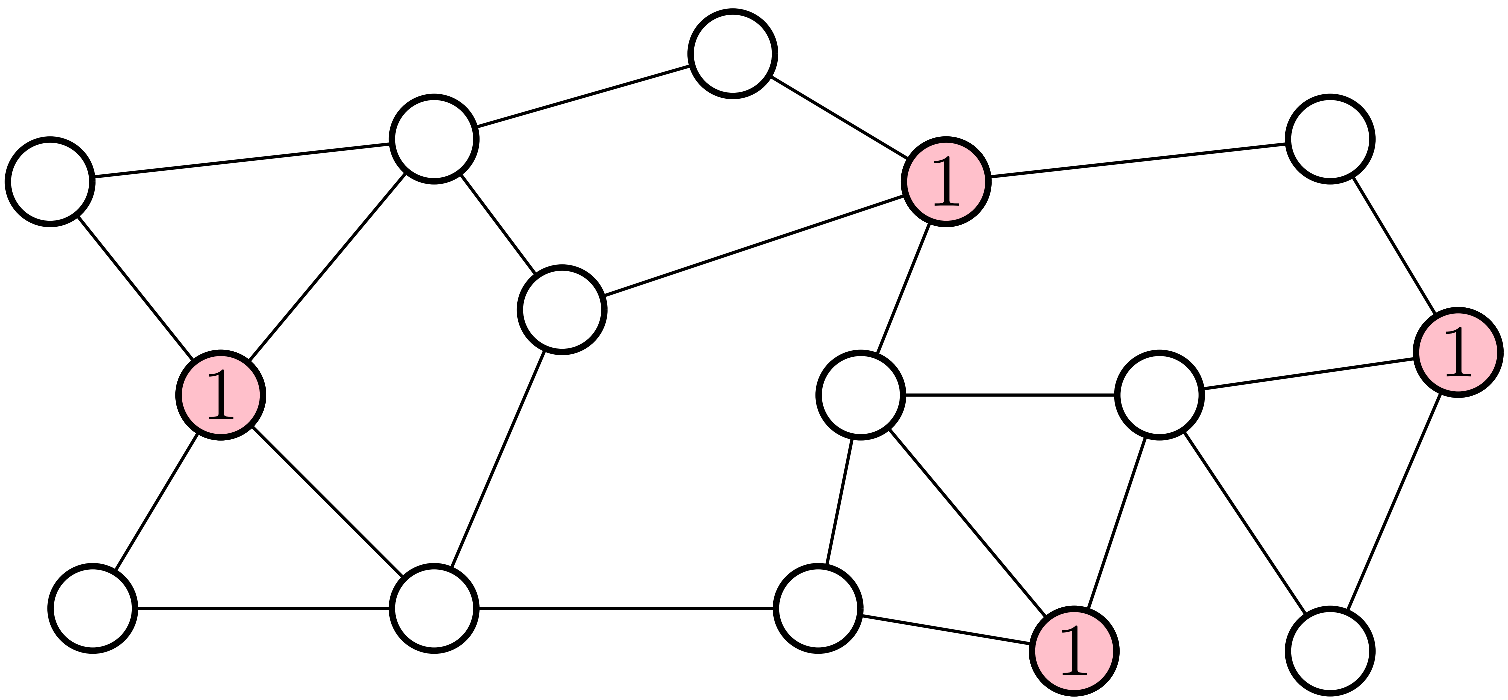
A sample execution

Initially all nodes are uncolored



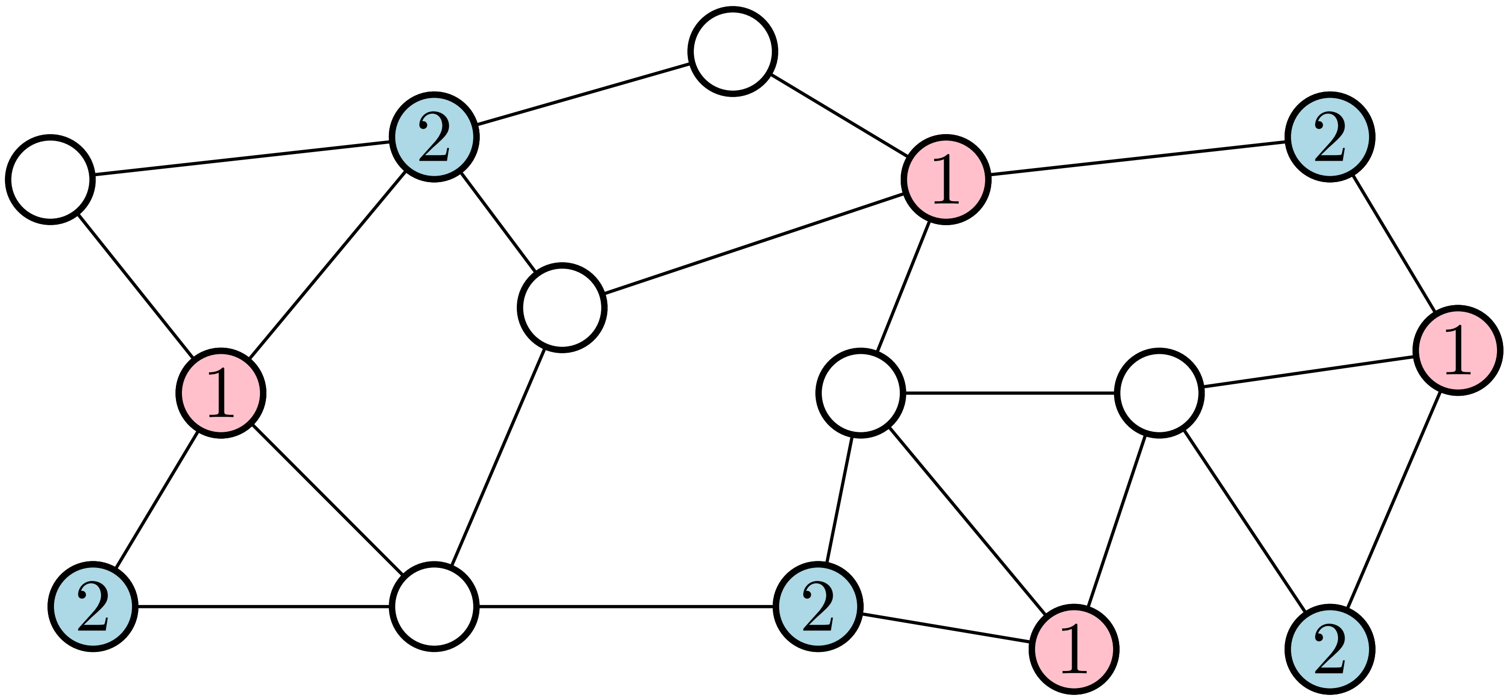
A sample execution

Iteration 1: Find a MIS $\mathcal{I}$ of the uncolored nodes and assign color 1 to the nodes in $\mathcal{I}$



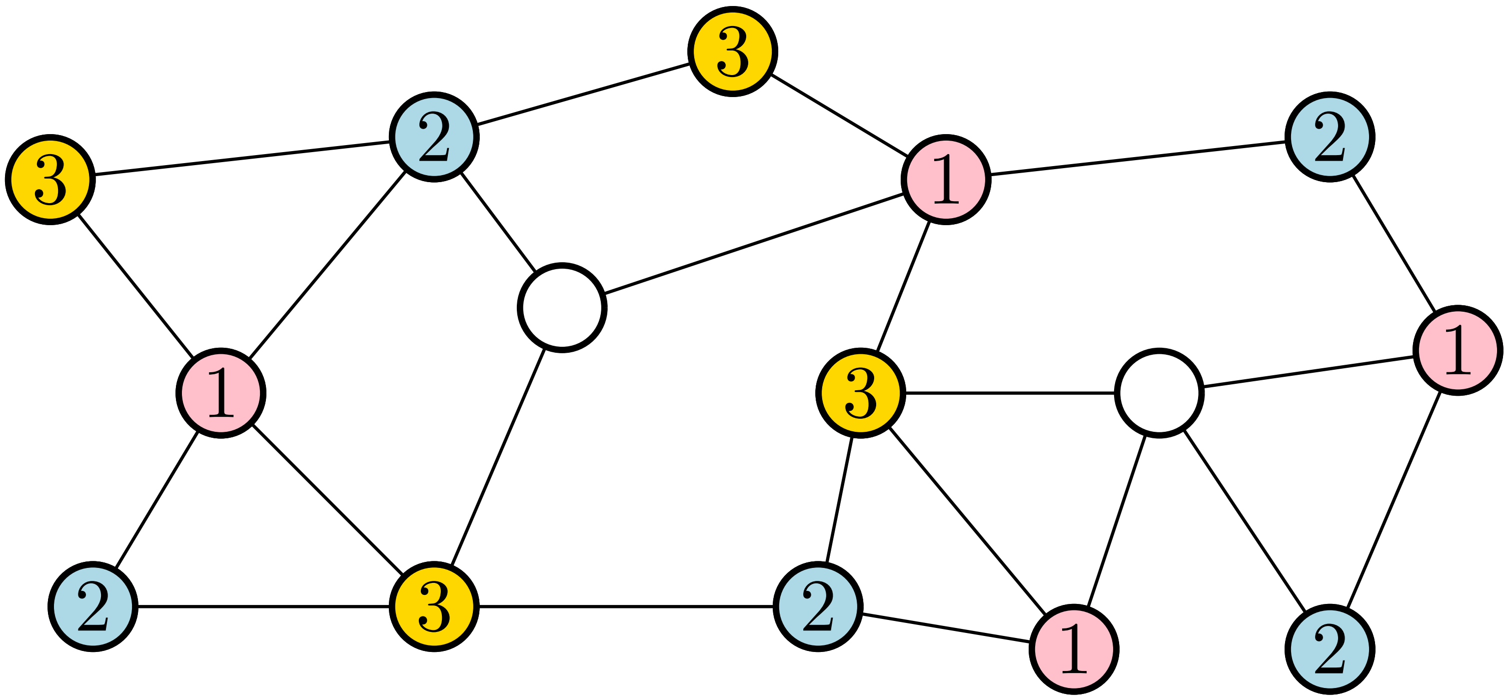
A sample execution

Iteration 2: Find a MIS $\mathcal{I}$ of the uncolored nodes and assign color 2 to the nodes in $\mathcal{I}$



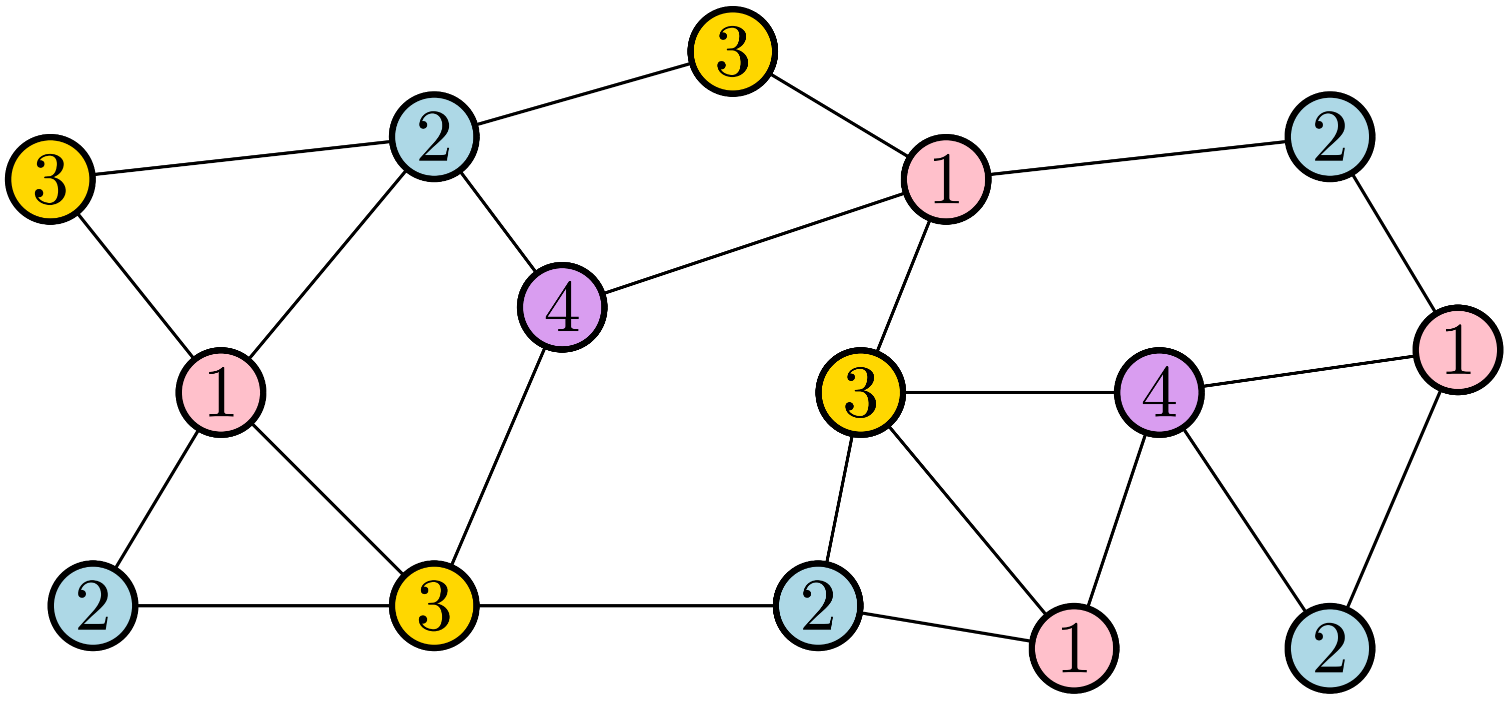
A sample execution

Iteration 3: Find a MIS $\mathcal{I}$ of the uncolored nodes and assign color 3 to the nodes in $\mathcal{I}$



