Maximum Edge Coloring

Prachi Goyal, Vikram Kamat and Neeldhara Misra

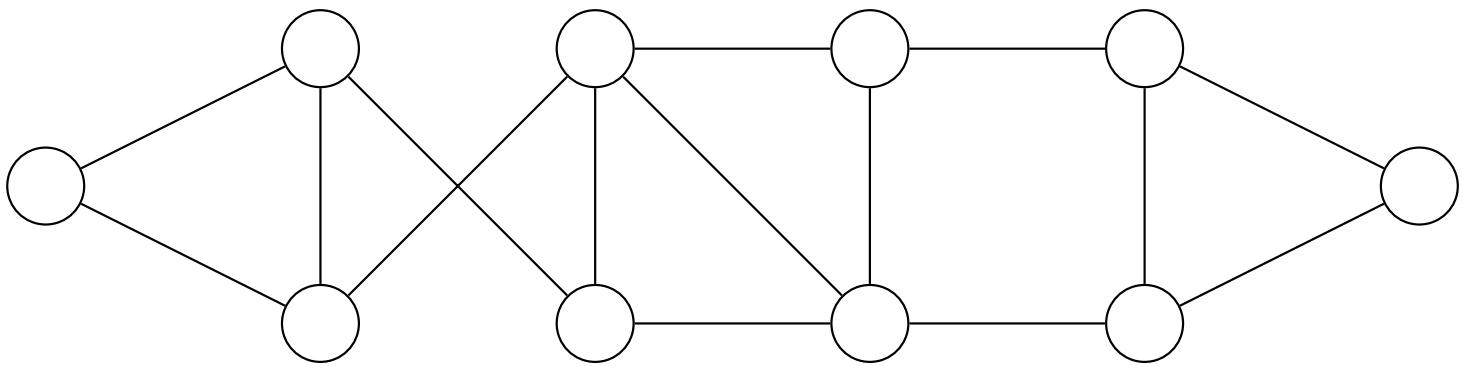
Department of Computer Science, Indian Institute of Science

Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex “sees” at most two colors.

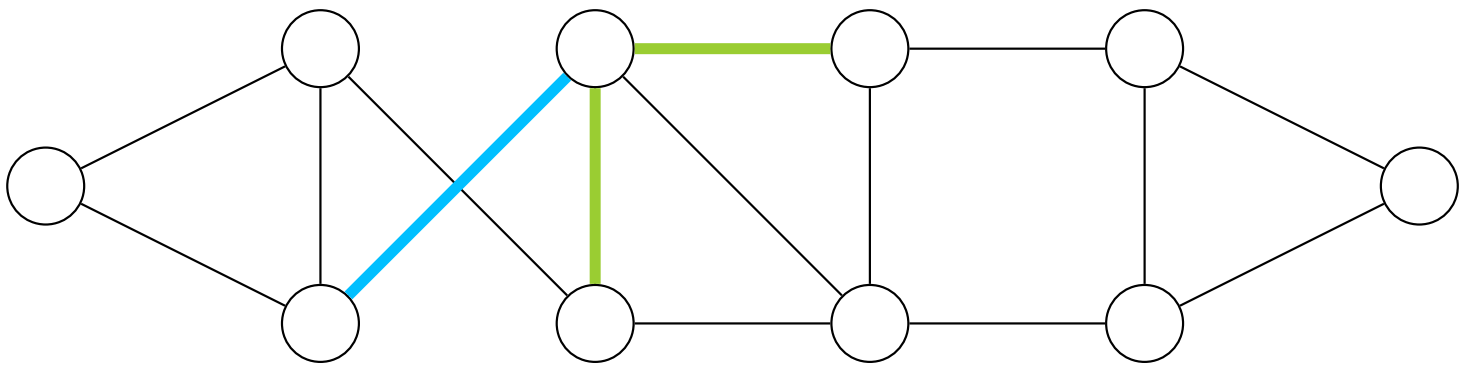
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



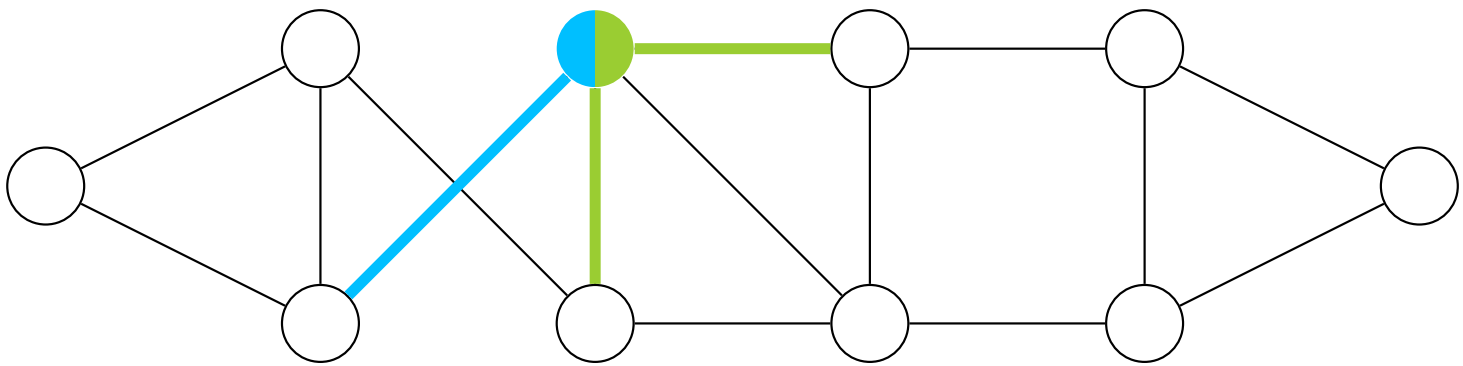
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



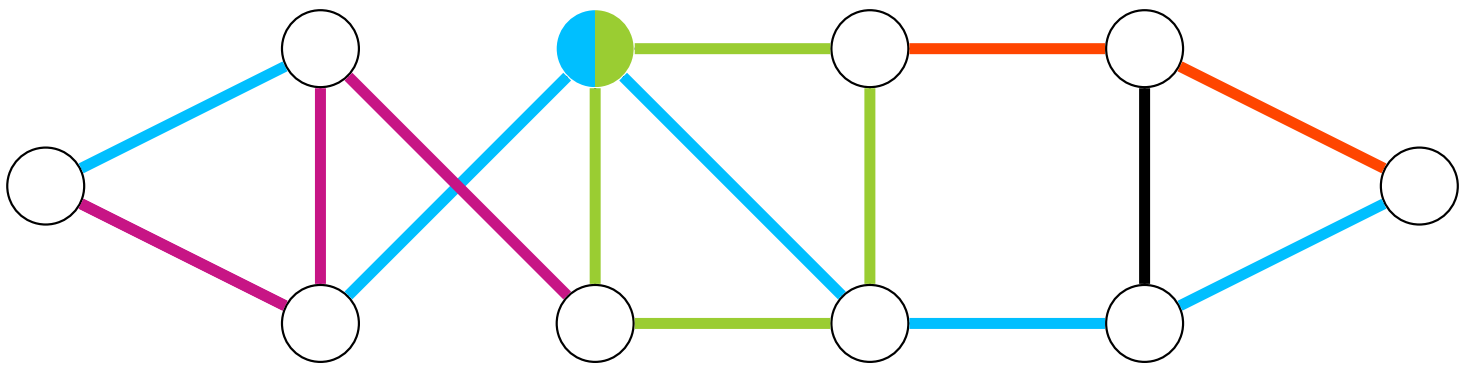
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



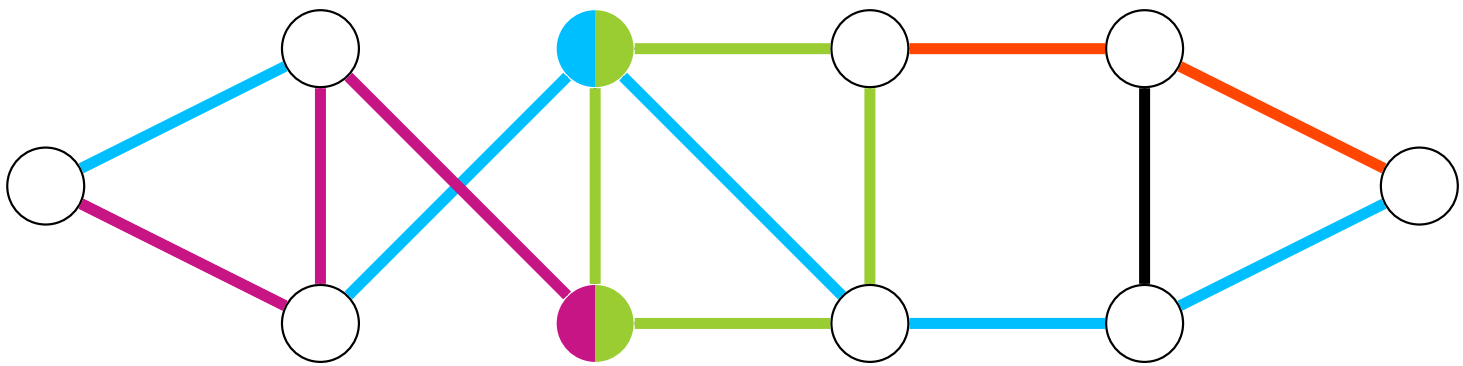
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



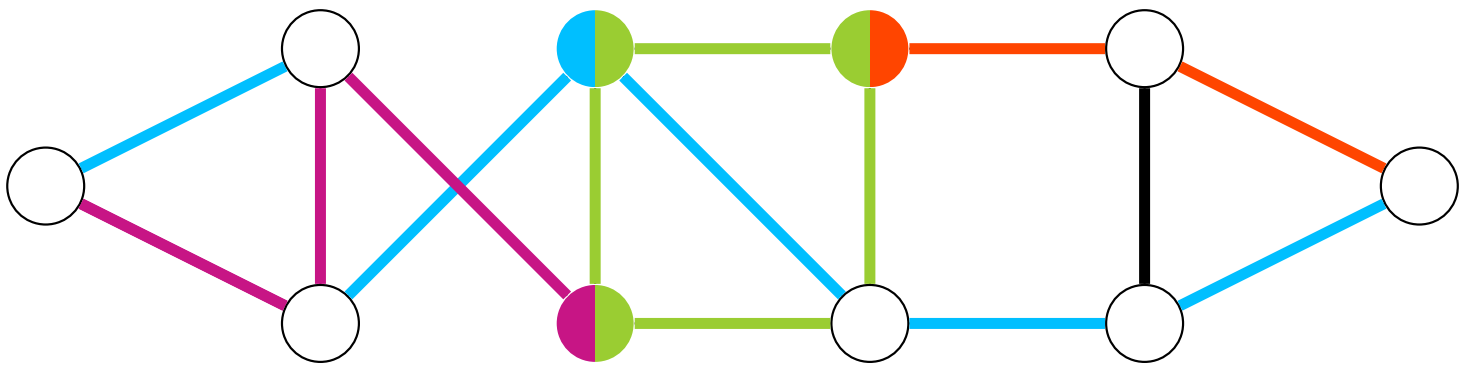
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



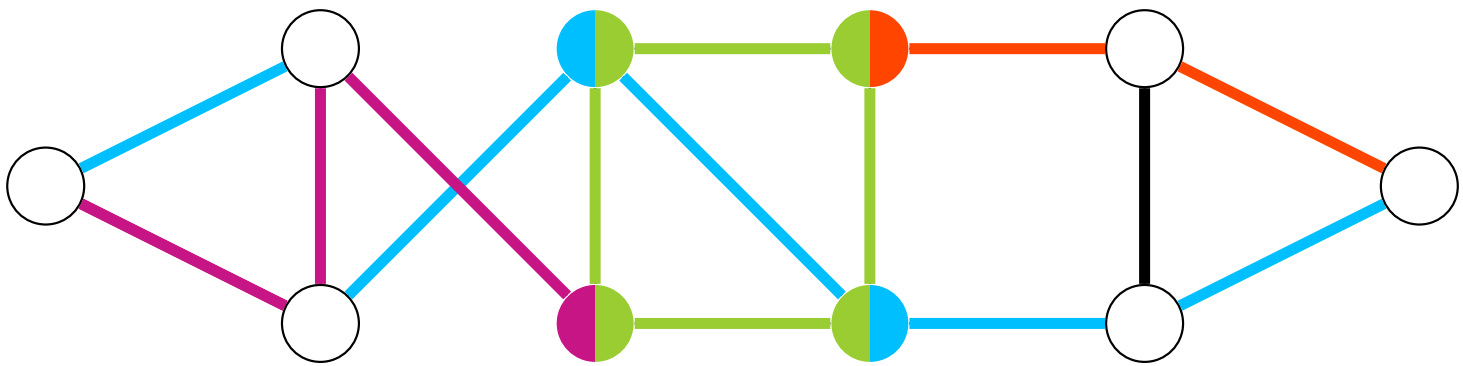
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



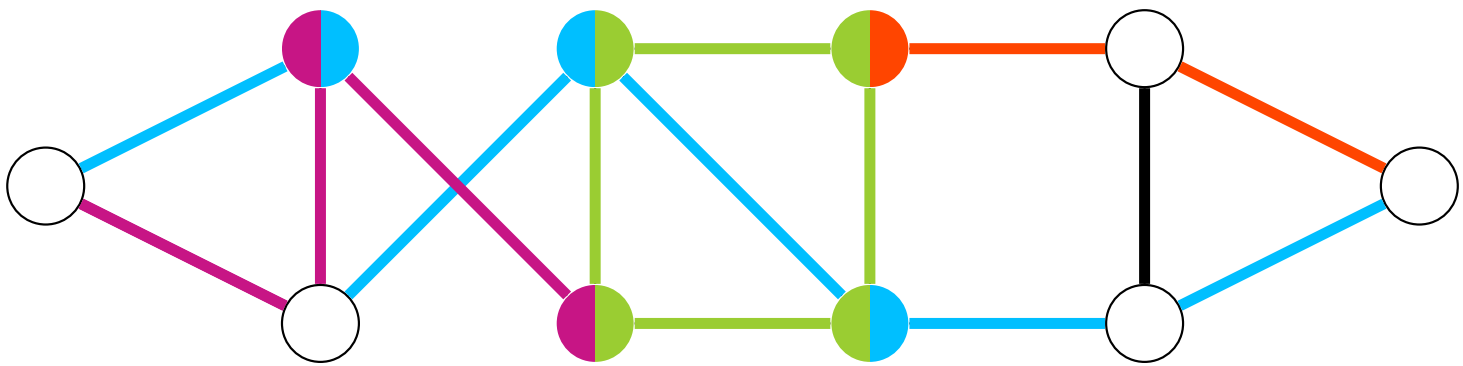
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



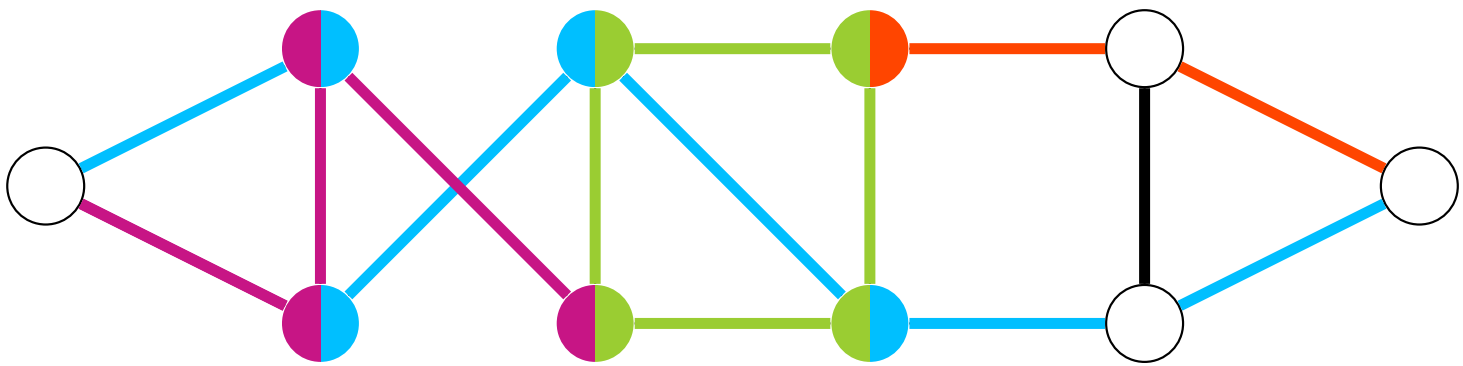
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



