A&W

Exercise Session 5

Coloring

Nil Ozer

A&W Overview

A&W

Connectivity
- Articulation Points
- Bridges
- Block - Decomposition
- Menger's Theorem

Cycles
- Eulerian Cycle
- Hamiltonian Cycle
- TSP

Matchings
- Definition
- Algorithms
- Hall's Theorem

Colorings
- Definition
- Algorithm
- Brooks's Theorem

Wahrscheinlichkeit
- Grundbegriffe und Notationen
- Bedingte Wahrscheinlichkeiten
- Unabhangigkeiten
- Zufallsvariablen
- Wichtige Diskrete Verteilungen
- Abschatzen von Wahrscheinlichkeiten

Randomized Algorithms
- Las-Vegas
- Monte-Carlo
- Primality Test
- Target - Shooting
- Finding Duplicates
- Longest Path Problem

Graph Algorithms

Flow
- Definition
- Maxflow - Mincut
- Ford - Fulkerson
- Matching w. Flow
- Edge-disjoint paths w. Flow

Minimum Cut
- Definition
- Cut(G) Algorithm
- Bootstrapping

Geometric Algorithms

Convex Hull
- Definition
- Jarvis Wrap
- Local optimization

Smallest Enclosing Circle
- Definition
- First Algorithm
- Final Algorithm

Outline

- T1 Discussion
- Matching Kahoot
- Coloring
- Minitest 2 - coloring discussion

T1
Feedback + Discussion

- 1.c : Reflexivity argument accepted
- Watch out for the comments !
- Keep up the good work ! 👏🏻👏🏻👏🏻
- Questions, issues ... Let me know !

Some Announcments

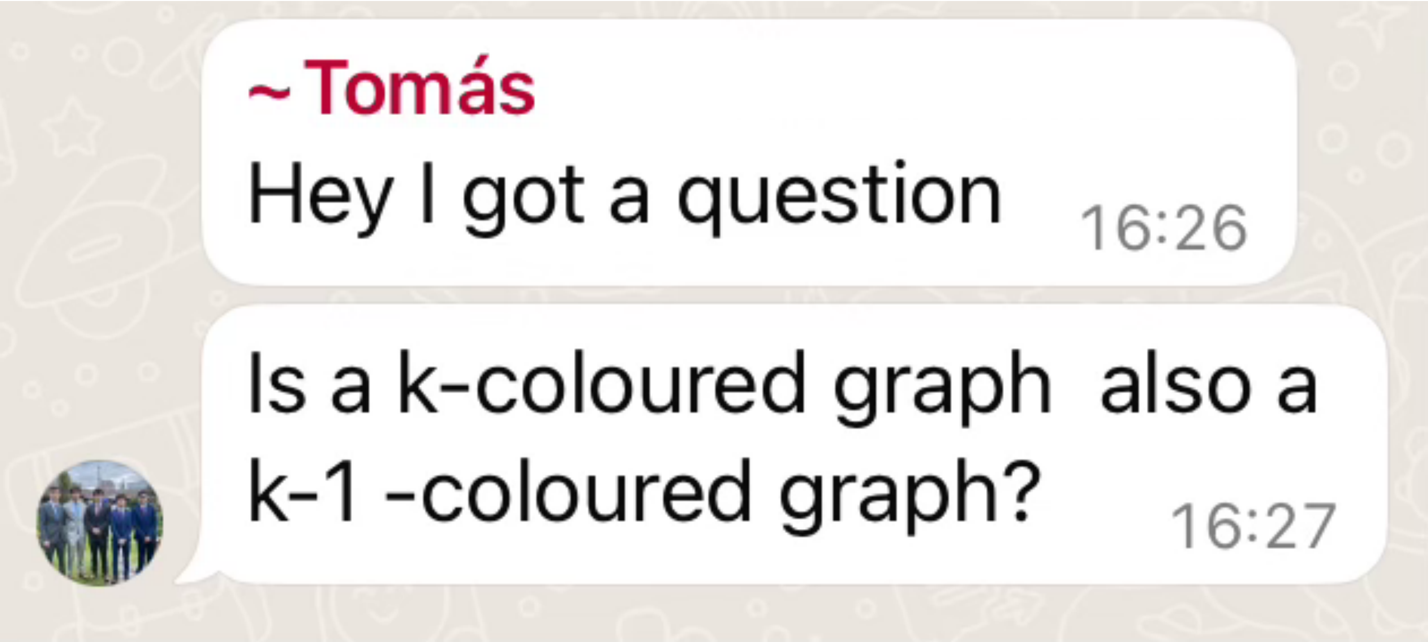
- Anki card approach changed (instead we have kahoots for now)
  - Matching kahoot this week
  - Cycles + TSP kahoot next week ...

- T2 (peer grading 1) ??
- Namings:
  - T1(theoretical exercise 1) , T2(peer grading 1), T3(theoretical exercise 2) ...

Matching Kahoot

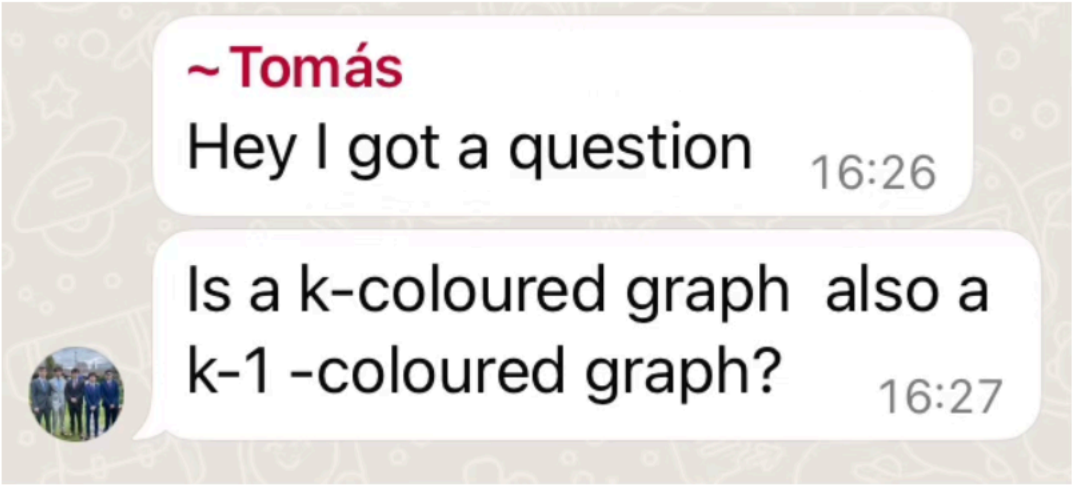
Coloring

Coloring
Intuition



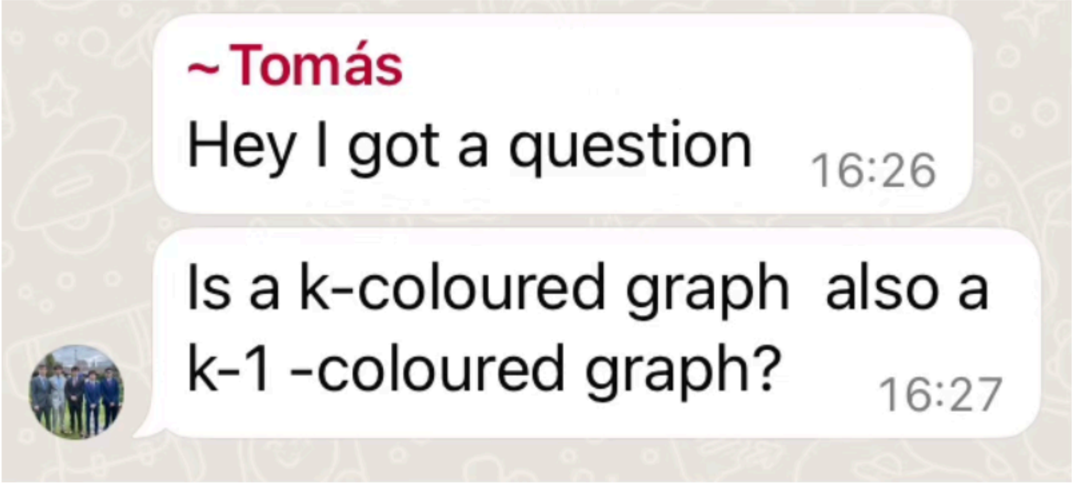
Coloring

Intuition



Matching	Coloring
pairing adjacent vertices without conflicts	seperating adjacent vertices
(pairing non-adjacent edges)	
ensure that selected edges don’t touch the same vertex	ensure that the connected vertices have distinct colors
edges are our friends	edges are our enemies

Coloring
Intuition



Matching

pairing adjacent vertices without conflicts

(pairing non-adjacent edges)

ensure that selected edges don't touch the same vertex

edges are our friends

k-matched G is also (k-1)-matched

simply remove 1 edge

Coloring

seperating adjacent vertices

ensure that the connected vertices have distinct colors

edges are our enemies

(k-1)-colored G is also k-colored

simply change one node's color

Coloring
Definitions

- (Vertex-) Coloring :
  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

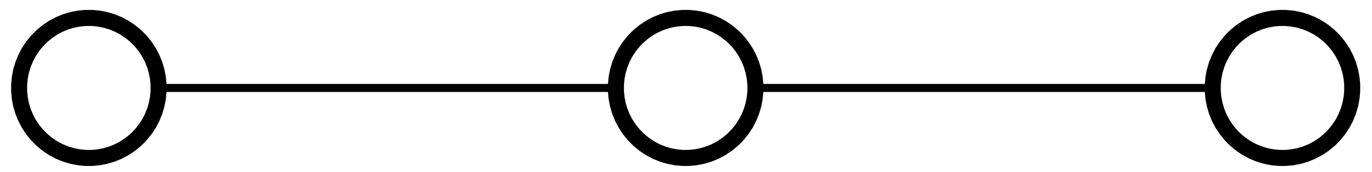
Color vertices in a way that no two vertices that share an edge are of the same color

Coloring
Examples

Color vertices in a way that no two vertices that share an edge are of the same color

- (Vertex-) Coloring :
  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

Is this a coloring ?



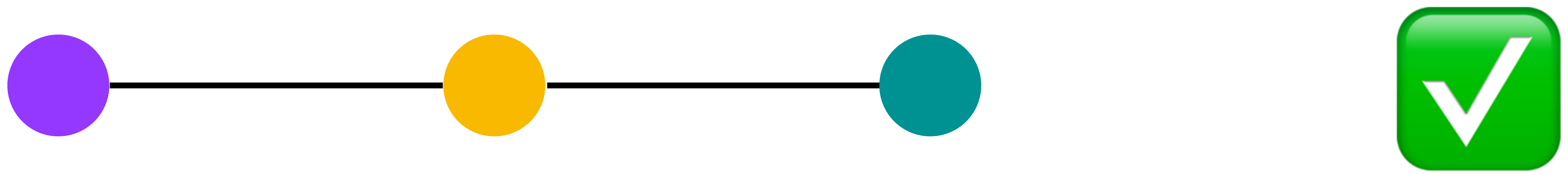
Coloring

Examples

Color vertices in a way that no two vertices that share an edge are of the same color

- (Vertex-) Coloring :
  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

Is this a coloring ?



Coloring

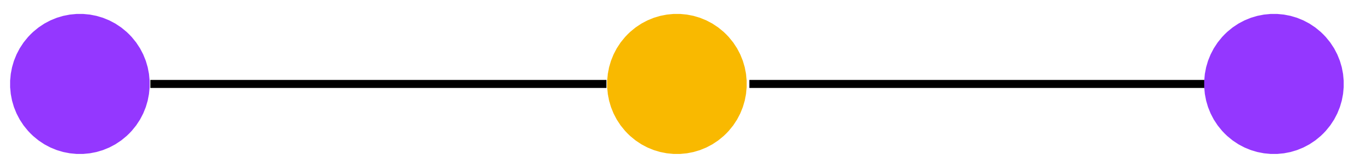
Examples

- (Vertex-) Coloring :

Color vertices in a way that no two vertices that share an edge are of the same color

  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \rightarrow [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

Is this a coloring ?



✅

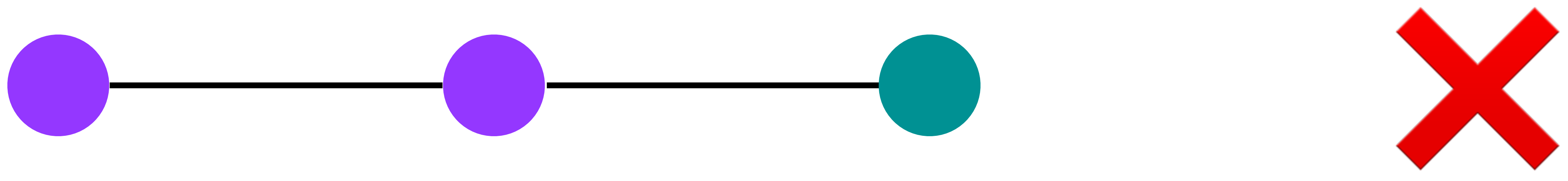
Coloring

Examples

Color vertices in a way that no two vertices that share an edge are of the same color

- (Vertex-) Coloring :
  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

Is this a coloring ?



Coloring
Definitions

Color vertices in a way that no two vertices that share an edge are of the same color

- (Vertex-) Coloring :
  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \rightarrow [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$
- Chromatic Number :
  - The chromatic number $\chi(G)$ is the minimum number of colors needed to color a graph
  - equivalent : $\chi(G) \leq k \iff G$ is k-partite

Coloring
k-partite

- General version of the bipartite

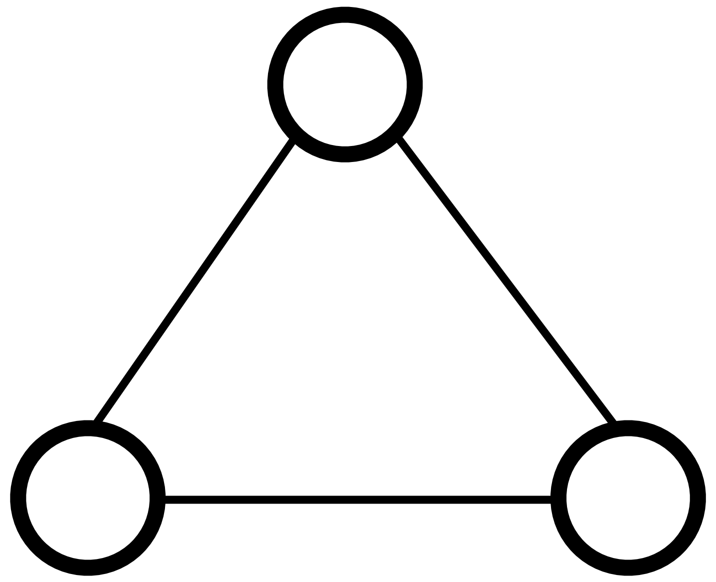
- A graph $G = (V, E)$ is called k-partite if
  - the vertex set V can be divided into k disjoint sets $V = V_1 \cup V_2 \cup \ldots \cup V_k$
  - s.t. for every edge $(u, v) \in E$, u and v belong to different sets V_i and V_j where $i \neq j$

Coloring
Examples

- (Vertex-) Coloring :
  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

Color vertices in a way that no two vertices that share an edge are of the same color

- Chromatic Number :
  - The chromatic number $\chi(G)$ is the minimum number of colors needed to color a graph
  - equivalent : $\chi(G) \leq k \iff G$ is k-partite



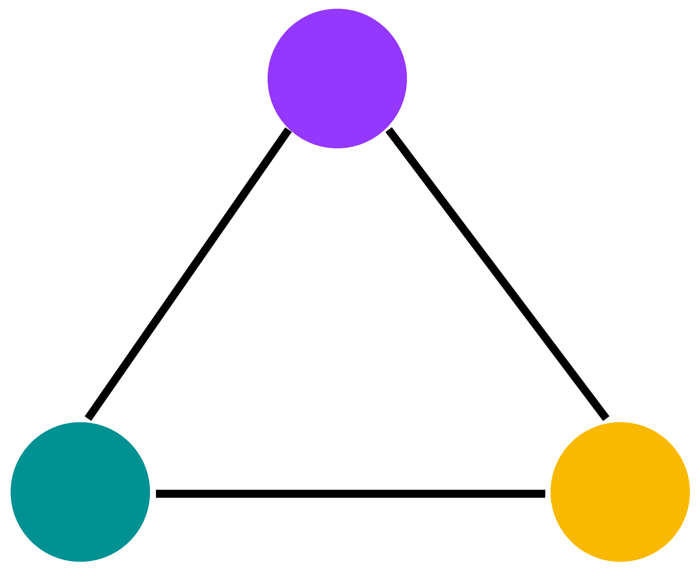
Coloring
Examples

- (Vertex-) Coloring :

Color vertices in a way that no two vertices that share an edge are of the same color

  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

- Chromatic Number :
  - The chromatic number $\chi(G)$ is the minimum number of colors needed to color a graph
  - equivalent : $\chi(G) \leq k \iff G$ is k-partite



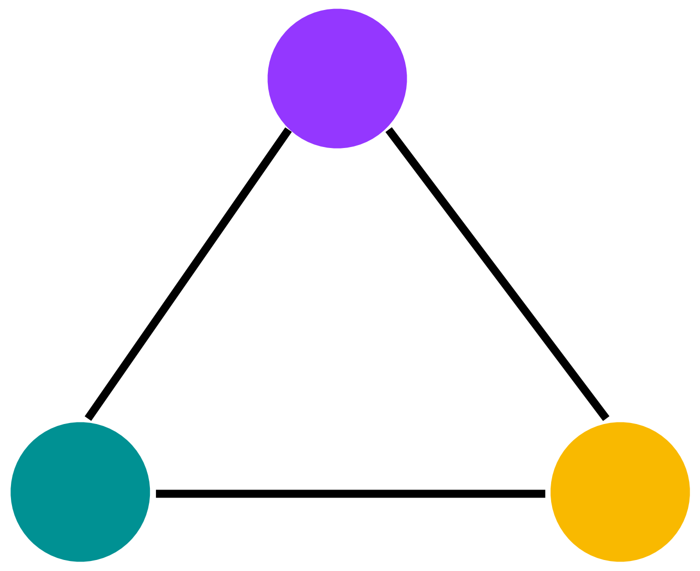
$$\chi(G_1) = 3$$

Coloring
Examples

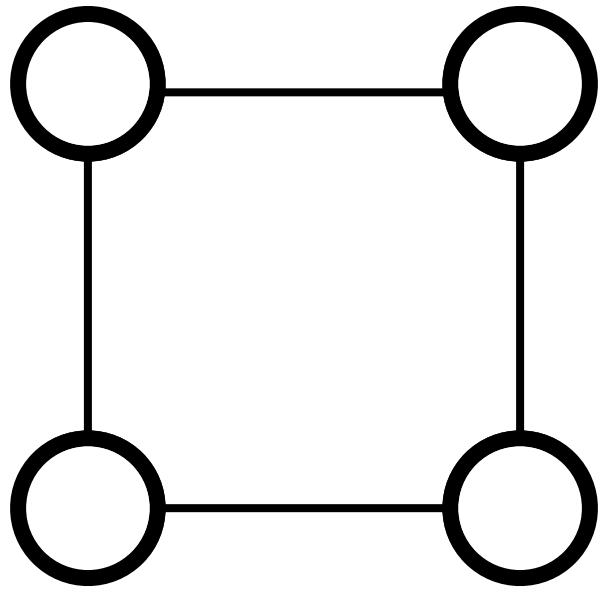
- (Vertex-) Coloring :
  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

Color vertices in a way that no two vertices that share an edge are of the same color

- Chromatic Number :
  - The chromatic number $\chi(G)$ is the minimum number of colors needed to color a graph
  - equivalent : $\chi(G) \leq k \iff G$ is k-partite



$\chi(G_1) = 3$



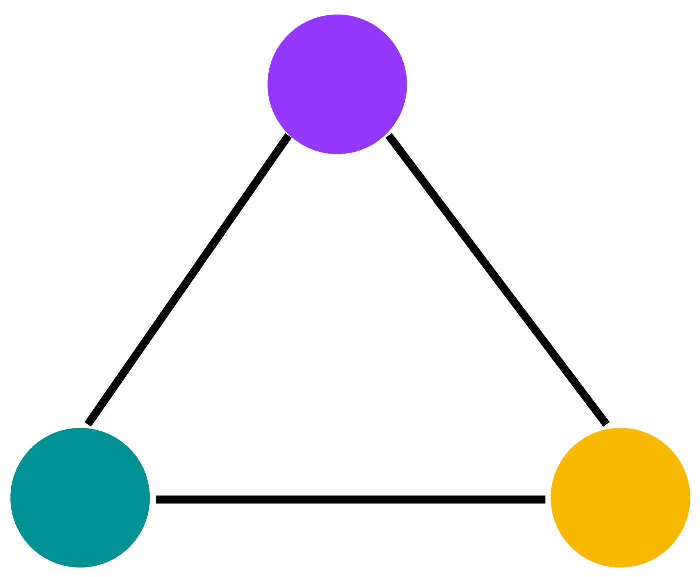
Coloring
Examples

- (Vertex-) Coloring :

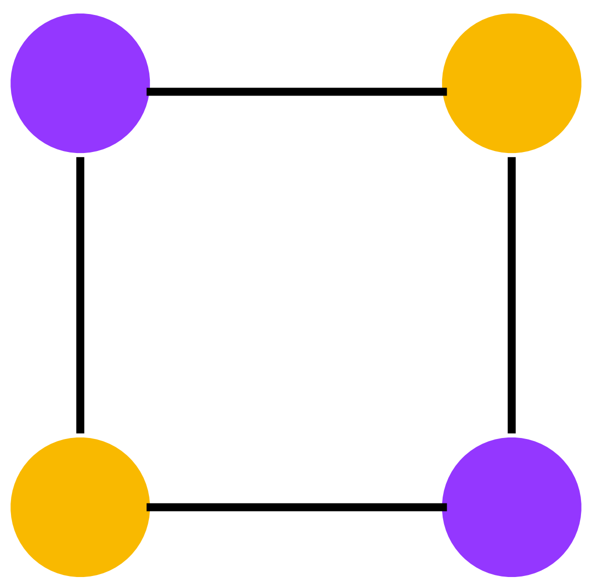
Color vertices in a way that no two vertices that share an edge are of the same color

  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

- Chromatic Number :
  - The chromatic number $\chi(G)$ is the minimum number of colors needed to color a graph
  - equivalent : $\chi(G) \leq k \iff G$ is k-partite