A sample execution

Iteration 4: Find a MIS $\mathcal{I}$ of the uncolored nodes and assign color 4 to the nodes in $\mathcal{I}$



Analysis

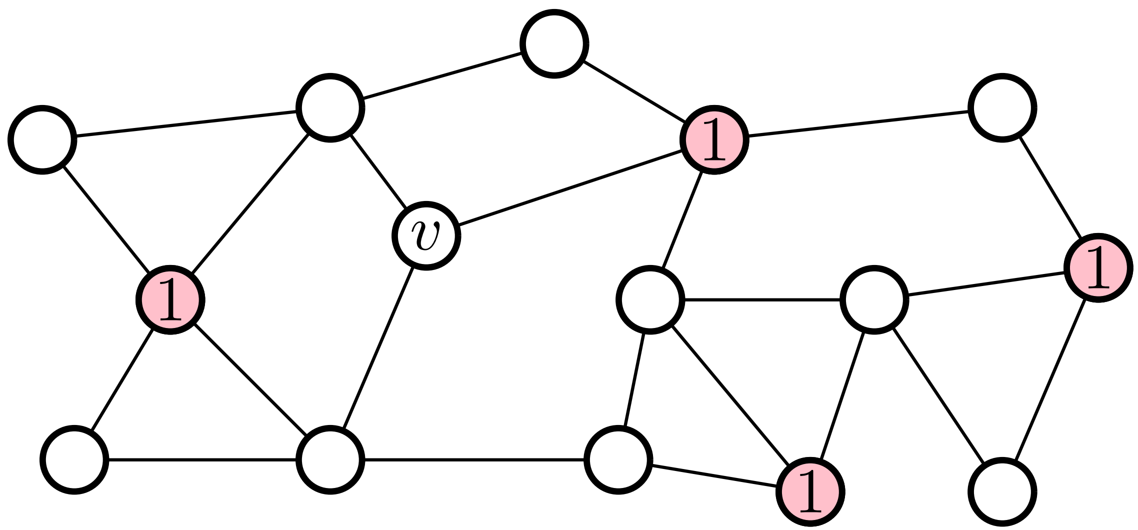
Lemma: The algorithm terminates in at most $\Delta + 1$ iterations (i.e., it uses at most $\Delta + 1$ colors).

Analysis

Lemma: The algorithm terminates in at most $\Delta + 1$ iterations (i.e., it uses at most $\Delta + 1$ colors).

Proof:

- At the end of each iteration, each uncolored node v is adjacent to at least one node in the MIS $\mathcal{I}$.

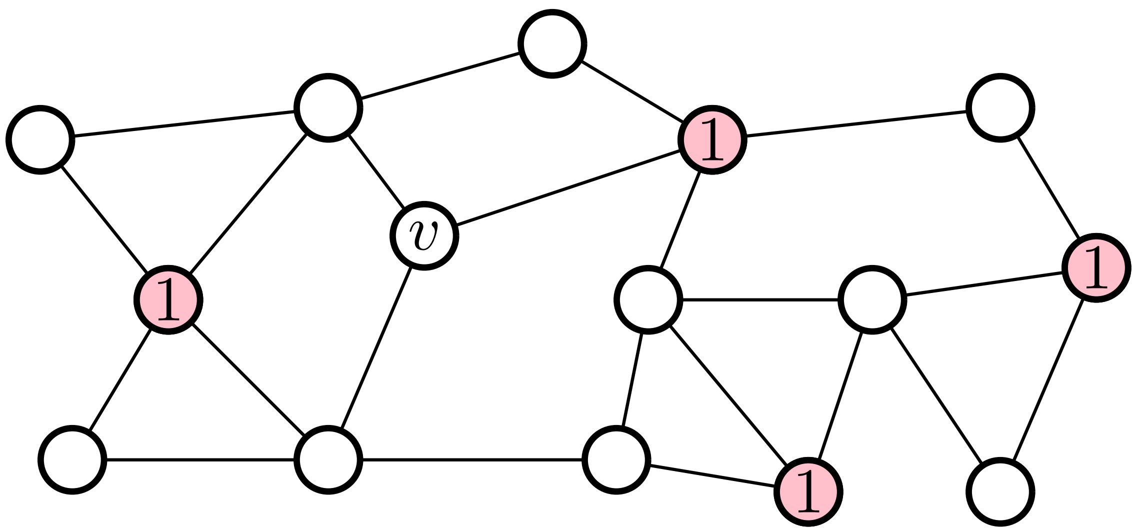


Analysis

Lemma: The algorithm terminates in at most $\Delta + 1$ iterations (i.e., it uses at most $\Delta + 1$ colors).

Proof:

- At the end of each iteration, each uncolored node v is adjacent to at least one node in the MIS $\mathcal{I}$.



Otherwise $\mathcal{I} \cup \{v\}$ would be an independent set, contradicting the maximality of $\mathcal{I}$.

Analysis

Lemma: The algorithm terminates in at most $\Delta + 1$ iterations (i.e., it uses at most $\Delta + 1$ colors).

Proof:

- At the end of each iteration, each uncolored node v is adjacent to at least one node in the MIS $\mathcal{I}$.
- Let the **effective degree** of a node be the number of its uncolored neighbors.
- At the end of each iteration, the effective degree of all uncolored nodes decreases by at least 1.

Analysis

Lemma: The algorithm terminates in at most $\Delta + 1$ iterations (i.e., it uses at most $\Delta + 1$ colors).

Proof:

- At the end of each iteration, each uncolored node v is adjacent to at least one node in the MIS $\mathcal{I}$.
- Let the **effective degree** of a node be the number of its uncolored neighbors.
- At the end of each iteration, the effective degree of all uncolored nodes decreases by at least 1.
- After at most Δ iterations the effective degree of all uncolored nodes becomes 0.

Analysis

Lemma: The algorithm terminates in at most $\Delta + 1$ iterations (i.e., it uses at most $\Delta + 1$ colors).

Proof:

- At the end of each iteration, each uncolored node v is adjacent to at least one node in the MIS $\mathcal{I}$.
- Let the **effective degree** of a node be the number of its uncolored neighbors.
- At the end of each iteration, the effective degree of all uncolored nodes decreases by at least 1.
- After at most Δ iterations the effective degree of all uncolored nodes becomes 0.
- At iteration $\Delta + 1$, each uncolored node enters the MIS $\mathcal{I}$. $\square$

A distributed (randomized) algorithm

Recall: Luby's MIS algorithm runs in $O(\log \Delta \cdot \log n)$ with high probability.

with high probability.

A distributed (randomized) algorithm

Recall: Luby's MIS algorithm runs in $O(\log \Delta \cdot \log n)$ with high probability.

Using Luby's algorithm to find the required MIS we obtain a $\Delta + 1$ coloring algorithm that runs in time:

$$O(\Delta \cdot \log \Delta \cdot \log n)$$

with high probability.

A distributed (randomized) algorithm

Recall: Luby's MIS algorithm runs in $O(\log \Delta \cdot \log n)$ with high probability.

Using Luby's algorithm to find the required MIS we obtain a $\Delta + 1$ coloring algorithm that runs in time:

$$O(\Delta \cdot \log \Delta \cdot \log n)$$

At most $\Delta + 1$ iterations

with high probability.

A distributed (randomized) algorithm

Recall: Luby's MIS algorithm runs in $O(\log\Delta \cdot \log n)$ with high probability.

Using Luby's algorithm to find the required MIS we obtain a $\Delta + 1$ coloring algorithm that runs in time:

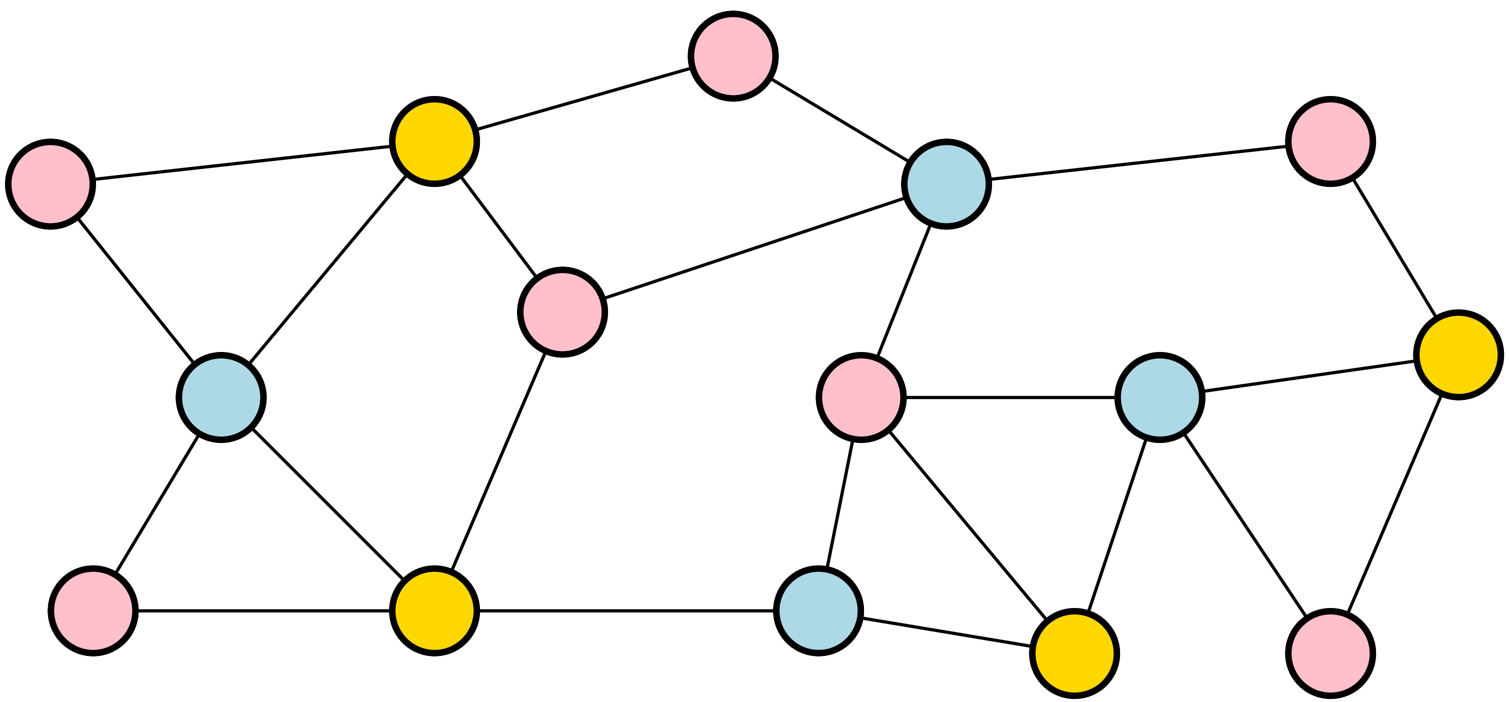
$$O(\underbrace{\Delta}_{\text{At most } \Delta+1 \text{ iterations}} \cdot \underbrace{\log\Delta \cdot \log n}_{\substack{\text{Time to compute a MIS w.h.p.} \\ \text{(using Luby's Algorithm)}}})$$

with high probability.