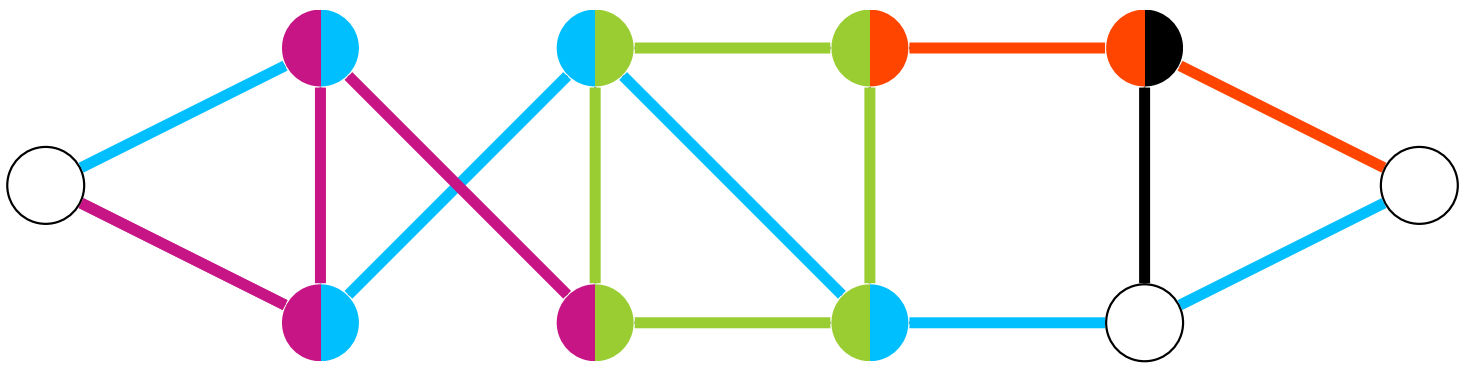
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



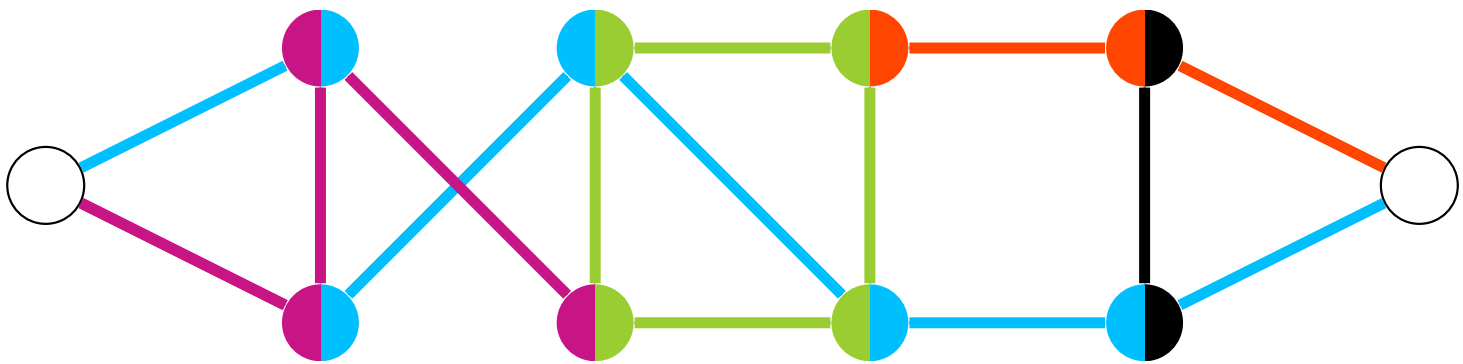
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



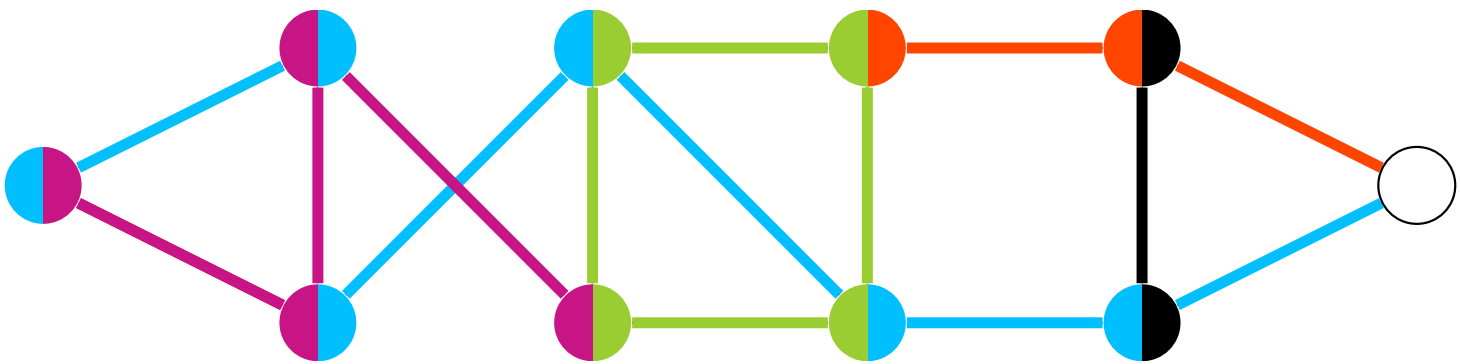
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



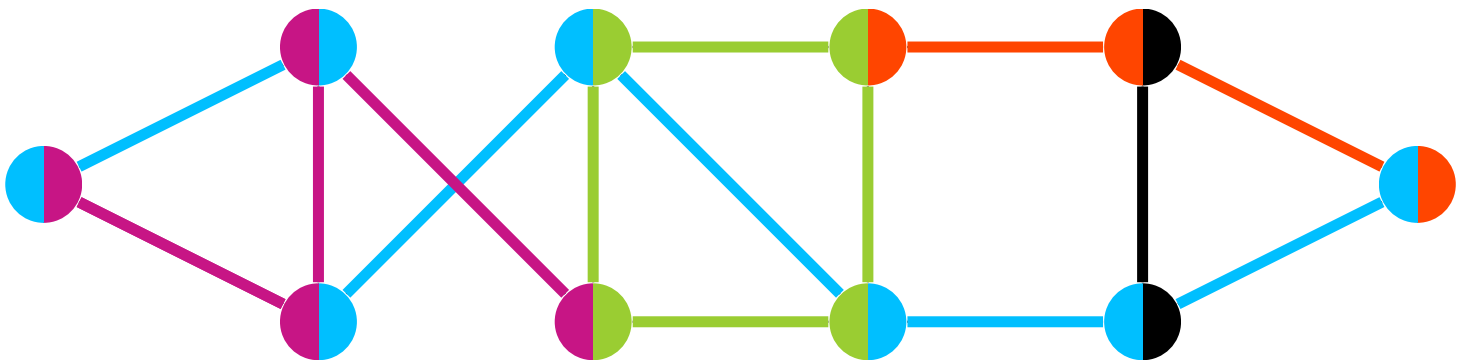
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



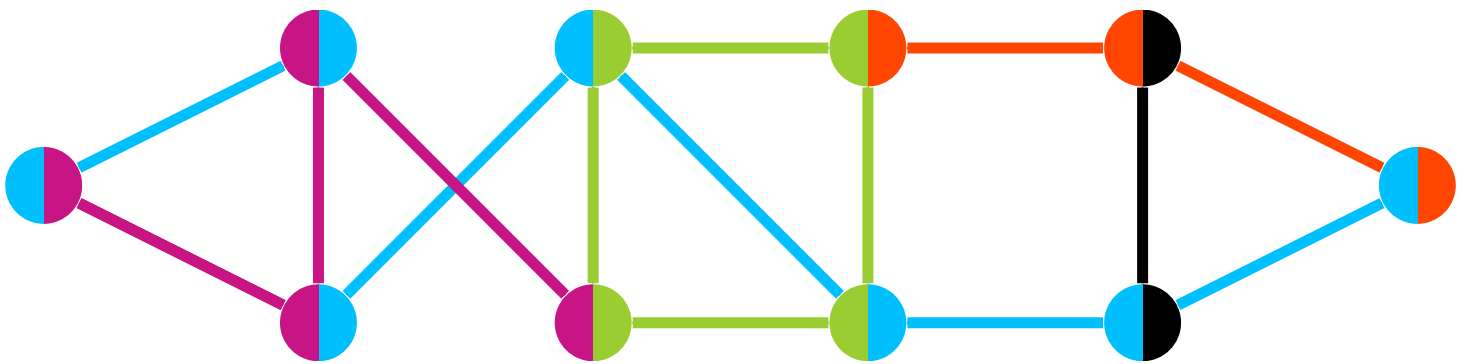
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



Maximum Edge Coloring

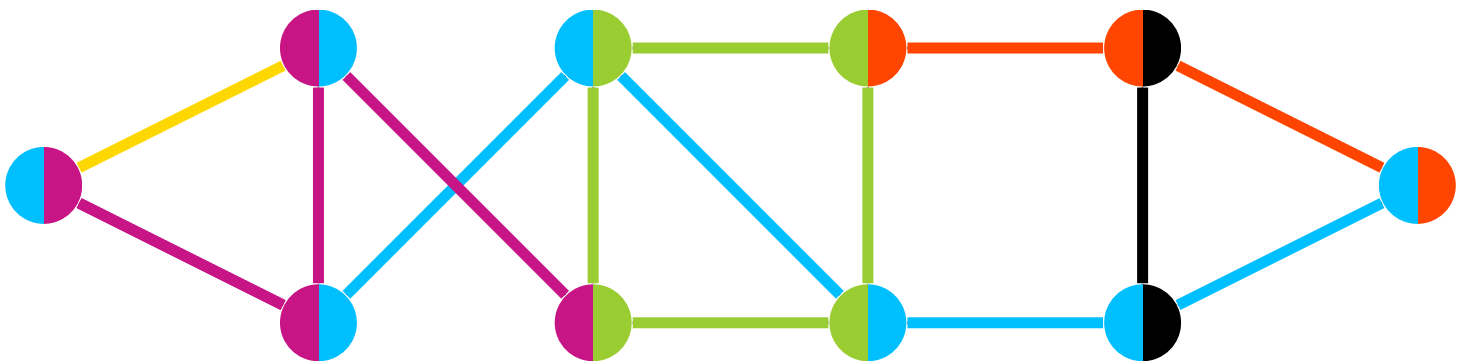
GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



This is not an optimal coloring yet.

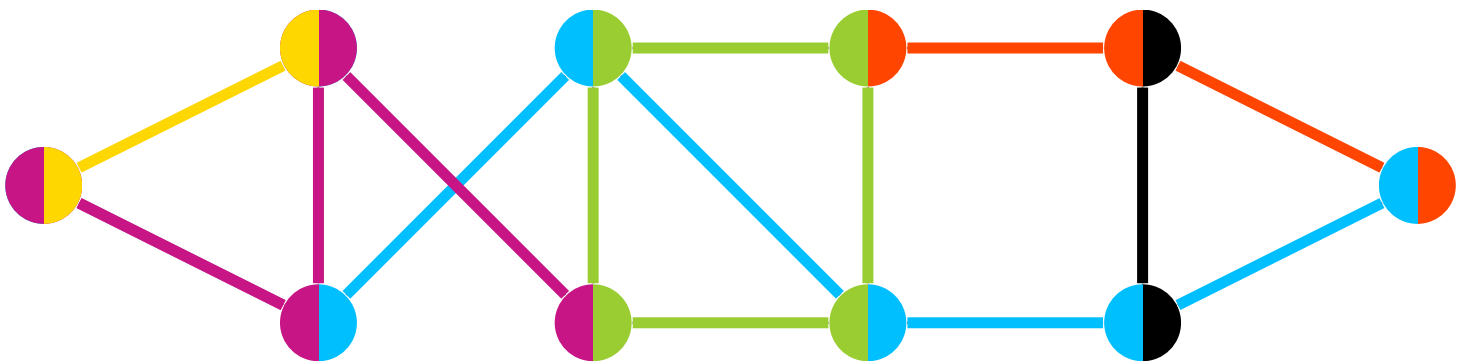
Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



Maximum Edge Coloring

GOAL. Color the edges of a graph so that each vertex "sees" at most two colors.



Motivation

In a network, every system has two interface cards.

Motivation

In a network, every system has two interface cards.

The goal is to assign frequency channels so that:

1. No system is assigned more than two channels.
2. The number of channels used overall is maximized.

Motivation

In a graph, every system has two interface cards.

The goal is to assign frequency channels so that:

1. No system is assigned more than two channels.
2. The number of channels used overall is maximized.

Motivation

In a graph, every vertex has two interface cards.

The goal is to assign frequency channels so that:

1. No system is assigned more than two channels.
2. The number of channels used overall is maximized.

Motivation

In a graph, every vertex has two interface cards.

The goal is to assign frequency channels so that:

1. No vertex sees more than two colors.
2. The number of channels used overall is maximized.

Motivation

In a graph, every vertex has two interface cards.

The goal is to assign frequency channels so that:

1. No vertex sees more than two colors.
2. The number of colors used overall is maximimized.

Past Work

Max Edge coloring is known to be NP-Complete and also APX-Hard (Adamaszek and Popa, 2010)

Past Work

Max Edge coloring is known to be NP-Complete and also APX-Hard (Adamaszek and Popa, 2010)

A 2-approximation algorithm is known on general graphs (Feng, Zhang and Wang, 2009)

Past Work

Max Edge coloring is known to be NP-Complete and also APX-Hard (Adamaszek and Popa, 2010)

A 2-approximation algorithm is known on general graphs (Feng, Zhang and Wang, 2009)

The problem is shown to have a polynomial time algorithm for complete graphs and trees (Feng, Zhang and Wang, 2009)

Past Work

Max Edge coloring is known to be NP-Complete and also APX-Hard (Adamaszek and Popa, 2010)

A 2-approximation algorithm is known on general graphs (Feng, Zhang and Wang, 2009)

The problem is shown to have a polynomial time algorithm for complete graphs and trees (Feng, Zhang and Wang, 2009)

There exists a $\frac{5}{3}$-approximation algorithm for graphs with perefect matching (Adamaszek and Popa, 2010)

Maximum Edge Coloring: The Decision Version

Can we color with at least k colors?

Maximum Edge Coloring: The Decision Version

Can we color with at least k colors?

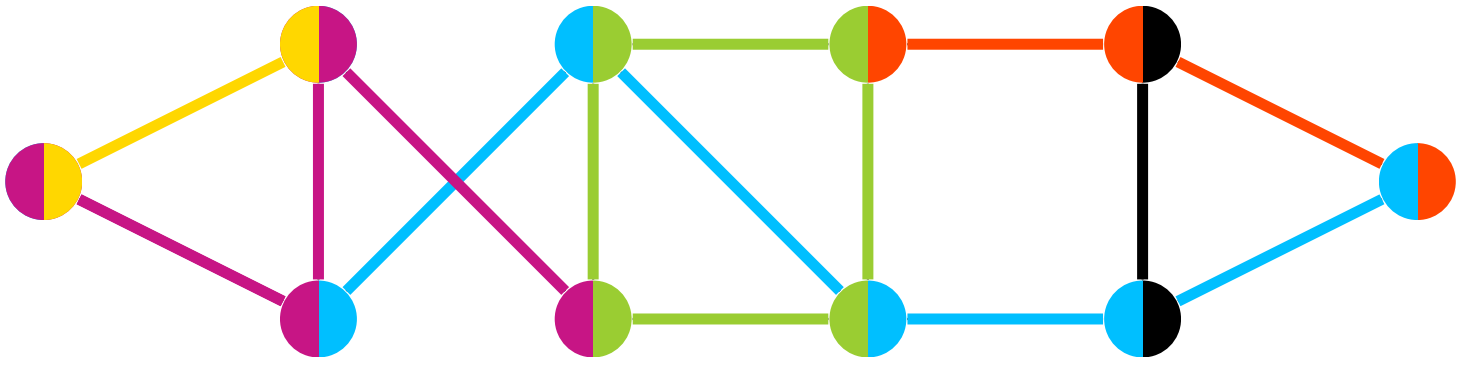
Maximum Edge Coloring: The Decision Version

Can we color with at least k colors?