$\chi(G_1) = 3$



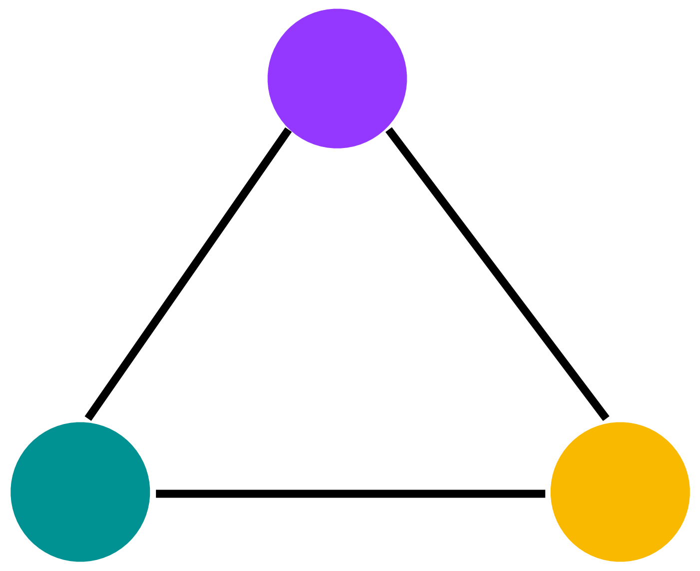
$\chi(G_1) = 2$

Coloring
Examples

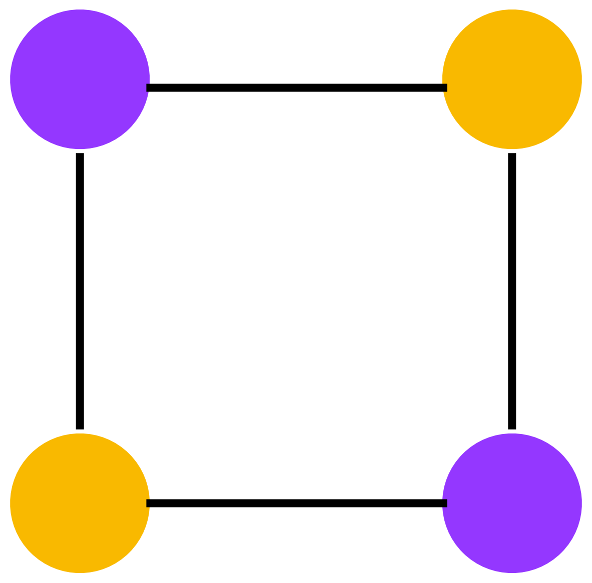
- (Vertex-) Coloring :
  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

Color vertices in a way that no two vertices that share an edge are of the same color

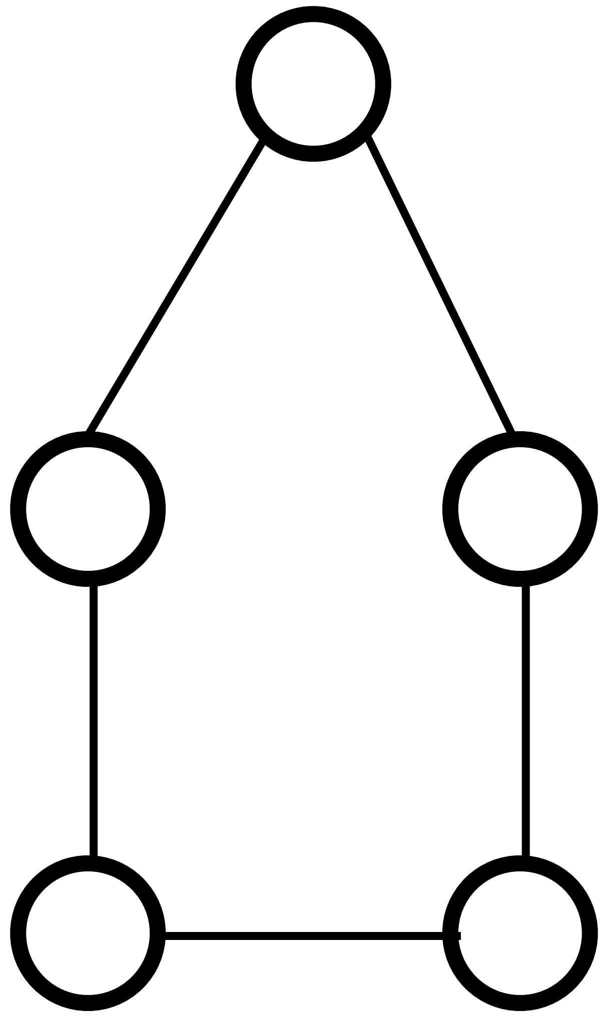
- Chromatic Number :
  - The chromatic number $\chi(G)$ is the minimum number of colors needed to color a graph
  - equivalent : $\chi(G) \leq k \iff G$ is k-partite



$\chi(G_1) = 3$



$\chi(G_1) = 2$



Coloring

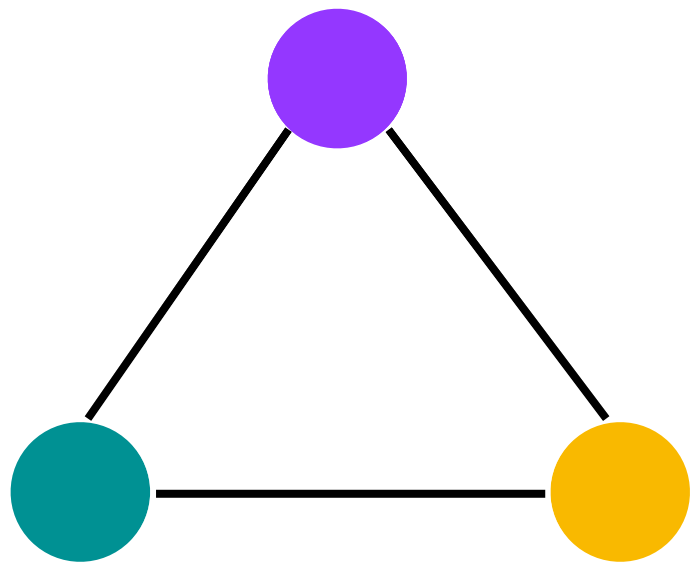
Examples

- (Vertex-) Coloring :

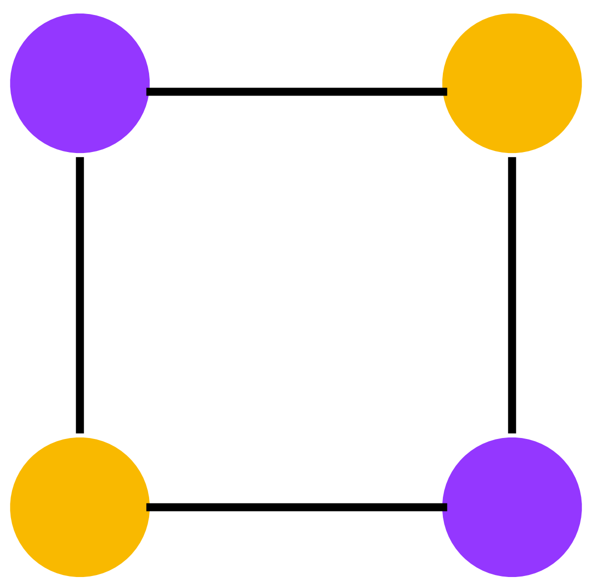
  Color vertices in a way that no two vertices that share an edge are of the same color

  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

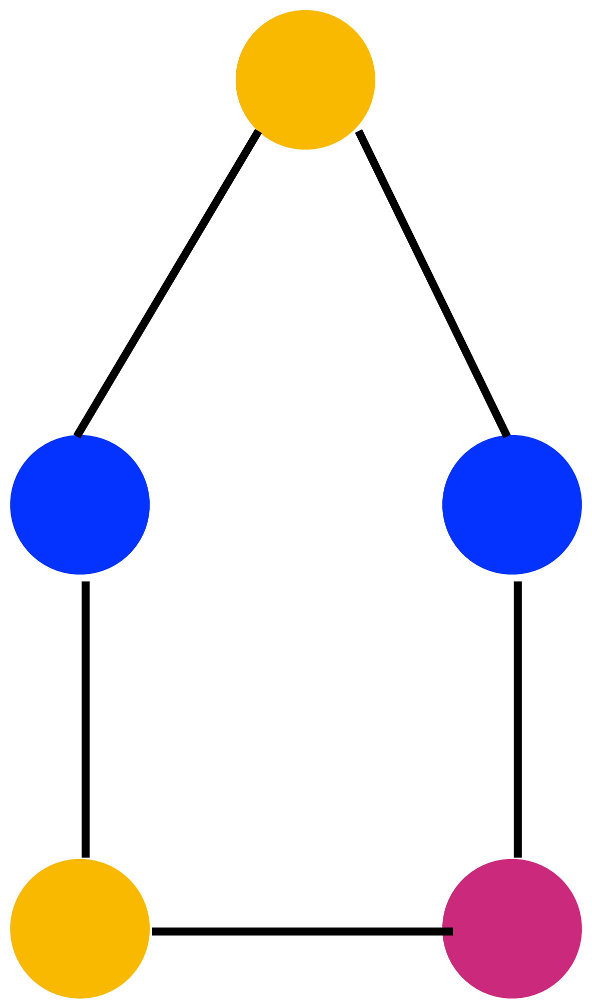
- Chromatic Number :
  - The chromatic number $\chi(G)$ is the minimum number of colors needed to color a graph
  - equivalent : $\chi(G) \leq k \iff G$ is k-partite



$\chi(G_1) = 3$



$\chi(G_1) = 2$



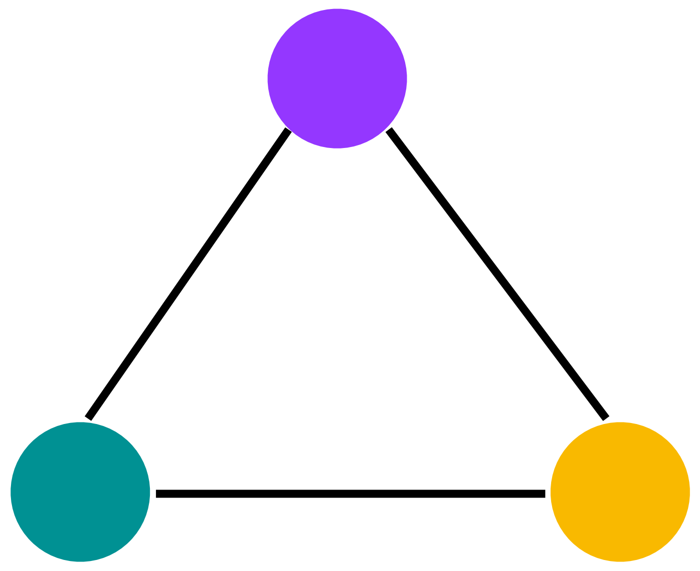
$\chi(G_1) = 3$

Coloring
Examples

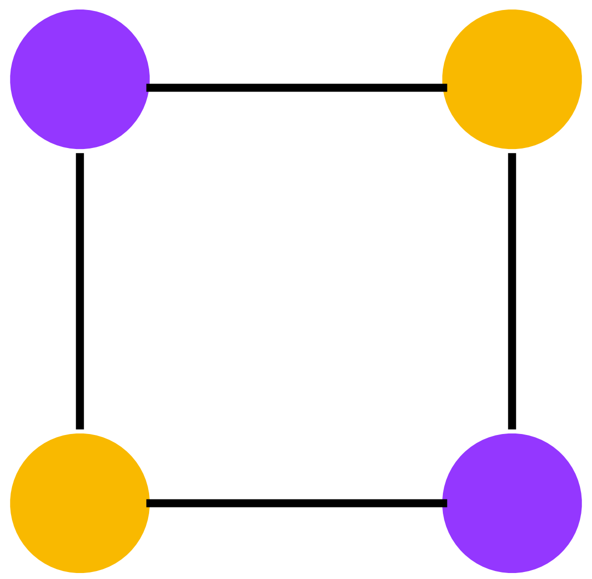
- (Vertex-) Coloring :
  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \rightarrow [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

Color vertices in a way that no two vertices that share an edge are of the same color

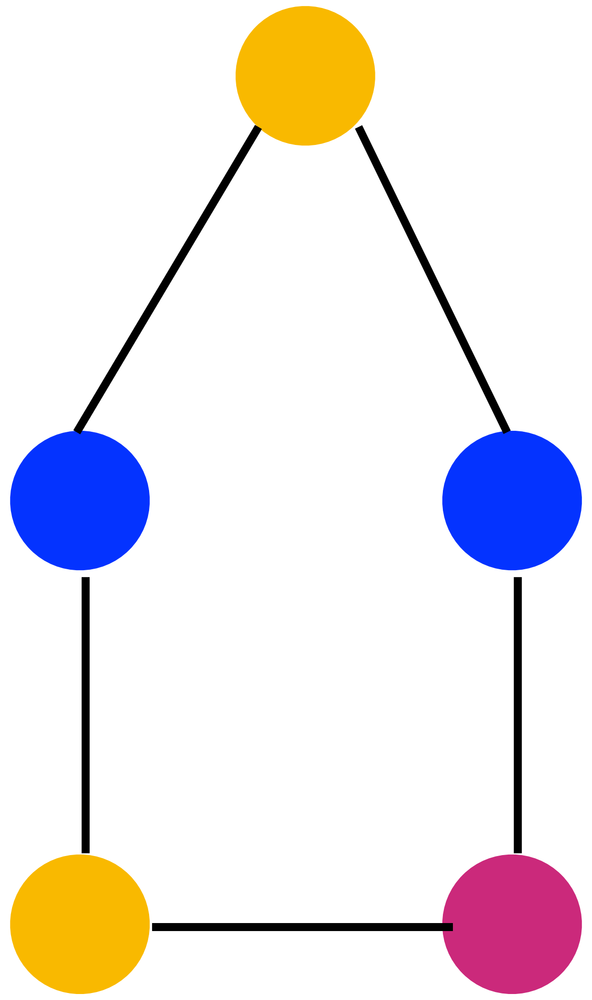
- Chromatic Number :
  - The chromatic number $\chi(G)$ is the minimum number of colors needed to color a graph
  - equivalent : $\chi(G) \leq k \iff G$ is k-partite



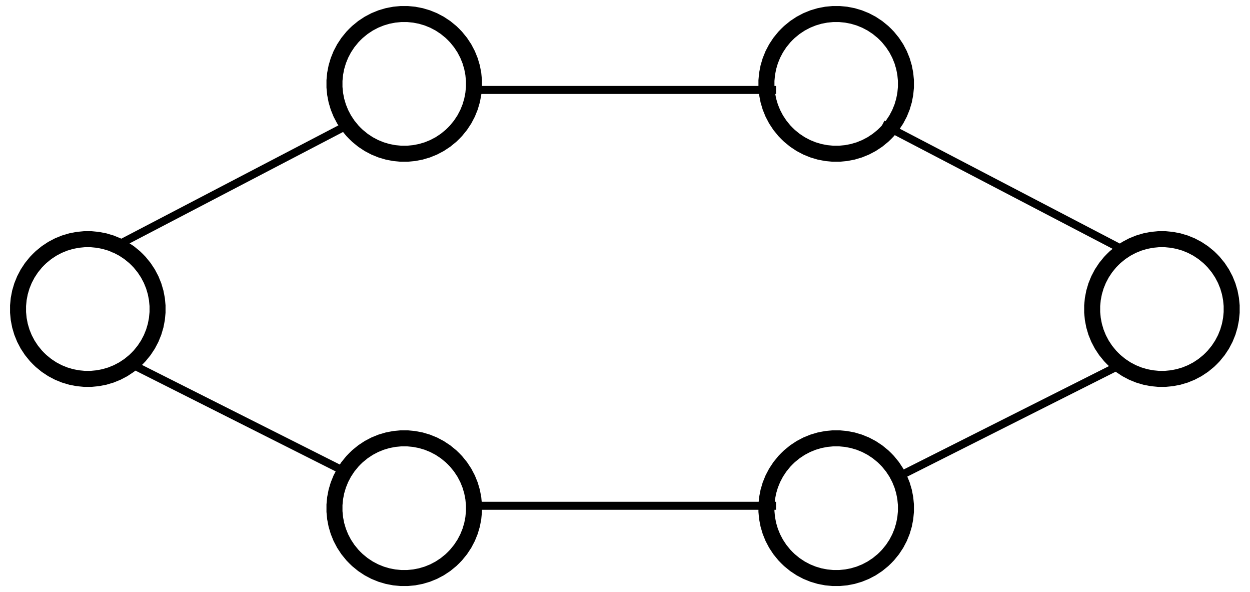
$\chi(G_1) = 3$



$\chi(G_1) = 2$



$\chi(G_1) = 3$



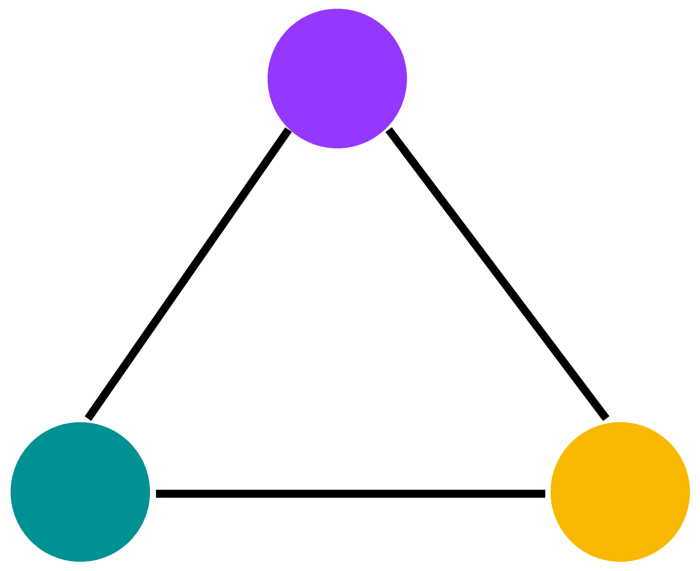
Coloring
Examples

- (Vertex-) Coloring :

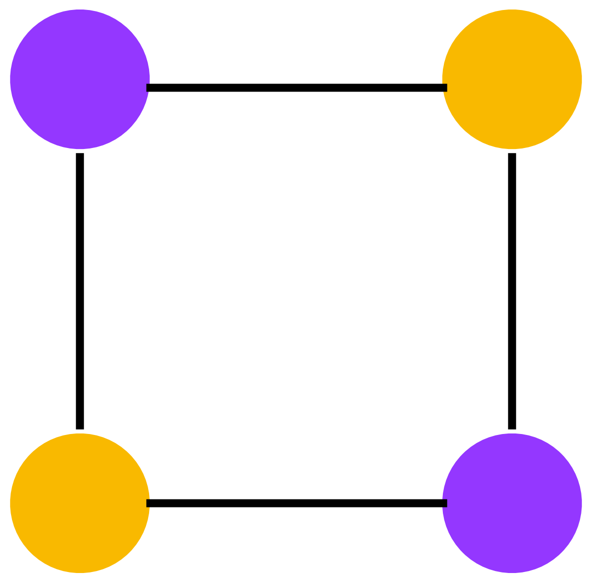
Color vertices in a way that no two vertices that share an edge are of the same color

  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

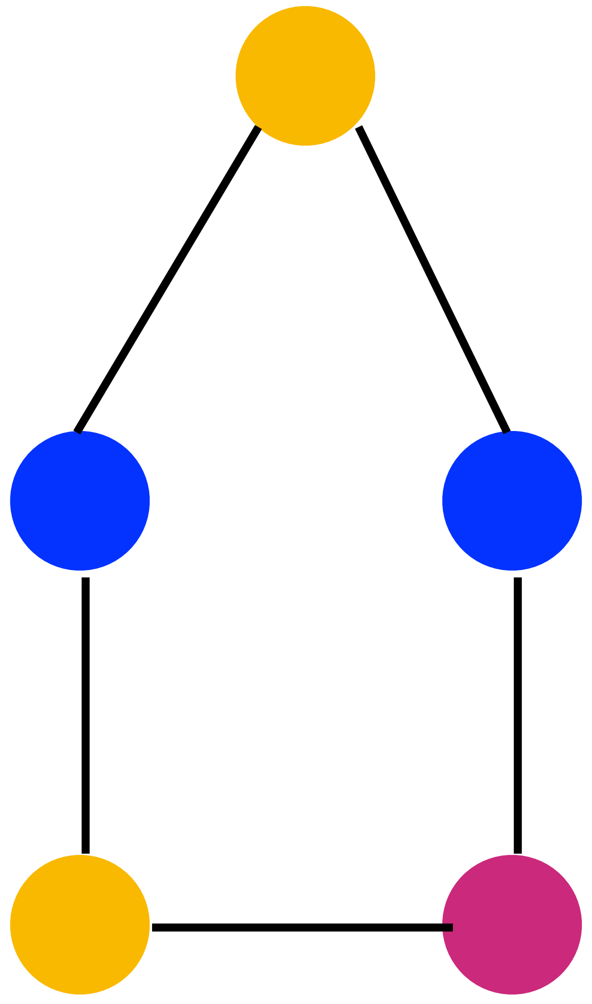
- Chromatic Number :
  - The chromatic number $\chi(G)$ is the minimum number of colors needed to color a graph
  - equivalent : $\chi(G) \leq k \iff G$ is k-partite



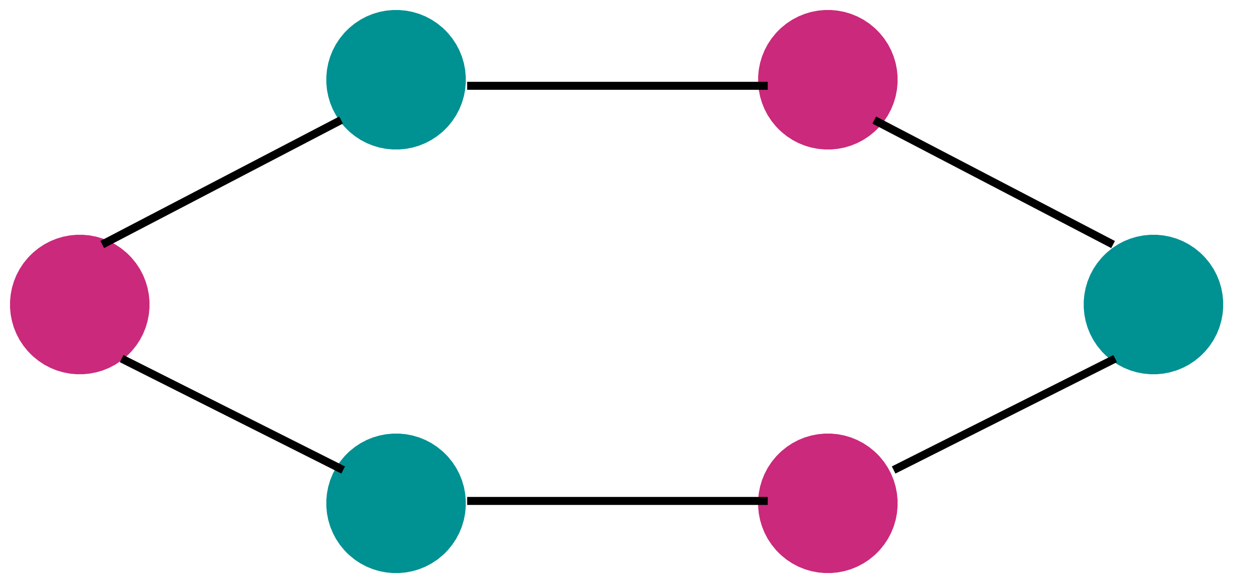
$\chi(G_1) = 3$



$\chi(G_1) = 2$



$\chi(G_1) = 3$



$\chi(G_1) = 2$

Coloring

Examples

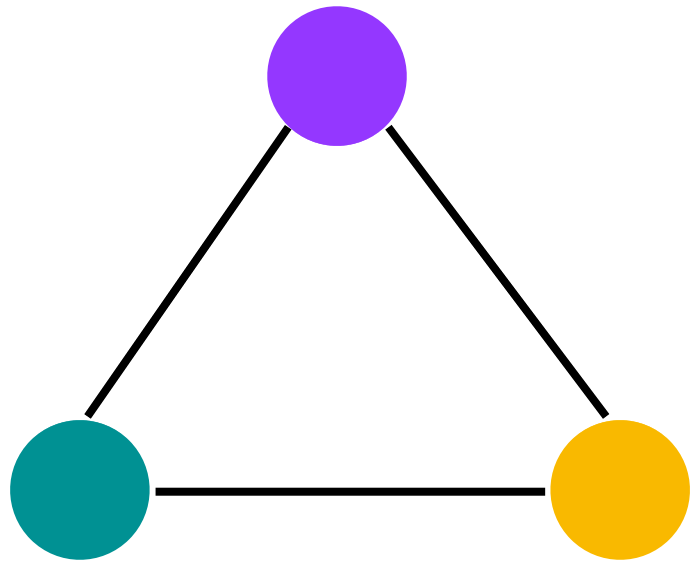
- (Vertex-) Coloring :