A faster algorithm (using more colors)

We will design a simple 2Δ-coloring algorithm:

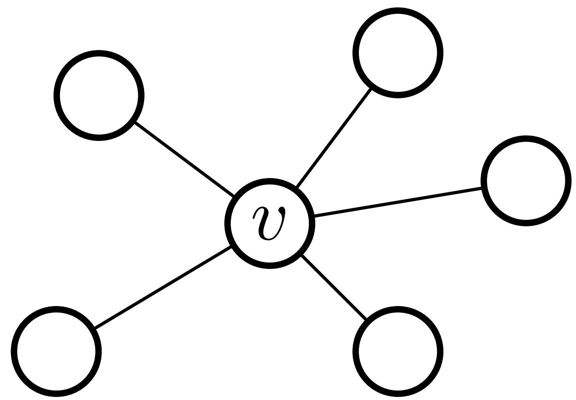
- Distributed
- Randomized
- Running time: $O(\log n)$ with high probability



A distributed 2Δ-coloring algorithm

Each node v mantains a palette of $2\,\delta(v)$ colors:

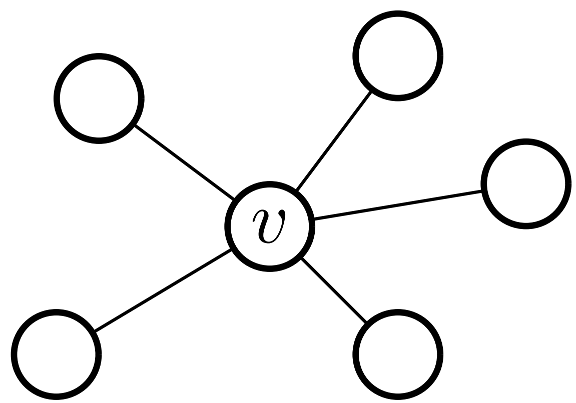
v's degree in G



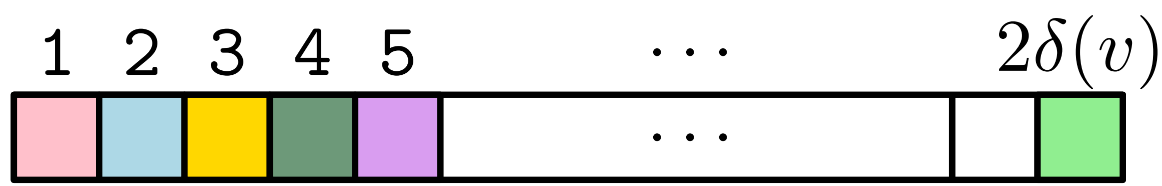
A distributed 2Δ-coloring algorithm

Each node v mantains a palette of $2\,\delta(v)$ colors:

v's degree in G



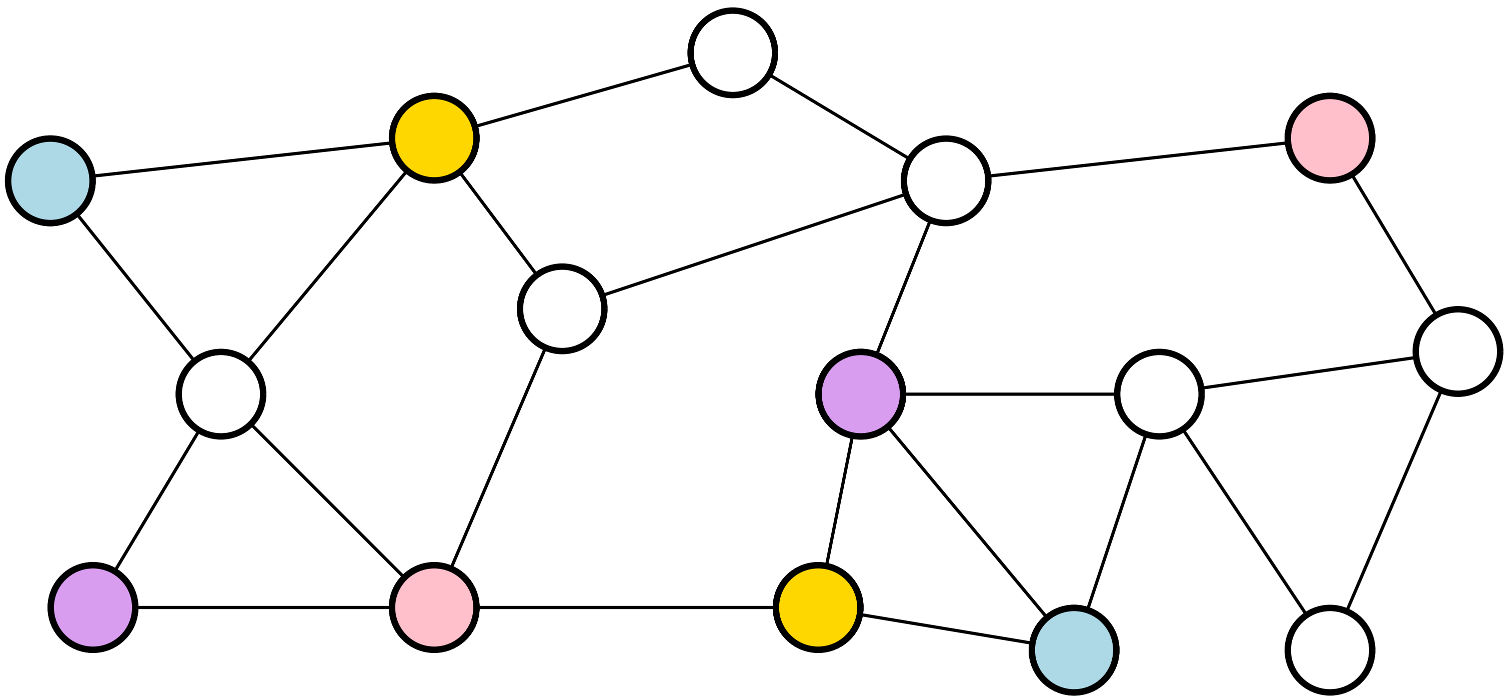
v's palette



Initially all colors are available.

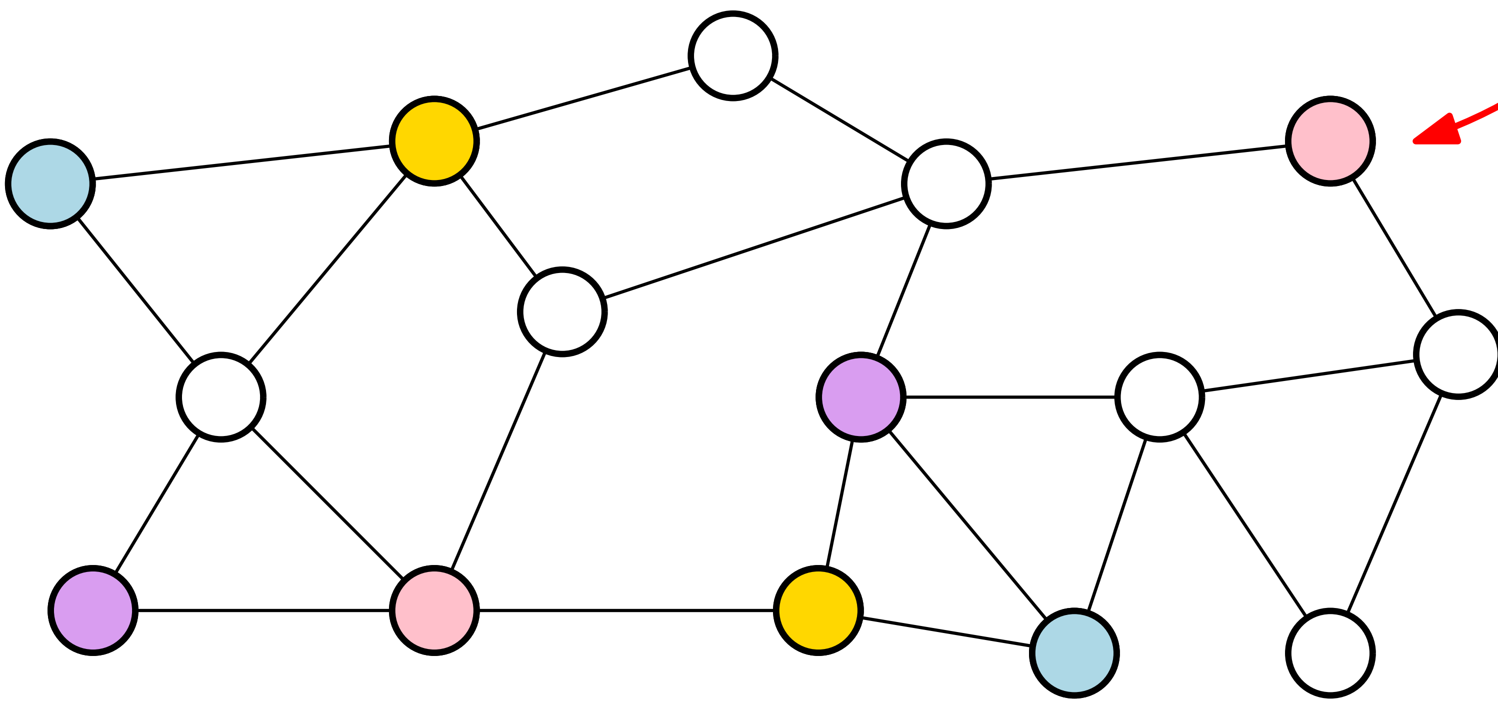
A distributed 2Δ-coloring algorithm

- The algorithm works in *phases*
- In each generic phase there are two kinds of nodes: colored and uncolored.



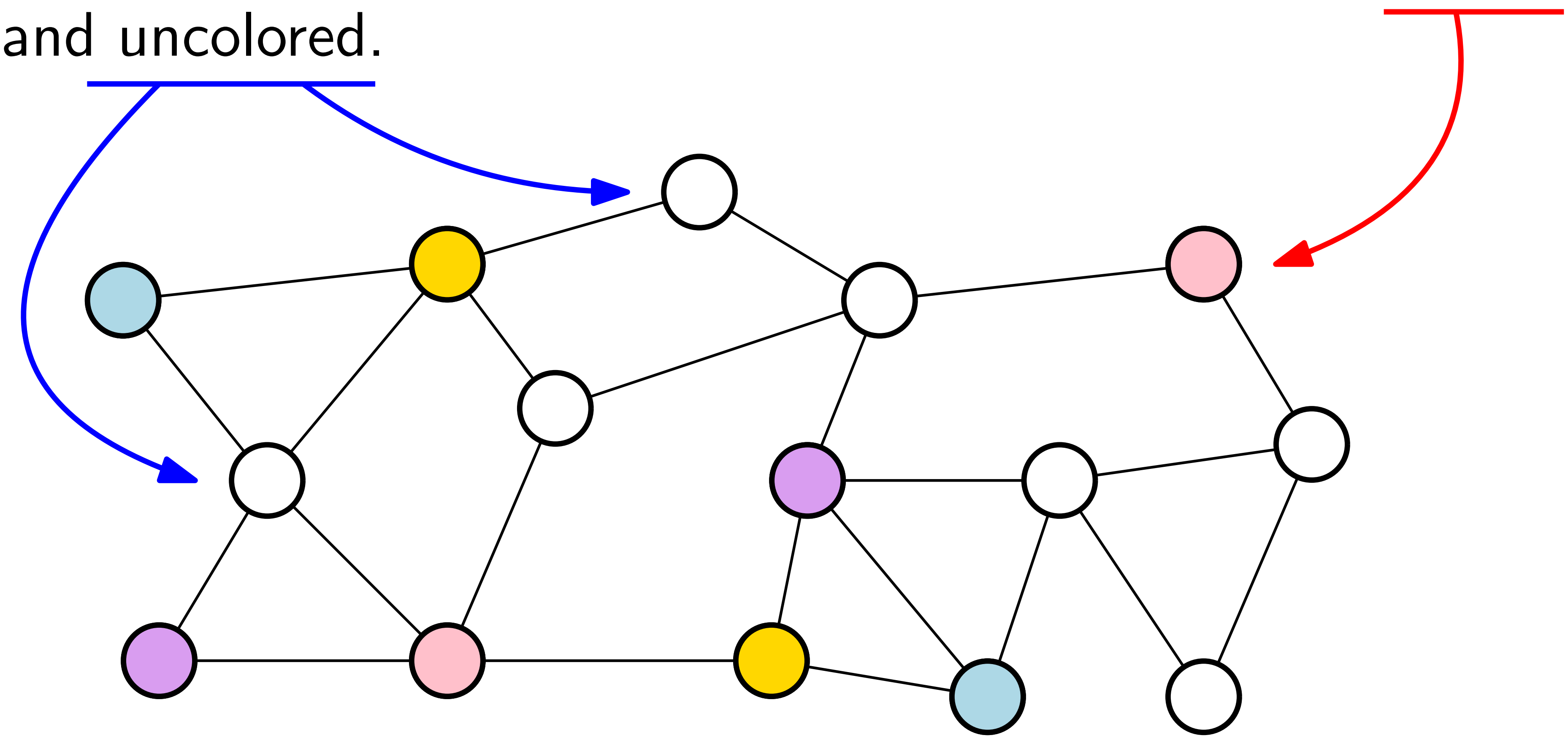
A distributed 2Δ-coloring algorithm

- The algorithm works in *phases*
- In each generic phase there are two kinds of nodes: colored and uncolored.