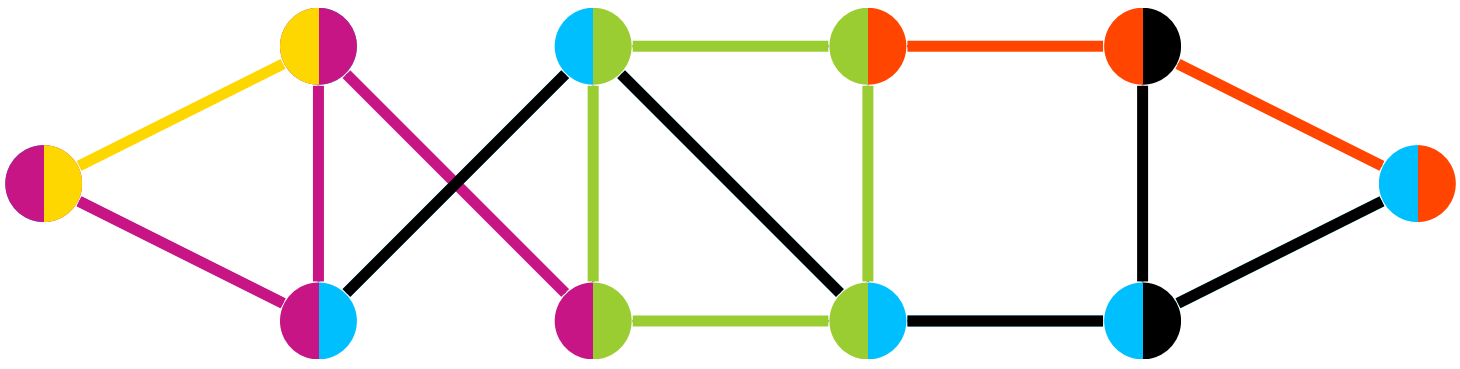
$\equiv$

Can we color with exactly k colors?

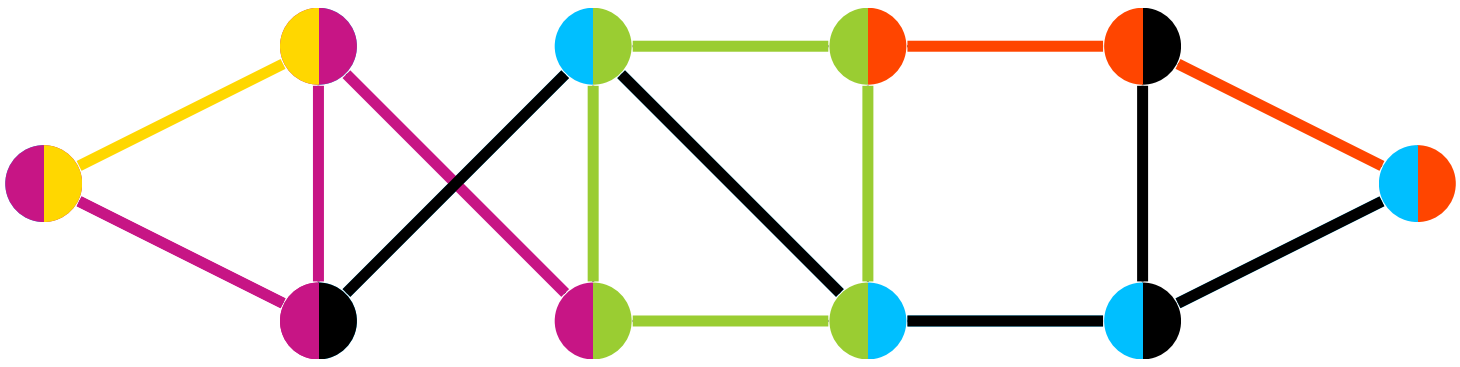
Blue $\longrightarrow$ Black.



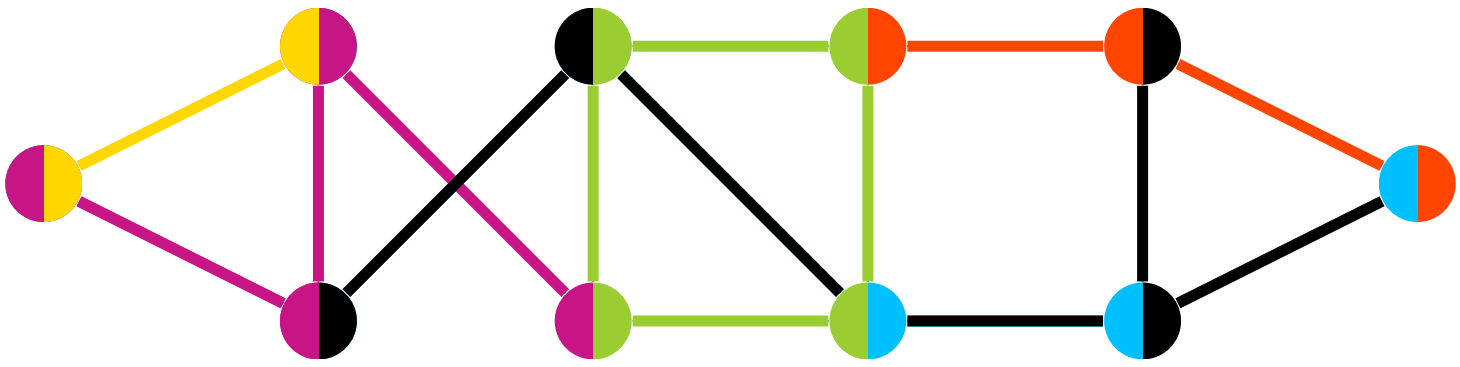
Blue $\longrightarrow$ Black.



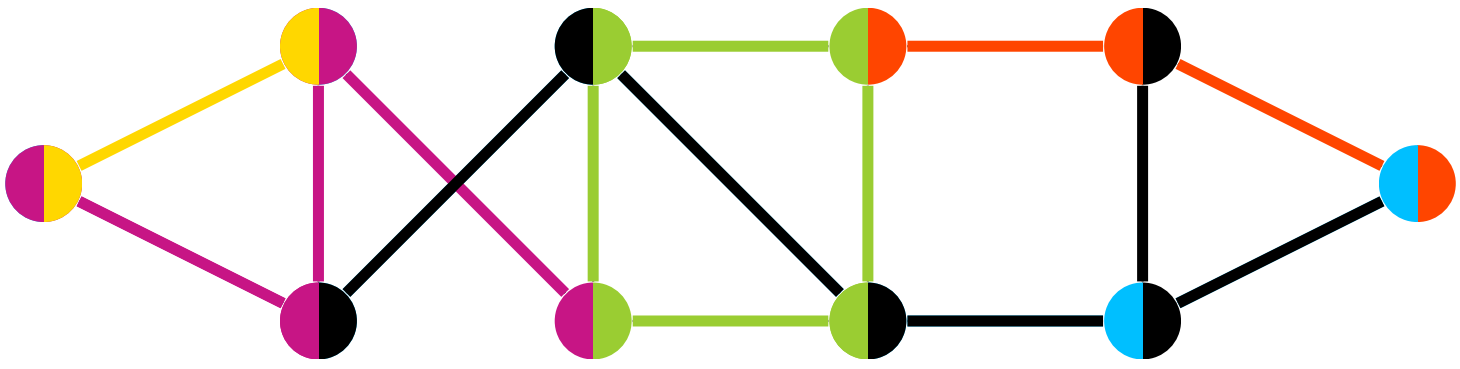
Blue $\longrightarrow$ Black.



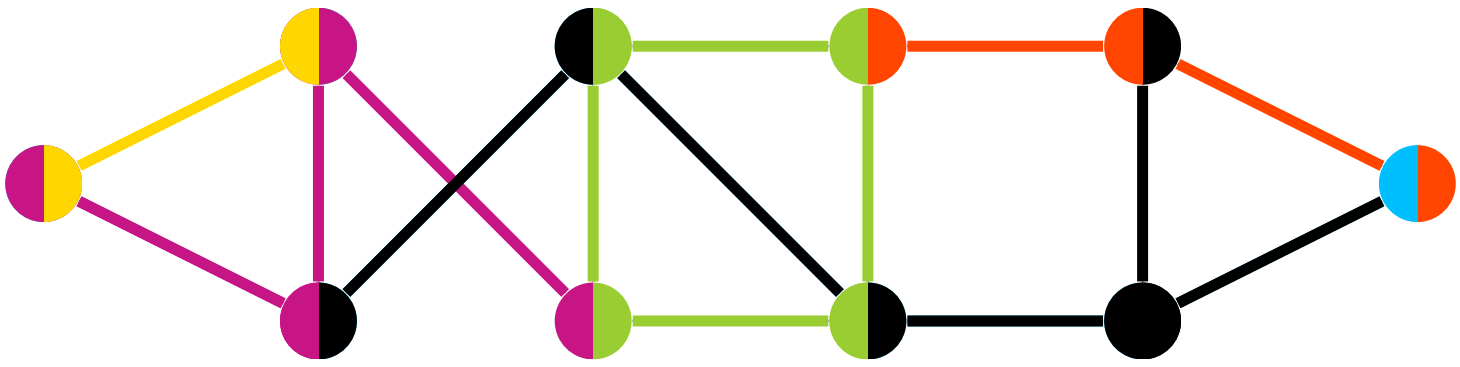
Blue $\longrightarrow$ Black.



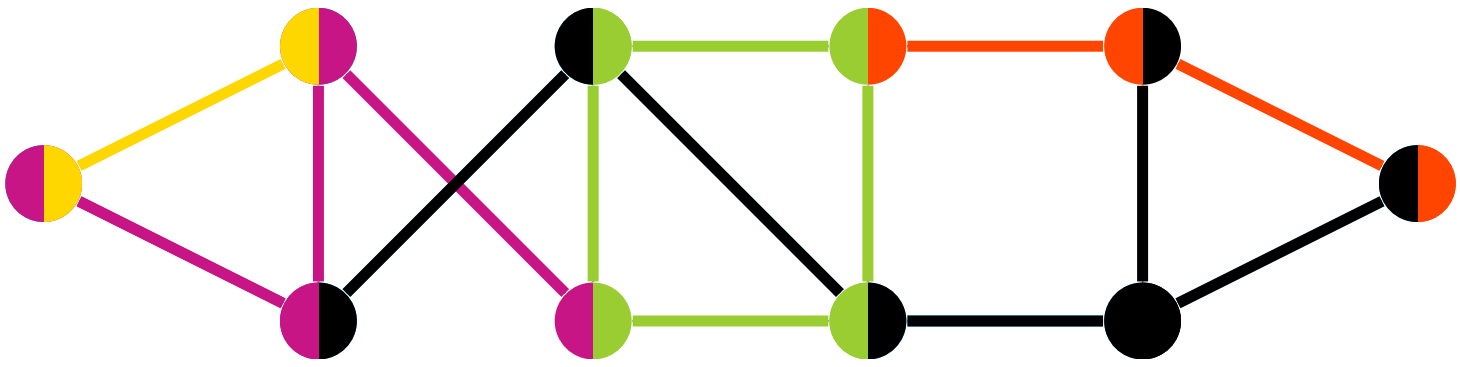
Blue $\longrightarrow$ Black.



Blue $\longrightarrow$ Black.



Blue $\longrightarrow$ Black.



The Maximum Edge Coloring Problem (Parameterized)

Input: A graph G and an integer k.

Question: Can the edges of G be colored with k colors so that no vertex sees more than two colors?

Parameter: k

The Maximum Edge Coloring Problem (Parameterized)

Input: A graph G and an integer k.

Question: Can the edges of G be colored with k colors so that no vertex sees more than two colors?

Parameter: k

A parameterized problem is denoted by a pair $(Q, k) \subseteq \Sigma^* \times \mathbb{N}$.

A parameterized problem is denoted by a pair $(Q, k) \subseteq \Sigma^* \times \mathbb{N}$.

The first component Q is a classical language, and the number k is called the parameter.

A parameterized problem is denoted by a pair $(Q, k) \subseteq \Sigma^* \times \mathbb{N}$.

The first component Q is a classical language, and the number k is called the parameter.

Such a problem is fixed-parameter tractable or FPT if there exists an algorithm that decides it in time $O(f(k)n^{O(1)})$ on instances of size n.

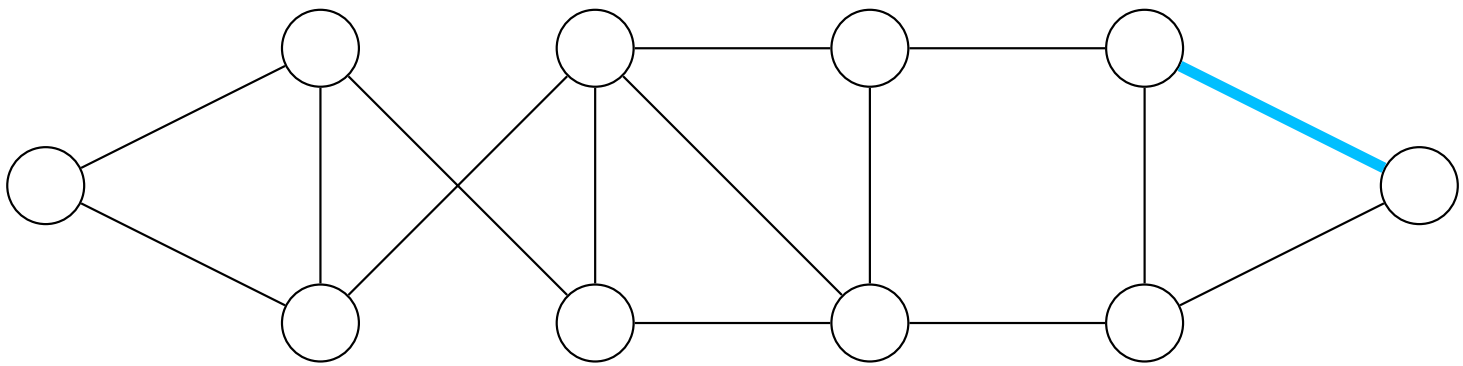
If there are less than k edges $\Rightarrow$ Say NO.

If there are less than k edges $\Rightarrow$ Say NO.

A matching of size at least $(k-1) \Rightarrow$ Say YES.

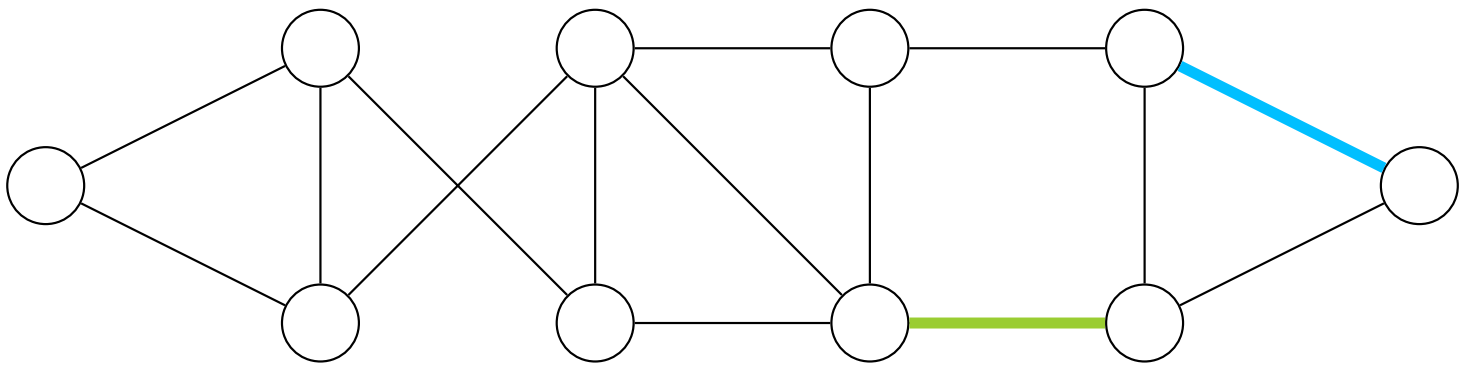
If there are less than k edges $\Rightarrow$ Say NO.

A matching of size at least $(k-1)$ $\Rightarrow$ Say YES.