  Color vertices in a way that no two vertices that share an edge are of the same color

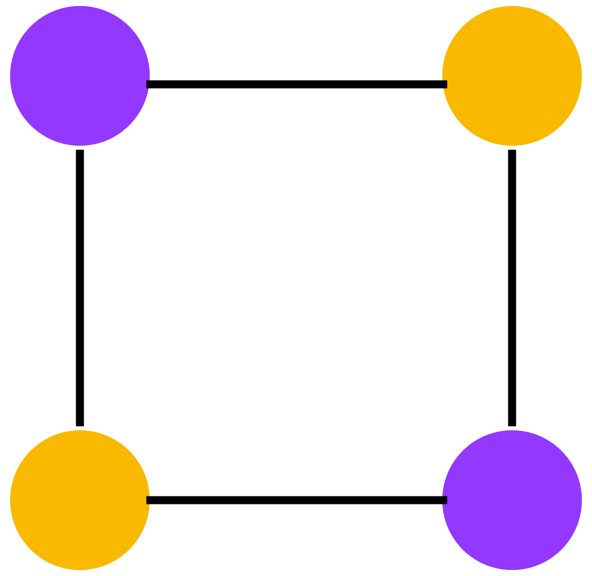
  - A (vertex-) coloring of $G = (V, E)$ with k colors is a mapping $c : V \to [k]$ s.t. $c(u) \neq c(v)$ for all edges $\{u, v\} \in E$

- Chromatic Number :
  - The chromatic number $\chi(G)$ is the minimum number of colors needed to color a graph
  - equivalent : $\chi(G) \leq k \iff G$ is k-partite

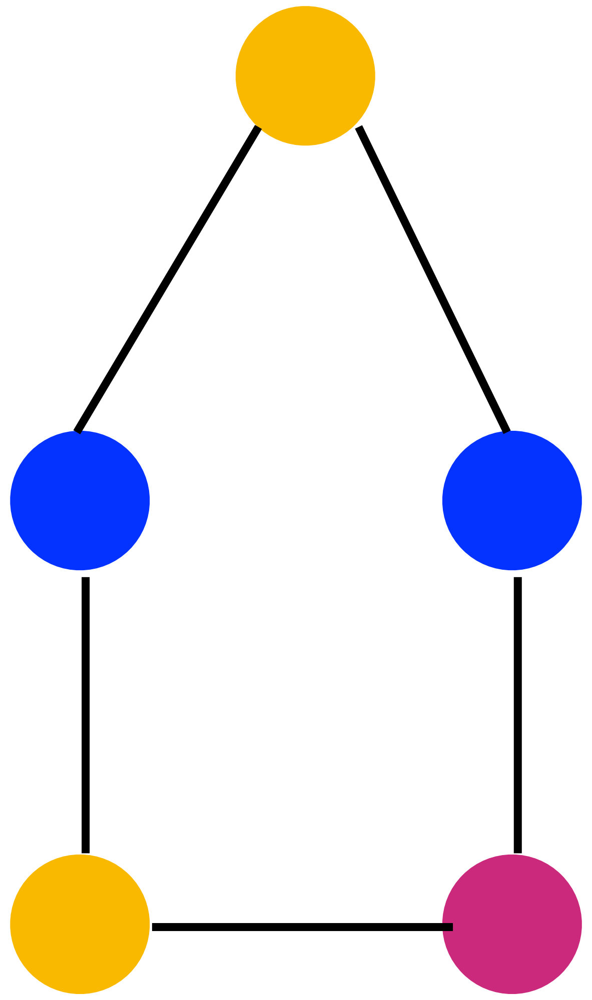
Do you notice something ?



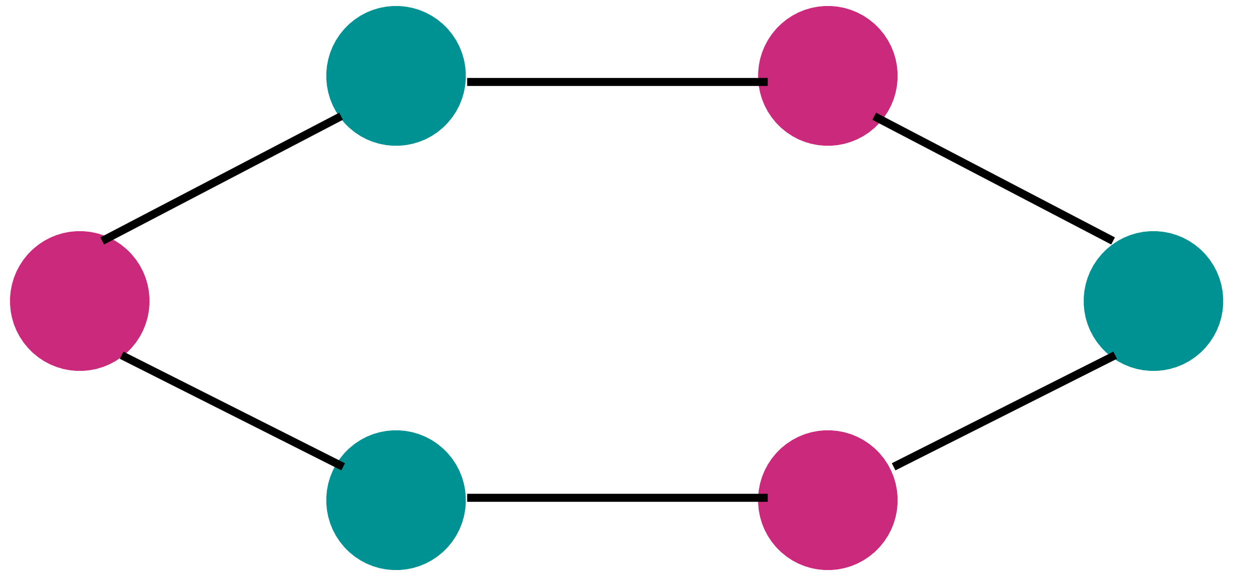
$\chi(G_1) = 3$



$\chi(G_1) = 2$



$\chi(G_1) = 3$



$\chi(G_1) = 2$

Coloring
Problem

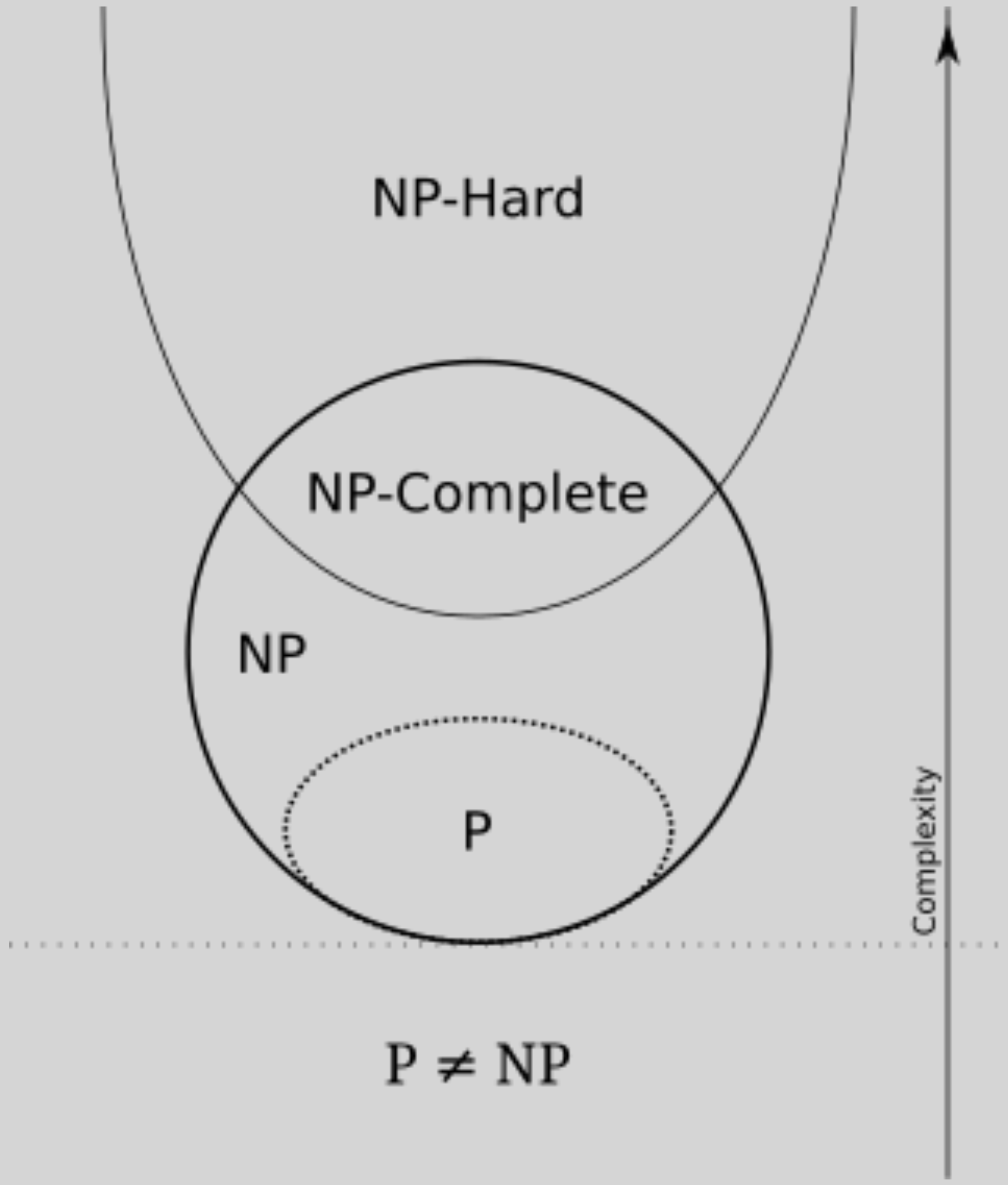
For all $k \geq 3$, given a graph $G = (V, E)$,

is $\chi(G) \leq k$?

NP - Complete

NP-Complete

For all $k \geq 3$, given a graph $G = (V, E)$, is $\chi(G) \leq k$?



P : polynomial

NP : non-deterministic polynomial

- NP is the set of decision problems *solvable* in polynomial time by a nondeterministic Turing machine.
- NP is the set of decision problems *verifiable* in polynomial time by a deterministic Turing machine.

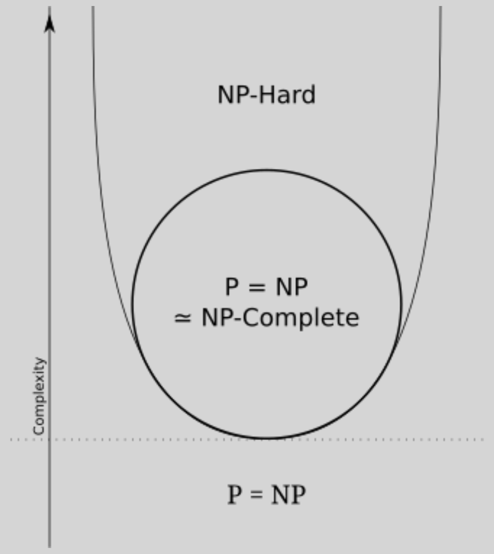
Complexity Theory

TI next semester

NP - Complete

A problem a in NP is NP-complete if :

$$a \in P \implies P = NP$$



?

Let’s take a break

A&W 😊

WhatsApp group

Coloring

Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

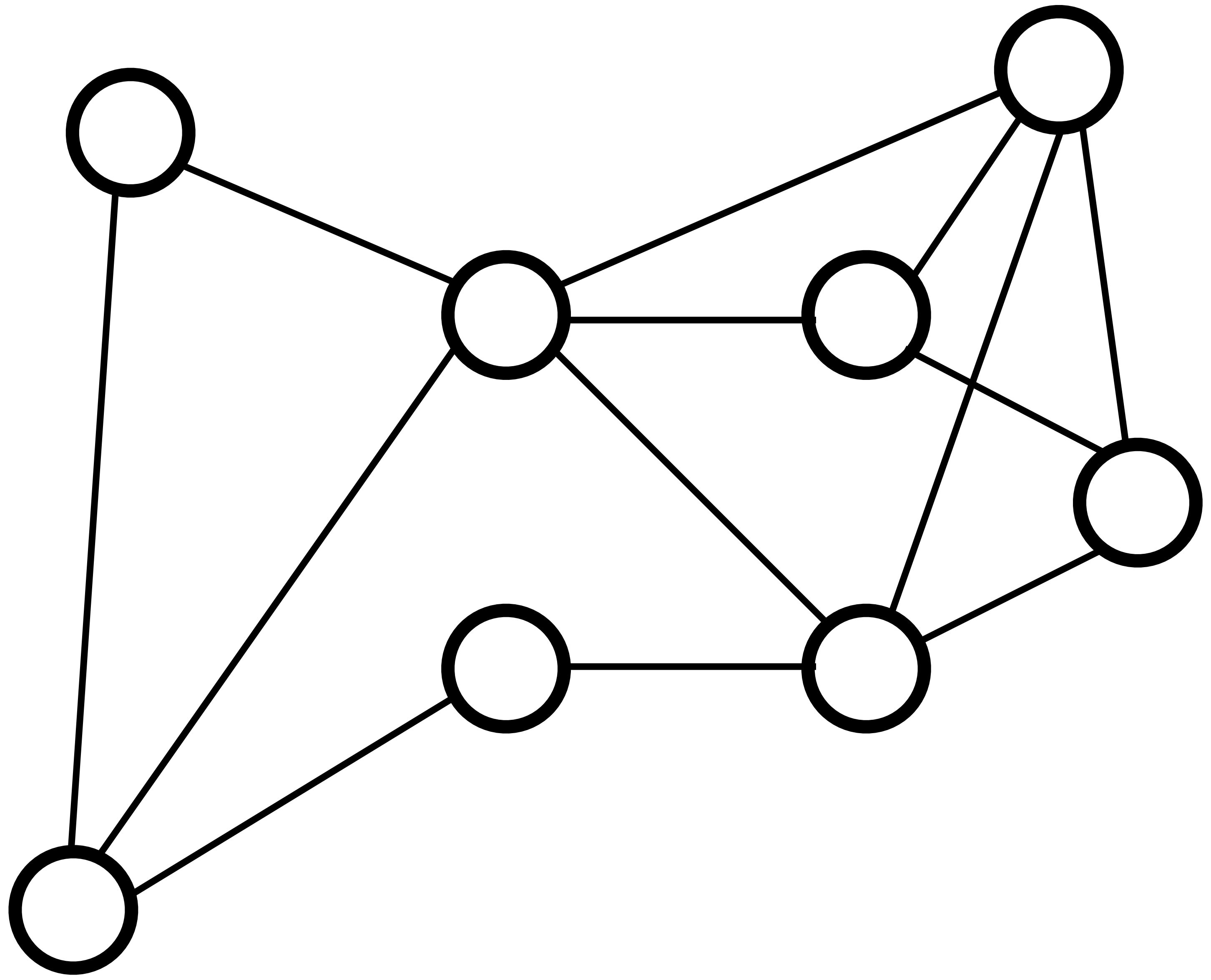
min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u] \text{ for all } u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :



C :

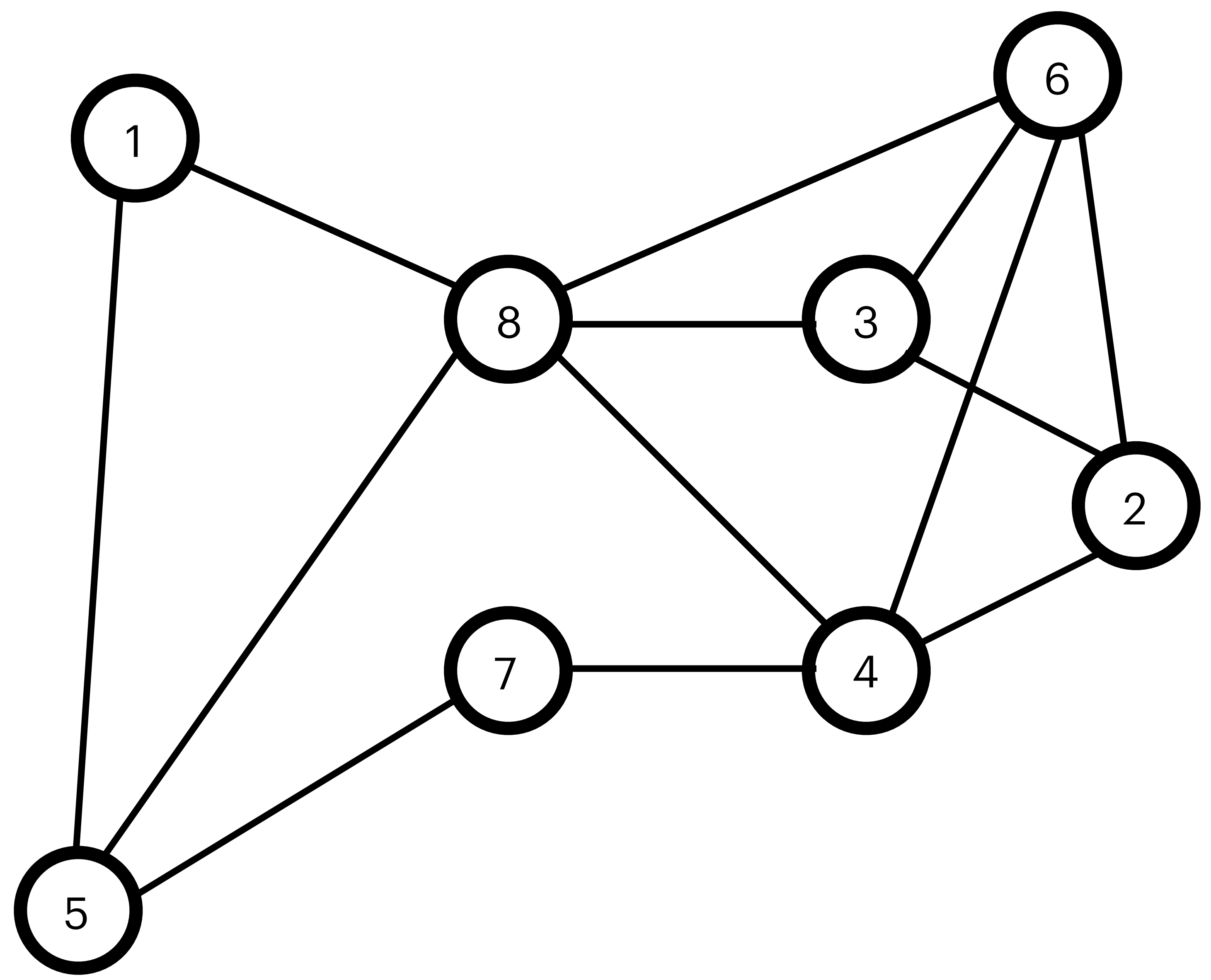
i	1	2	3	4	5	6	7	8
$c[v_i]$								

Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :



C :

i	1	2	3	4	5	6	7	8
$c[v_i]$								

Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

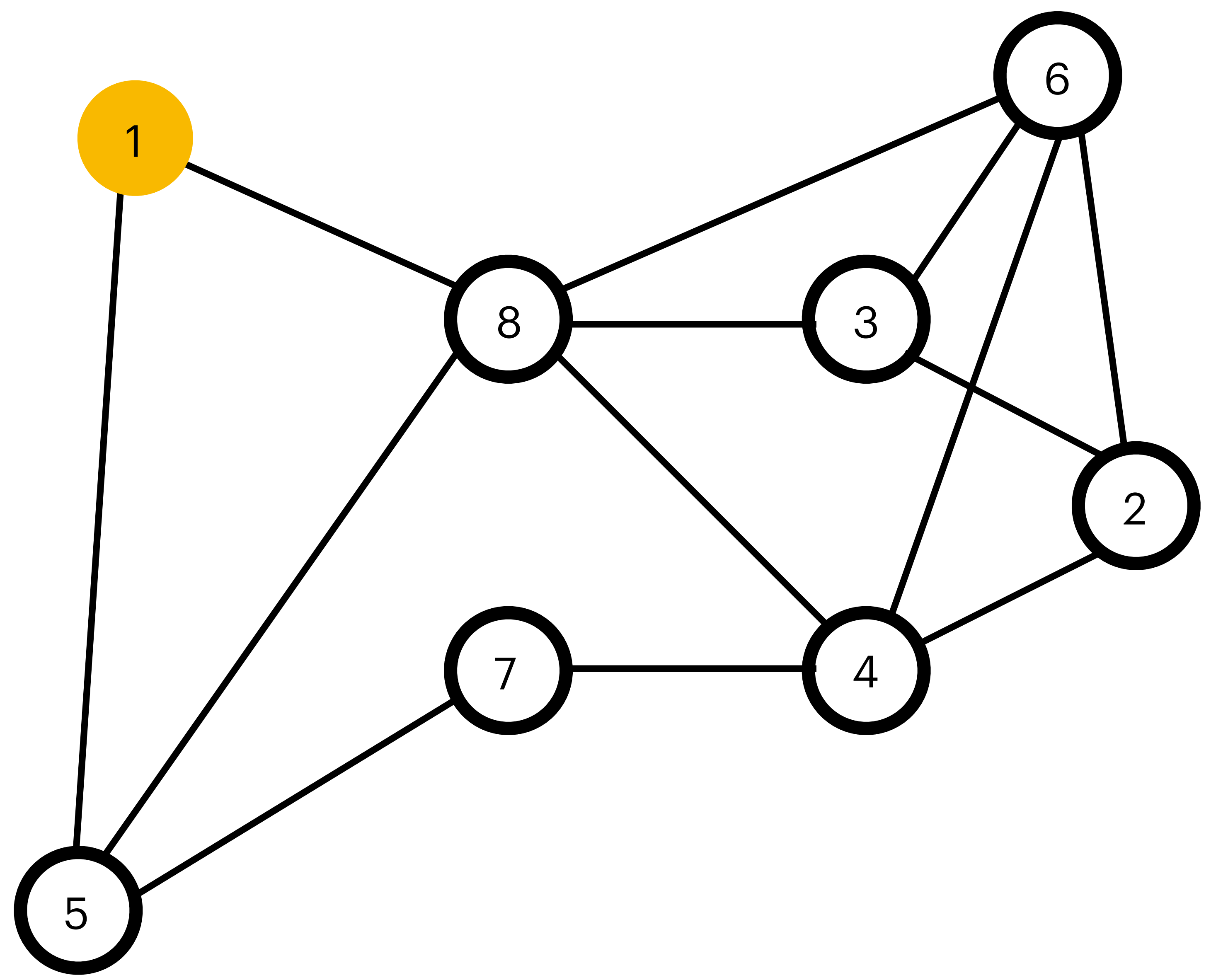
    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :