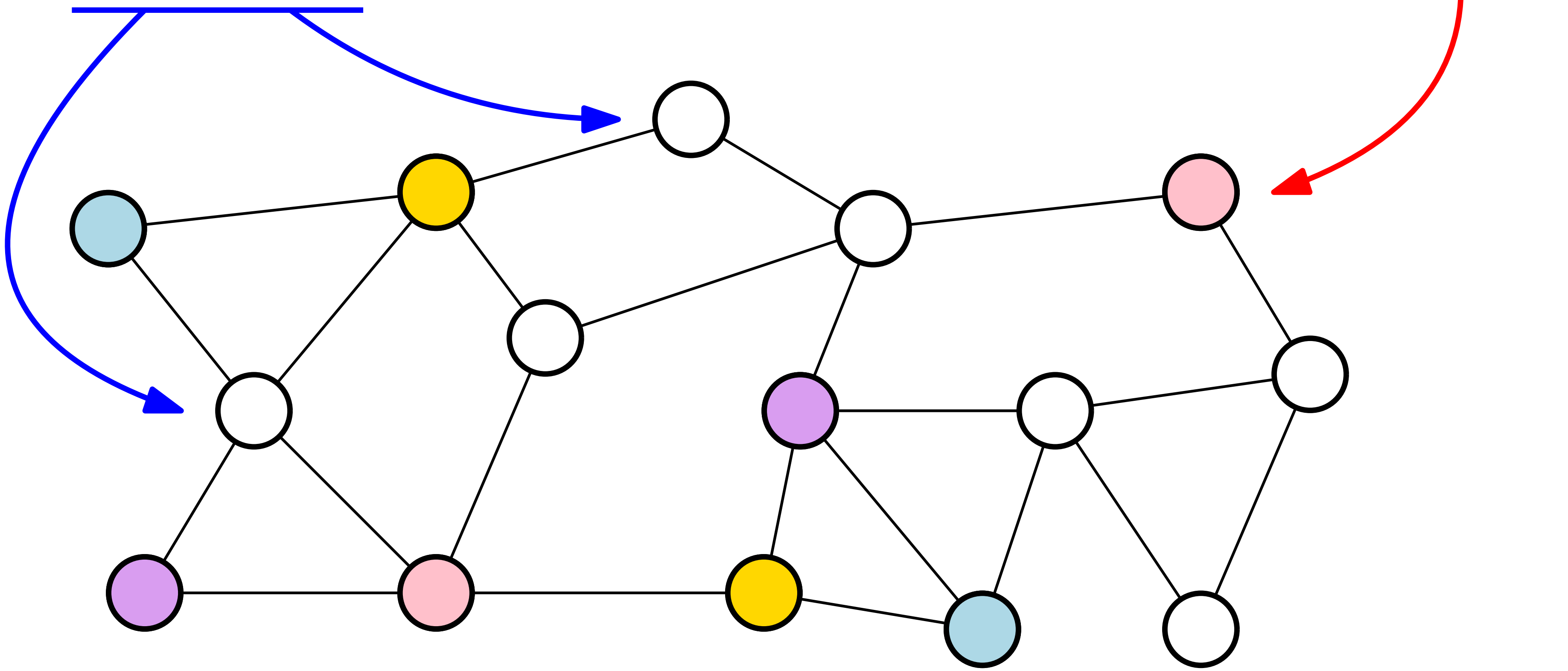
A distributed 2Δ-coloring algorithm

- The algorithm works in *phases*
- In each generic phase there are two kinds of nodes: colored and uncolored.



A distributed 2Δ-coloring algorithm

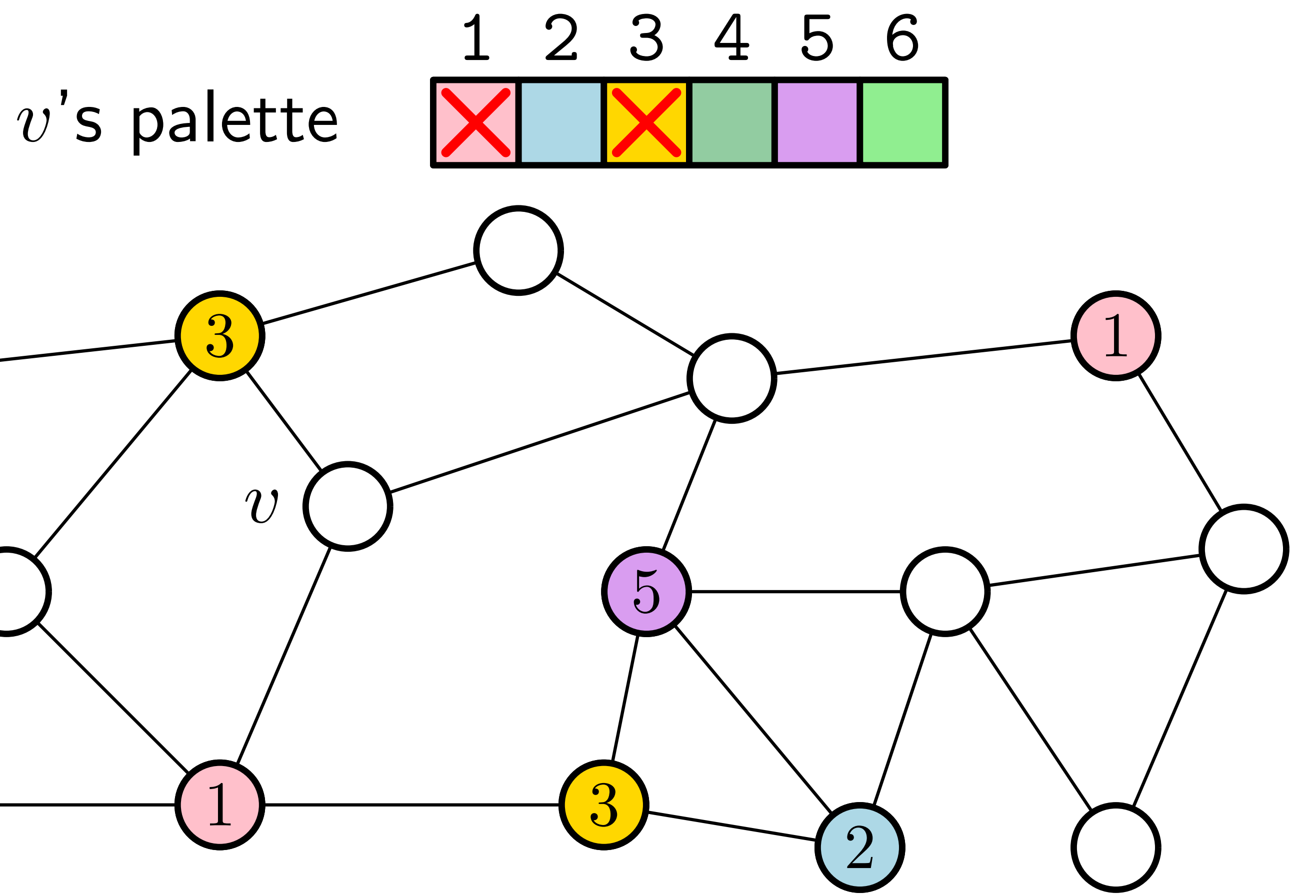
- The algorithm works in *phases*
- In each generic phase there are two kinds of nodes: colored and uncolored.



- Colors are **final**.

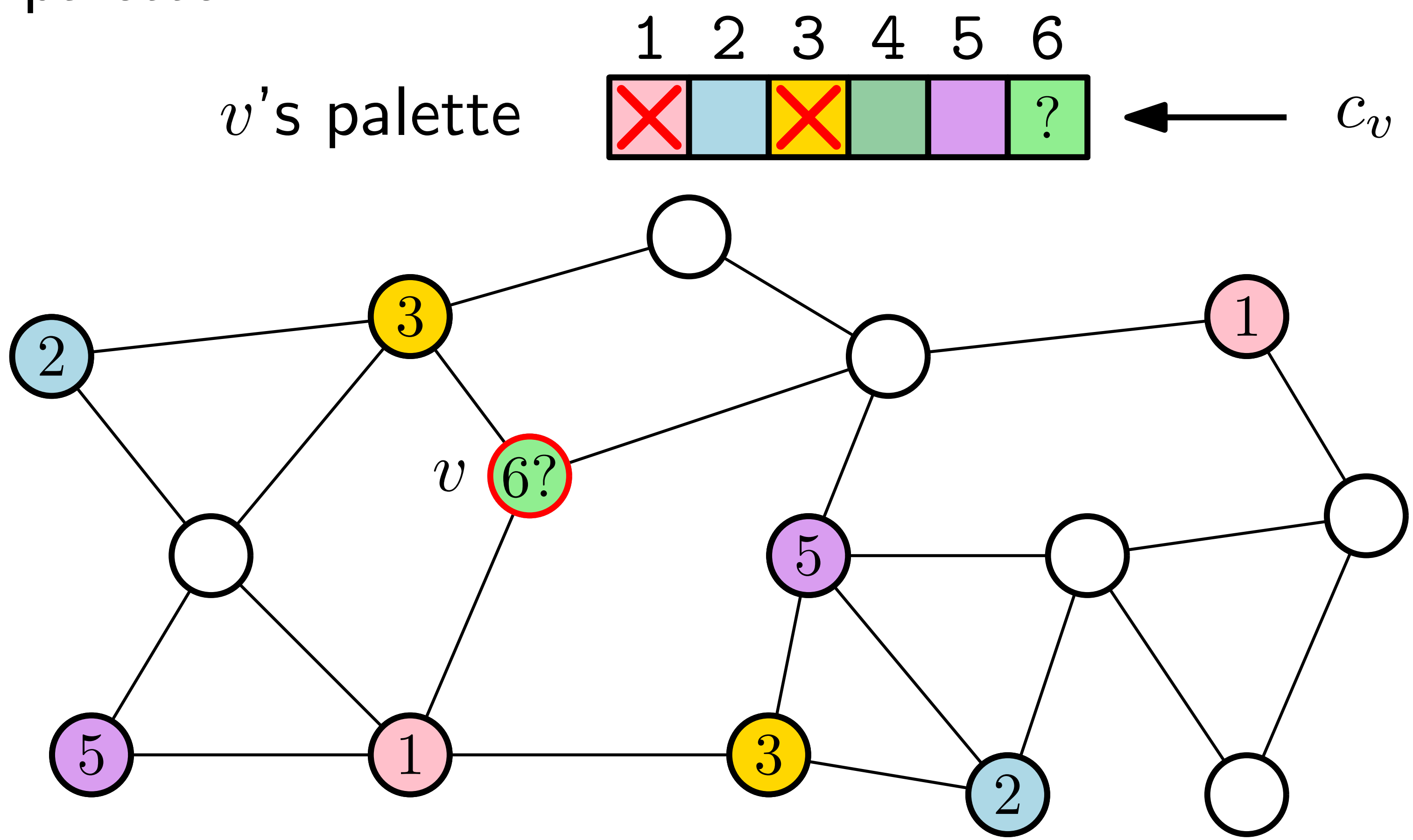
A distributed 2Δ-coloring algorithm

- Let v be an uncolored node in a generic phase.
- All the colors assigned to v's neighbors are unavailable in v's palette.



