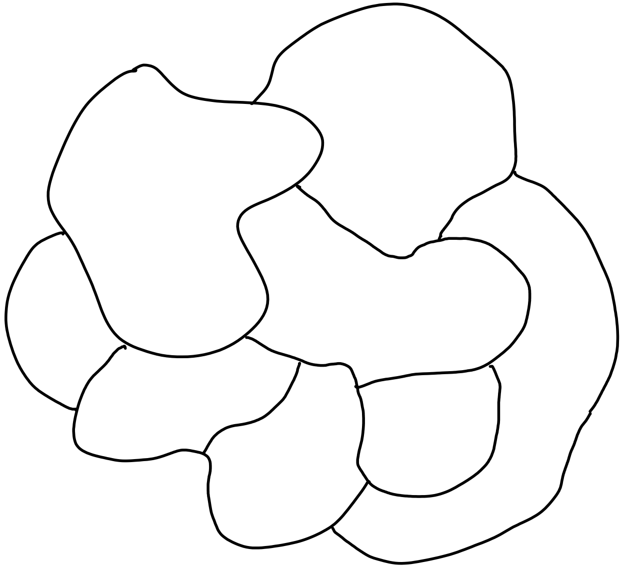
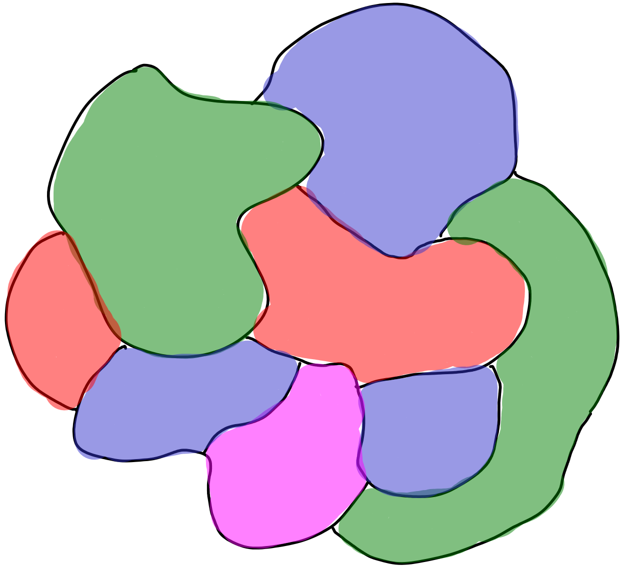
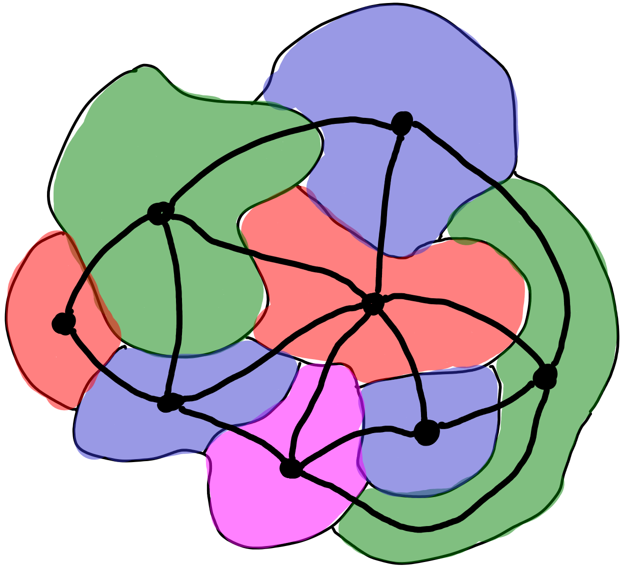
MAP COLORING