k = 1

C :

i	1	2	3	4	5	6	7	8
$c[v_i]$	1							



Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

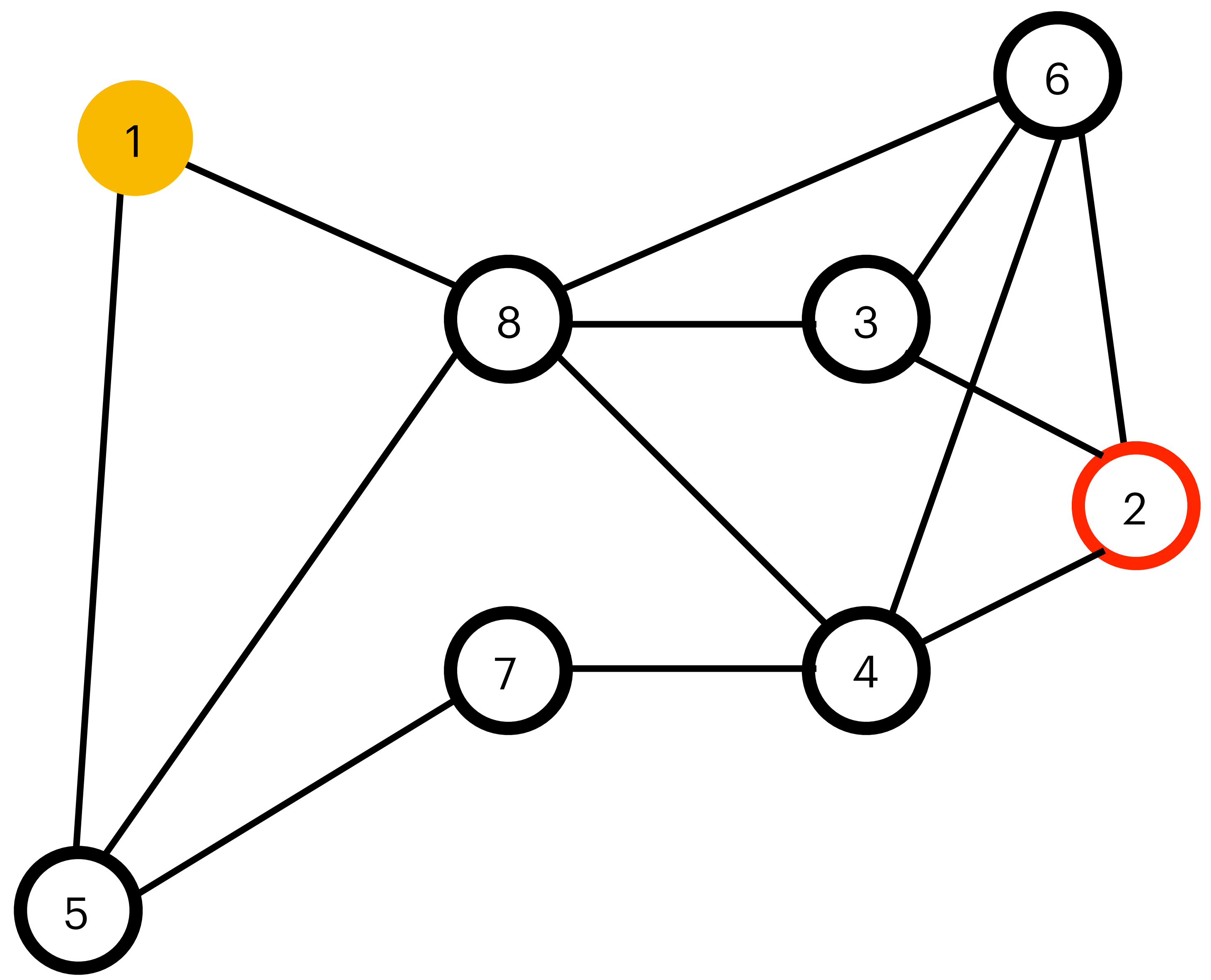
    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :

k = 1

considering ks

✅



C :

i	1	2	3	4	5	6	7	8
c[v_i]	1							

Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

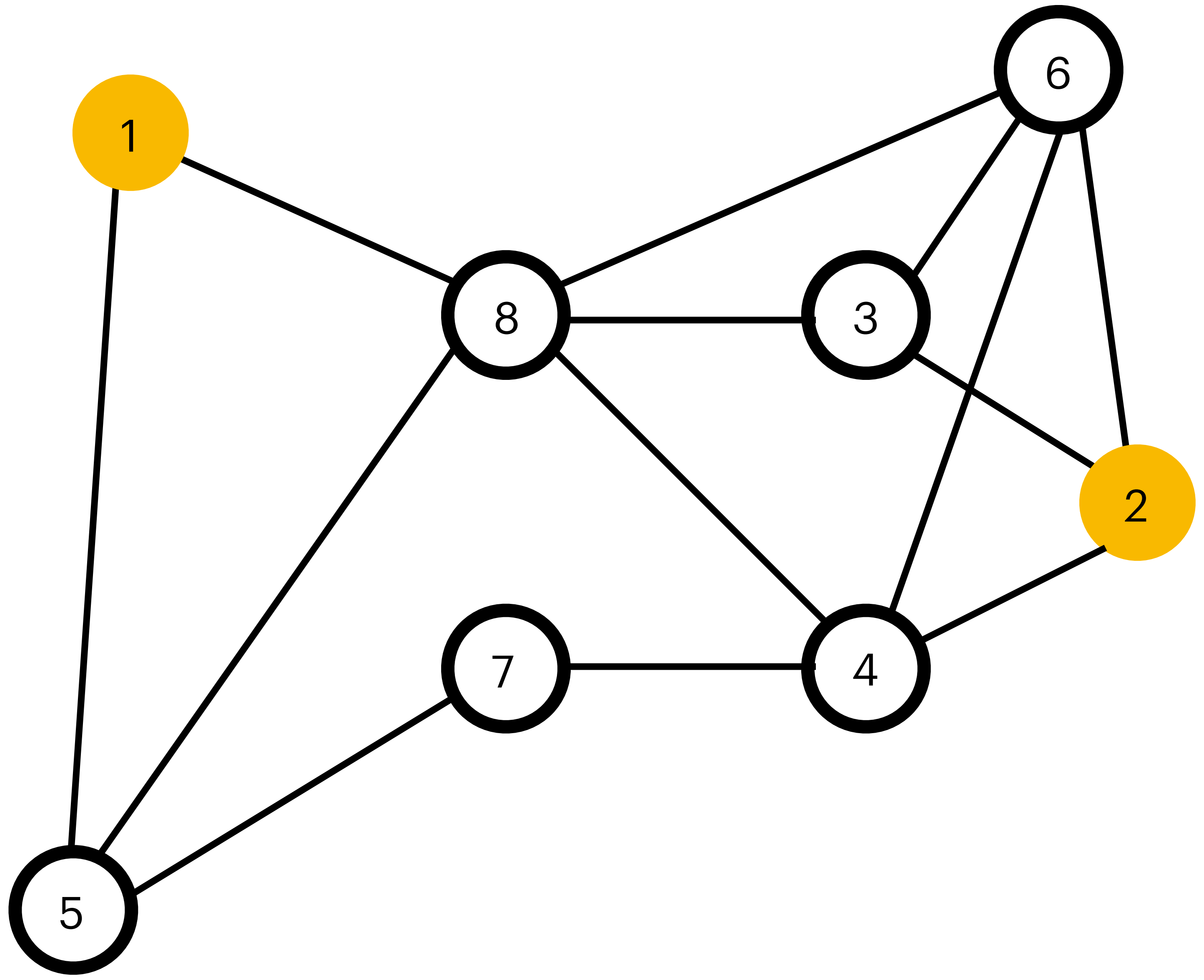
    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :

k = 1

considering ks

✅



C :

i	1	2	3	4	5	6	7	8
c[vᵢ]	1	1						

Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

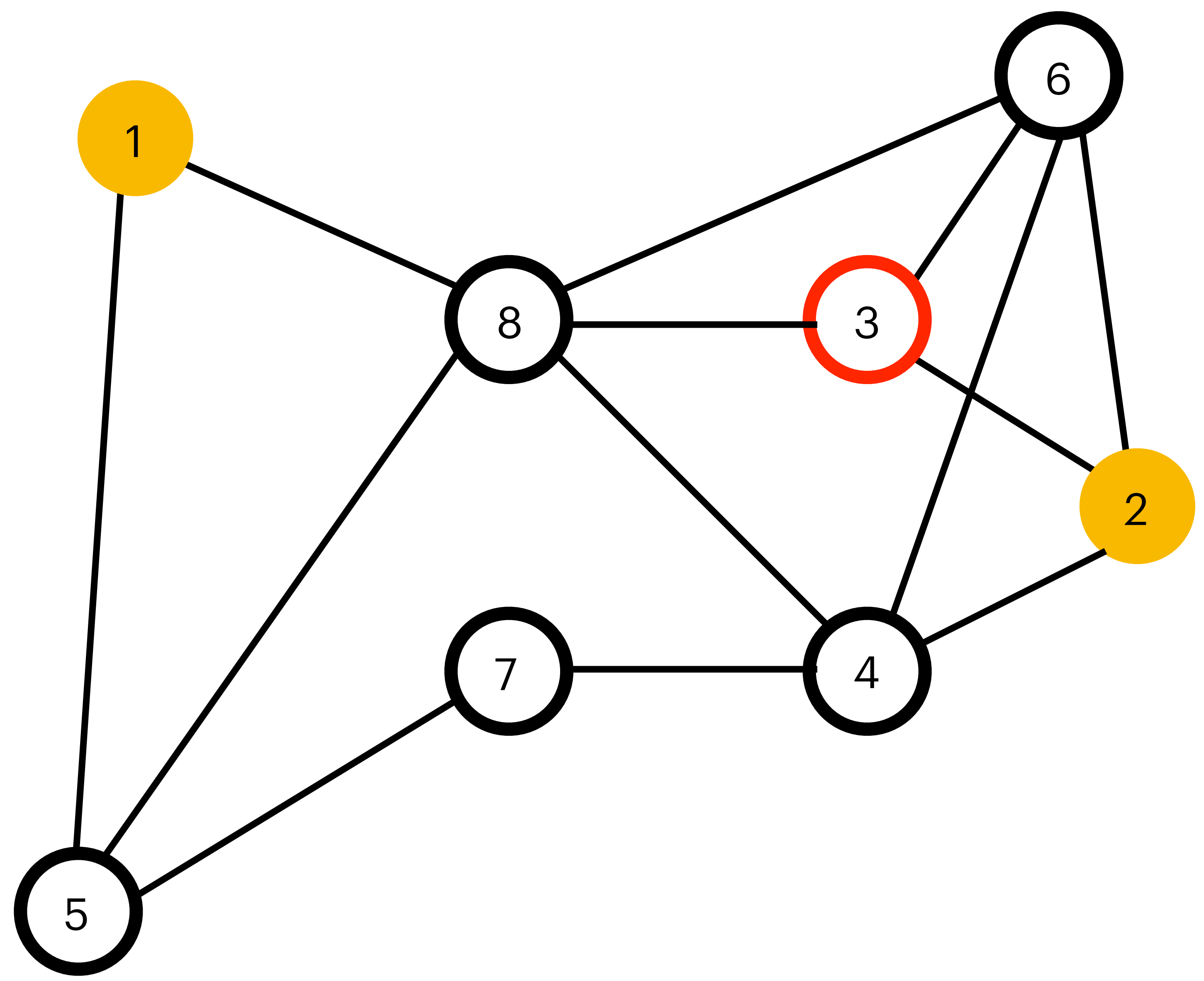
k indexing for colors :

k = 1

k = 2

considering ks

✅



C :

i	1	2	3	4	5	6	7	8
c[v_i]	1	1						

Coloring

Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \dots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u] \text{ for all } u \in N(v_i) \cap \{v_1, \dots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

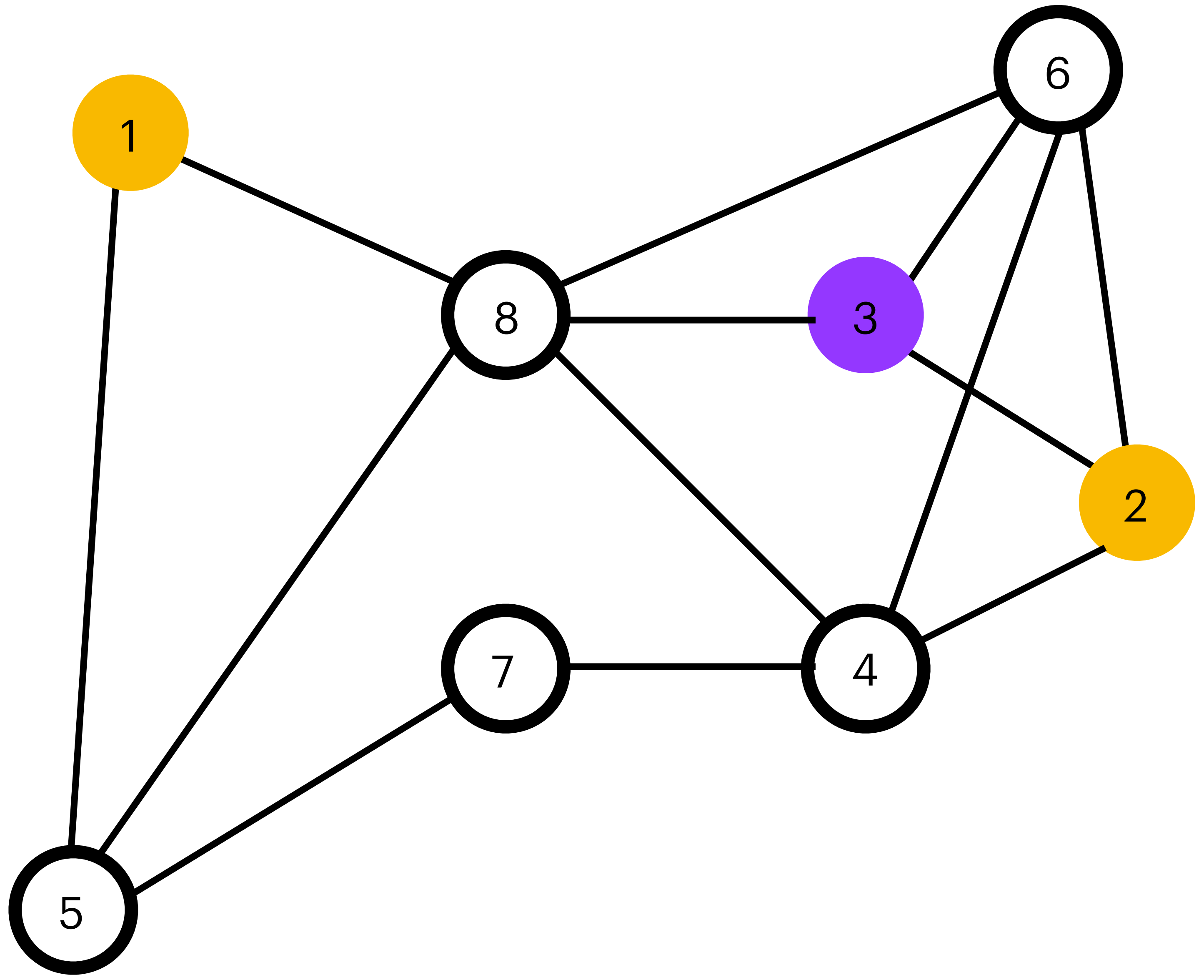
k indexing for colors :

k = 1

k = 2

considering ks

✅



C :

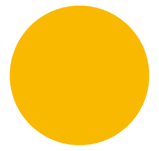
i	1	2	3	4	5	6	7	8
$c[v_i]$	1	1	2					

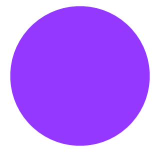
Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u] \text{ for all } u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

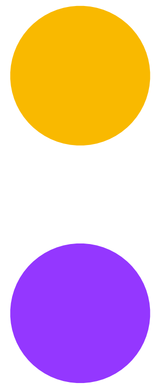
    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :

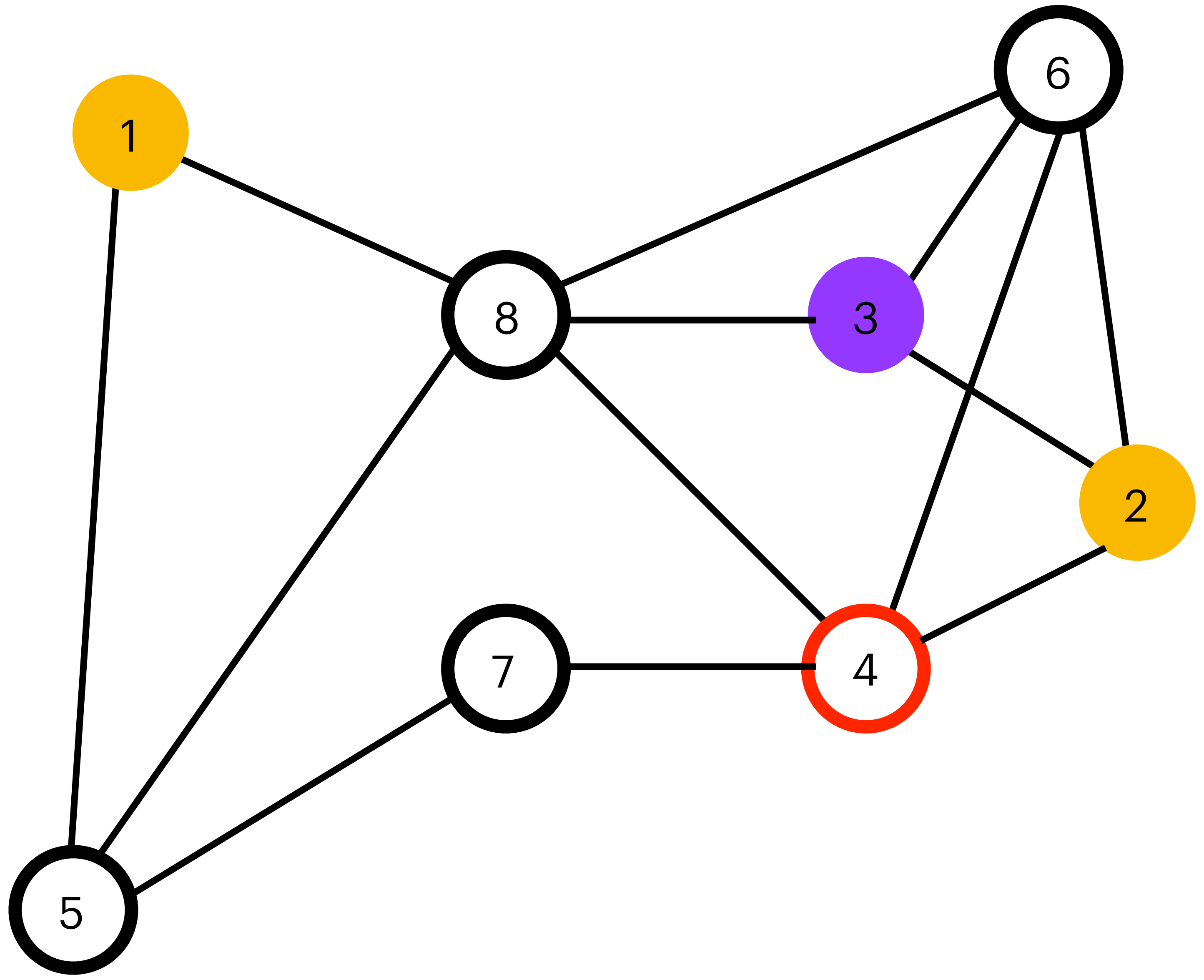
k = 1 

k = 2 

considering ks



✅



C :

i	1	2	3	4	5	6	7	8
c[vi]	1	1	2					

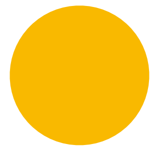
Coloring

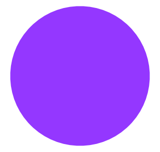
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

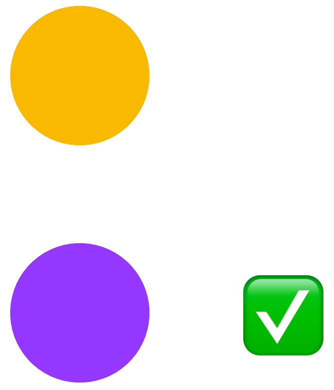
    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