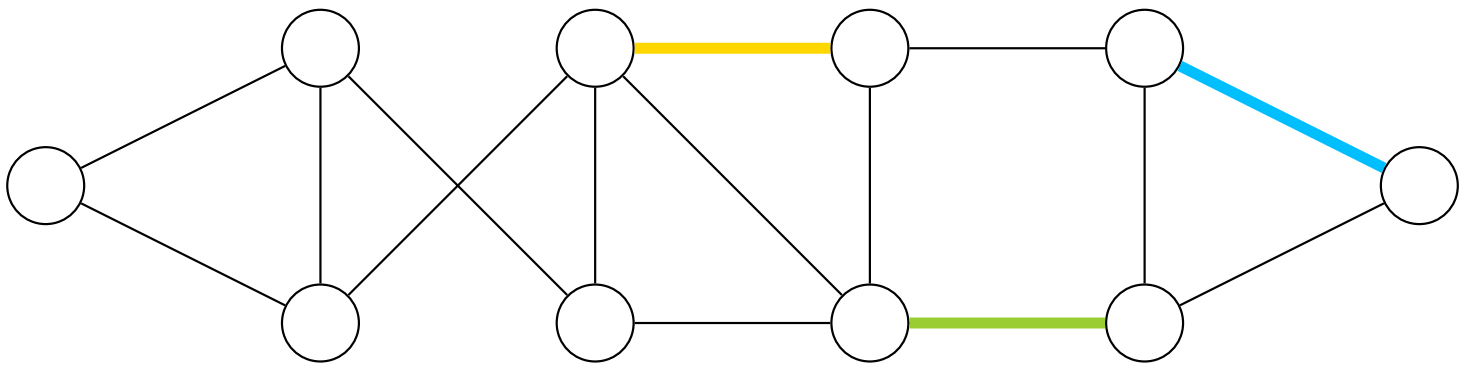
If there are less than k edges $\Rightarrow$ Say NO.

A matching of size at least $(k-1)$ $\Rightarrow$ Say YES.



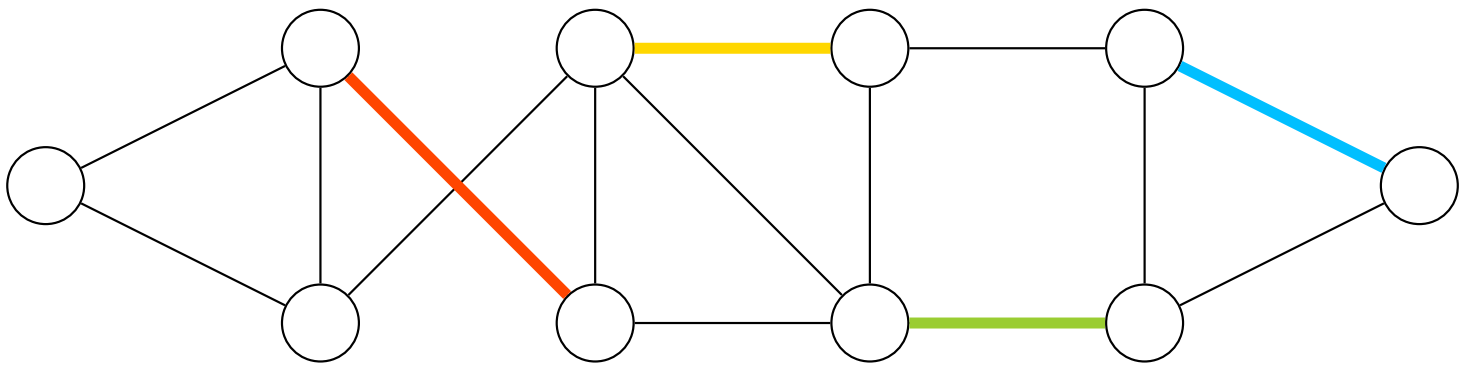
If there are less than k edges $\Rightarrow$ Say NO.

A matching of size at least $(k-1)$ $\Rightarrow$ Say YES.



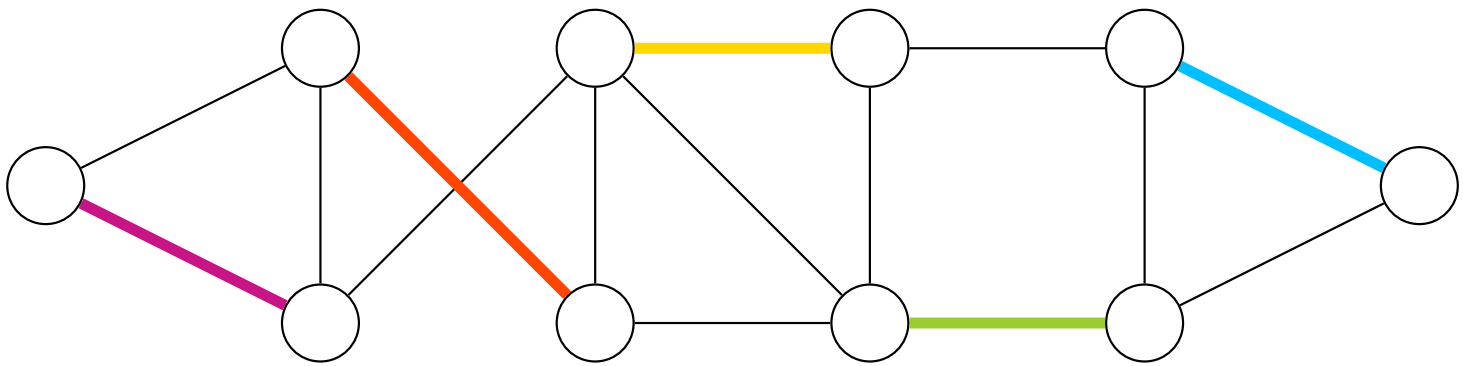
If there are less than k edges $\Rightarrow$ Say NO.

A matching of size at least $(k-1)$ $\Rightarrow$ Say YES.



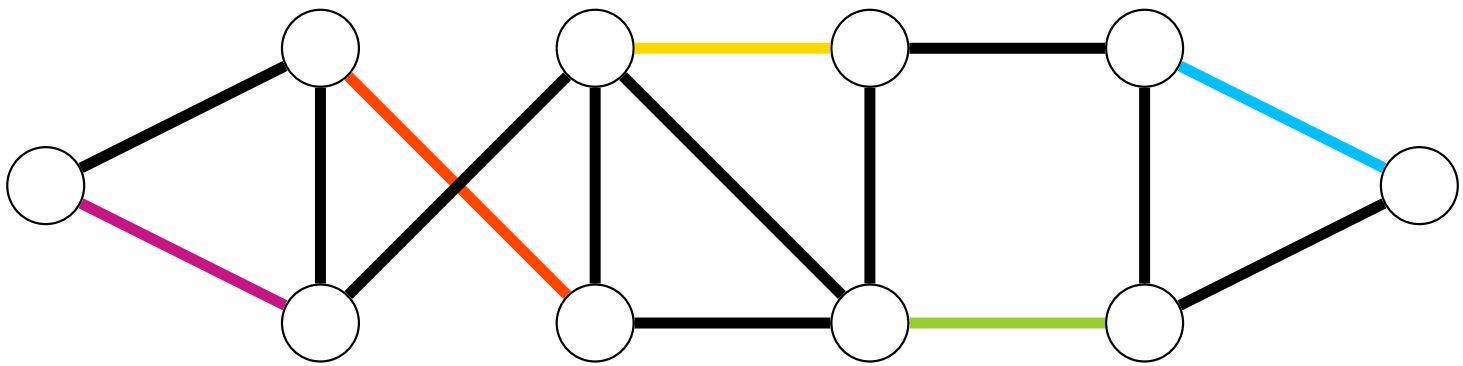
If there are less than k edges $\Rightarrow$ Say NO.

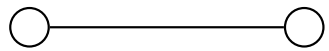
A matching of size at least $(k-1)$ $\Rightarrow$ Say YES.

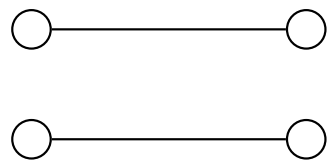


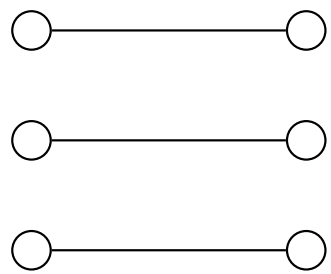
If there are less than k edges $\Rightarrow$ Say NO.

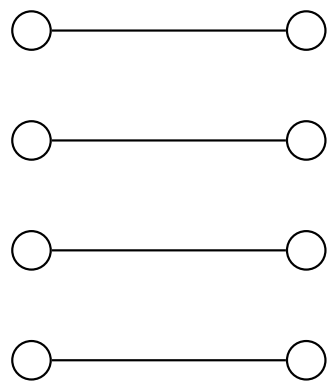
A matching of size at least $(k-1)$ $\Rightarrow$ Say YES.

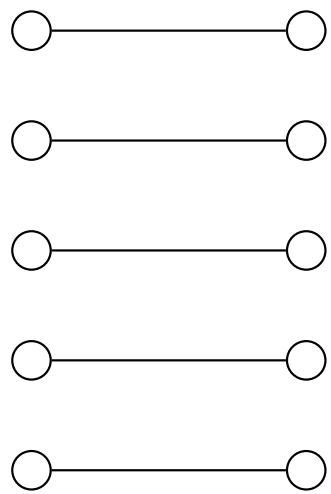


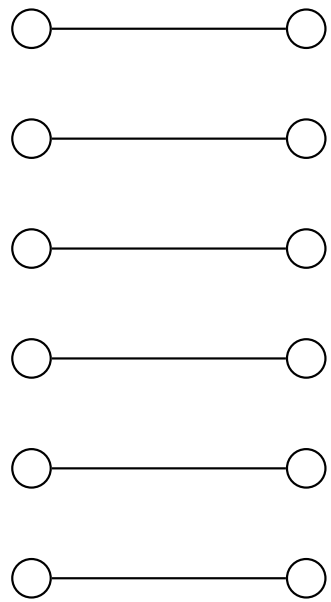


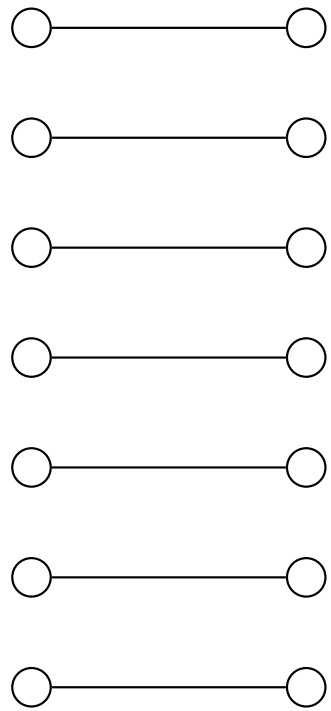


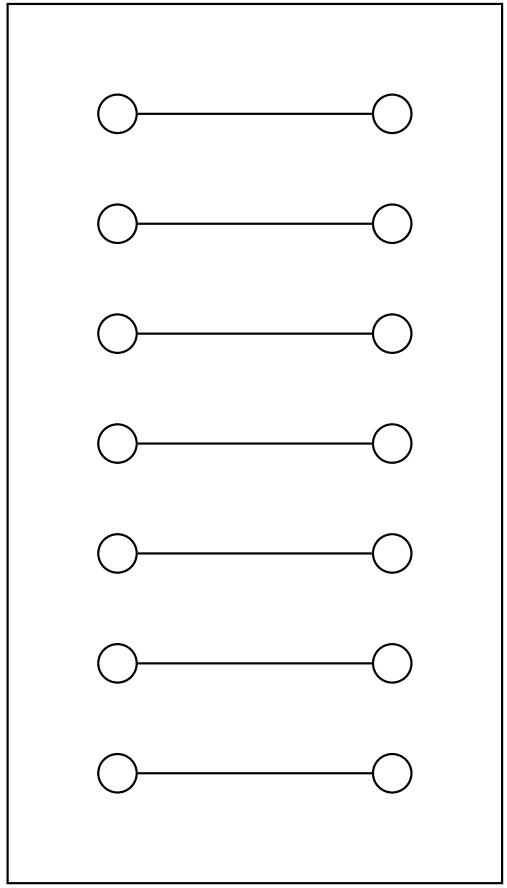


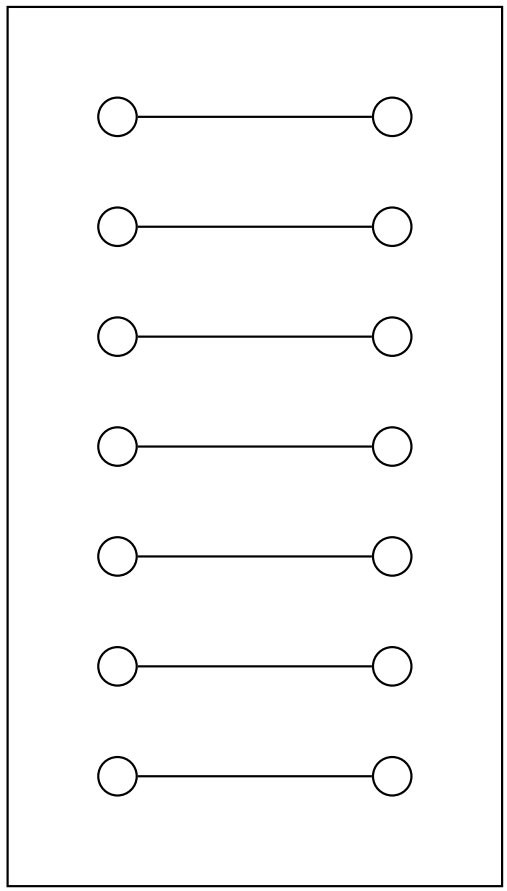






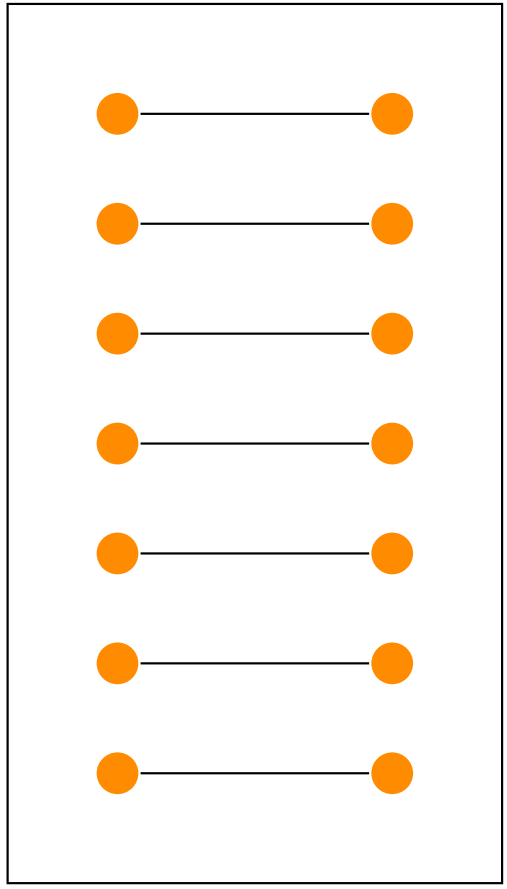






We have a vertex cover

of size at most $2k$.

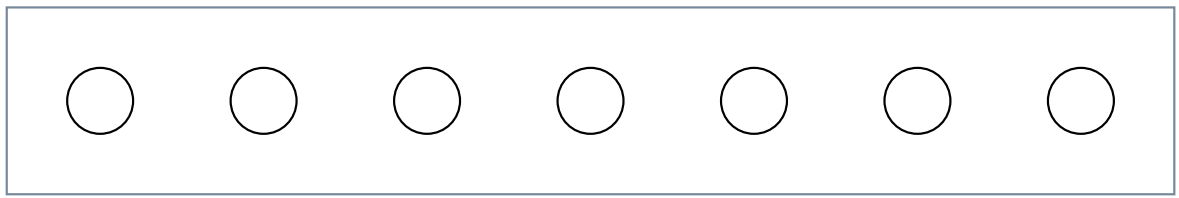


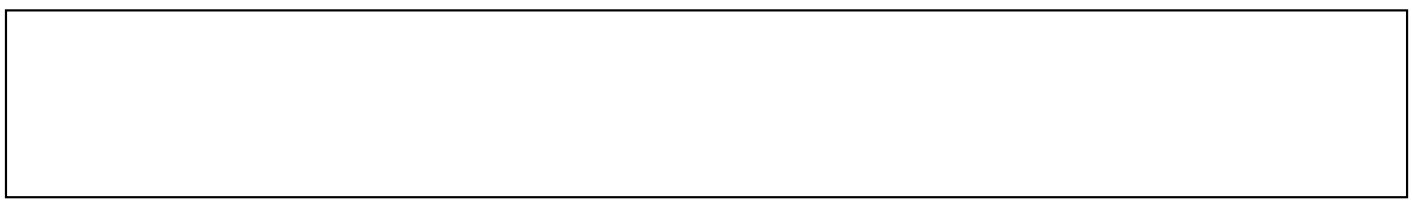
We have a vertex cover

of size at most $2k$.