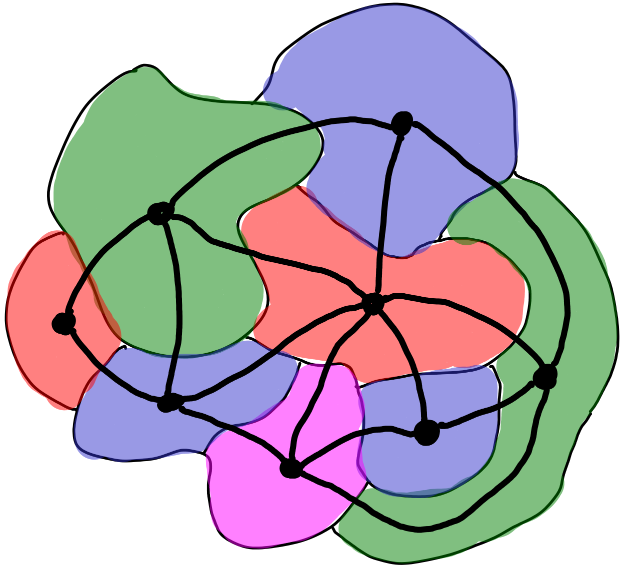
MAP COLORING



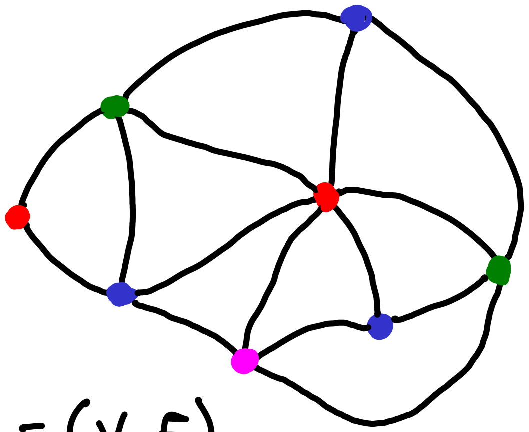
MAP COLORING



MAP COLORING



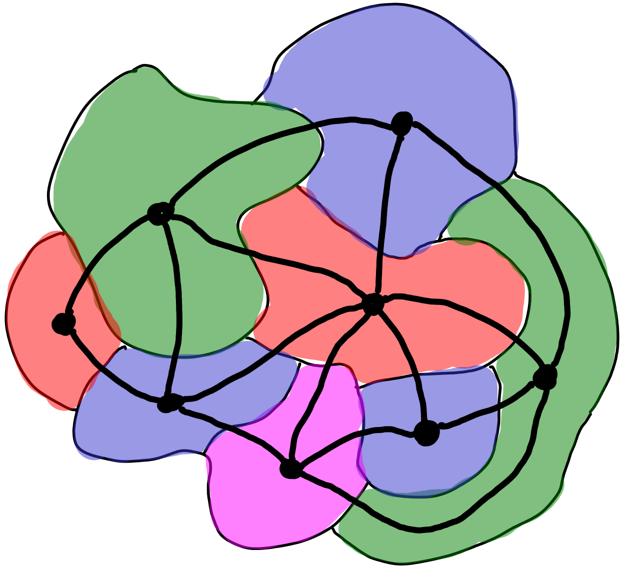
$\Rightarrow$



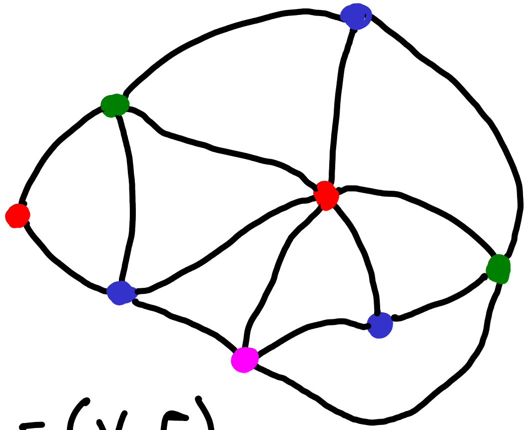
$G = (V, E)$

COLORING G $\rightarrow$ no adjacent vertices get same color

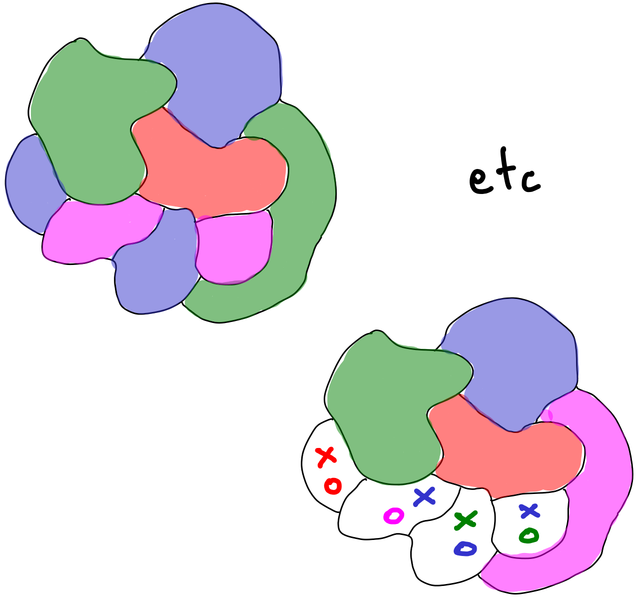
MAP COLORING



$\Rightarrow$