A distributed 2Δ-coloring algorithm

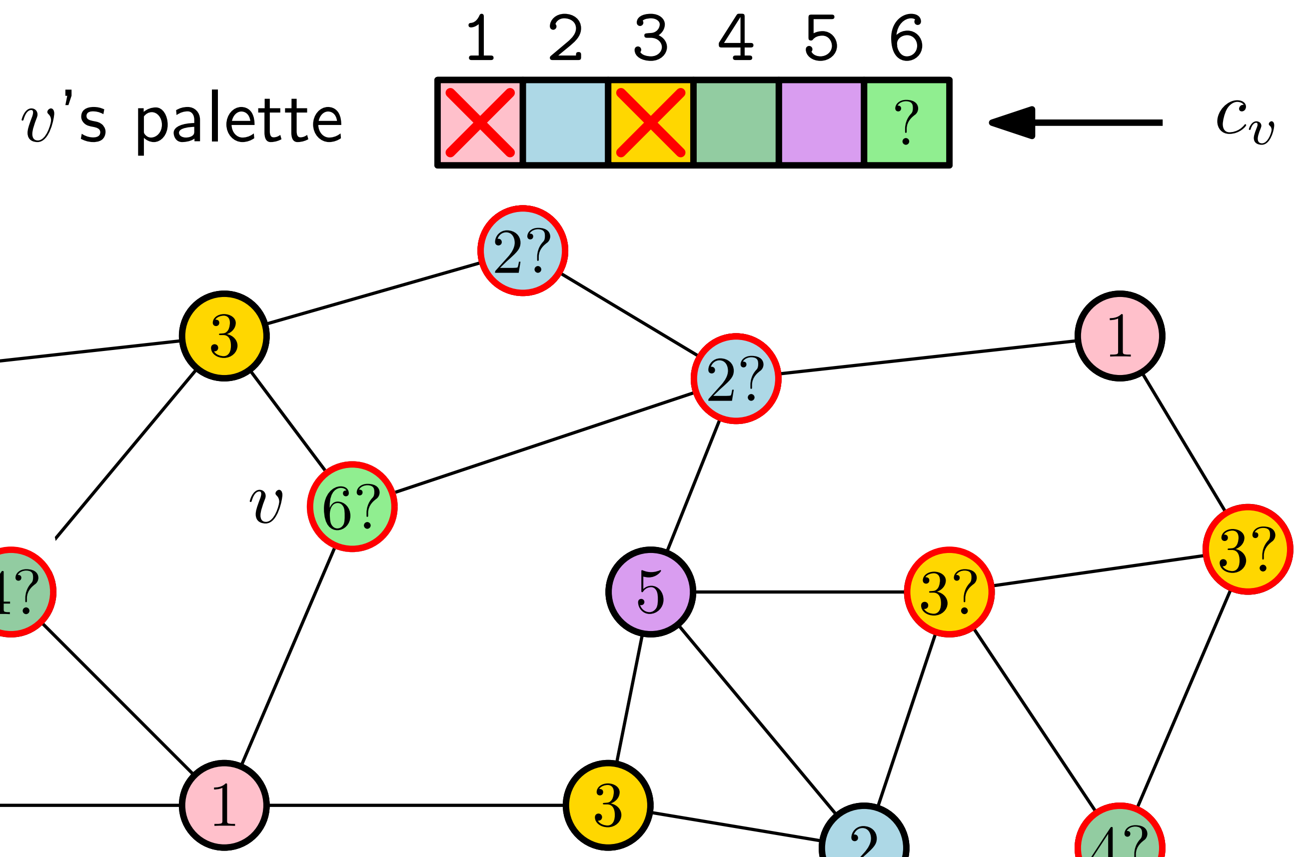
- Let v be an uncolored node in a generic phase.
- All the colors assigned to v's neighbors are unavailable in v's palette.



- Each node v choses uniformly at random a **candidate color** c_v among the available colors in its palette.

A distributed 2Δ-coloring algorithm

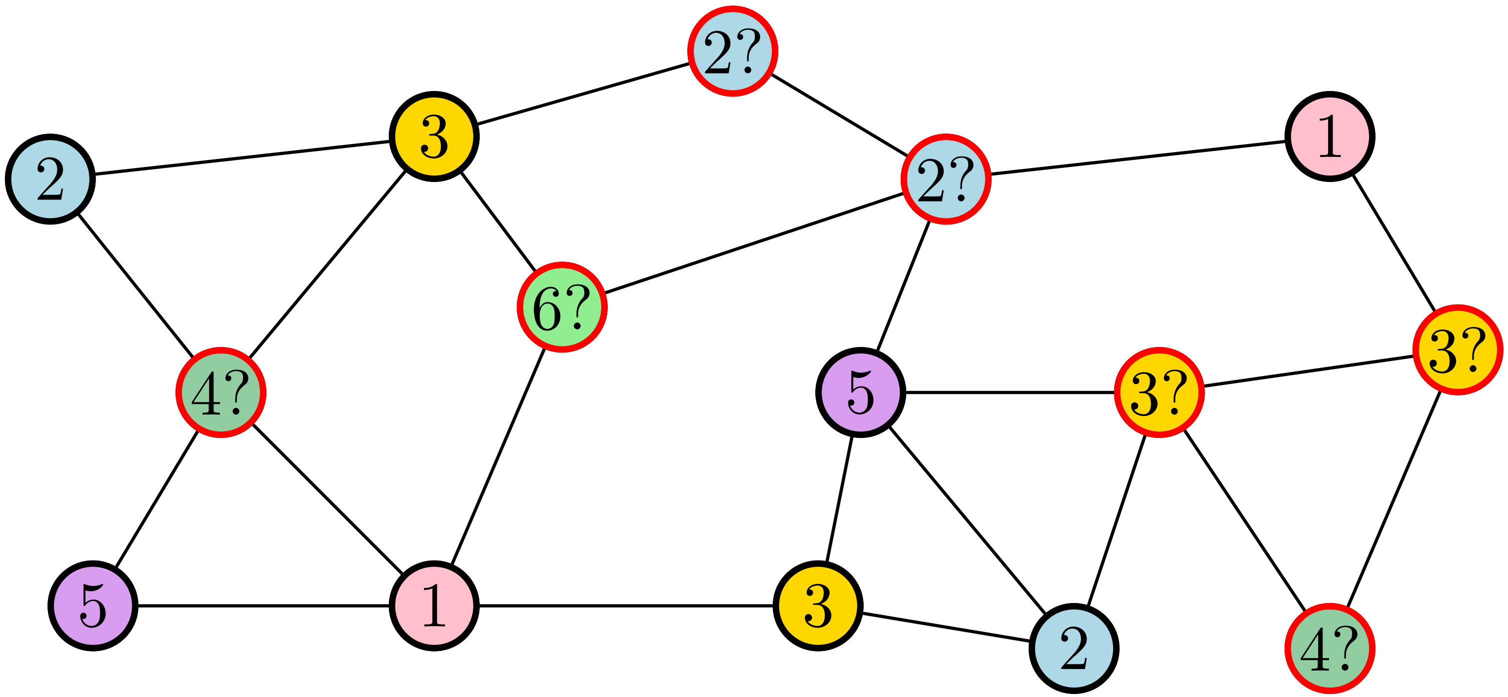
- Let v be an uncolored node in a generic phase.
- All the colors assigned to v's neighbors are unavailable in v's palette.



- Each node v choses uniformly at random a **candidate color** c_v among the available colors in its palette.

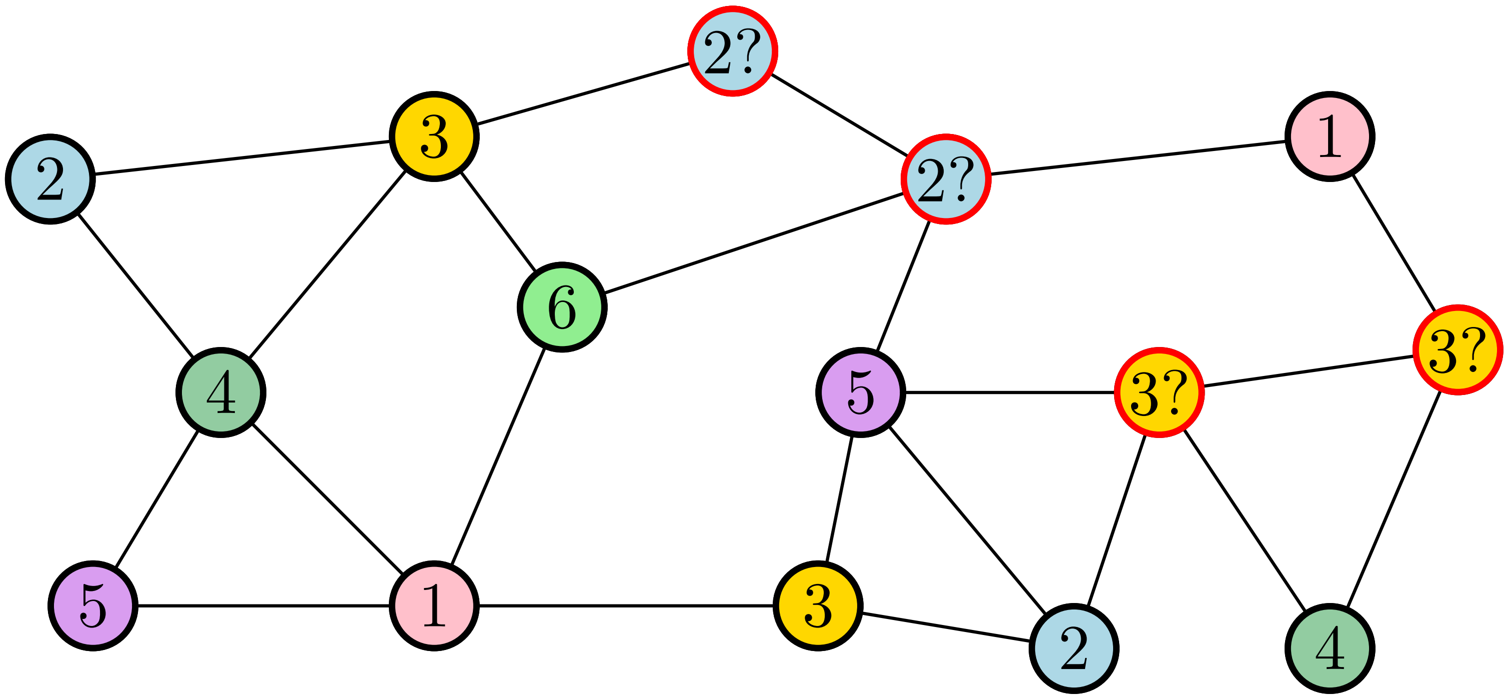
A distributed 2Δ-coloring algorithm

- If $c_v \neq c_u$ for all neighbors u of v:
  - v **accepts** its candidate color: c_v becomes the **final** color of v, and v exits the algorithm.