COLOR PALETTE

VERTEX COVER

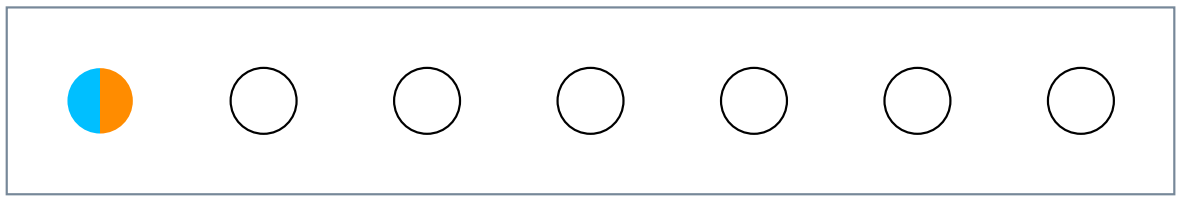


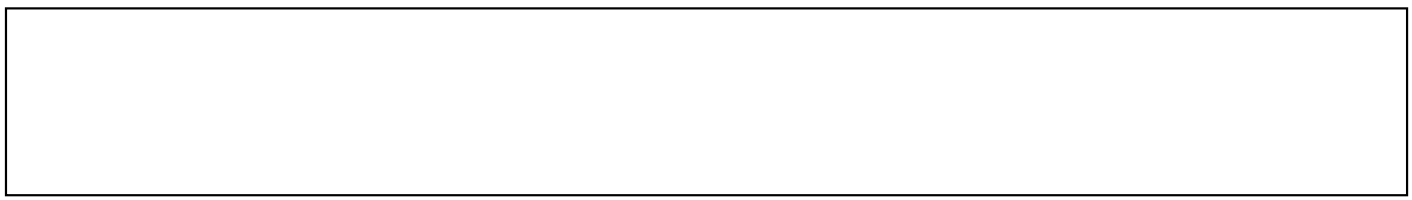


INDEPENDENT SET

COLOR PALETTE

VERTEX COVER

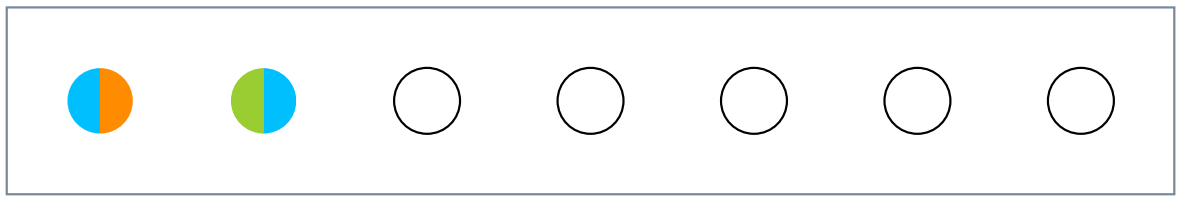


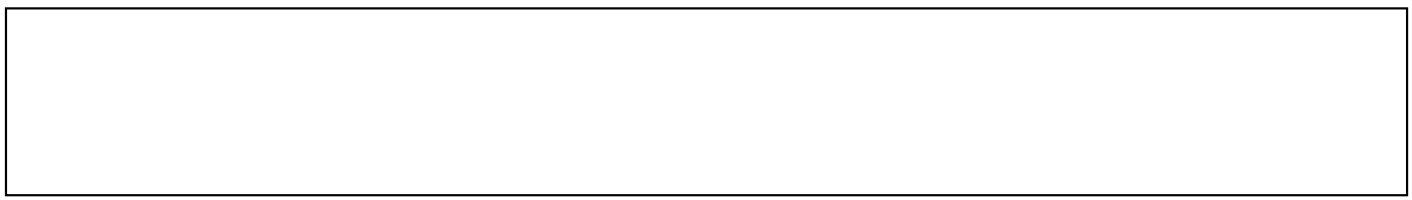


INDEPENDENT SET

COLOR PALETTE

VERTEX COVER

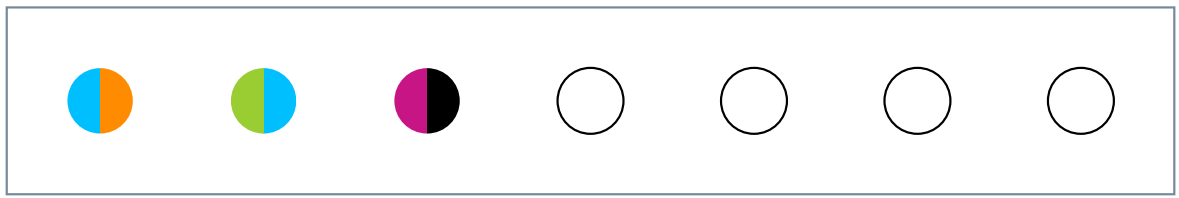


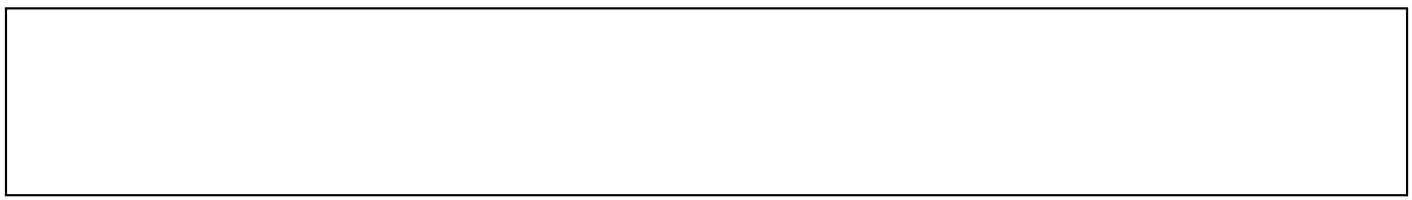


INDEPENDENT SET

COLOR PALETTE

VERTEX COVER

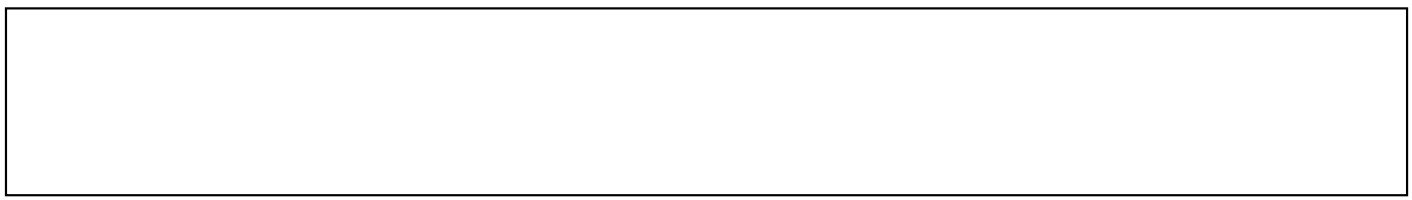




INDEPENDENT SET

COLOR PALETTE

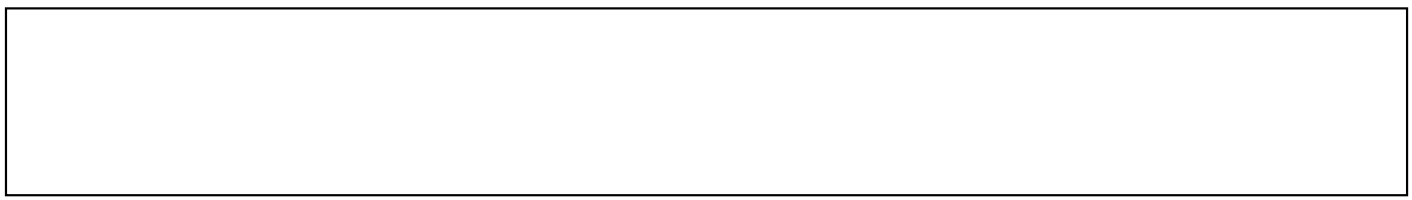
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

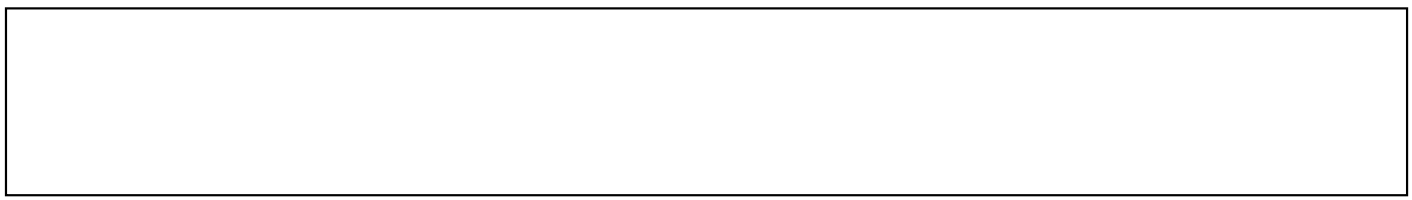
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

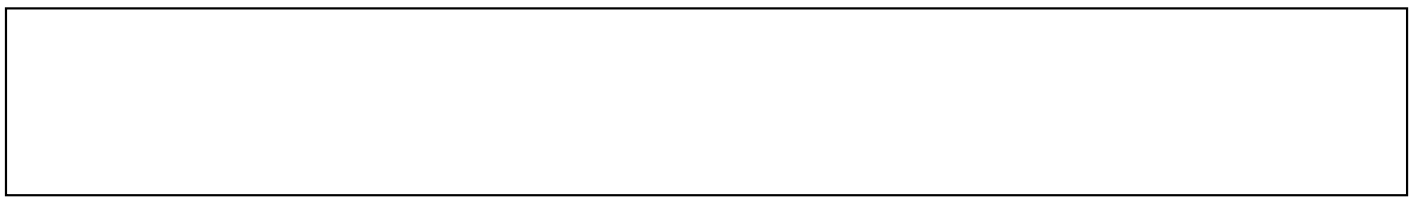
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

VERTEX COVER



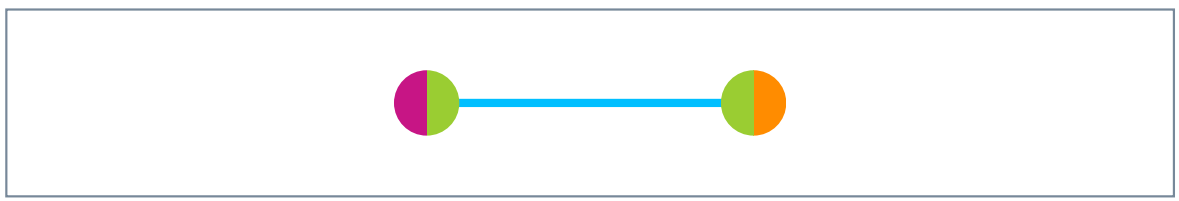
INDEPENDENT SET

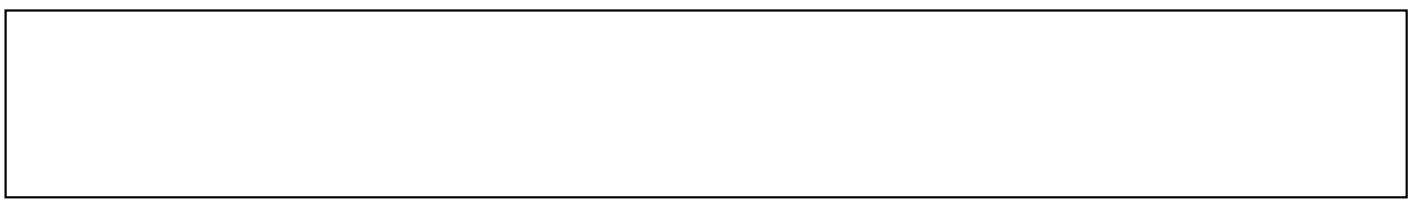
To realize a palette assignment, we must assign colors so that:

To realize a palette assignment, we must assign colors so that:

1. Every edge respects the palette.

VERTEX COVER

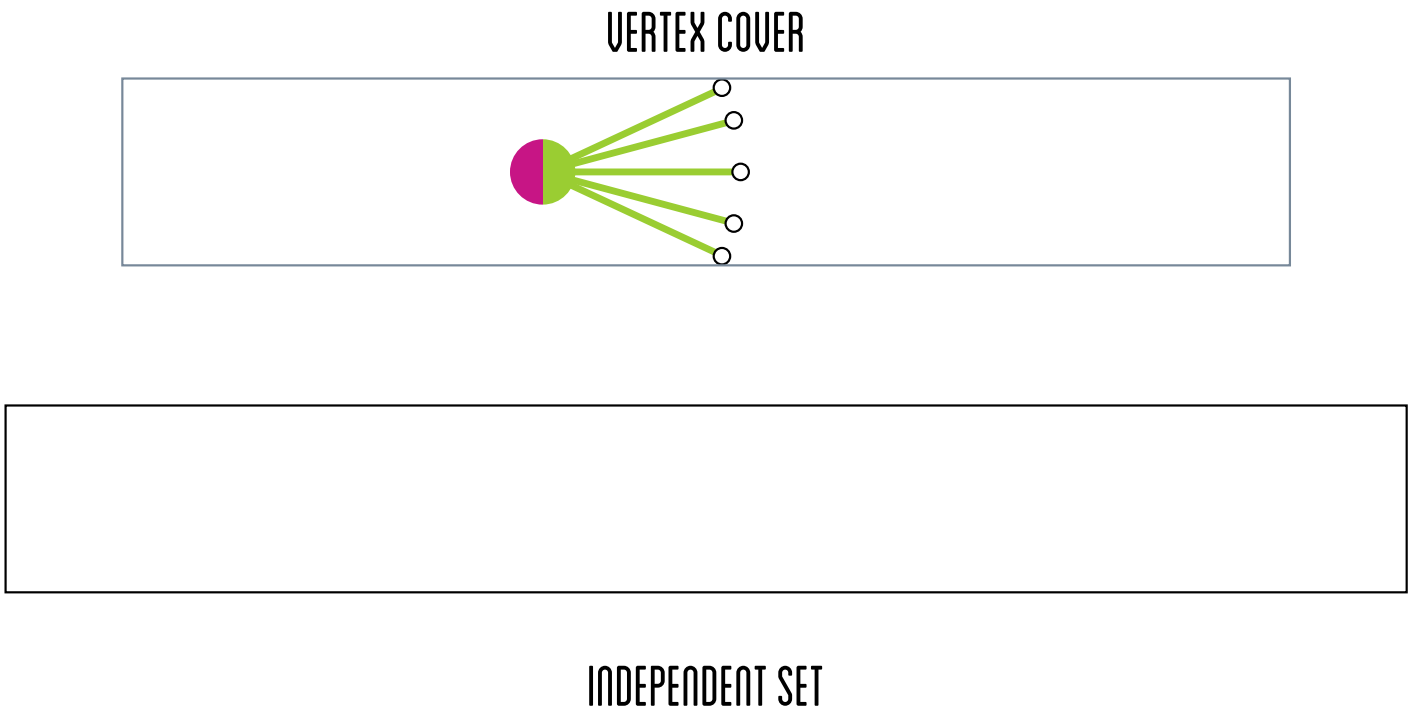




INDEPENDENT SET

To realize a palette assignment, we must assign colors so that:

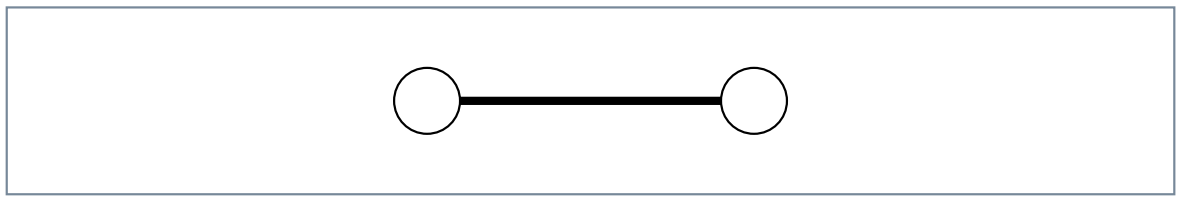
1. Every edge respects the palette.
2. Every palette is satisified.