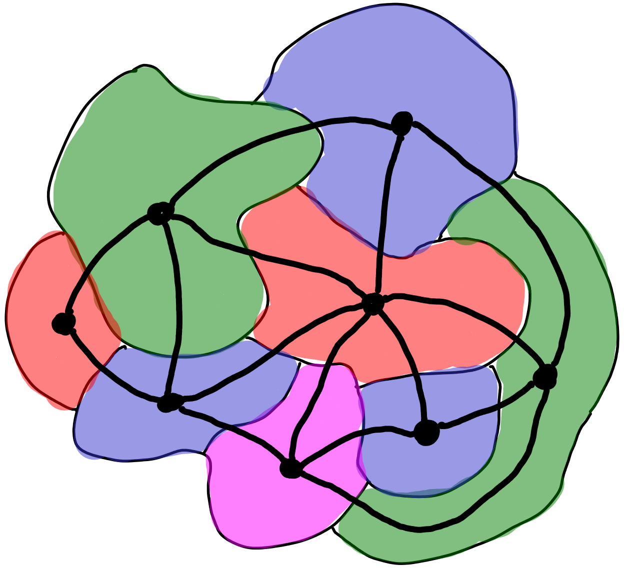


$G = (V, E)$

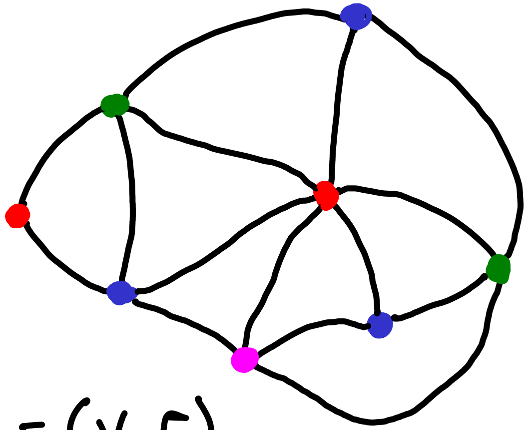


COLORING G → no adjacent vertices get same color

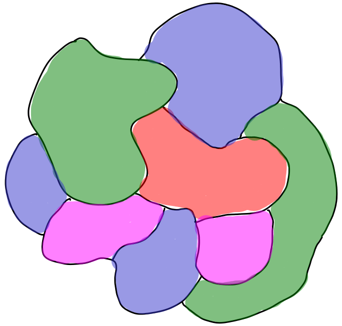
MAP COLORING



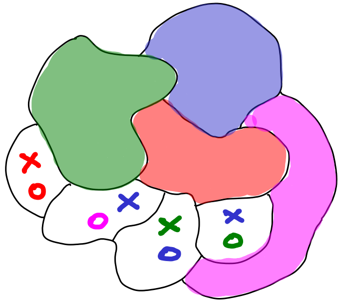
$\Rightarrow$



$G = (V, E)$



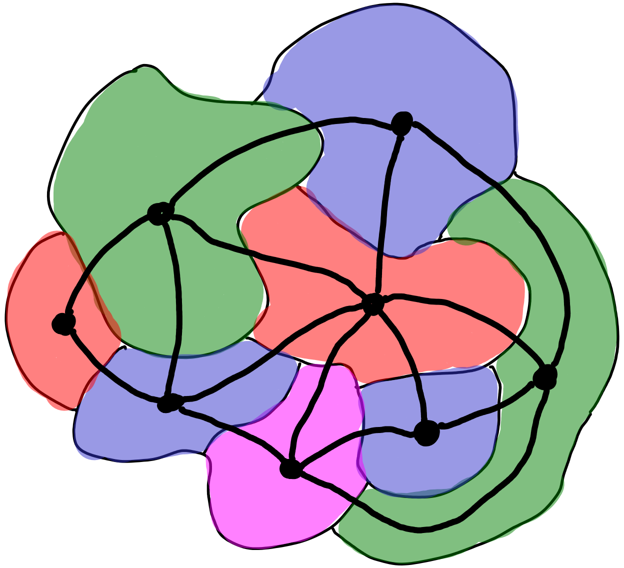
etc



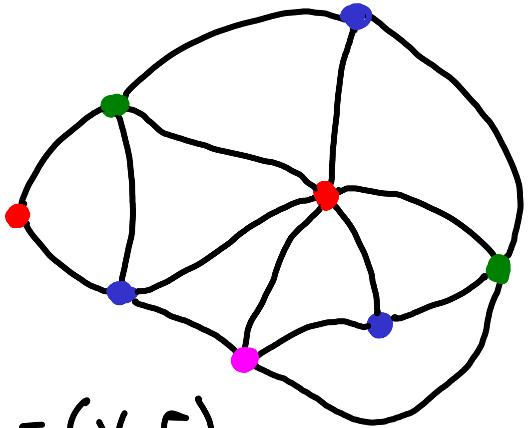