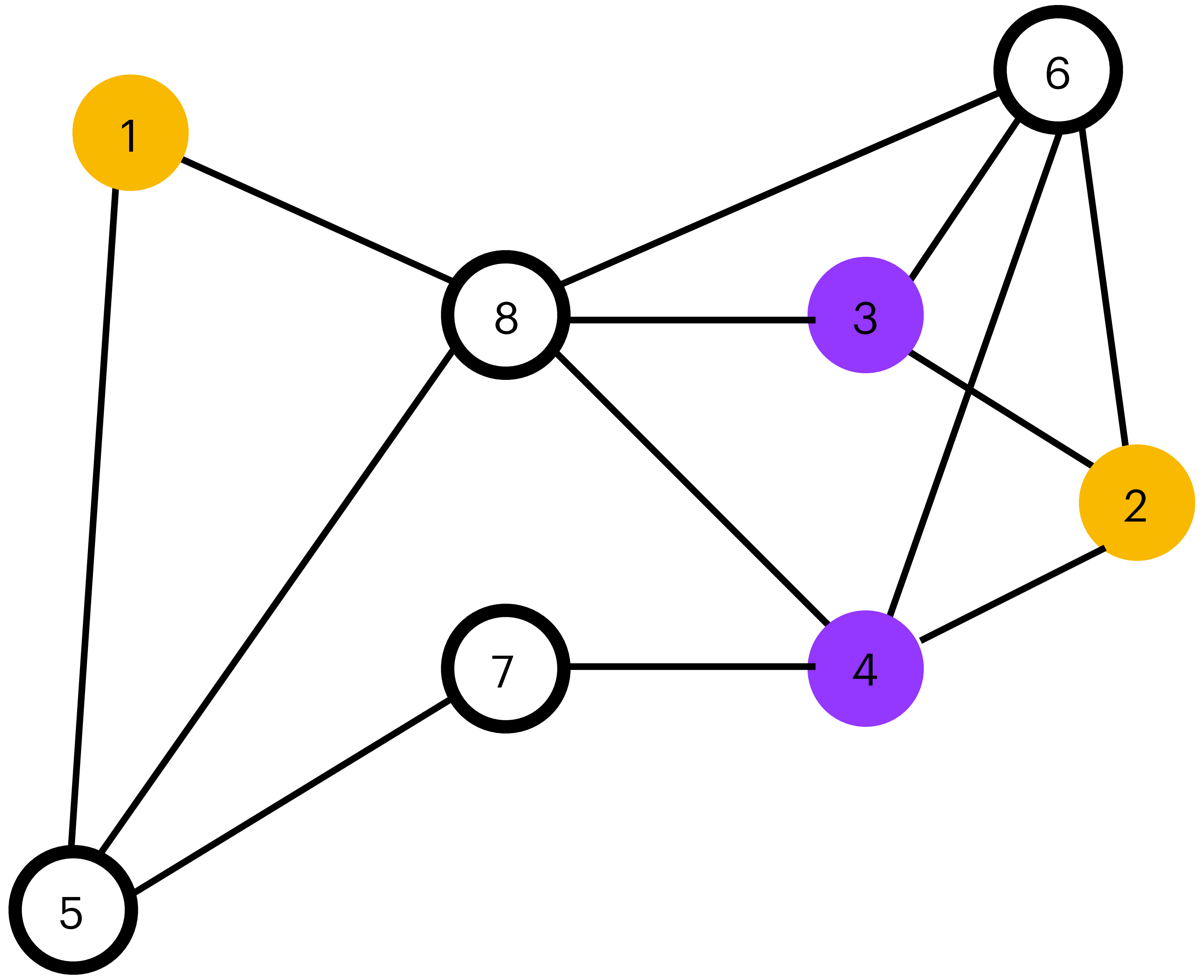
k indexing for colors :

k = 1 

k = 2 

considering ks





C :

i	1	2	3	4	5	6	7	8
$c[v_i]$	1	1	2	2				

Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

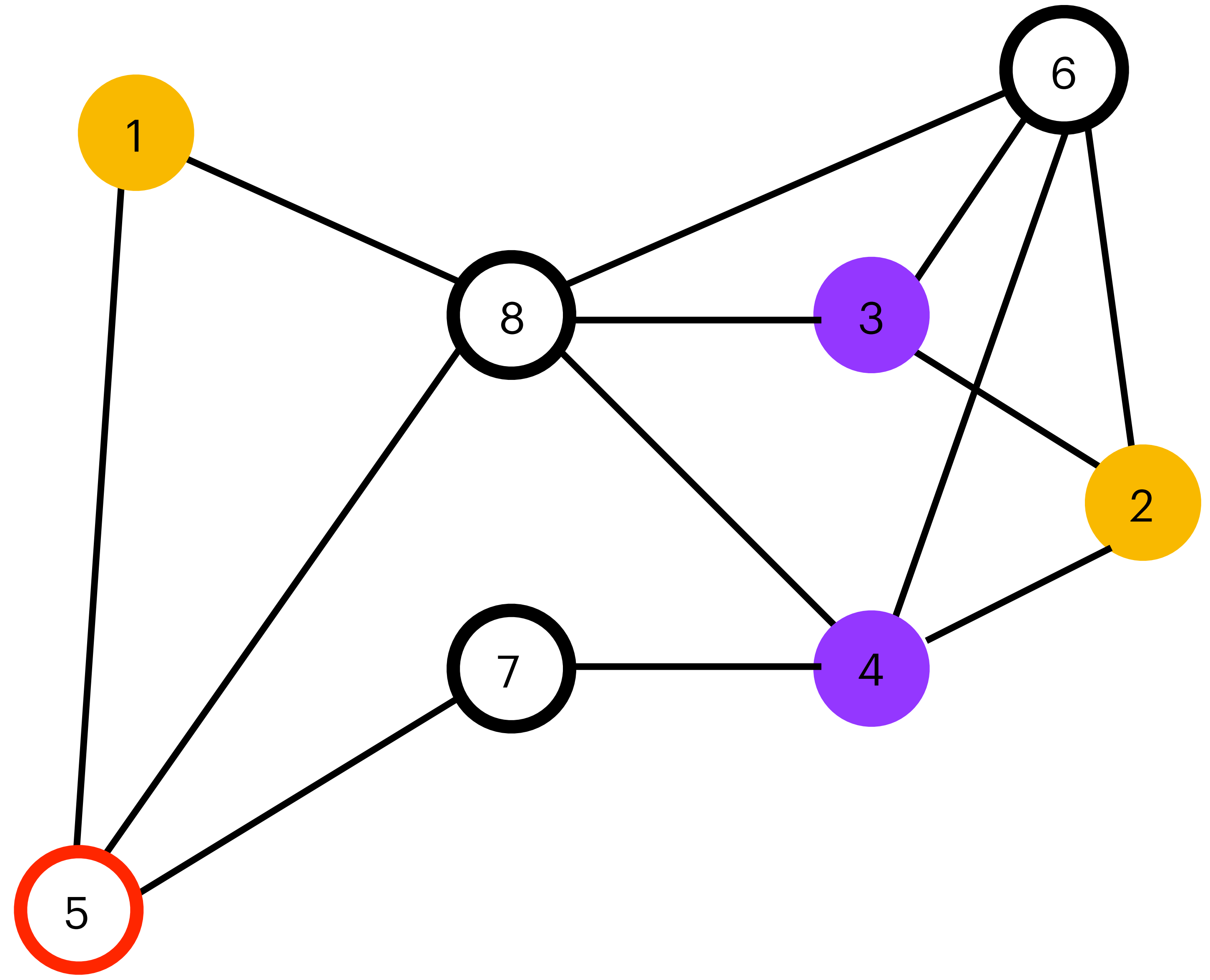
k indexing for colors :

k = 1

k = 2

considering ks

✅



C :

i	1	2	3	4	5	6	7	8
$c[v_i]$	1	1	2	2				

Coloring
Greedy Algorithm

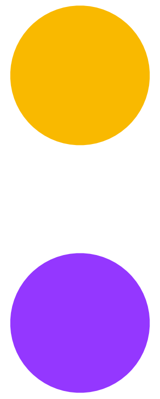
- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u] \text{ for all } u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

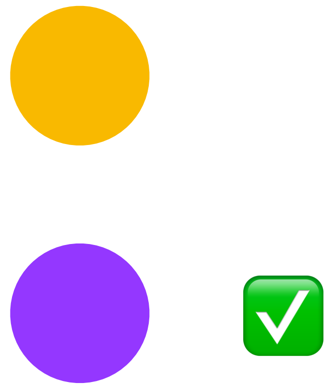
k indexing for colors :

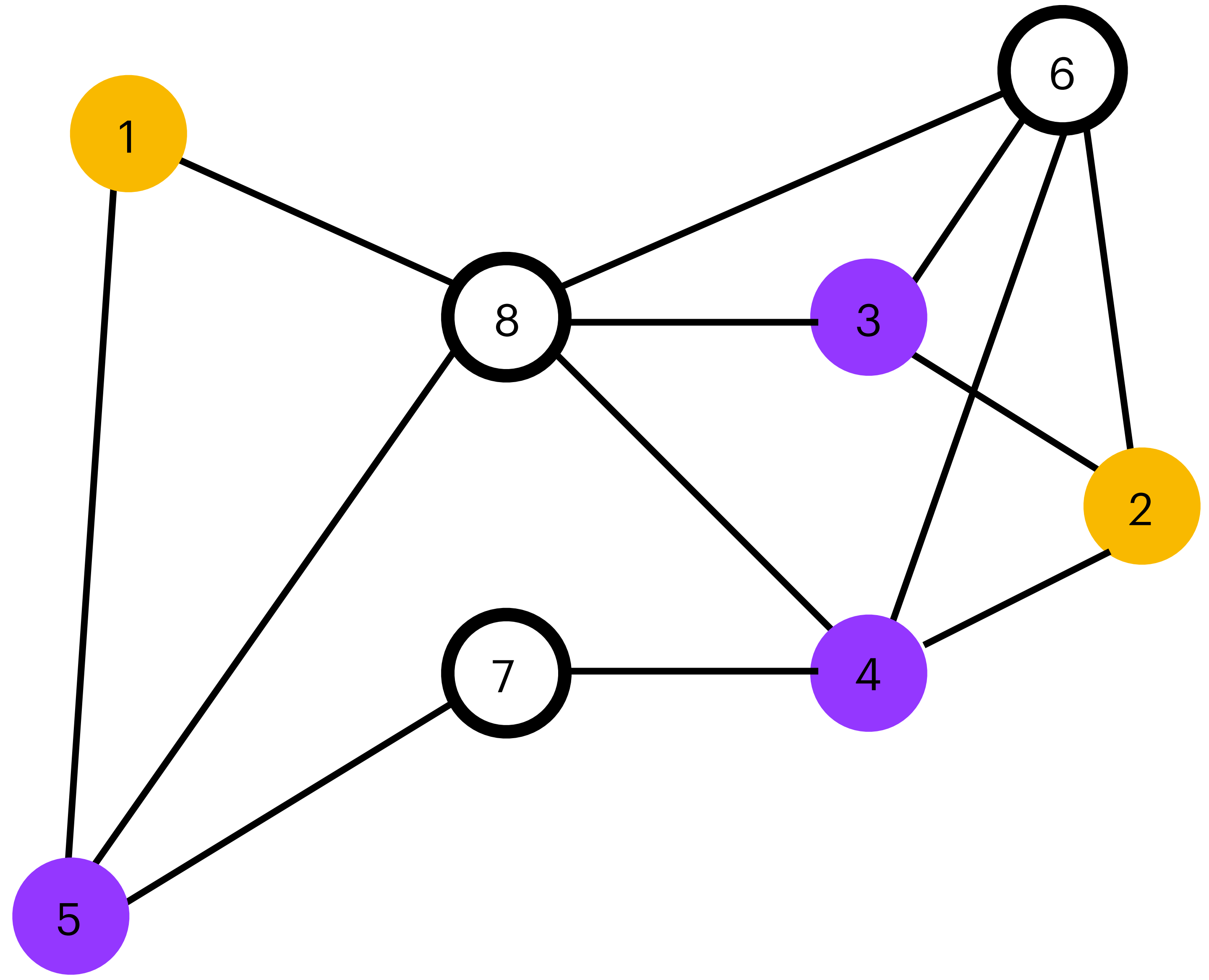
k = 1

k = 2



considering ks





C :

i	1	2	3	4	5	6	7	8
$c[v_i]$	1	1	2	2	2			

Coloring

Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

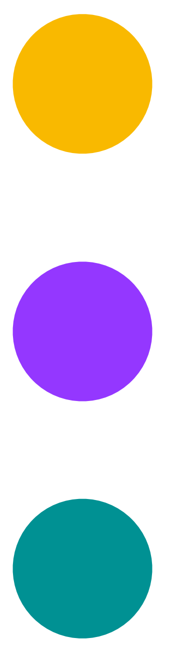
    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :

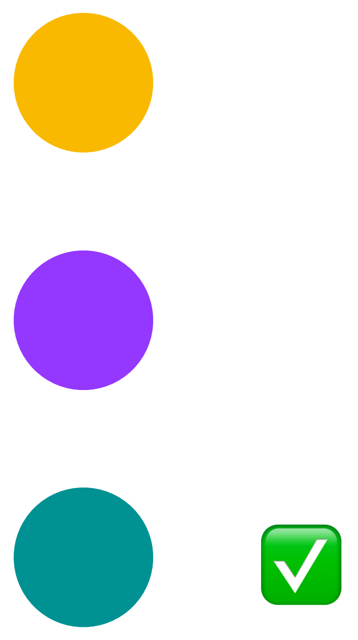
k = 1

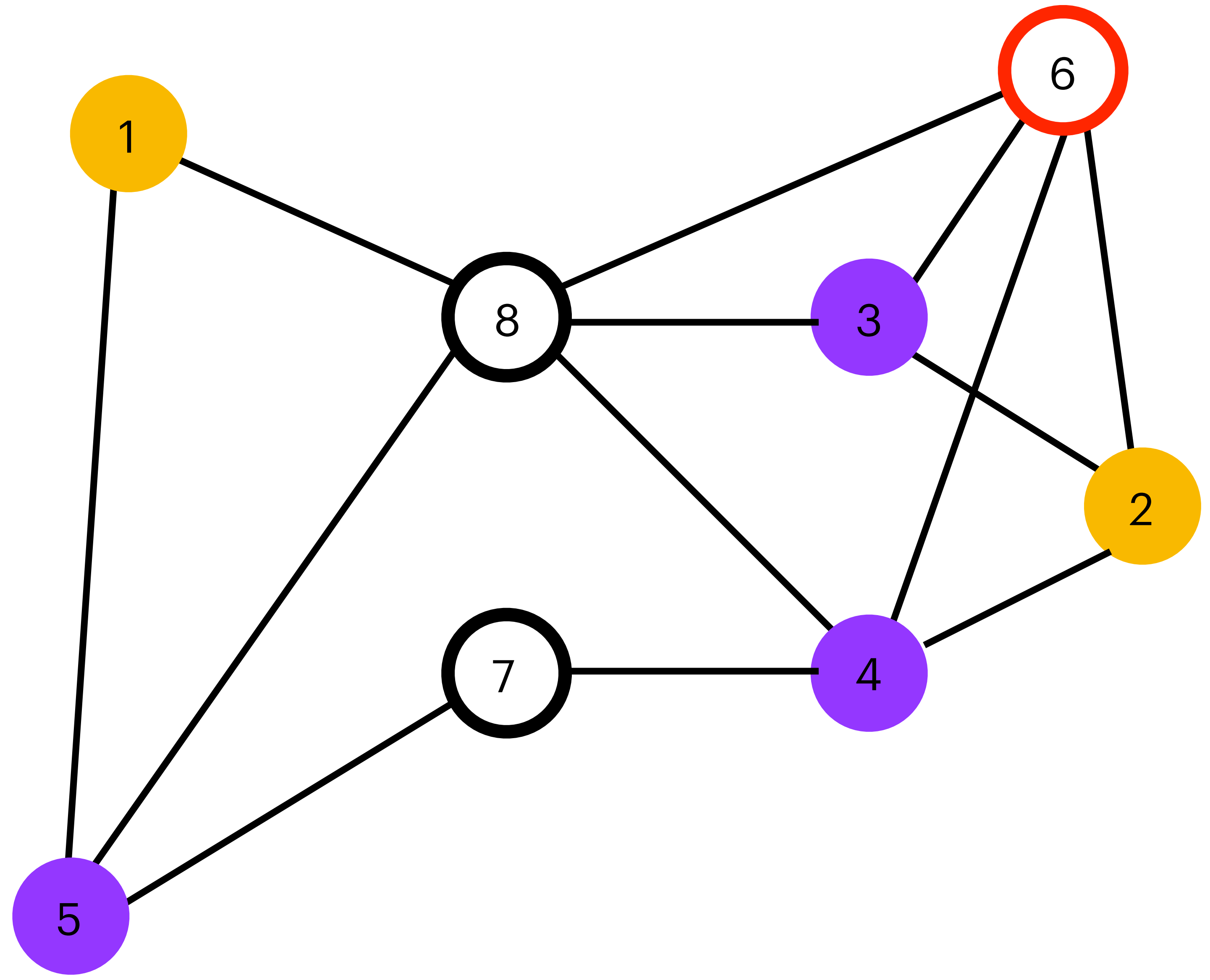
k = 2

k = 3



considering ks





C :

i	1	2	3	4	5	6	7	8
$c[v_i]$	1	1	2	2	2			

Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$
    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :

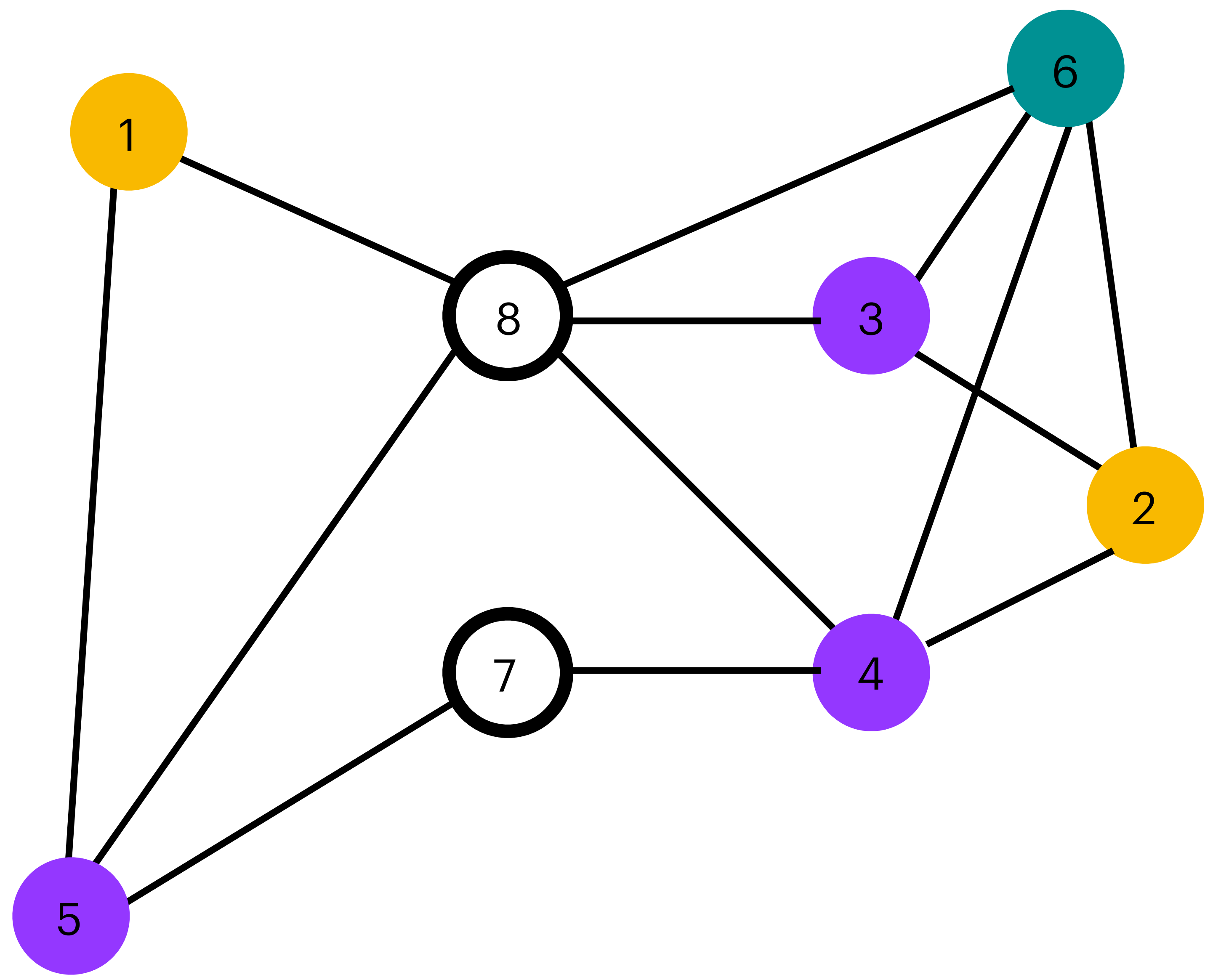
k = 1

k = 2

k = 3

considering ks

✅



C :

i	1	2	3	4	5	6	7	8
c[v_i]	1	1	2	2	2	3		

Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :

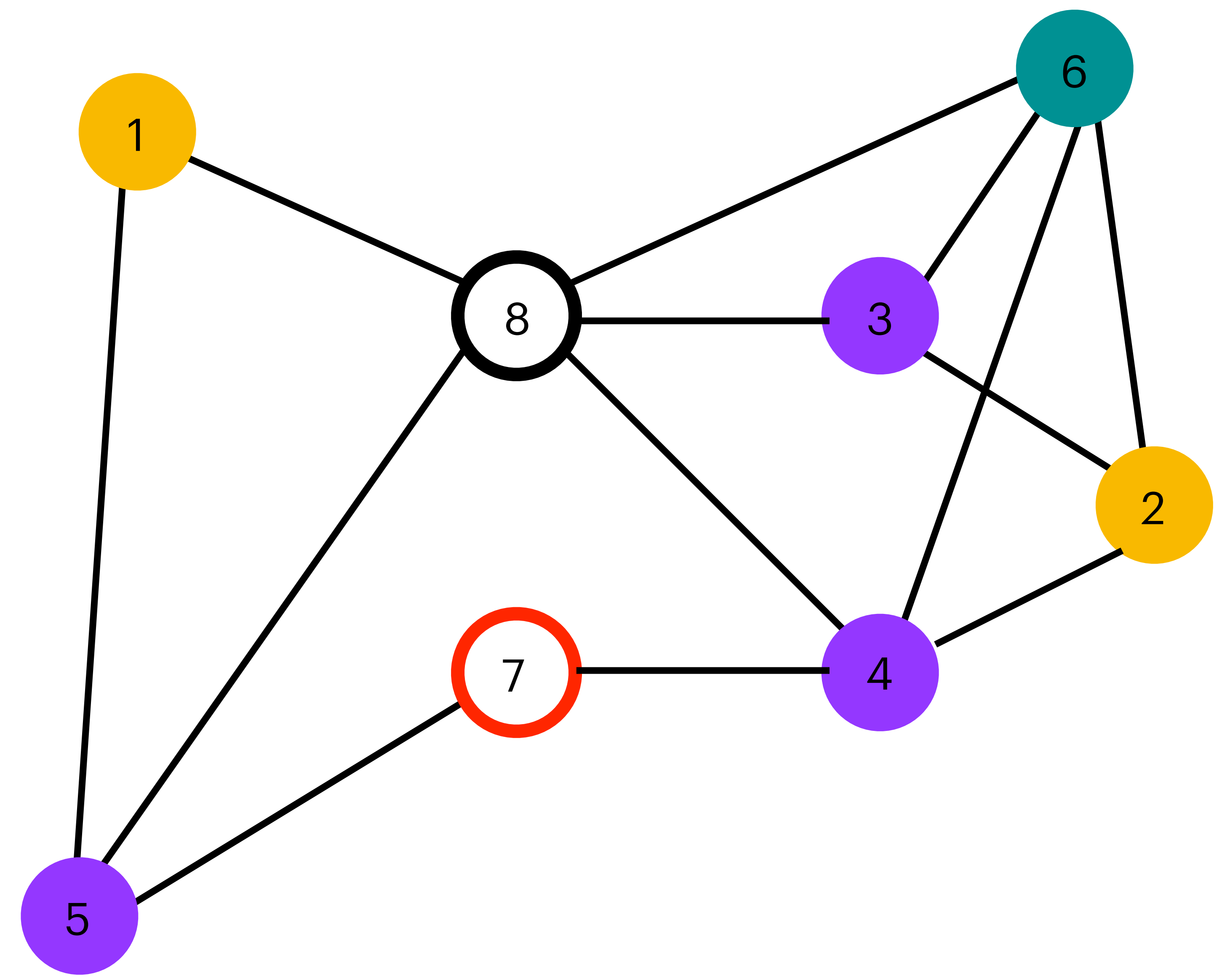
k = 1

k = 2

k = 3

considering ks

✅



C :

i	1	2	3	4	5	6	7	8
$c[v_i]$	1	1	2	2	2	3		

Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :