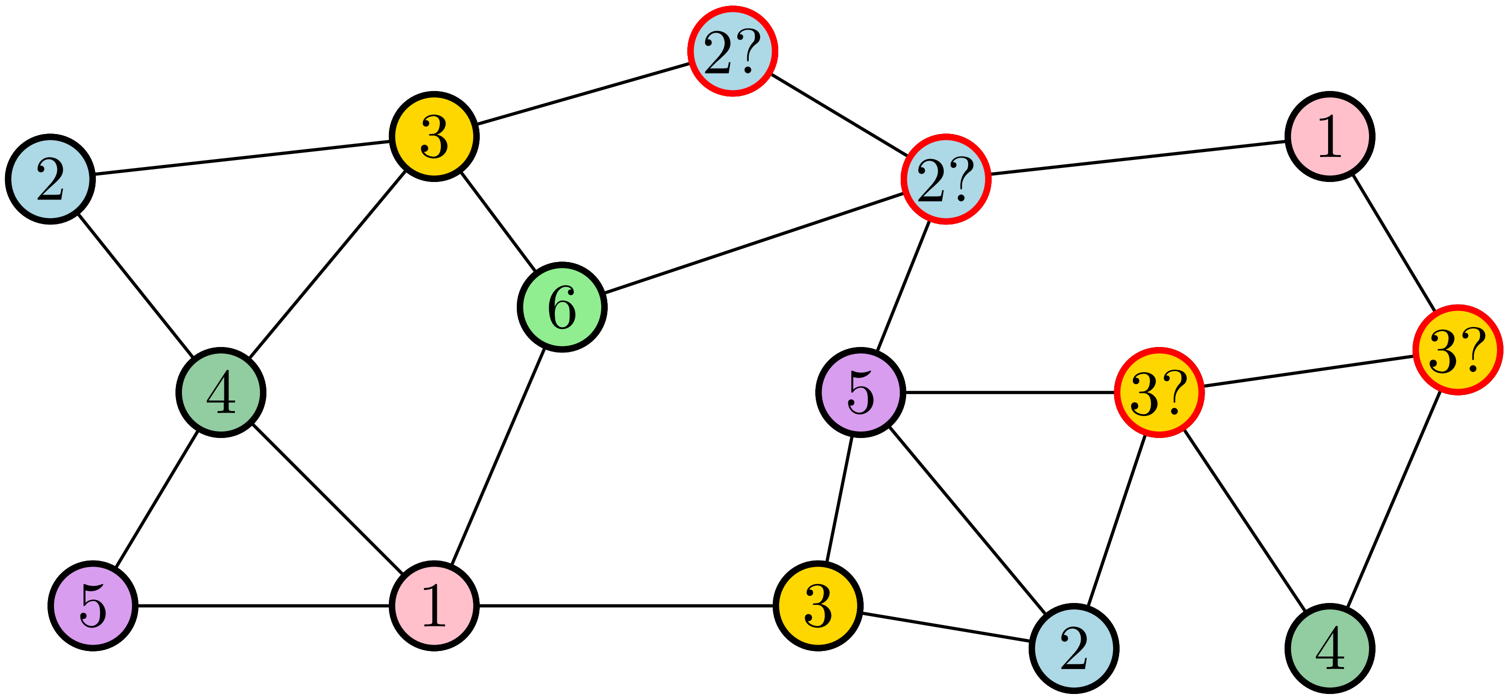
A distributed 2Δ-coloring algorithm

- If $c_v \neq c_u$ for all neighbors u of v:
  - v **accepts** its candidate color: c_v becomes the **final** color of v, and v exits the algorithm.



A distributed 2Δ-coloring algorithm

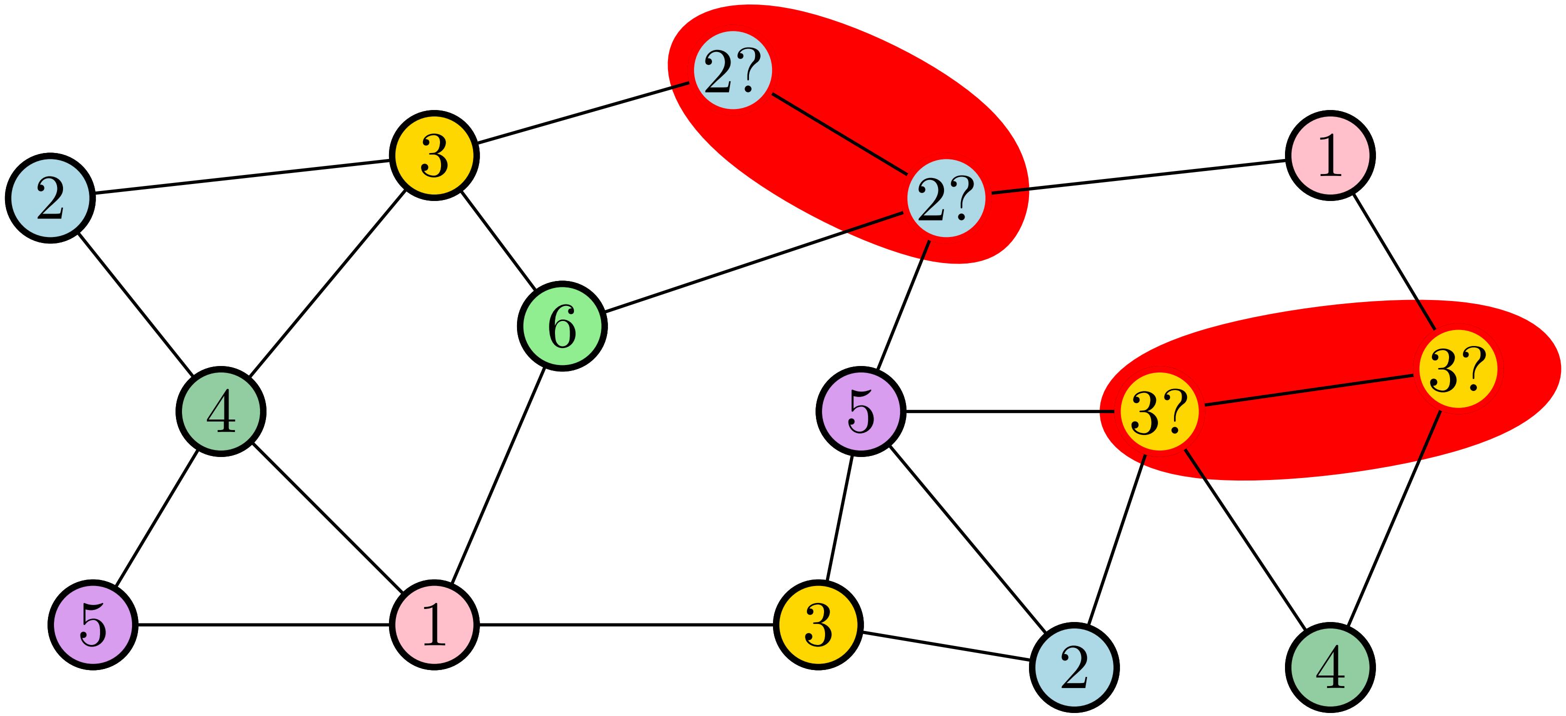
- If $c_v \neq c_u$ for all neighbors u of v:
  - v **accepts** its candidate color: c_v becomes the **final** color of v, and v exits the algorithm.



- If $c_v = c_u$ for for some neighbor u of v:
  - v **rejects** the candidate color c_v.

A distributed 2Δ-coloring algorithm

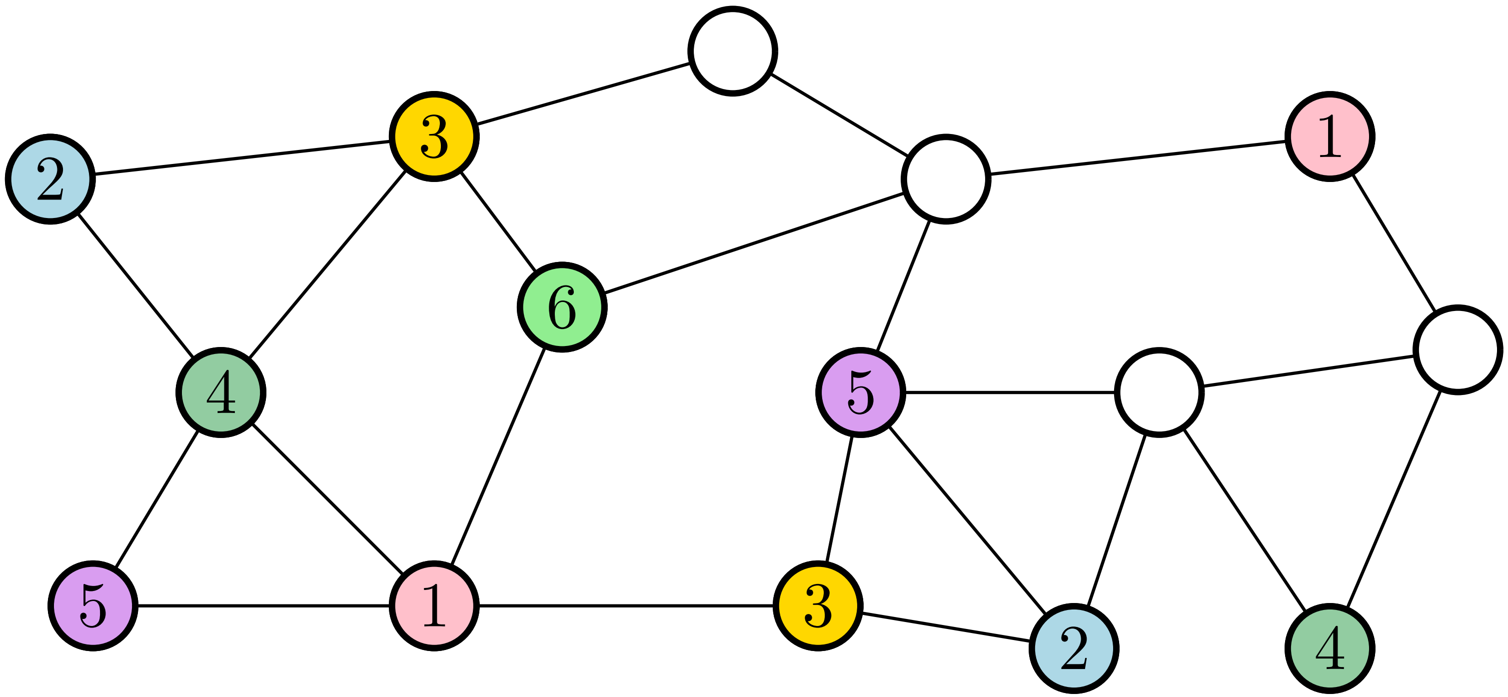
- If $c_v \neq c_u$ for all neighbors u of v:
  - v **accepts** its candidate color: c_v becomes the **final** color of v, and v exits the algorithm.



- If $c_v = c_u$ for for some neighbor u of v:
  - v **rejects** the candidate color c_v.

A distributed 2Δ-coloring algorithm

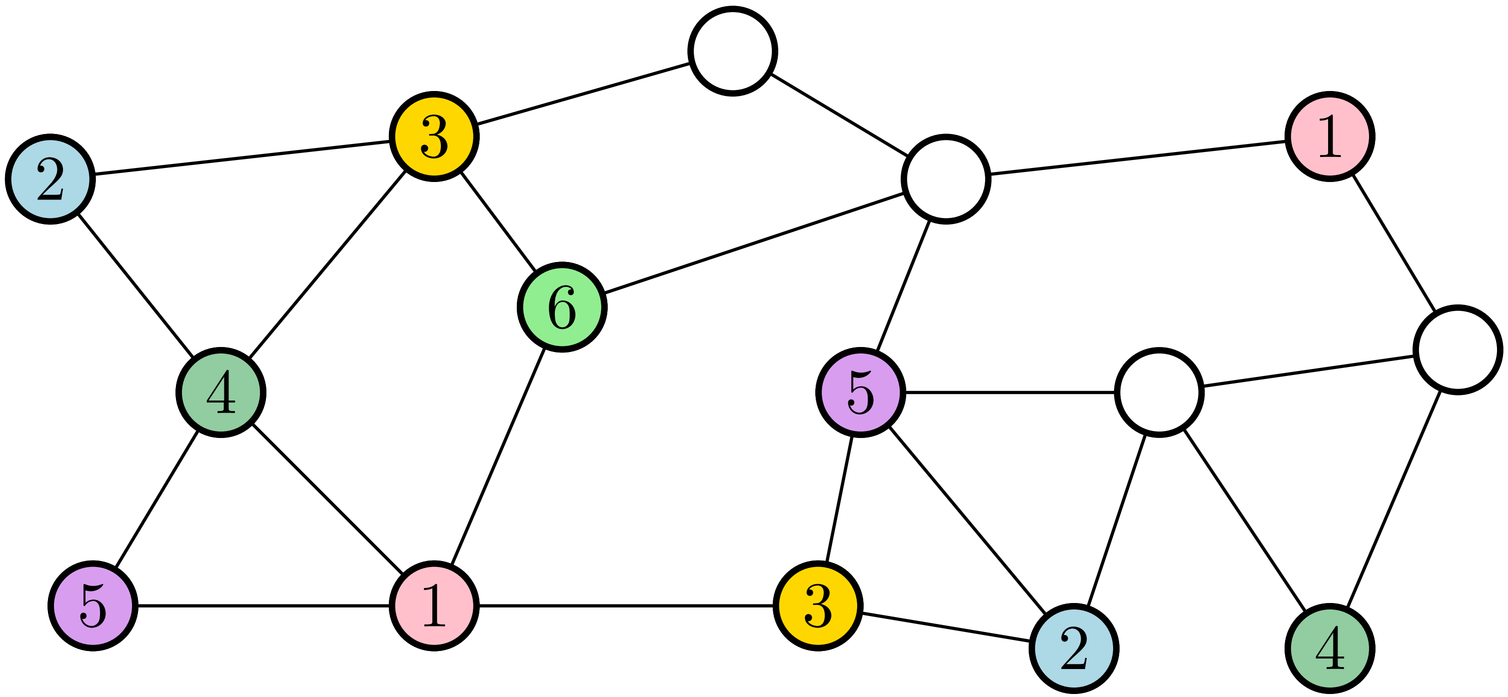
- If $c_v \neq c_u$ for all neighbors u of v:
  - v **accepts** its candidate color: c_v becomes the **final** color of v, and v exits the algorithm.