COLORING G $\rightarrow$ no adjacent vertices get same color

G is k-colorable if we can use $\leq k$ colors

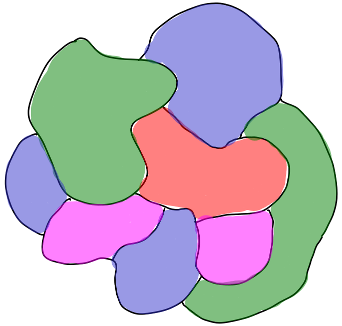
MAP COLORING

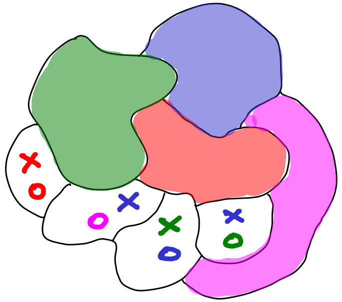


$\Rightarrow$



$G=(V,E)$





etc

COLORING G → no adjacent vertices get same color

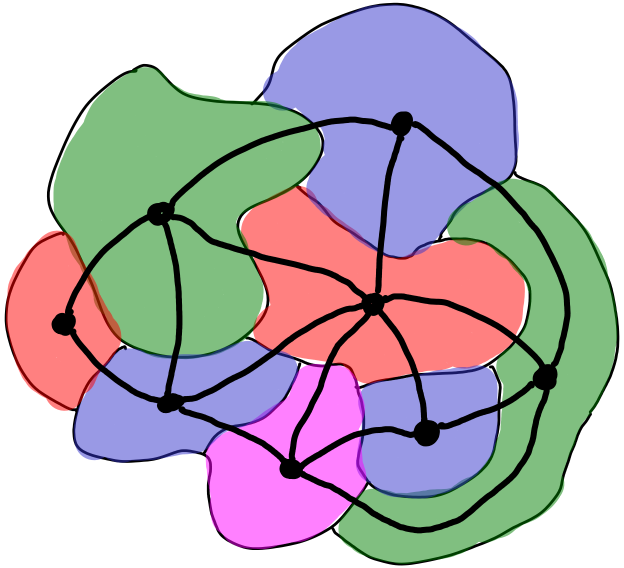
G is k-colorable if we can use $\leq k$ colors