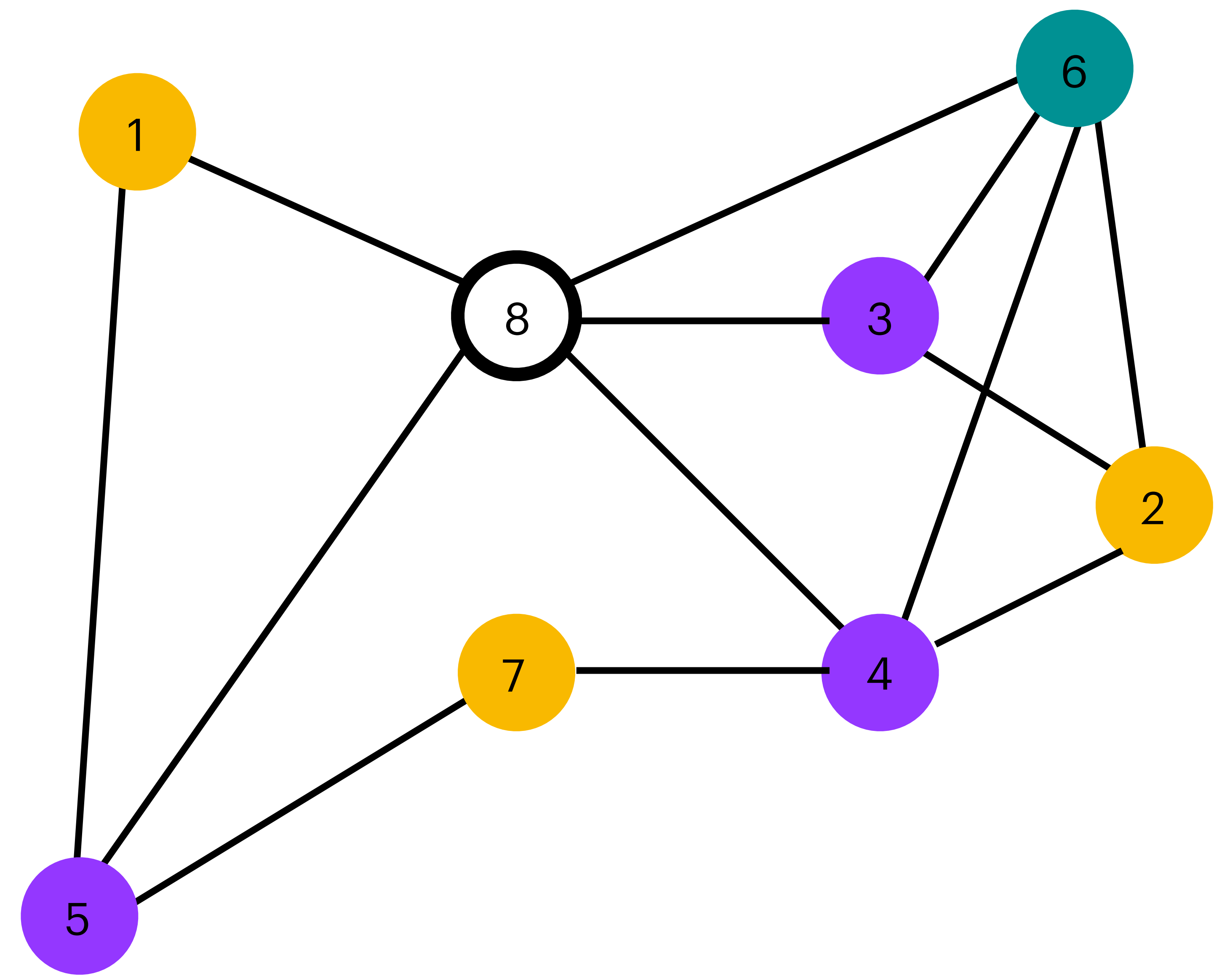
k = 1

k = 2

k = 3

considering ks

✅



C :

i	1	2	3	4	5	6	7	8
$c[v_i]$	1	1	2	2	2	3	1	

Coloring

Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 to n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :

k = 1

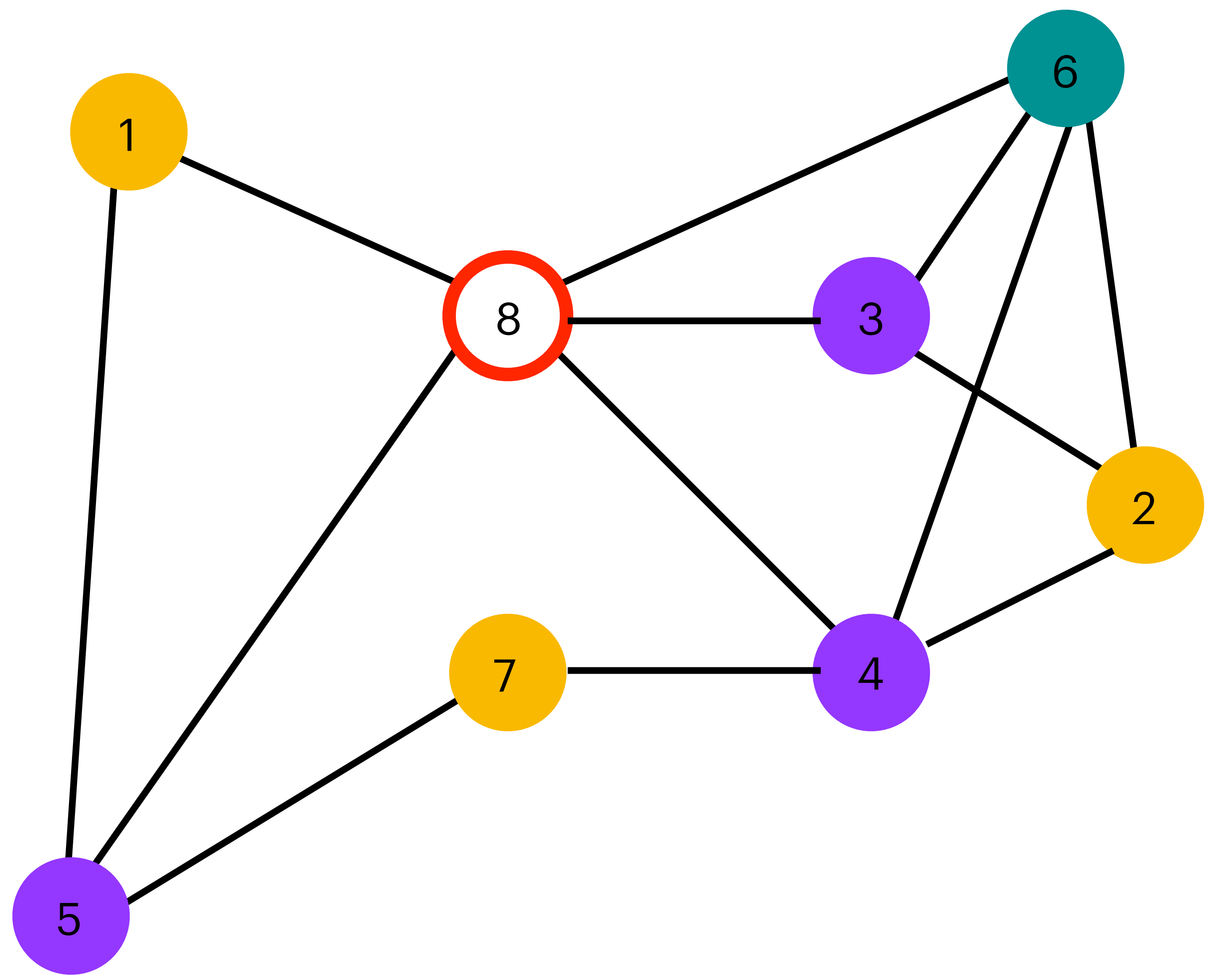
k = 2

k = 3

k = 4

considering ks

✅



C :

i	1	2	3	4	5	6	7	8
c[vᵢ]	1	1	2	2	2	3	1	

Coloring
Greedy Algorithm

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

k indexing for colors :

k = 1

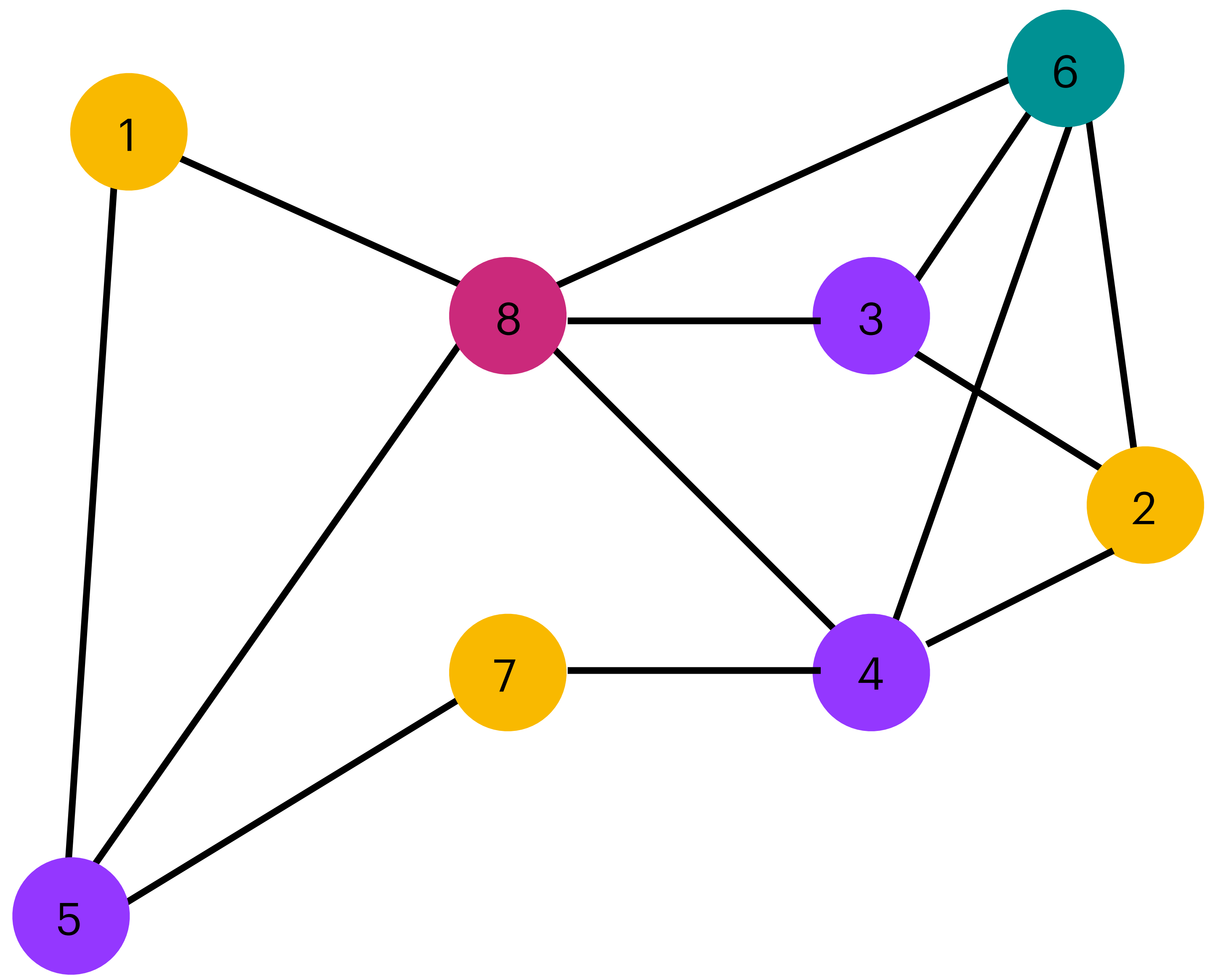
k = 2

k = 3

k = 4

considering ks

✅



C :

i	1	2	3	4	5	6	7	8
c[vᵢ]	1	1	2	2	2	3	1	4

done

Coloring

Greedy Algorithm - Observations

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

- For all orders of vertices $V = \{v_1, \ldots, v_n\}$
  - the Greedy Algorithm needs $\Delta(G)$ +1 colors
- There exists an order of vertices $V = \{v_1, \ldots, v_n\}$ for which
  - the Greedy Algorithm needs $\chi(G)$ colors

$\Delta(G) :=$ maximum degree in G

Coloring
Greedy Algorithm - Observations

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

- For all orders of vertices $V = \{v_1, \ldots, v_n\}$
  - the Greedy Algorithm needs $\Delta(G)$ +1 colors
- There exists an order of vertices $V = \{v_1, \ldots, v_n\}$ for which
  - the Greedy Algorithm needs $\chi(G)$ colors

$\chi(G) \leq k \iff G$ is k-partite

$\Delta(G) :=$ maximum degree in G

Coloring

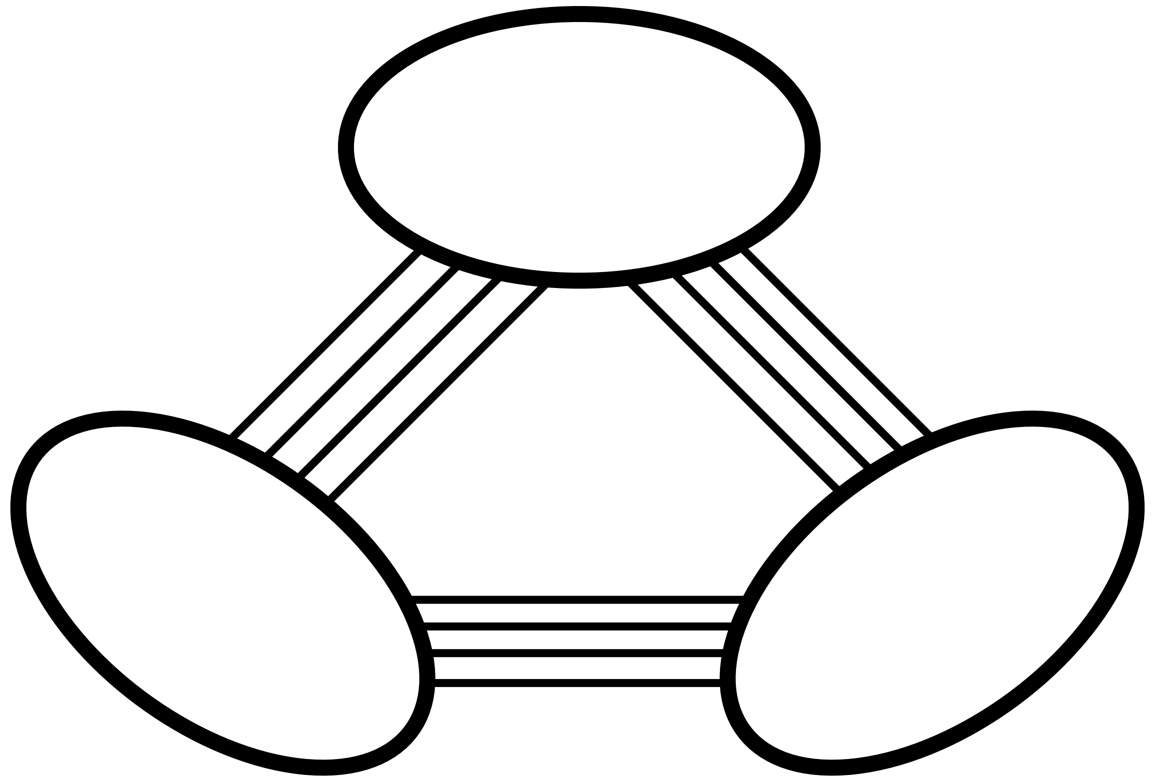
Greedy Algorithm - Observations

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

- For all orders of vertices $V = \{v_1, \ldots, v_n\}$
  - the Greedy Algorithm needs $\Delta(G)$ +1 colors
- There exists an order of vertices $V = \{v_1, \ldots, v_n\}$ for which
  - the Greedy Algorithm needs $\chi(G)$ colors

$\chi(G) \leq k \iff G$ is k-partite



$\Delta(G) :=$ maximum degree in G

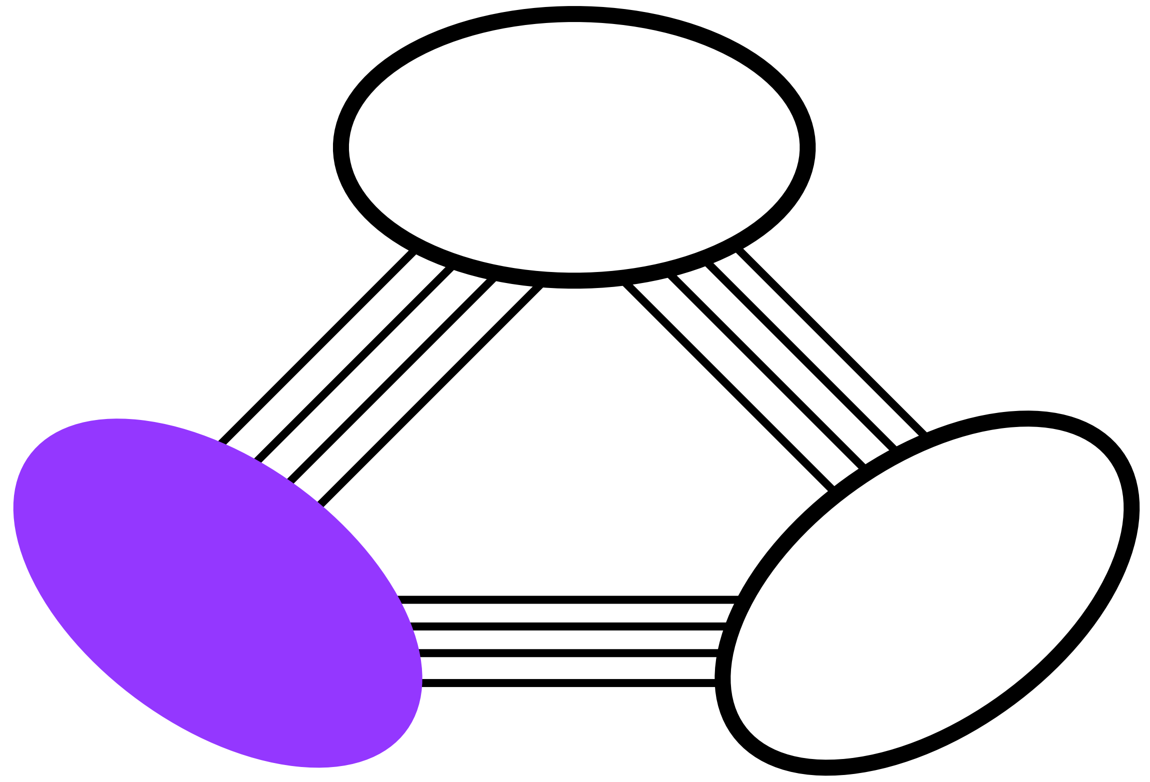
Coloring
Greedy Algorithm - Observations

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

- For all orders of vertices $V = \{v_1, \ldots, v_n\}$
  - the Greedy Algorithm needs $\Delta(G)$ +1 colors
- There exists an order of vertices $V = \{v_1, \ldots, v_n\}$ for which
  - the Greedy Algorithm needs $\chi(G)$ colors

$\chi(G) \leq k \iff G$ is k-partite



$\Delta(G) :=$ maximum degree in G

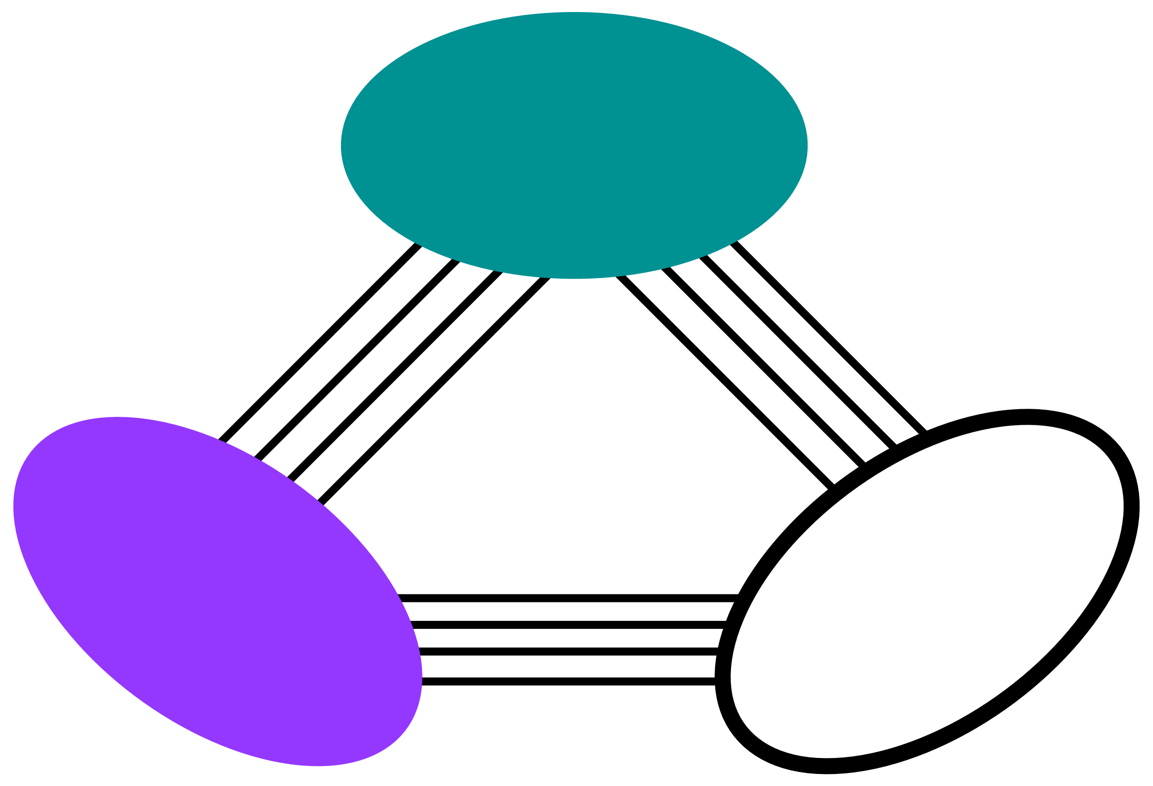
Coloring
Greedy Algorithm - Observations

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

- For all orders of vertices $V = \{v_1, \ldots, v_n\}$
  - the Greedy Algorithm needs $\Delta(G)$ +1 colors
- There exists an order of vertices $V = \{v_1, \ldots, v_n\}$ for which
  - the Greedy Algorithm needs $\chi(G)$ colors

$\chi(G) \leq k \iff G$ is k-partite



$\Delta(G) :=$ maximum degree in G

Coloring

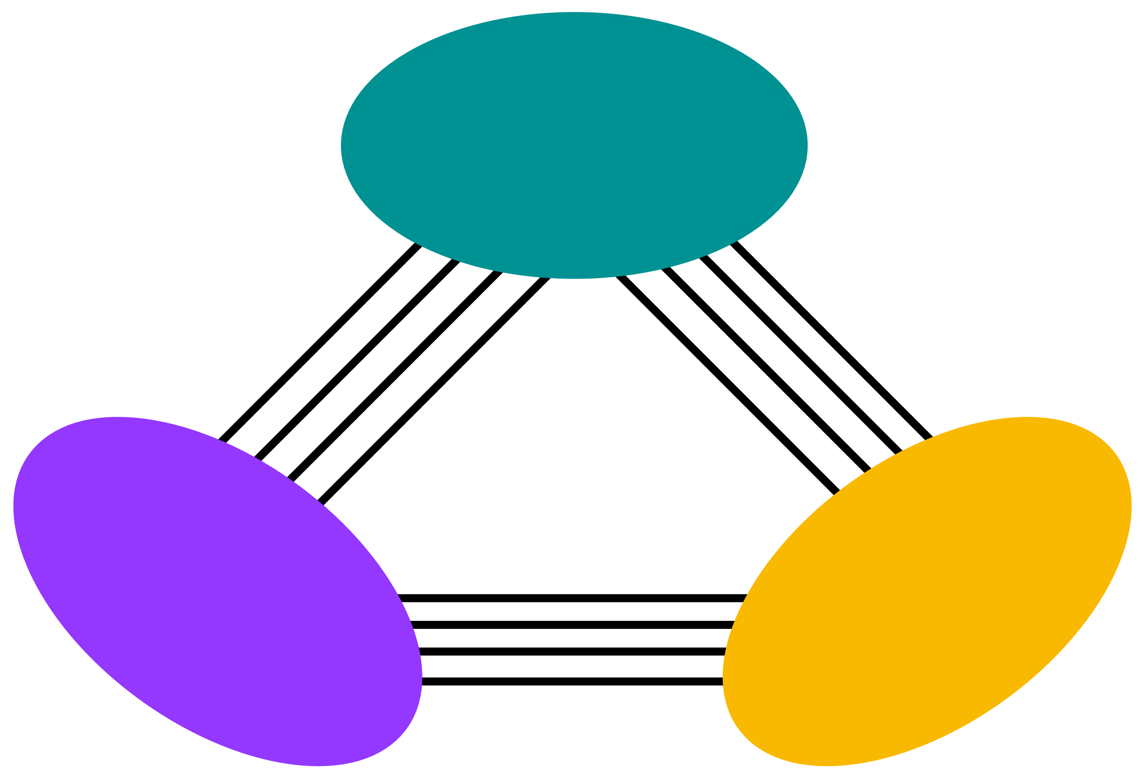
Greedy Algorithm - Observations

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

- For all orders of vertices $V = \{v_1, \ldots, v_n\}$
  - the Greedy Algorithm needs $\Delta(G)$ +1 colors
- There exists an order of vertices $V = \{v_1, \ldots, v_n\}$ for which
  - the Greedy Algorithm needs $\chi(G)$ colors