Sanity Checks

COLOR PALETTE

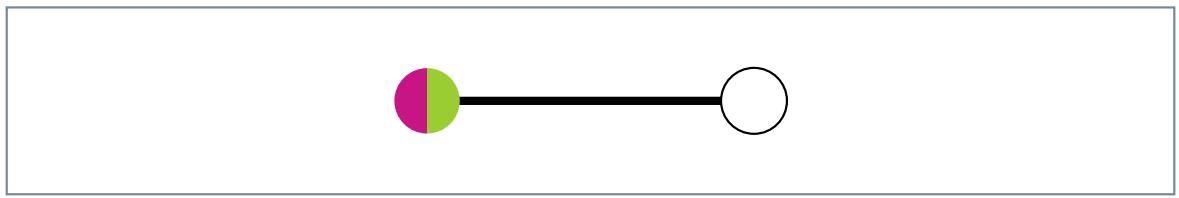
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

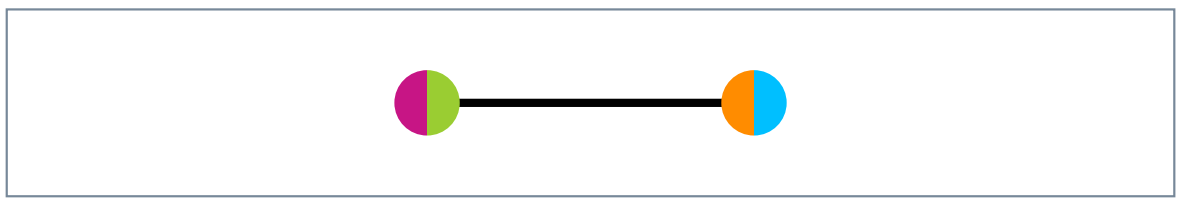
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

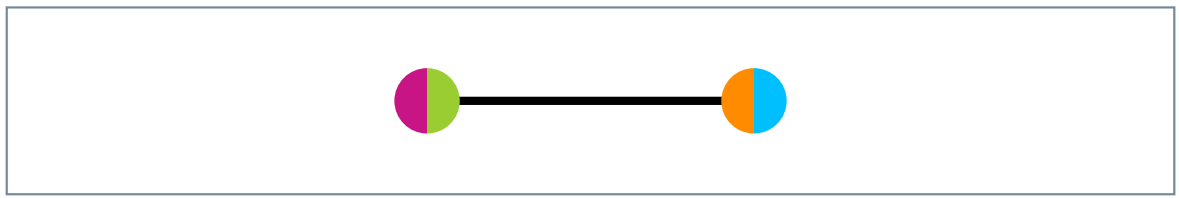
VERTEX COVER



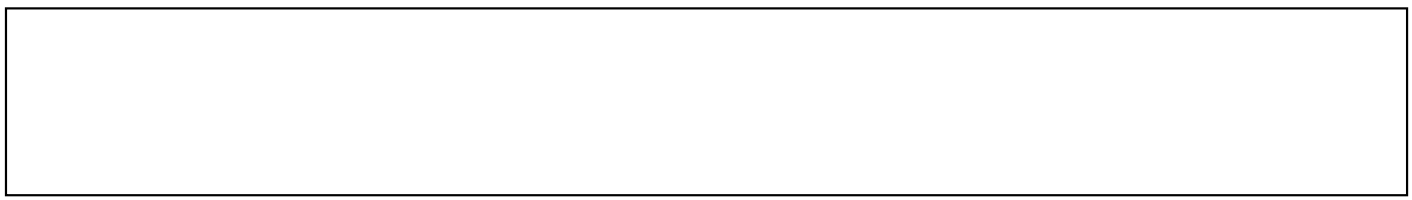
INDEPENDENT SET

COLOR PALETTE

VERTEX COVER



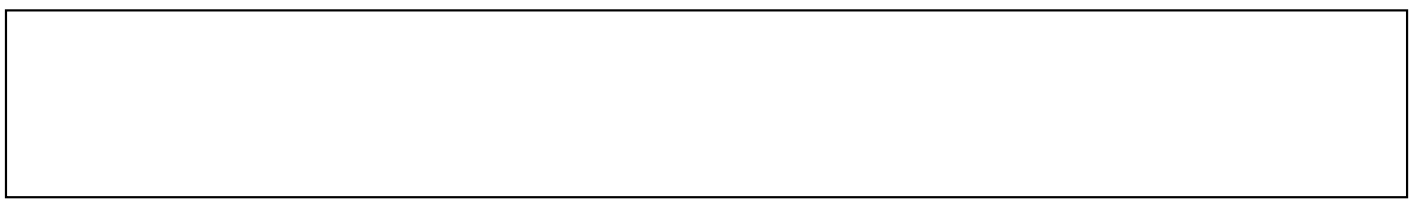
Reject this palette assignment...



INDEPENDENT SET

COLOR PALETTE

VERTEX COVER

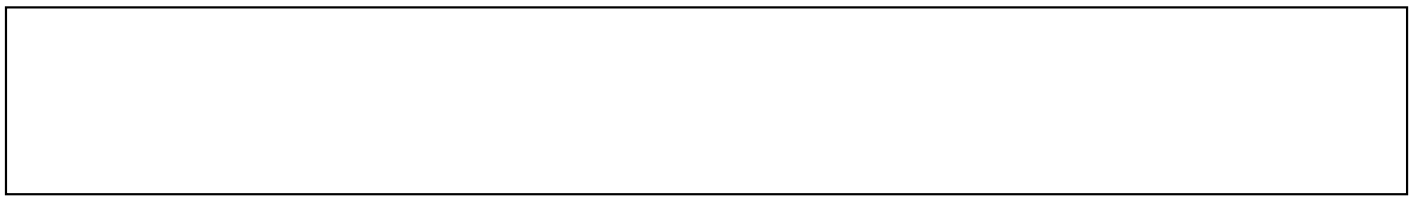


INDEPENDENT SET

COLOR PALETTE

VERTEX COVER

Every color must be realized in the palettes of the vertex cover vertices.



INDEPENDENT SET

COLOR PALETTE

VERTEX COVER

Reject this assignment.

INDEPENDENT SET

Guess a split of the Palette

COLOR PALETTE

Guess X: the set of colors assigned to edges within the vertex cover.

VERTEX COVER

INDEPENDENT SET

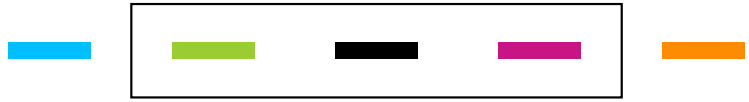
COLOR PALETTE

Guess X: the set of colors assigned to edges within the vertex cover.

VERTEX COVER

INDEPENDENT SET

COLOR PALETTE

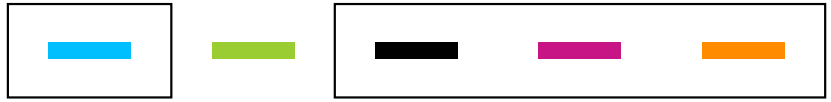


Guess X: the set of colors assigned to edges within the vertex cover.

VERTEX COVER

INDEPENDENT SET

COLOR PALETTE



Guess X: the set of colors assigned to edges within the vertex cover.

VERTEX COVER

INDEPENDENT SET

COLOR PALETTE

$$|X| \leqslant k.$$

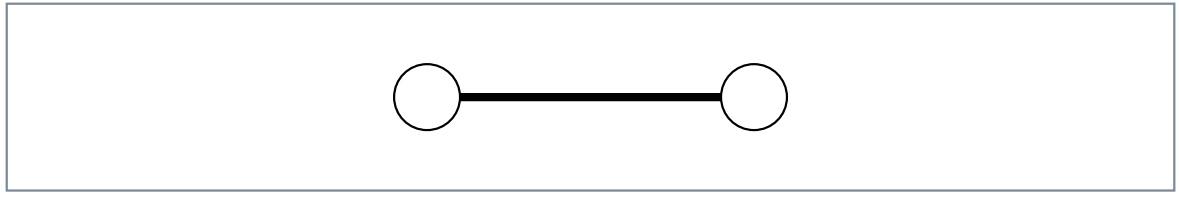
VERTEX COVER

INDEPENDENT SET

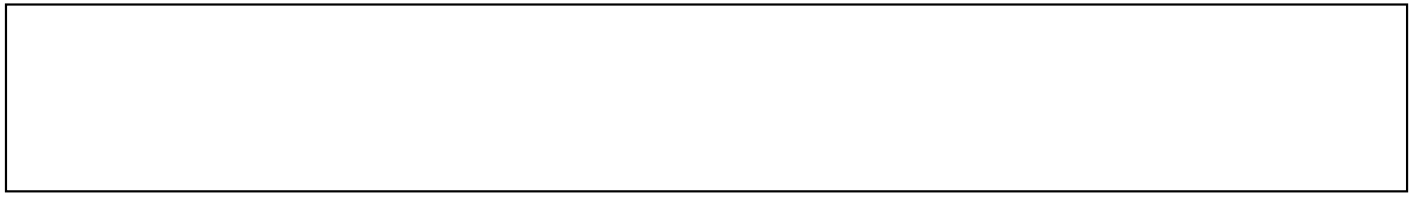
Assign Colors Within the Vertex Cover

COLOR PALETTE WITH X FIXED

VERTEX COVER



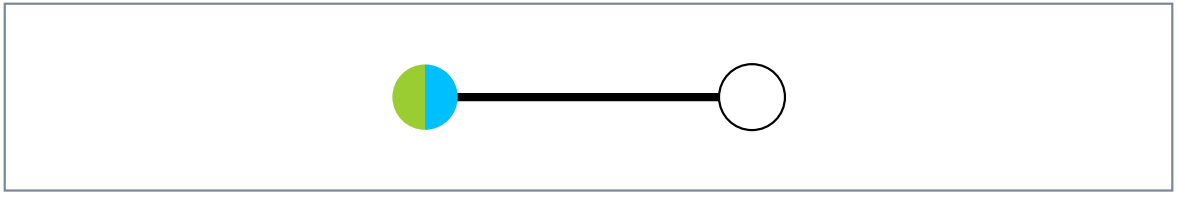
Case 1: The palettes intersect at one color.



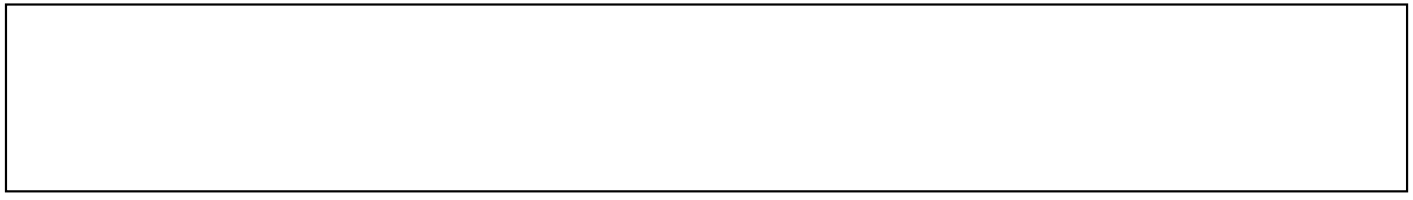
INDEPENDENT SET

COLOR PALETTE WITH X FIXED

VERTEX COVER



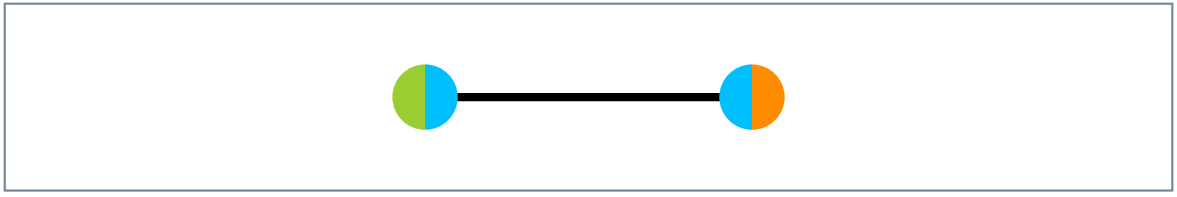
Case 1: The palettes intersect at one color.



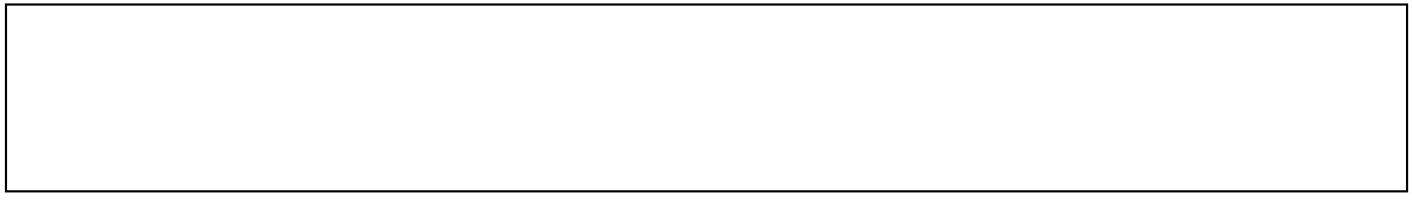
INDEPENDENT SET

COLOR PALETTE WITH X FIXED

VERTEX COVER



Case 1: The palettes intersect at one color.

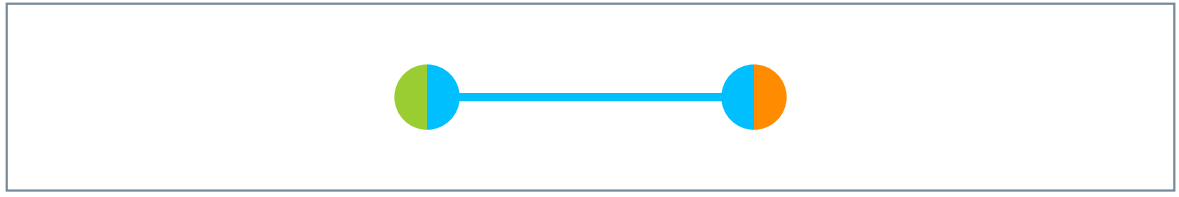


INDEPENDENT SET

COLOR PALETTE WITH X FIXED

The edge gets that color.

VERTEX COVER

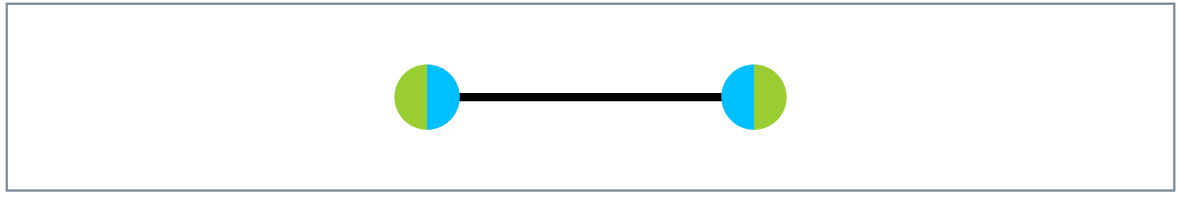


Case 1: The palettes intersect at one color.

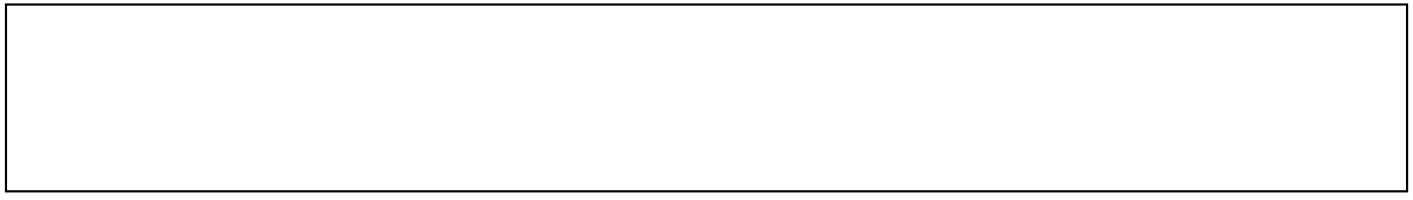
INDEPENDENT SET

COLOR PALETTE WITH X FIXED

VERTEX COVER



Case 2: The palettes are the same.