$\chi(G)$: min # colors we can use to color G

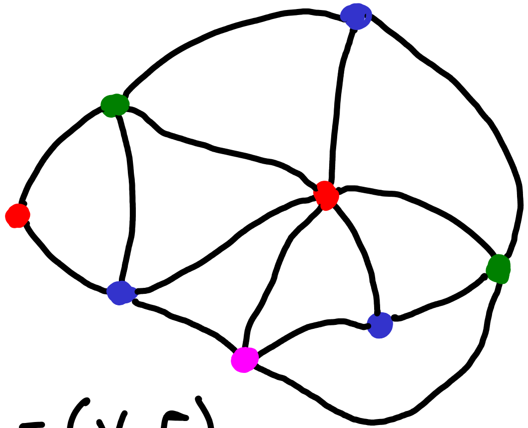
chromatic number

χρῶμα = color

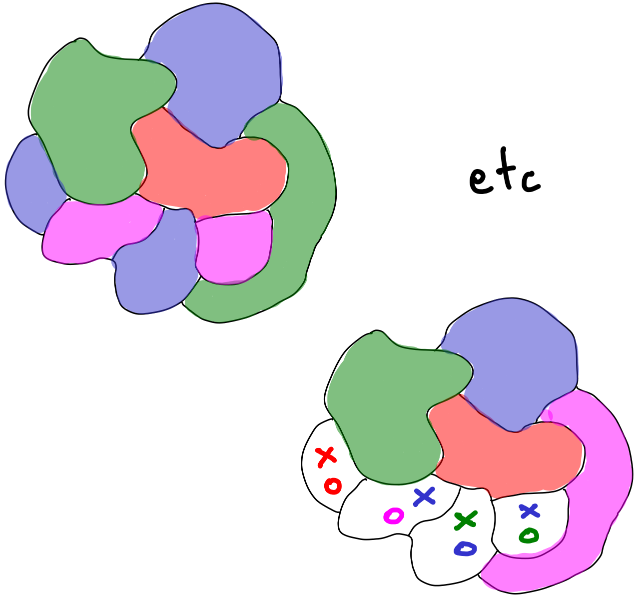
MAP COLORING



$\Rightarrow$



$G=(V,E)$



etc

COLORING G $\rightarrow$ no adjacent vertices get same color

G is k-colorable if we can use $\leq k$ colors

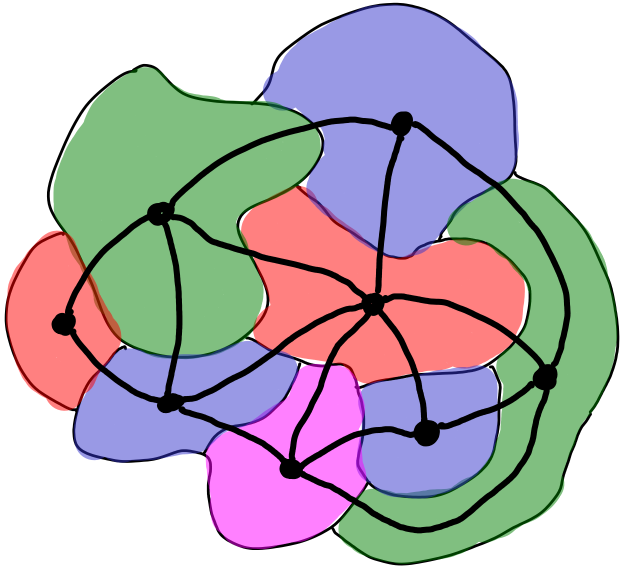
$\chi(G)$: min # colors we can use to color G