- If $c_v = c_u$ for for some neighbor u of v:
  - v **rejects** the candidate color c_v.

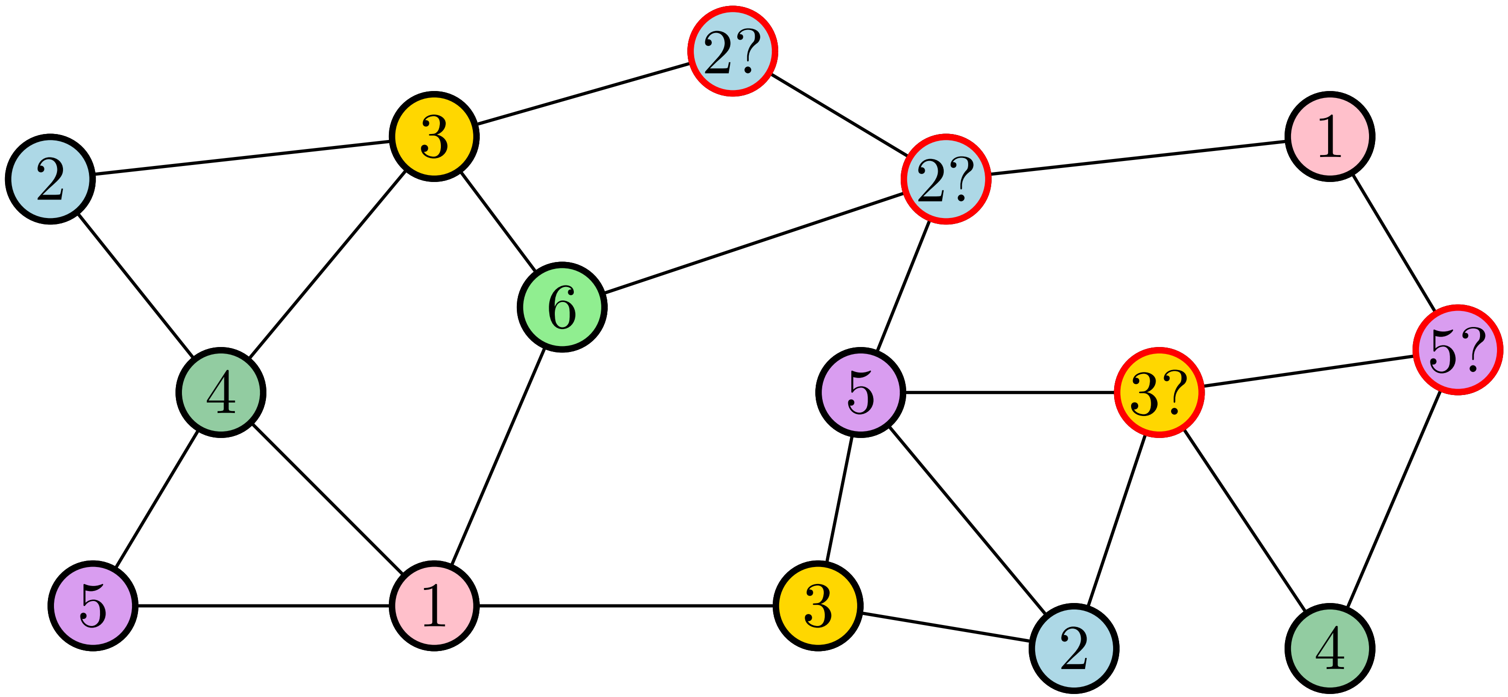
A distributed 2Δ-coloring algorithm

All uncolored nodes advance to the next phase (and try again with another candidate color)



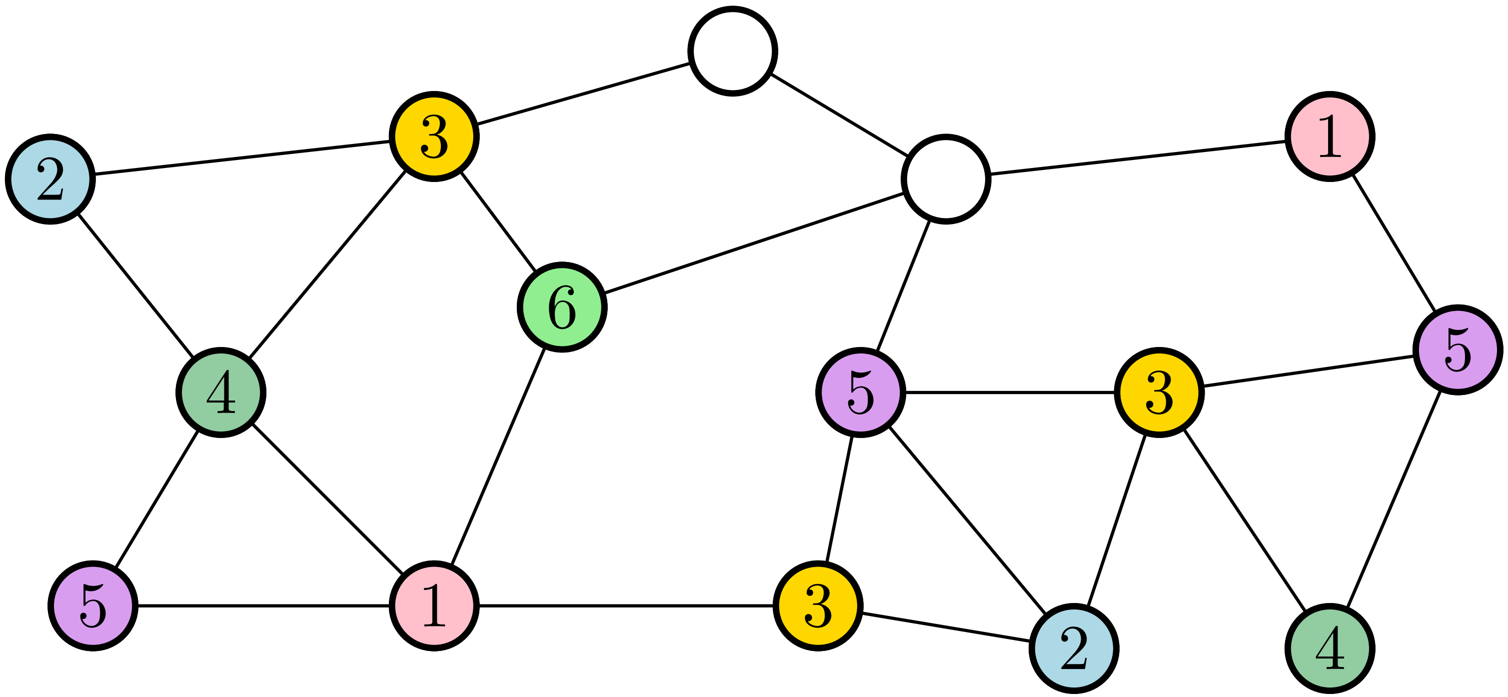
A distributed 2Δ-coloring algorithm

All uncolored nodes advance to the next phase (and try again with another candidate color)



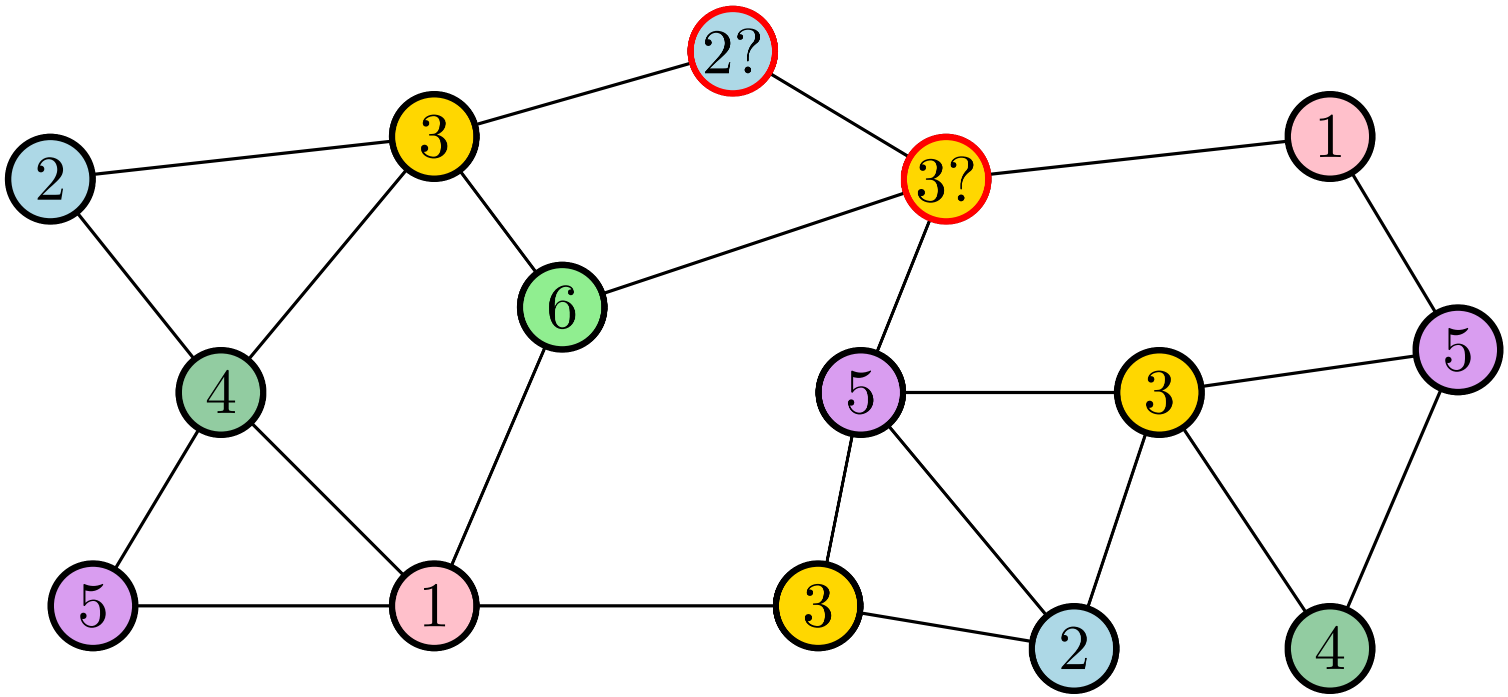
A distributed 2Δ-coloring algorithm

All uncolored nodes advance to the next phase (and try again with another candidate color)