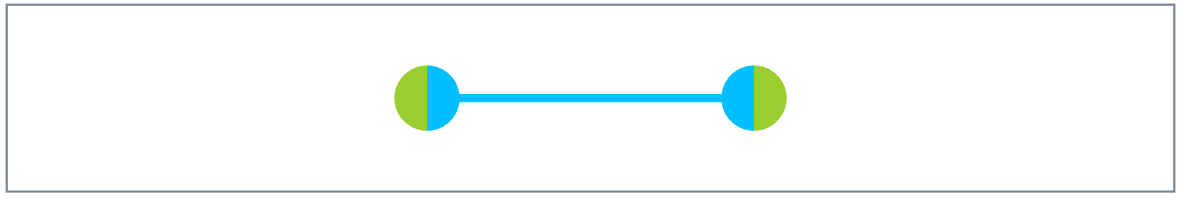


INDEPENDENT SET

COLOR PALETTE WITH X FIXED

If only one of the colors is in X, assign that color.

VERTEX COVER



Case 2: The palettes are the same.

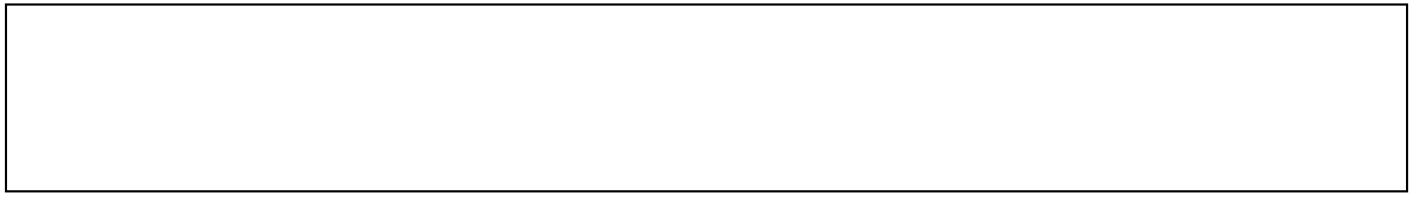
INDEPENDENT SET

COLOR PALETTE WITH X FIXED

If both colors are in X, branch.

VERTEX COVER

Case 2: The palettes are the same.



INDEPENDENT SET

Whenever a color in X is assigned to an edge, mark it as used.

Whenever a color in X is assigned to an edge, mark it as used.

Branch only over unused colors.

Whenever a color in X is assigned to an edge, mark it as used.

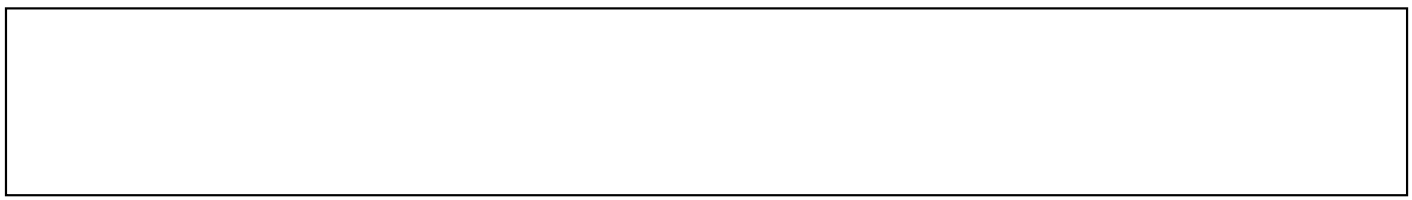
Branch only over unused colors.

Once all colors in X are used, assign colors arbitrarily.

Assign Colors Outside the Vertex Cover

COLOR PALETTE

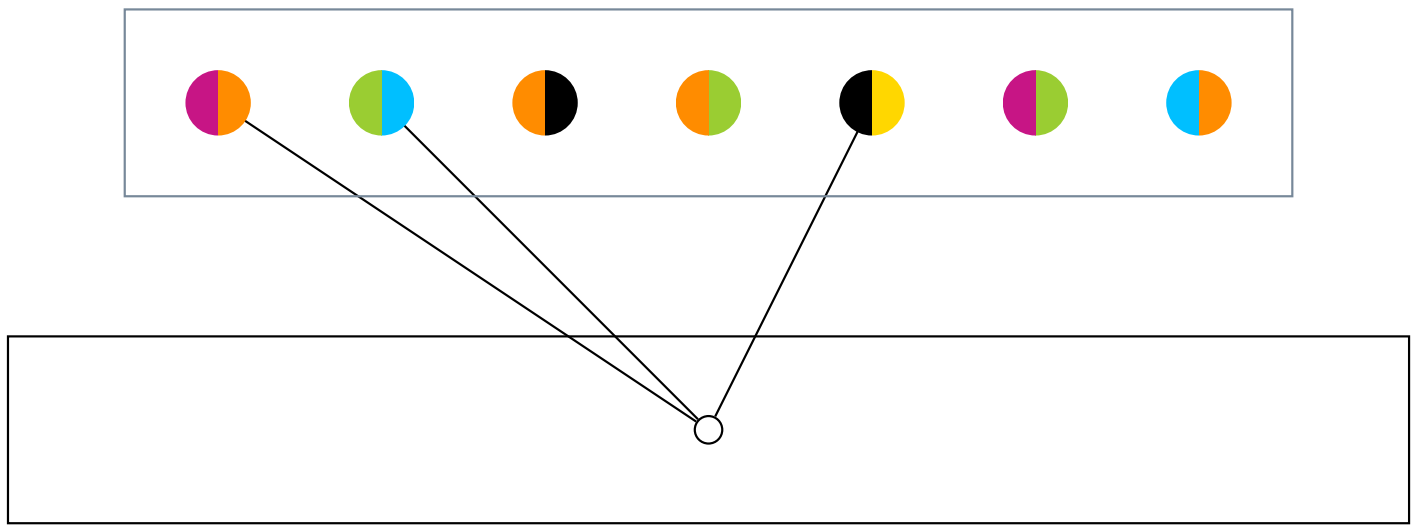
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

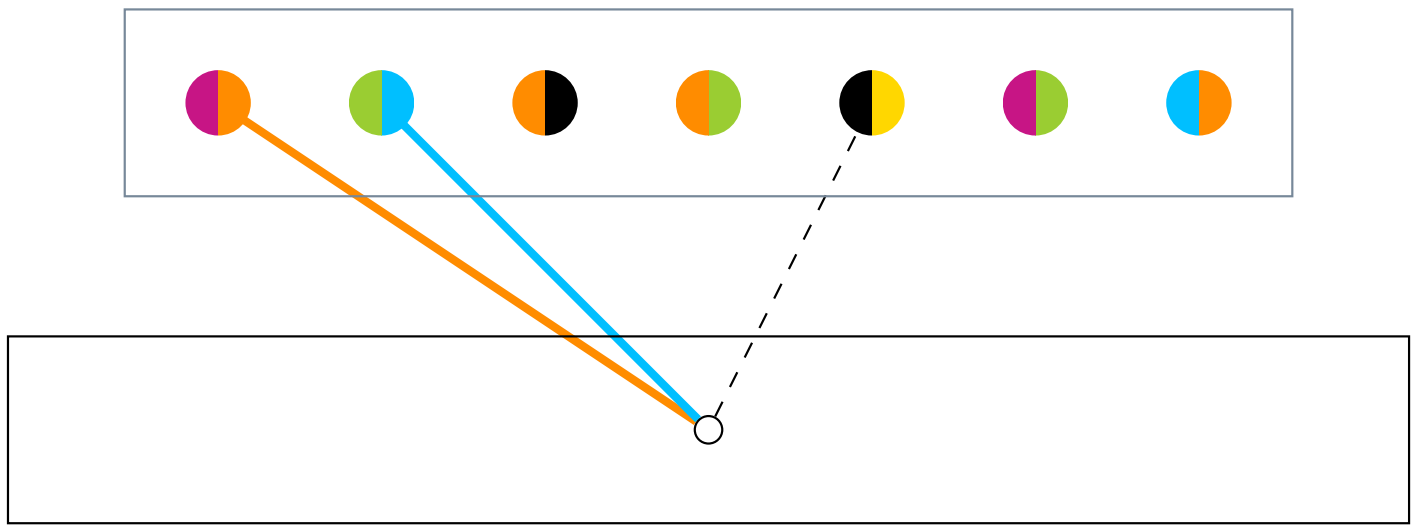
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

VERTEX COVER



INDEPENDENT SET

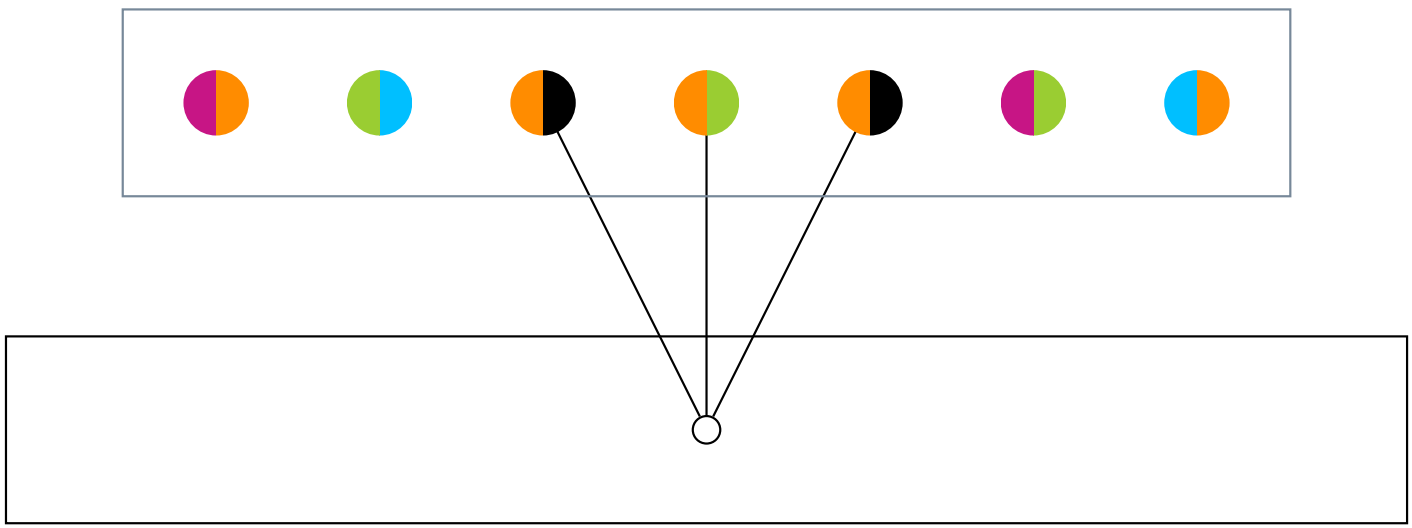
COLOR PALETTE

VERTEX COVER

INDEPENDENT SET

COLOR PALETTE

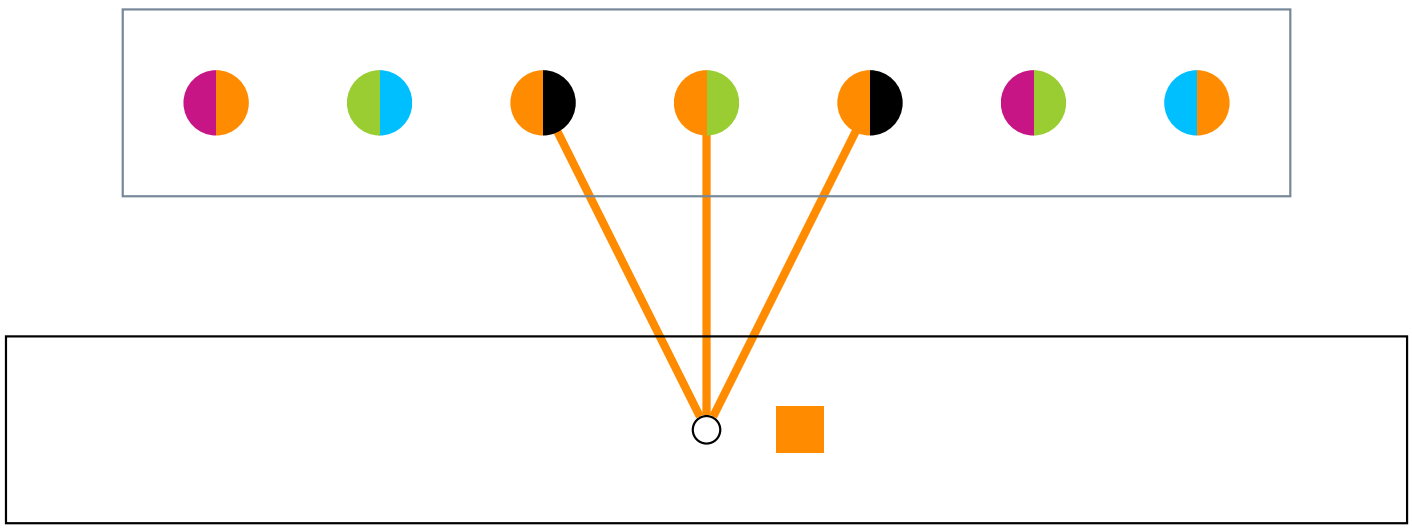
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

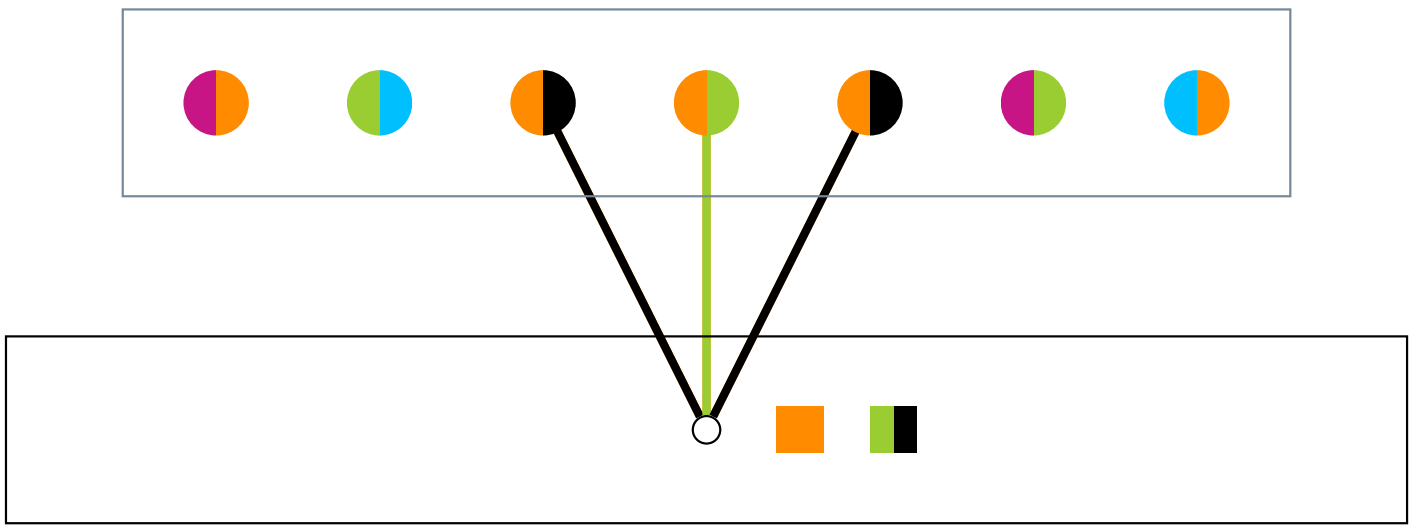
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