$\chi(G) \leq k \iff G$ is k-partite



$\Delta(G) :=$ maximum degree in G

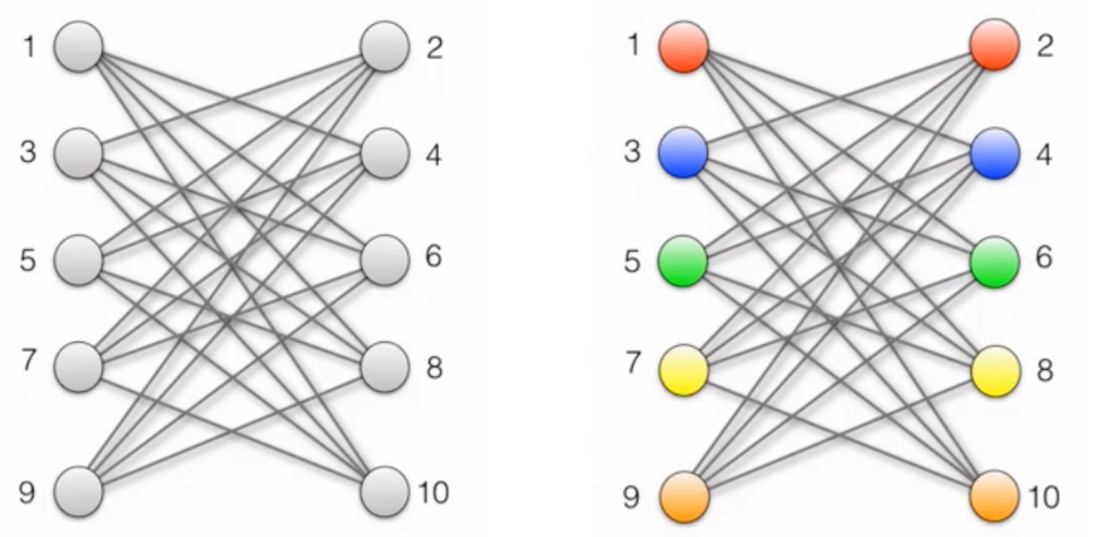
Coloring

Greedy Algorithm - Observations

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

- For all orders of vertices $V = \{v_1, \ldots, v_n\}$
  - the Greedy Algorithm needs $\Delta(G)$ +1 colors
- There exists an order of vertices $V = \{v_1, \ldots, v_n\}$ for which
  - the Greedy Algorithm needs $\chi(G)$ colors
- There exists bipartite Graphs and order of vertices $V = \{v_1, \ldots, v_n\}$ for which
  - the Greedy Algorithm needs $|V| / 2$ colors



Coloring

Greedy Algorithm - Observations

- For the chosen order of vertices $V = \{v_1, \ldots, v_n\}$ s.t.
  - $|N(v_i) \cap \{v_1, \ldots, v_{i-1}\}| \leq k \qquad \forall\, 2 \leq i \leq n$
  - the Greedy Algorithm needs at most k +1 colors

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**

k gets increased here $\longrightarrow$
- $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

Heuristic Meaning

🔎 Heuristic in Algorithms:

A **heuristic** in algorithms refers to a **problem-solving technique** that:

- Uses a **practical approach** to find a solution quickly.
- Sacrifices **accuracy** or **completeness** for **speed**.
- Aims for a **good enough** solution rather than the **perfect** one.
- Often relies on **experience, intuition, or patterns** rather than formal mathematical proofs.

Coloring
Heuristic + Greedy Algorithm

- For the chosen order of vertices $V = \{v_1, \ldots, v_n\}$ s.t.
  - $|N(v_i) \cap \{v_1, \ldots, v_{i-1}\}| \leq k \quad \forall 2 \leq i \leq n$
  - the Greedy Algorithm needs at most k +1 colors

- Pick the order of the vertices using the heuristic : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

- Heuristic :
  - $v_n :=$ Vertex with the smallest degree. Delete v_n
  - $v_{n-1} :=$ Vertex with the smallest degree in the remaining G. Delete v_{n-1}
  - Iterate

Coloring

Heuristic + Greedy Algorithm - Observations

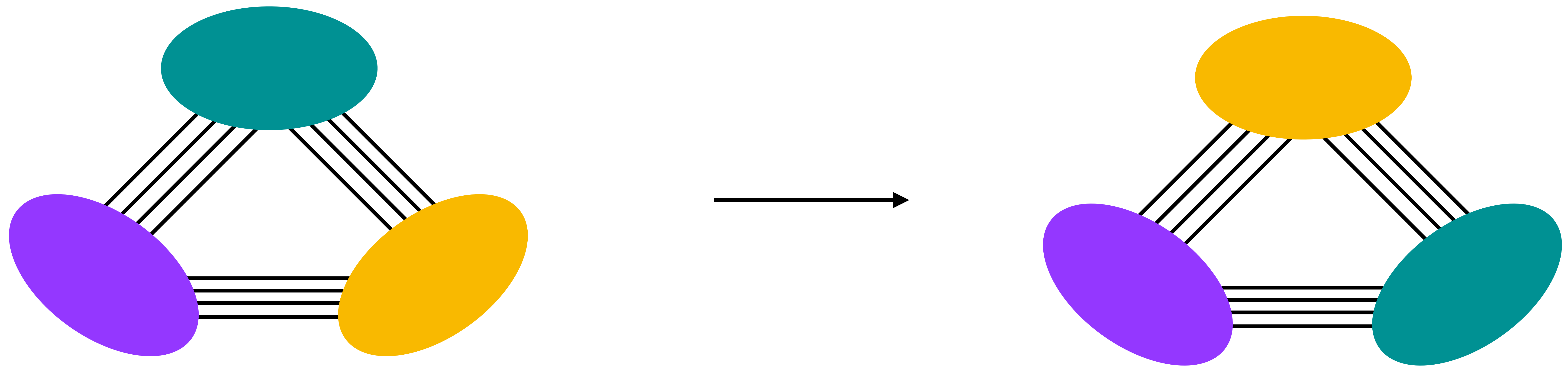
- If in every subgraph of G , there exists a vertex with degree $\leq$ k
  - heuristic provides an order $v_1, ... v_n$ s.t. the Greedy Algorithm needs k+1 colors

- For trees heuristic+greedy finds a coloring with 2 colors
- For planar graphs heuristic+greedy finds a coloring with $\leq$ 6 colors
- If G is connected and there exists $v \in G$ with deg(v) < $\Delta(G)$ → it doesn't hold only when the graph is regular: $\forall v \in V$ (deg(v) = $\Delta(G)$

  heuristic (or bfs/dfs)+ greedy finds a coloring with $\leq \Delta(G)$ colors
- If the G is 3-colorable, then one can color it in O(|V| + |E|) time with O($\sqrt{|V|}$) colors

Coloring

Swapping Color Classes Trick



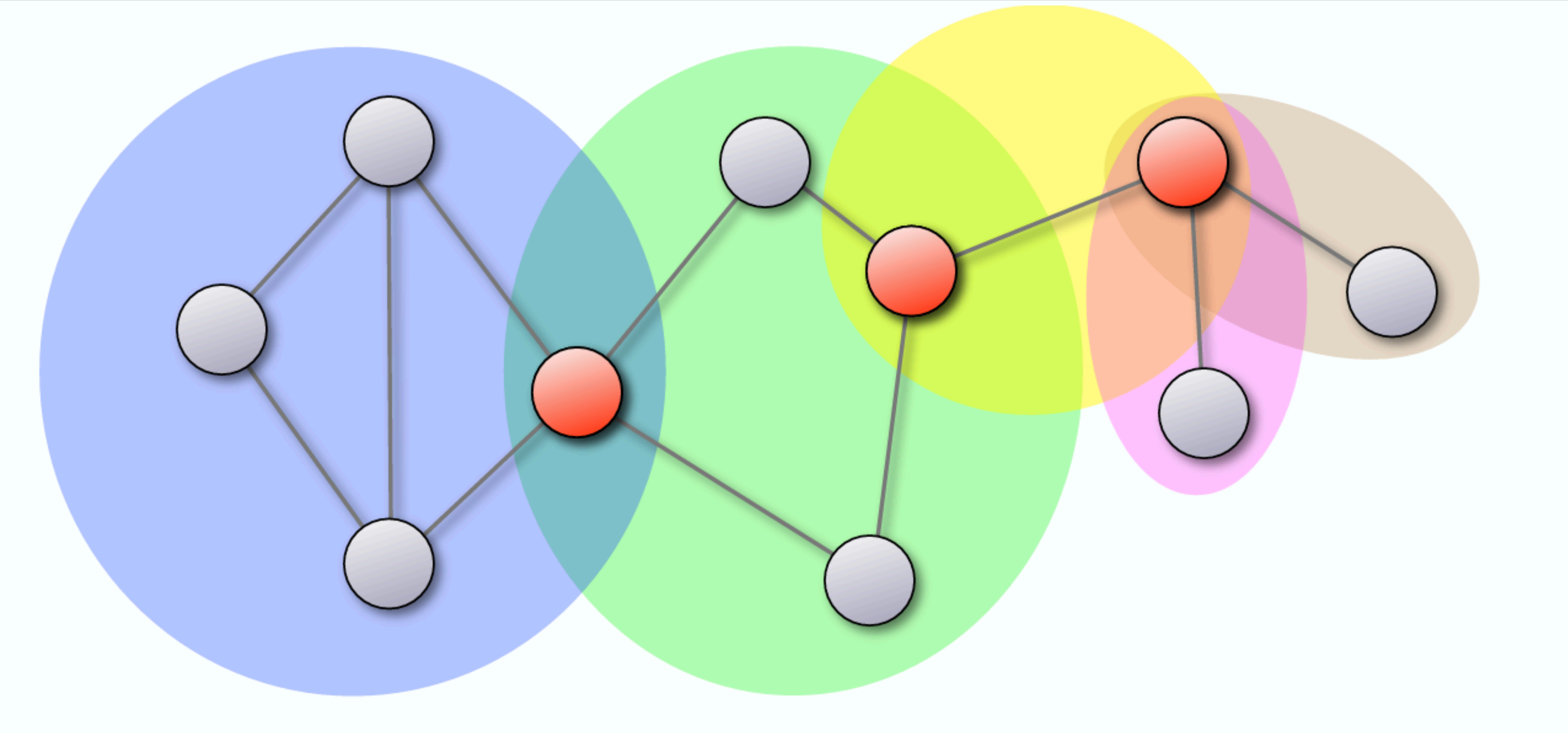
Articulation Points and Bridges
Definition

- Let $G = (V, E)$ be a graph.

  The equivalence relation $\sim$ on E is defined as :

$$e \sim f \quad := \begin{cases} e = f & \text{or} \\ e \text{ and } f \text{ are on a common cycle} \end{cases}$$

- The equivalence classes are named as Blocks



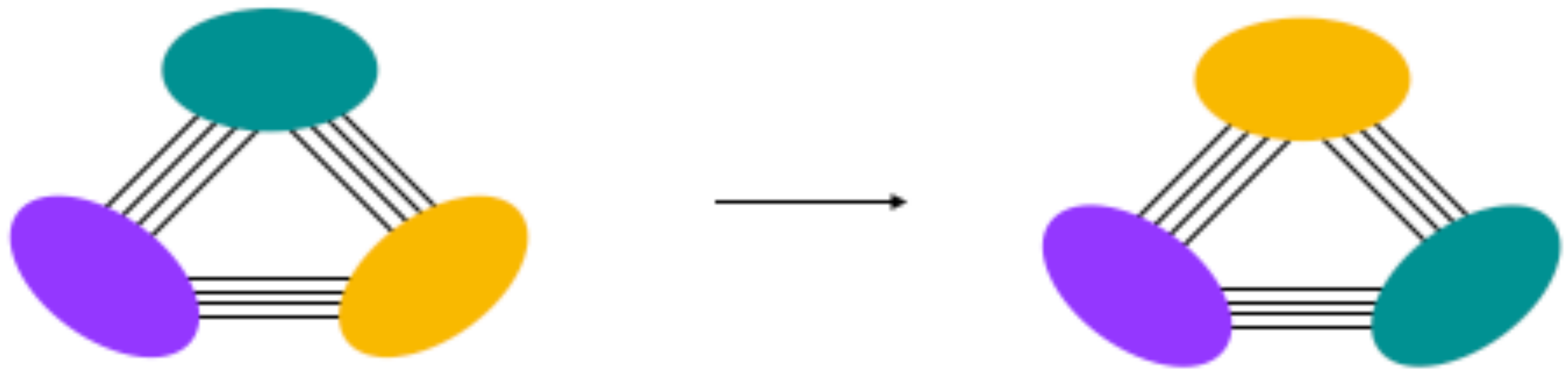
Lemma :

2 blocks always intersect at an articulation point.

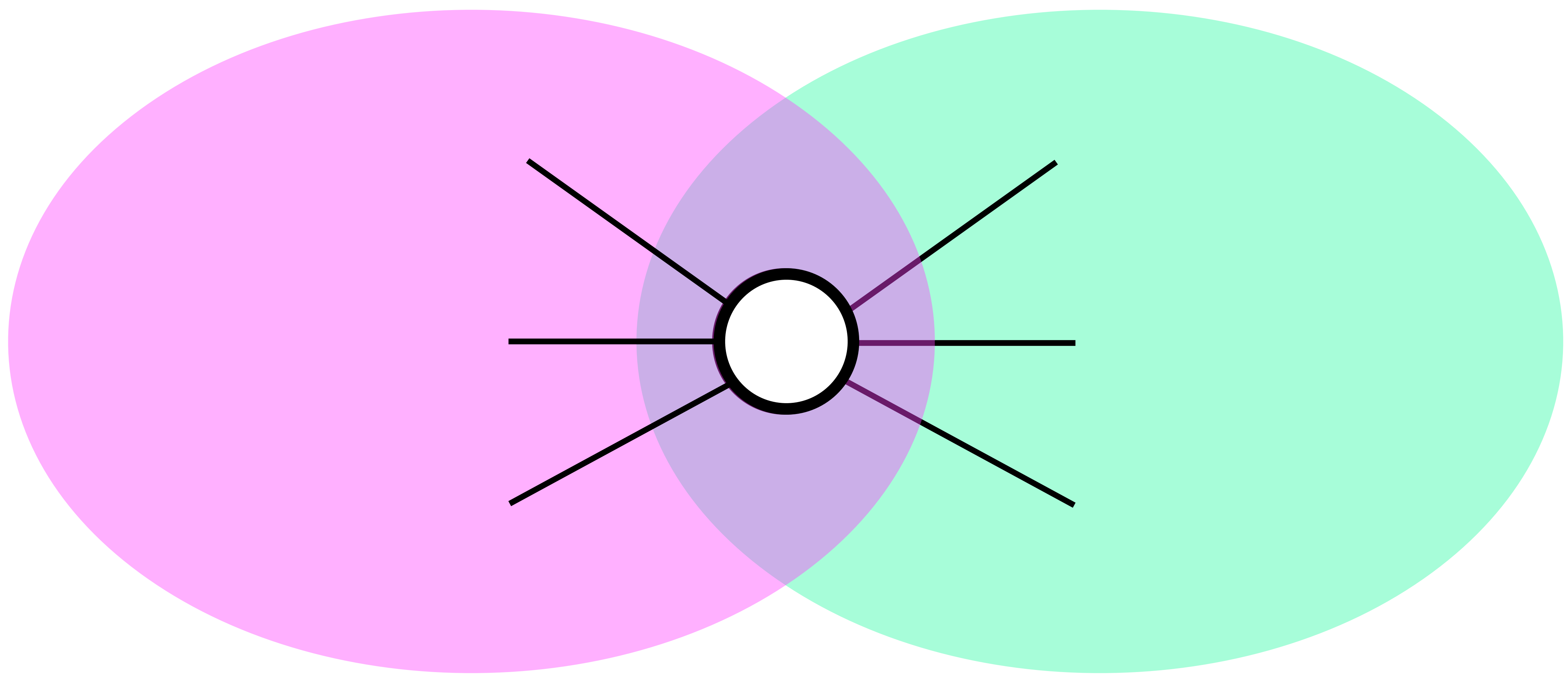
Articulation point is the critical point that holds blocks together. If a graph has an articulation point, it serves as the **only connection** between two or more blocks.

Coloring

Swapping Color Classes Trick

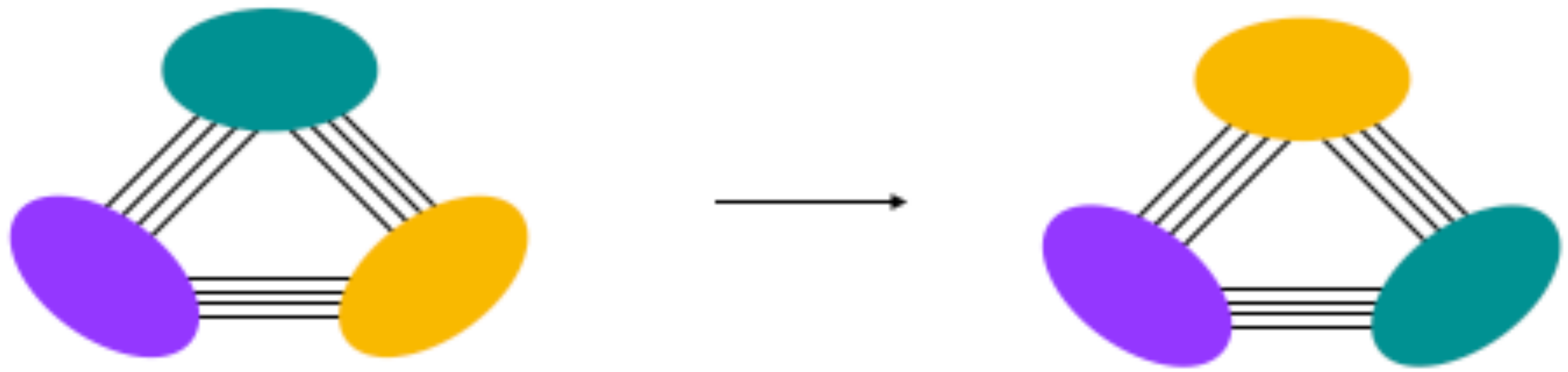


- For all Block-Graphs , if every block can be colored with k colors
  - G can be colored with k colors