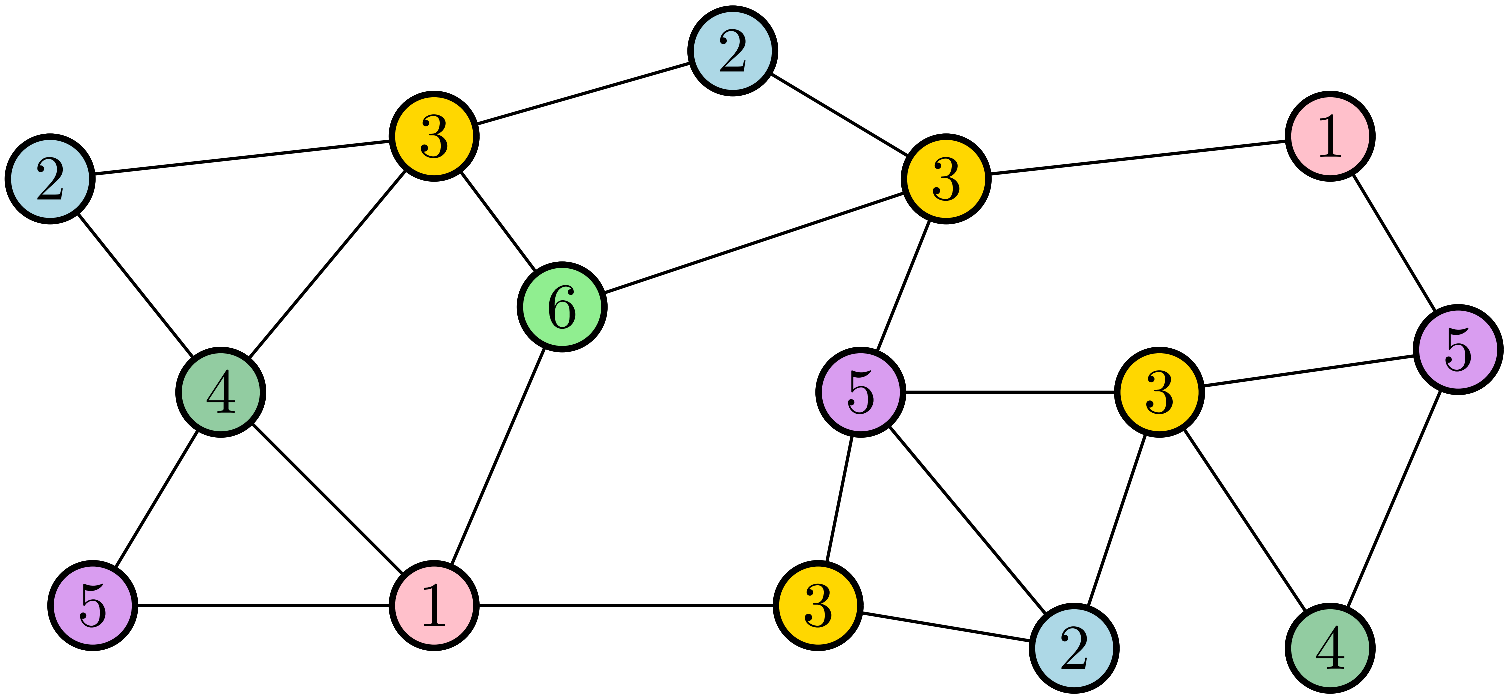
A distributed 2Δ-coloring algorithm

All uncolored nodes advance to the next phase (and try again with another candidate color)



A distributed 2Δ-coloring algorithm

All uncolored nodes advance to the next phase (and try again with another candidate color)



...until all nodes are colored.

A distributed 2Δ-coloring algorithm

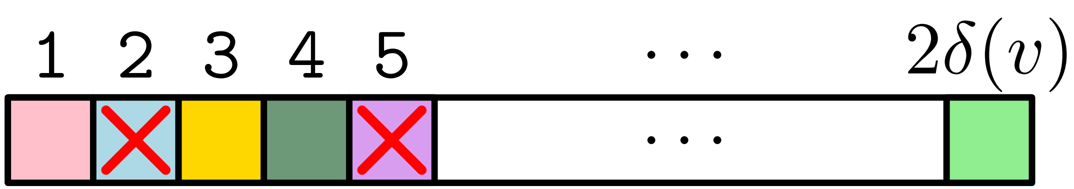
Algorithm for node v:

- **While v is uncolored** (iteration# $=$ phase#):
  - Pick a color c_v uniformly at random from the available palette colors
  - Send c_v to uncolored neighbors (and receive neighbors' colors)
  - **If** some uncolored neighbor u chose $c_u = c_v$:
    - Reject color c_v (do nothing)
  - **Else**
    - Accept color c_v
    - Inform neighbors about color c_v
      (neighbors mark color c_v as unavailable in their palettes)

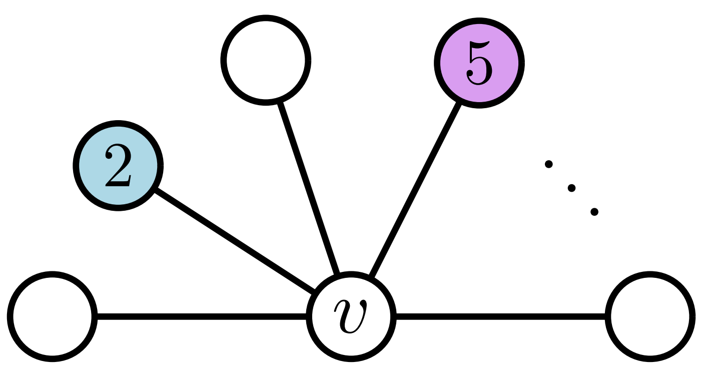
Analysis

Consider a generic uncolored node v in a generic phase k.

Let $A(v)$ be the set of available colors in v's palette at the beginning of phase k.



$$A(v) = \{1, 3, 4, \dots\}$$

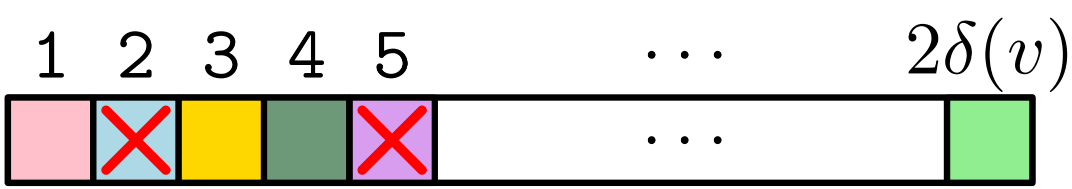


Analysis

Consider a generic uncolored node v in a generic phase k.

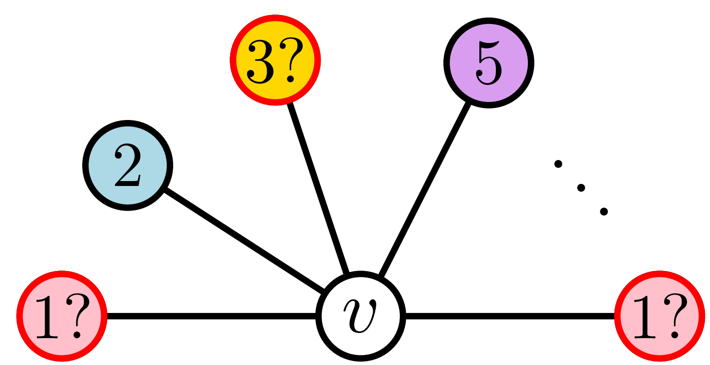
Let $A(v)$ be the set of available colors in v's palette at the beginning of phase k.

Let $U(v) = \{c_u : (v, u) \in E\}$ be the set of **candidate colors** chosen by the neighbors of v that were uncolored at the beginning of phase k.



$$A(v) = \{1, 3, 4, \dots\}$$

$$U(v) = \{1, 3, \dots\}$$



Analysis

Let $A'(v)$ be the set of available colors in v's palette that are not chosen as candidates by any uncolored neighbor of v.

$A'(v) = A(v) \setminus U(v)$

Analysis

Let $A'(v)$ be the set of available colors in v's palette that are not chosen as candidates by any uncolored neighbor of v.

$$A'(v) = A(v) \setminus U(v)$$

Let ℓ the number of uncolored neighbors of v (at the beginning of phase k)

$$|A'(v)| = |A(v) \setminus U(v)| \geq |A(v)| - |U(v)|$$

Analysis

Let $A'(v)$ be the set of available colors in v's palette that are not chosen as candidates by any uncolored neighbor of v.

$$A'(v) = A(v) \setminus U(v)$$

Let ℓ the number of uncolored neighbors of v (at the beginning of phase k)

$$\begin{aligned} |A'(v)| = |A(v) \setminus U(v)| &\geq |A(v)| - |U(v)| \\ &\geq (\underbrace{2\delta(v)}_{\text{Colors in } v\text{'s palette}} - \underbrace{(\delta(v) - \ell)}_{\text{Colored neighbors of } v}) - \underbrace{\ell}_{|U(v)| \leq \ell} \end{aligned}$$

Analysis

Let $A'(v)$ be the set of available colors in v's palette that are not chosen as candidates by any uncolored neighbor of v.

$$A'(v) = A(v) \setminus U(v)$$

Let ℓ the number of uncolored neighbors of v (at the beginning of phase k)

$$\begin{aligned} |A'(v)| = |A(v) \setminus U(v)| &\geq |A(v)| - |U(v)| \\ &\geq \underbrace{(2\delta(v)}_{\text{Colors in } v\text{'s palette}} - \underbrace{(\delta(v) - \ell))}_{\text{Colored neighbors of } v} - \underbrace{\ell}_{|U(v)| \leq \ell} \quad = \delta(v) \end{aligned}$$

Analysis

The probability that the candidate color c_v chosen by v is accepted is:

Number of successful choices for c_v

$$\frac{|A'(v)|}{|A(v)|}$$

Number of available choices for c_v

Analysis

The probability that the candidate color c_v chosen by v is accepted is:

Number of successful choices for c_v

$$\frac{|A'(v)|}{|A(v)|} \geq \frac{\delta(v)}{2\delta(v)} = \frac{1}{2}$$

Number of available choices for c_v

Analysis

The probability that the candidate color c_v chosen by v is accepted is:

Number of successful choices for c_v

$$\frac{|A'(v)|}{|A(v)|} \geq \frac{\delta(v)}{2\delta(v)} = \frac{1}{2}$$

Number of available choices for c_v

Probability that v **succeeds** during phase k: at least $\frac{1}{2}$.
(becomes colored)

Analysis

Probability that v **succeeds** in a (generic) phase: at least $\frac{1}{2}$.

Analysis

Probability that v **succeeds** in a (generic) phase: at least $\frac{1}{2}$.

Probability that v **fails** for $2\log n$ consecutive phases:

$$\leq \left(1 - \frac{1}{2}\right)^{2\log n} = \left(\frac{1}{2}\right)^{2\log n} = \frac{1}{2^{2\log n}} = \frac{1}{n^2}.$$

Analysis

Probability that v **succeeds** in a (generic) phase: at least $\frac{1}{2}$.

Probability that v **fails** for $2\log n$ consecutive phases:

$$\leq \left(1-\frac{1}{2}\right)^{2\log n} = \left(\frac{1}{2}\right)^{2\log n} = \frac{1}{2^{2\log n}} = \frac{1}{n^2}.$$

Probability that **at least one node** fails for $2\log n$ consecutive phases:

$$\leq n \cdot \frac{1}{n^2} = \frac{1}{n}$$

Analysis

Probability that v **succeeds** in a (generic) phase: at least $\frac{1}{2}$.

Probability that v **fails** for $2\log n$ consecutive phases:

$$\leq \left(1-\frac{1}{2}\right)^{2\log n} = \left(\frac{1}{2}\right)^{2\log n} = \frac{1}{2^{2\log n}} = \frac{1}{n^2}.$$

Probability that **at least one node** fails for $2\log n$ consecutive phases:

$$\leq n \cdot \frac{1}{n^2} = \frac{1}{n}$$

With probability at least $1-\frac{1}{n}$ all nodes succeed within $2\log(n)$ phases.

Analysis

With probability at least $1 - \frac{1}{n}$ a valid 2Δ-coloring is computed in at most $2 \log n$ phases.

Analysis

With probability at least $1 - \frac{1}{n}$ a valid 2Δ-coloring is computed in at most $2 \log n$ phases.

Each phase requries $O(1)$ time steps.

Analysis

With probability at least $1 - \frac{1}{n}$ a valid 2Δ-coloring is computed in at most $2\log n$ phases.

Each phase requries $O(1)$ time steps.

The algorithm computes a valid 2Δ-coloring in time $O(\log n)$, with high probability.