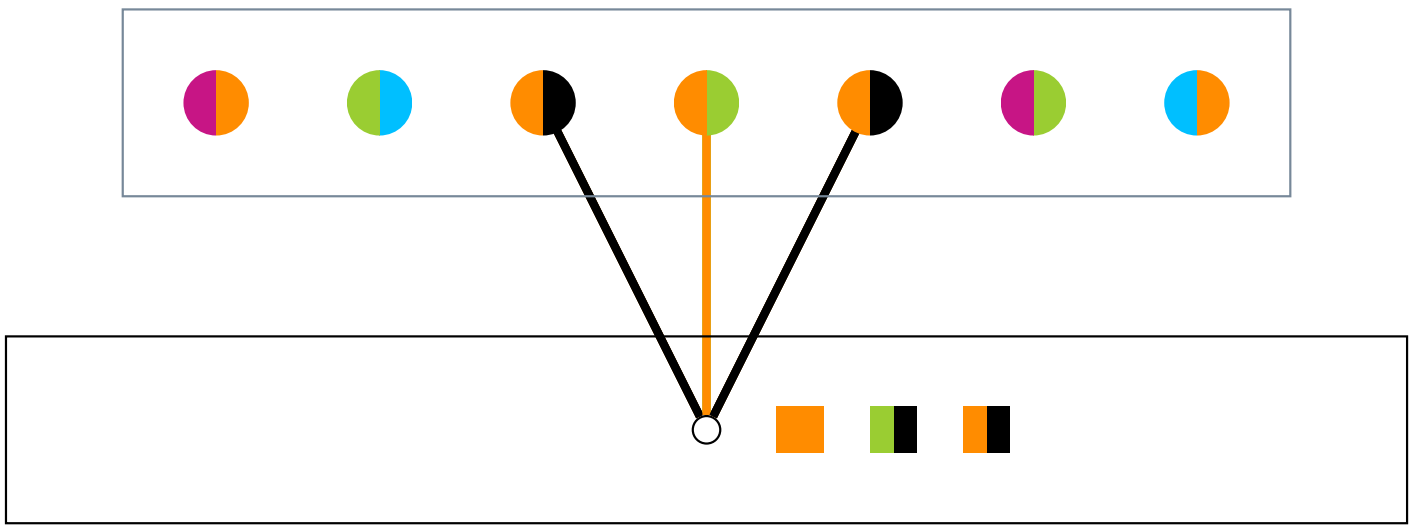
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

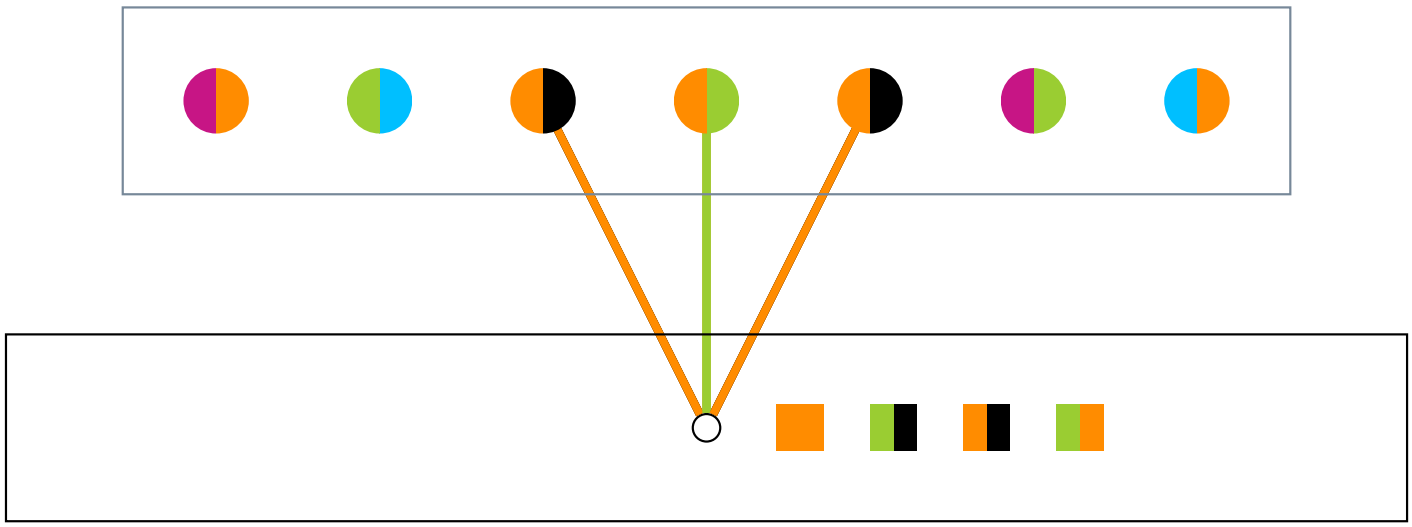
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

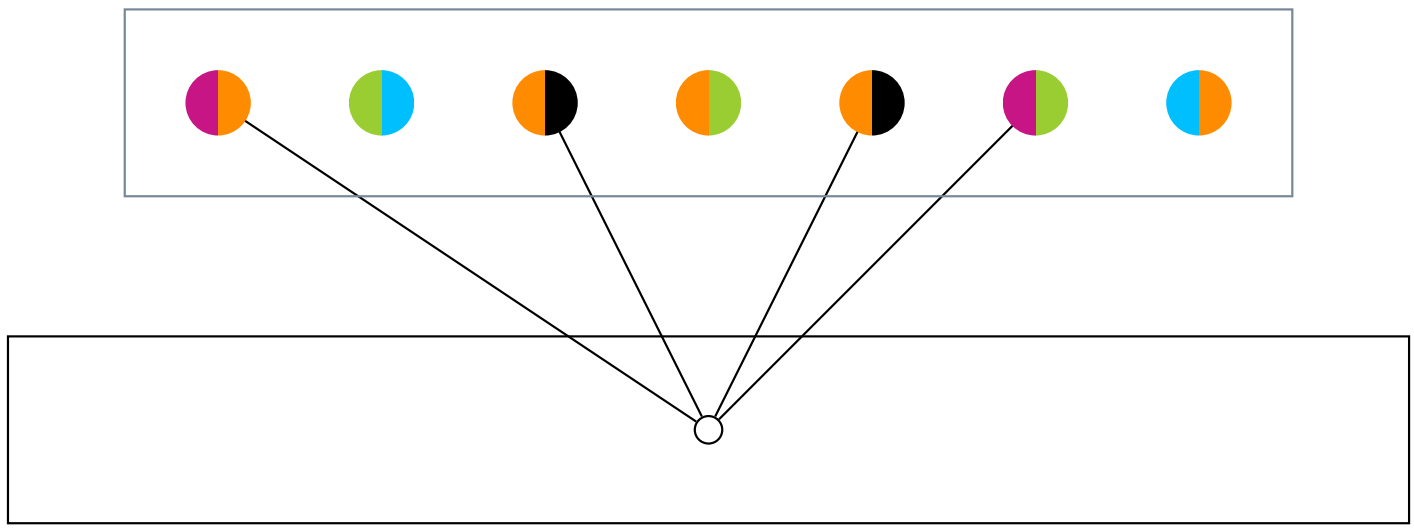
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

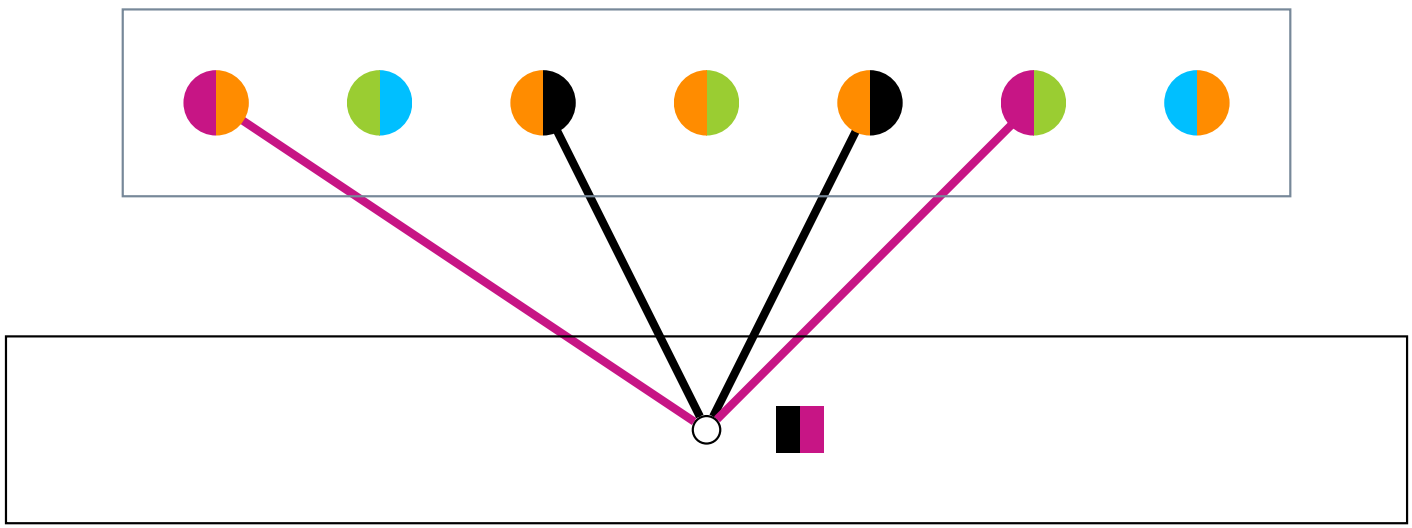
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

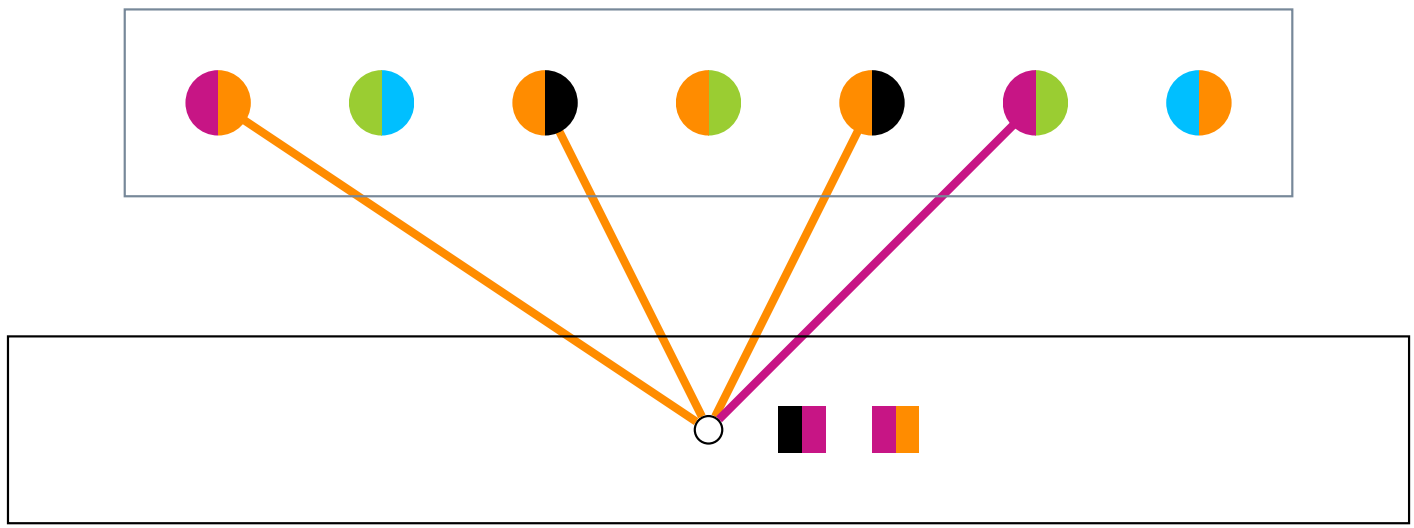
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

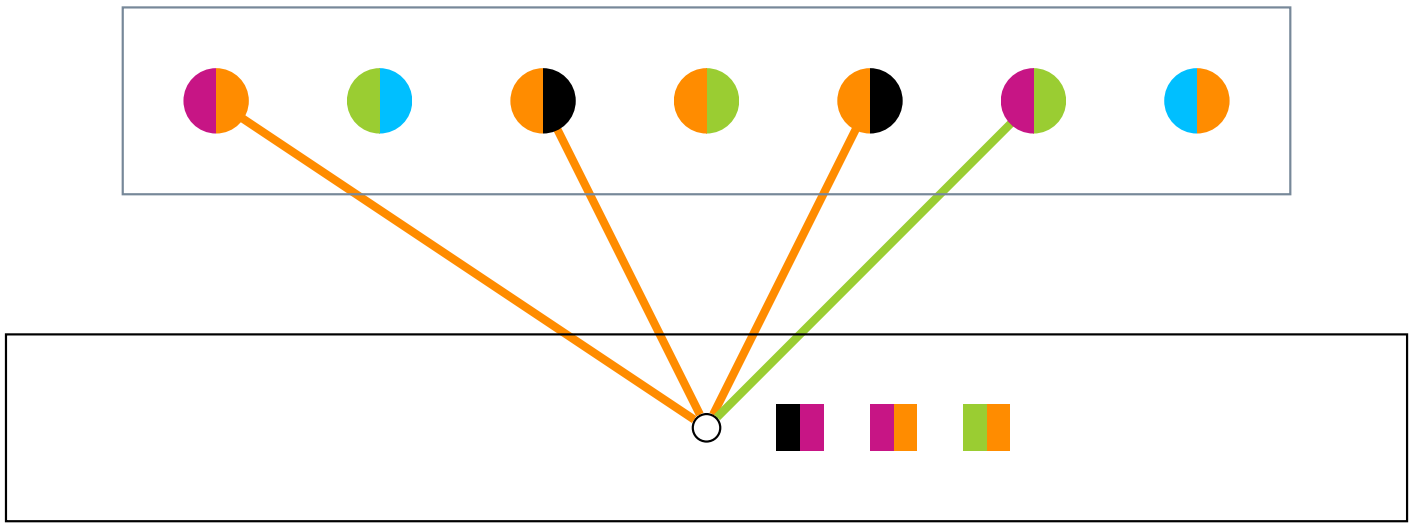
VERTEX COVER



INDEPENDENT SET

COLOR PALETTE

VERTEX COVER



INDEPENDENT SET

As it turns out, there are only two kinds of lists:

As it turns out, there are only two kinds of lists:

1. Those with constant size.
2. Those with a common color.

As it turns out, there are only two kinds of lists:

1. Those with constant size.
   Continue to branch.
2. Those with a common color.

As it turns out, there are only two kinds of lists:

1. Those with constant size.
   Continue to branch.
2. Those with a common color.
   Reduces to a maximum matching problem.

Running time?

Palette

k^{2k}

Palette $\times$ Guess X

$$k^{2k} \cdot 2^k$$

$$\text{Palette} \times \text{Guess } X \times \text{Branching}$$

$$k^{2k} \cdot 2^k \cdot 10^k$$

Palette $\times$ Guess X $\times$ Branching

$$k^{2k} \cdot 2^k \cdot 10^k$$

Overall: $\mathcal{O}^*((20k)^k)$

Other Results

Other Results

1. We show an explicit exponential kernel by the application of some simple reduction rules.

Other Results

1. We show an explicit exponential kernel by the application of some simple reduction rules.
2. We also show NP-hardness and polynomial kernels for restricted graph classes (constant maximum degree, and C_4-free graphs).

Other Results

1. We show an explicit exponential kernel by the application of some simple reduction rules.
2. We also show NP-hardness and polynomial kernels for restricted graph classes (constant maximum degree, and C_4-free graphs).
3. We consider the dual parameter [1] and show a polynomial kernel in this setting.

[1] Can we color with at least $(n - k)$ colors?

Several Open Problems!

Several Open Problems!

1. Can the algorithm be improved to a running time of $\mathcal{O}(c^k)$ for some constant c?

Several Open Problems!

1. Can the algorithm be improved to a running time of $\mathcal{O}(c^k)$ for some constant c?
2. Does the problem admit a polynomial kernel?

Several Open Problems!

1. Can the algorithm be improved to a running time of $\mathcal{O}(c^k)$ for some constant c?
2. Does the problem admit a polynomial kernel?
3. A natural extension would be the above-guarantee version: can we color with at least $(t + k)$ colors, where t is the size of a maximum matching?

Several Open Problems!

1. Can the algorithm be improved to a running time of $\mathcal{O}(c^k)$ for some constant c?
2. Does the problem admit a polynomial kernel?
3. A natural extension would be the above-guarantee version: can we color with at least $(t + k)$ colors, where t is the size of a maximum matching?
4. Is there an explicit FPT algorithm for the dual parameter?

Thank You.