chromatic number

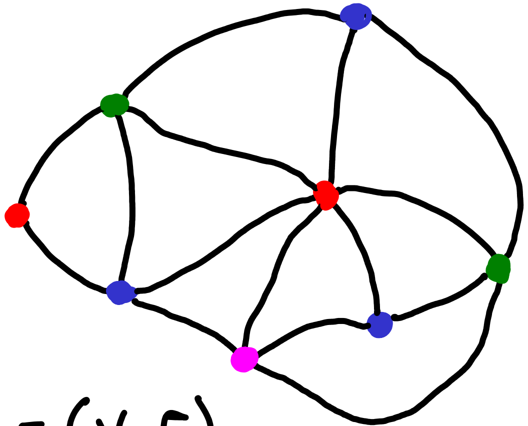
χρῶμα = color

Our map is 4-colorable $\Big\}\ \chi \leq 4$

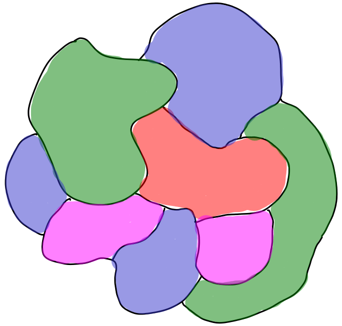
MAP COLORING



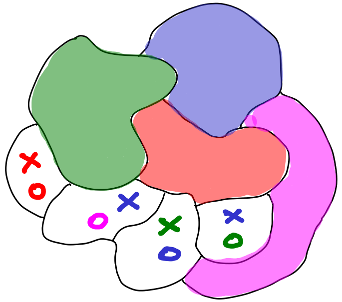
$\Rightarrow$



$G=(V,E)$



etc



COLORING G → no adjacent vertices get same color

G is k-colorable if we can use $\leq k$ colors

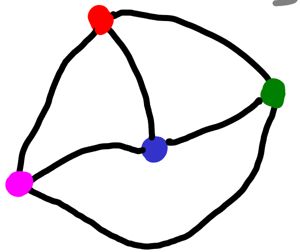
$\chi(G)$: min # colors we can use to color G

chromatic number

χρῶμα = color

Our map is 4-colorable $\Big\}\ \chi \leq 4$

subgraph K_4



...but not 3-colorable

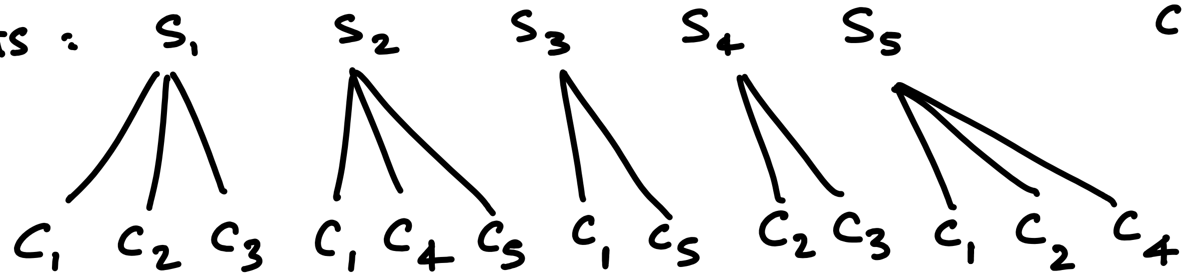
so $\chi \geq 4$

EXAM SCHEDULING

students : s_1 s_2 s_3 s_4 s_5 classes c_1 c_2 c_3 c_4 c_5

EXAM SCHEDULING

students :