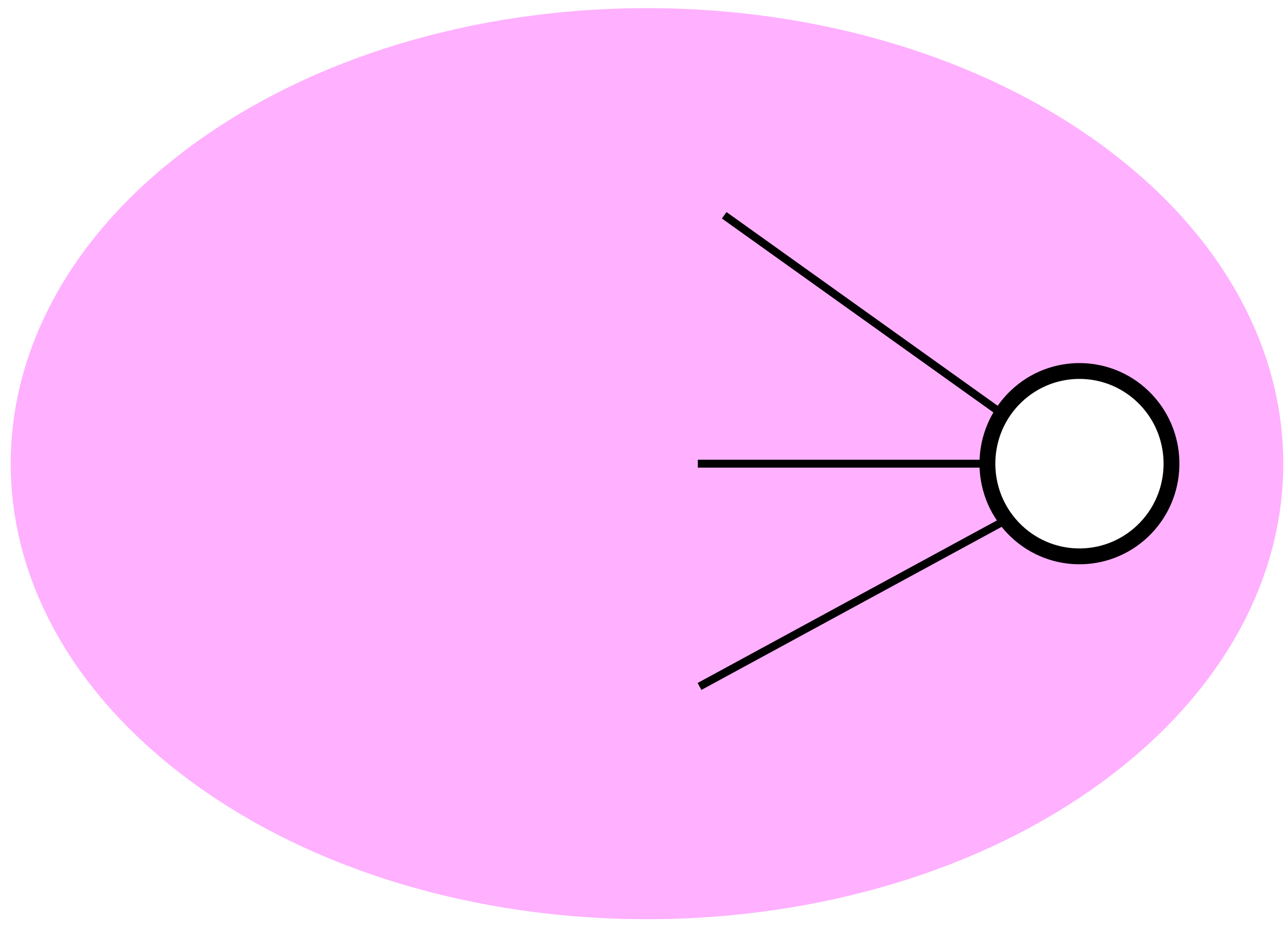


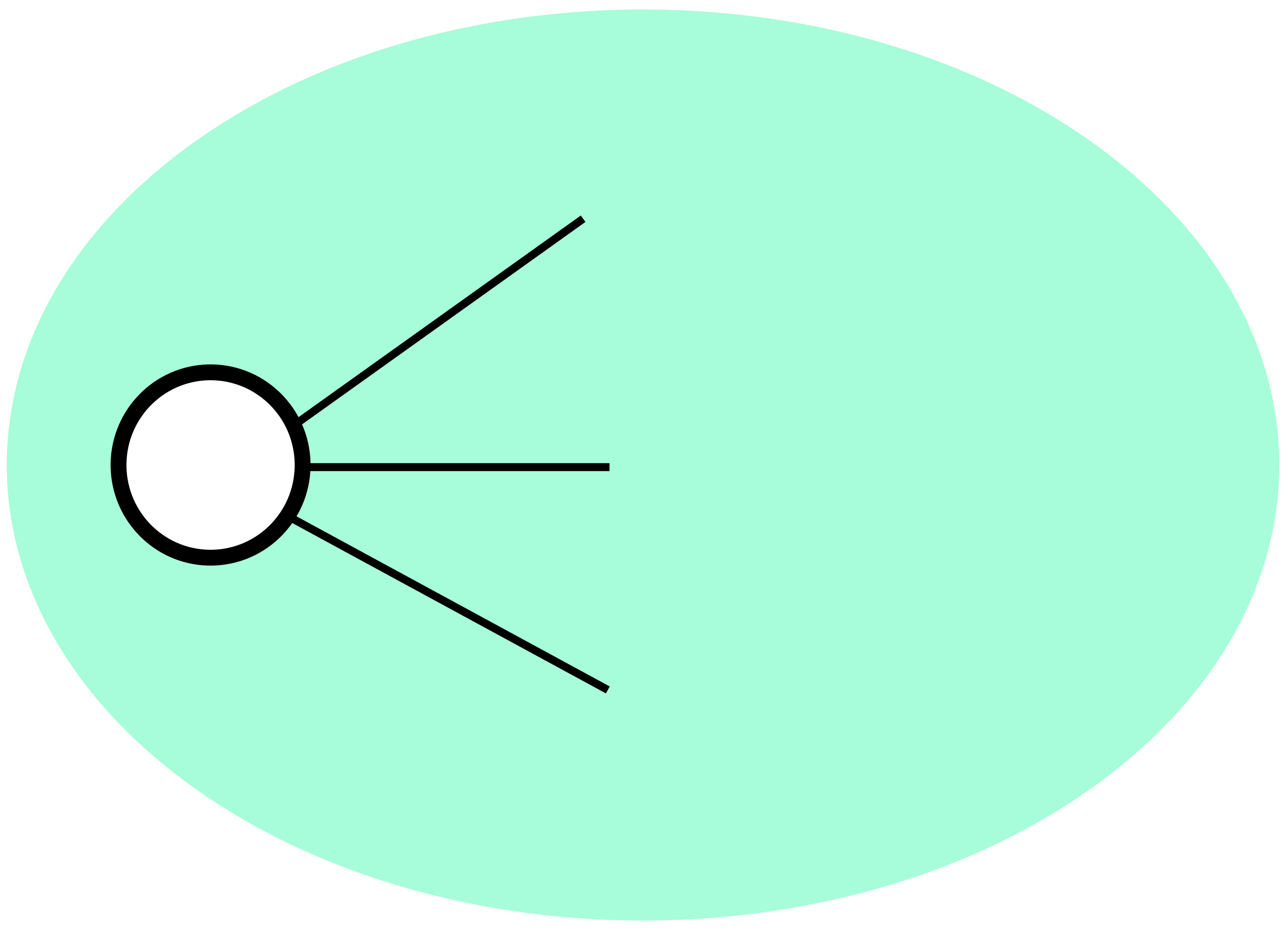
Coloring

Swapping Color Classes Trick



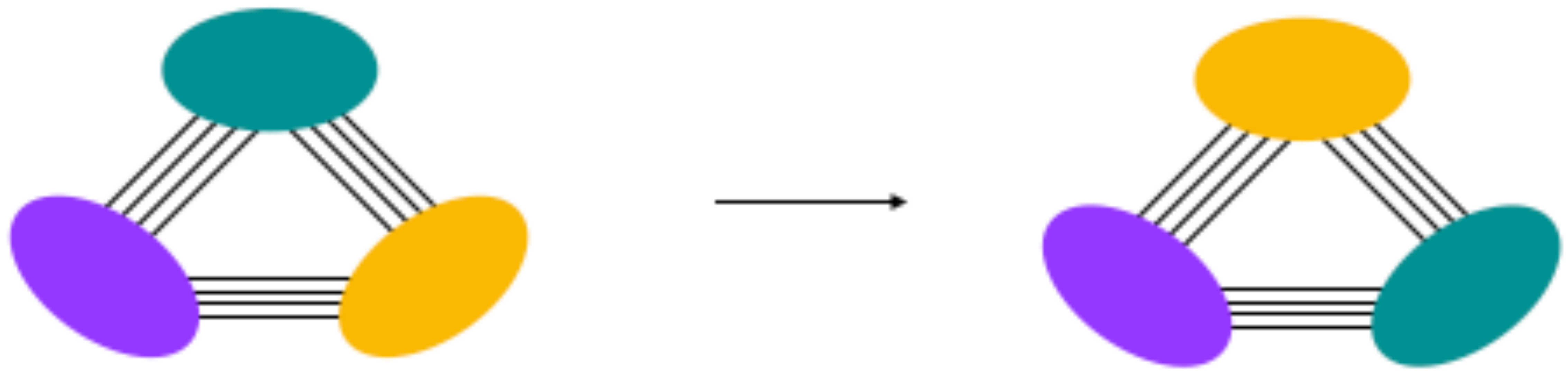
- For all Block-Graphs , if every block can be colored with k colors
    - G can be colored with k colors



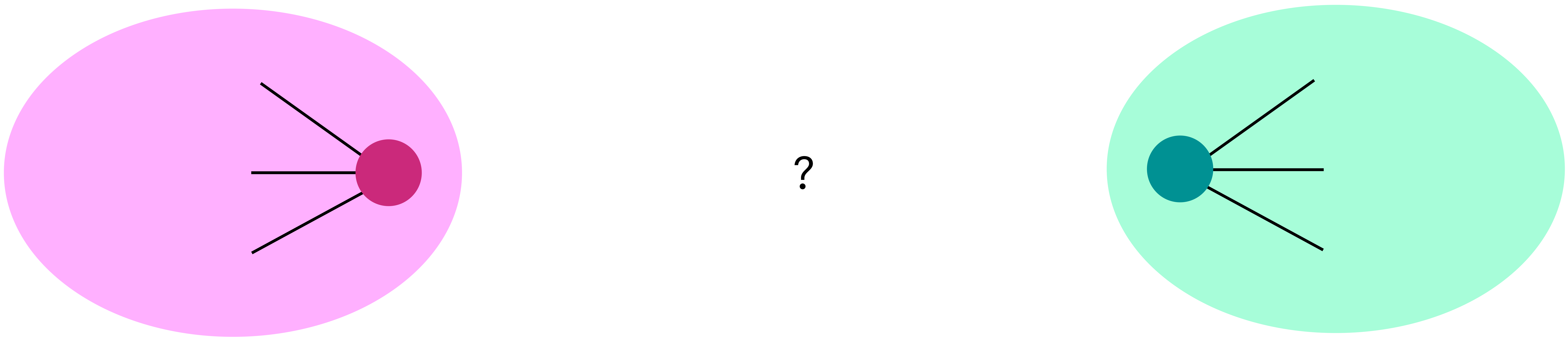


Coloring

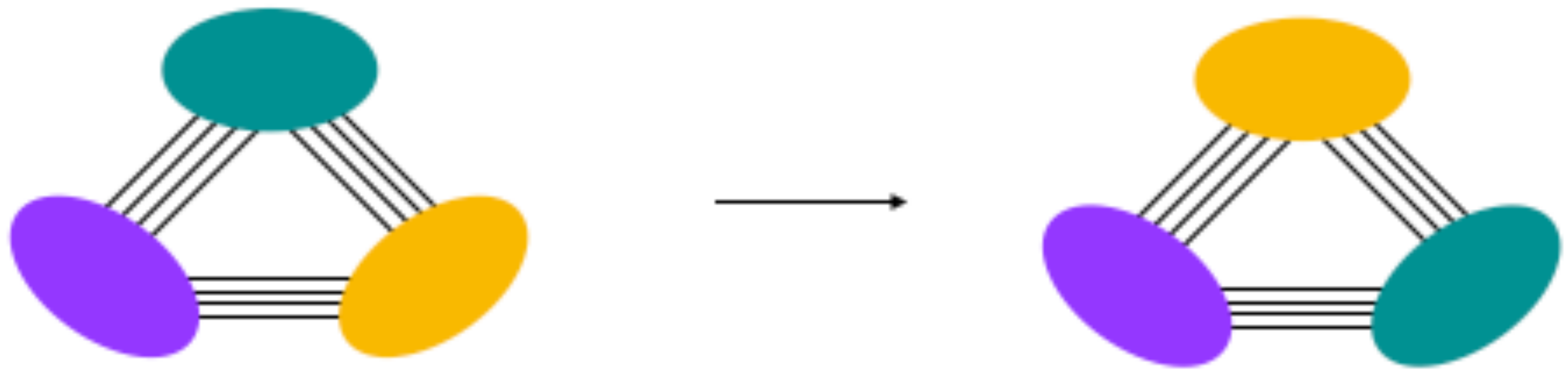
Swapping Color Classes Trick



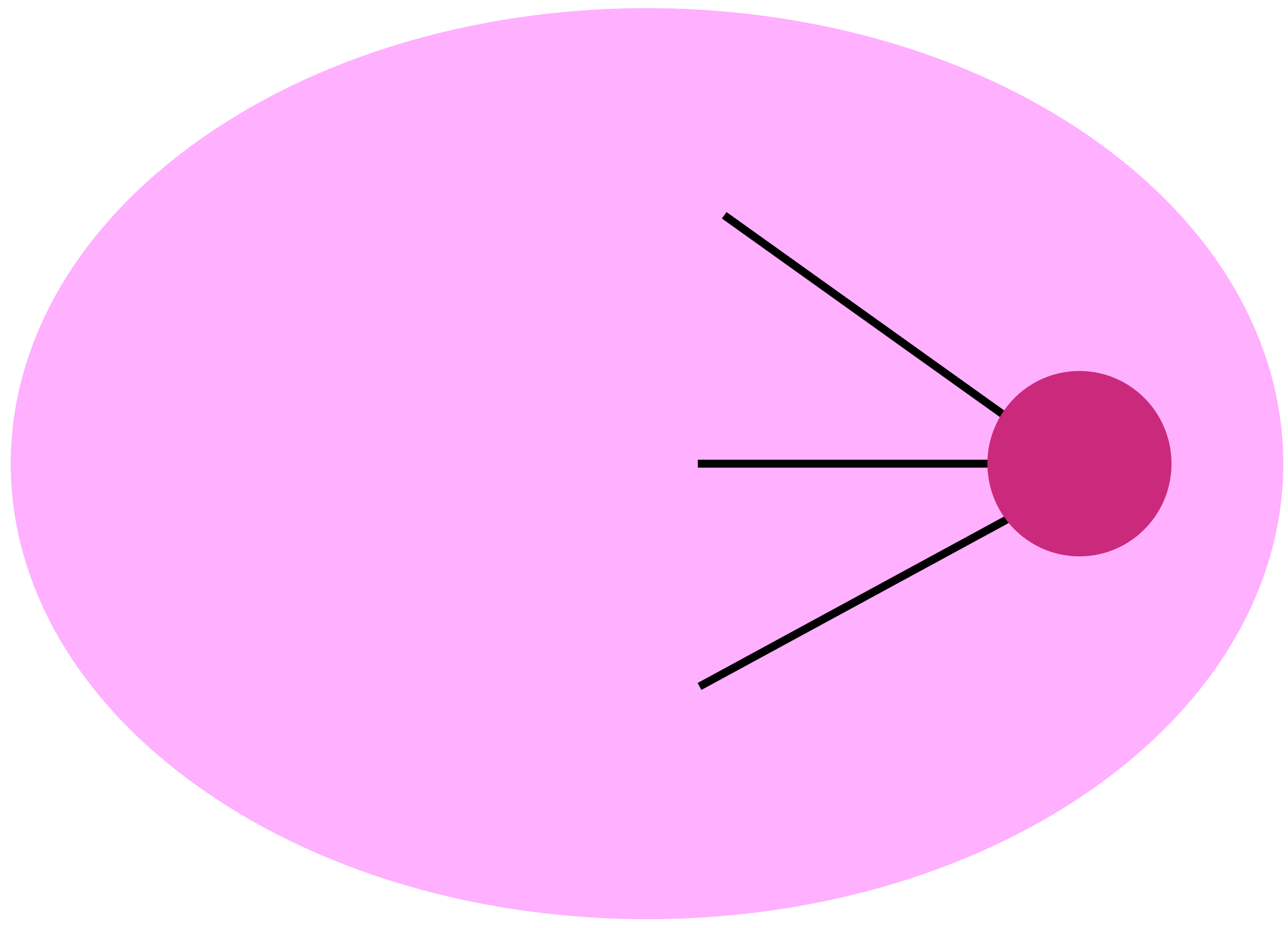
- For all Block-Graphs , if every block can be colored with k colors
    - G can be colored with k colors

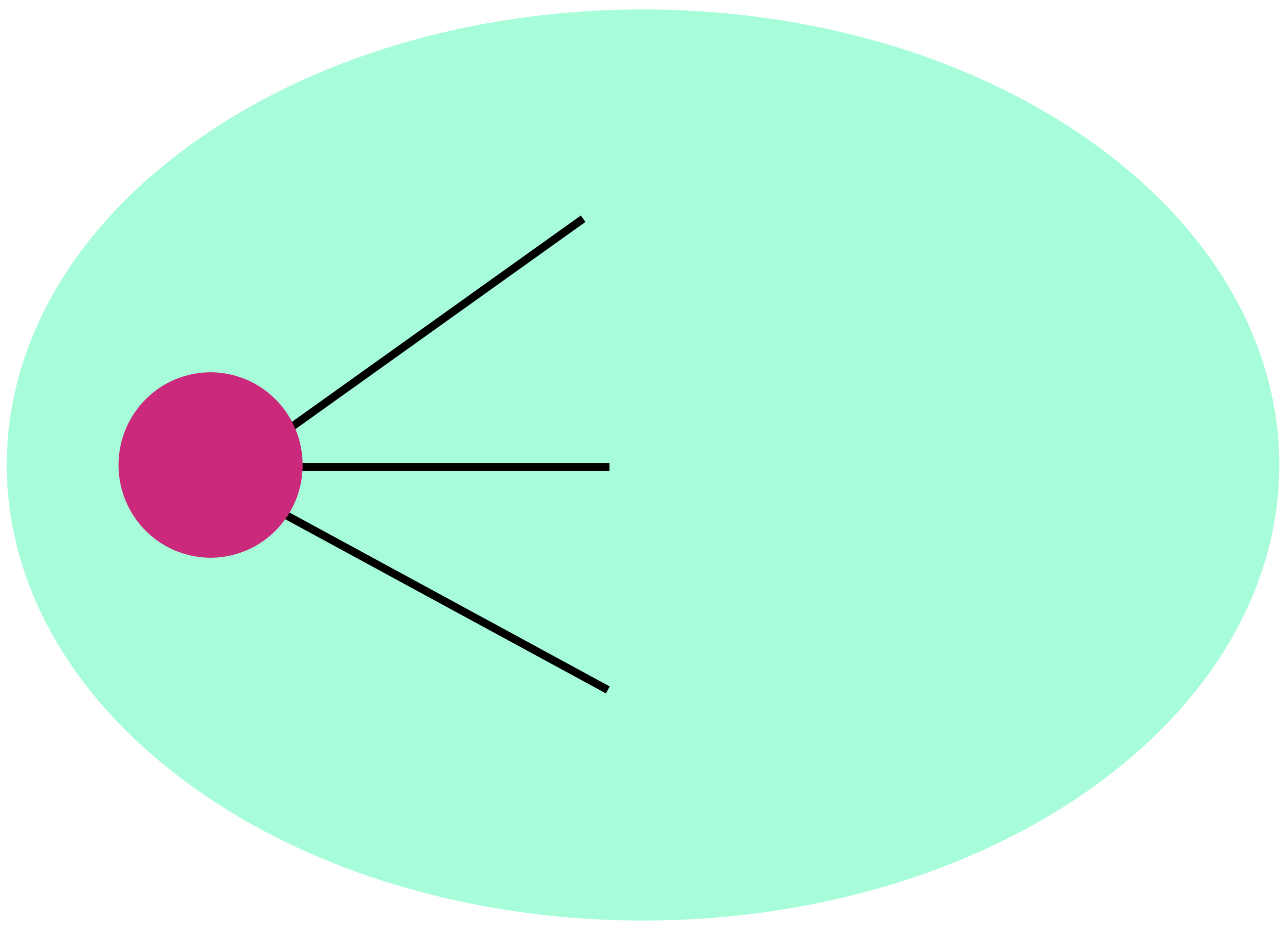


Coloring
Swapping Color Classes Trick



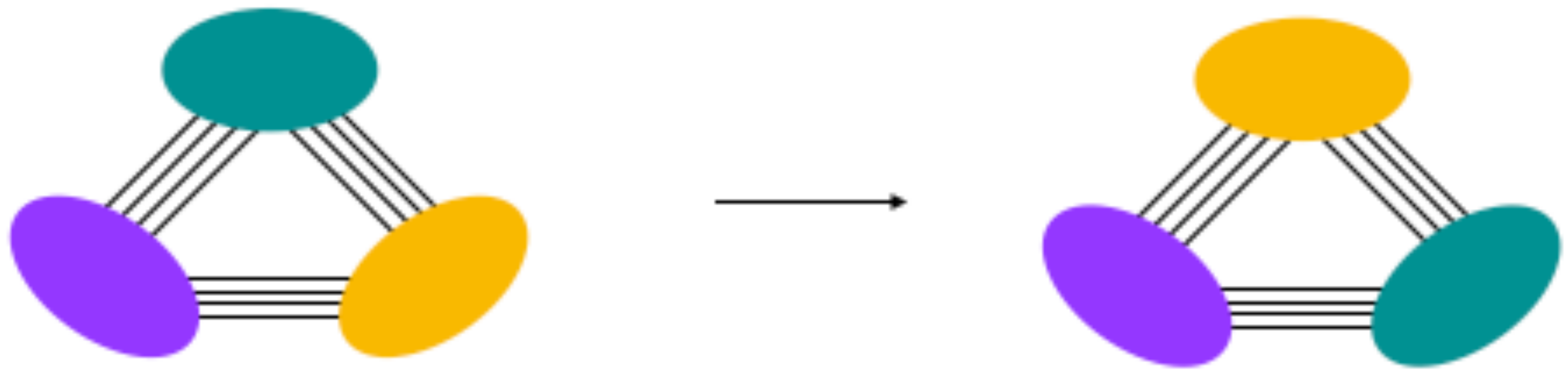
- For all Block-Graphs , if every block can be colored with k colors
    - G can be colored with k colors



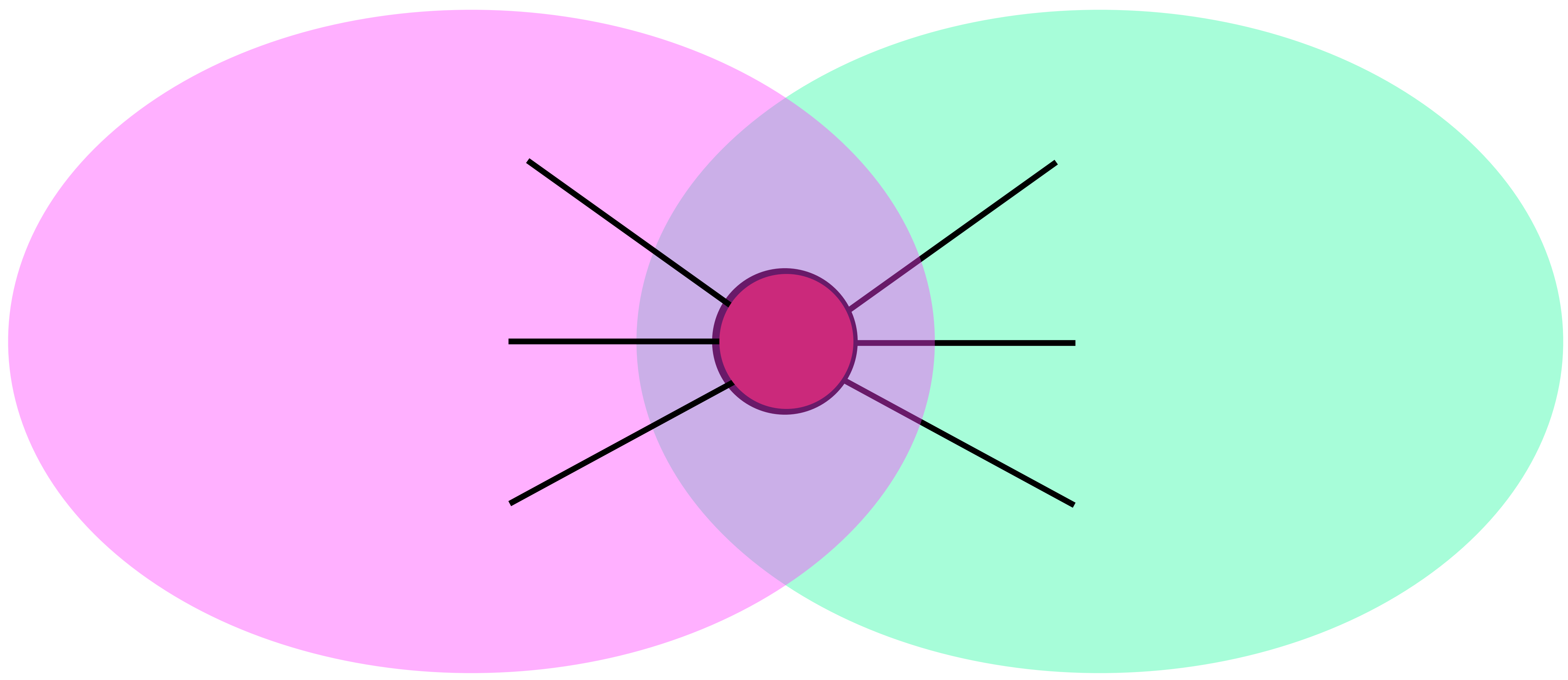


Coloring

Swapping Color Classes Trick



- For all Block-Graphs , if every block can be colored with k colors
  - G can be colored with k colors



Coloring

Brook's Theorem

- Pick an arbitrary order of the vertices : $V = \{v_1, \ldots, v_n\}$
- $c[v_1] \leftarrow 1$ color the first vertex with color 1
- **for** i = 2 **to** n **do**
  - $c[v_i] \leftarrow \min\{k \in \mathbb{N} \mid k \neq c[u]$ for all $u \in N(v_i) \cap \{v_1, \ldots, v_{i-1}\}\}$

    min color k s.t. it's not equal to the color of the neighbors of v_i that are already colored

- All Graphs
  - can be colored in O(|E|) time with $\Delta(G) + 1$ colors

Brook's Theorem

- $G \neq K_n$, $G \neq C_{2n+1}$, G connected
  - can be colored in O(|E|) time with $\Delta(G)$ colors

Minitest 2 - rest

Questions

Feedbacks , Recommendations

Nil Ozer