classes c_1 c_2 c_3 c_4 c_5

EXAM SCHEDULING

students: s_1 s_2 s_3 s_4 s_5 classes c_1 c_2 c_3 c_4 c_5

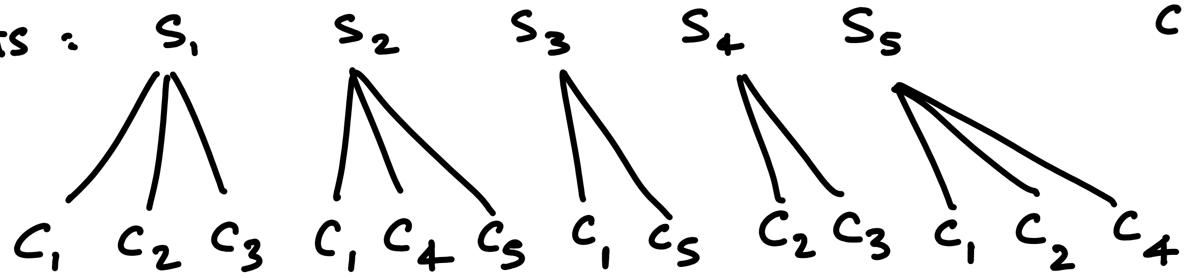
s_1	s_2	s_3	s_4	s_5
c_1 c_2 c_3	c_1 c_4 c_5	c_1 c_5	c_2 c_3	c_1 c_2 c_4

Can't schedule exam simultaneously for classes taken by s_i

Want to minimize exam slots.

EXAM SCHEDULING

students:

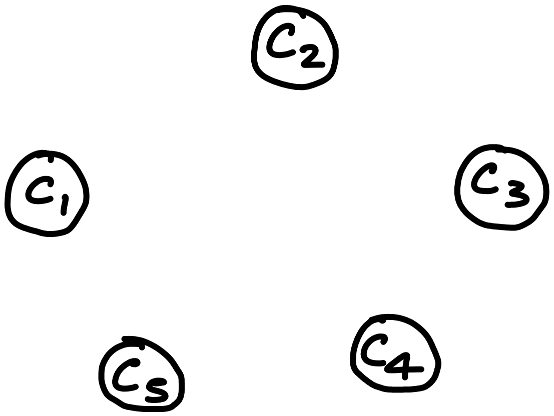


classes c_1 c_2 c_3 c_4 c_5

Can't schedule exam simultaneously for classes taken by s_i

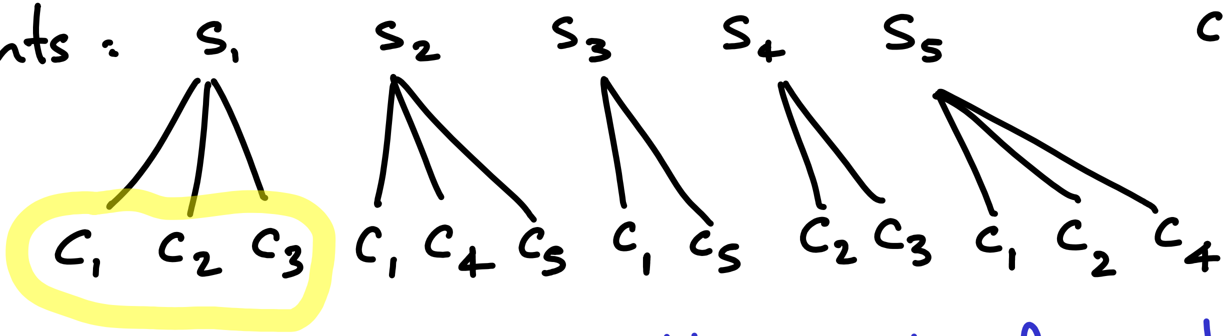
Want to minimize exam slots.

Make G: V = classes E = conflicts



EXAM SCHEDULING

students:

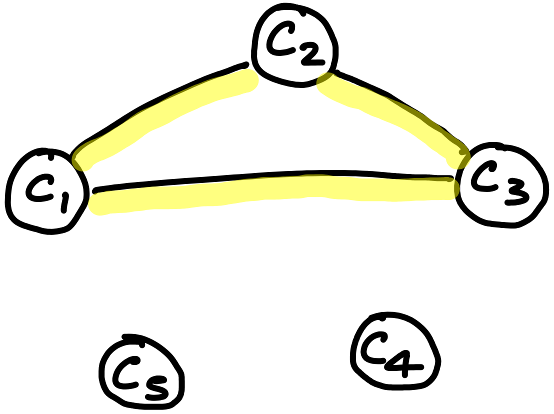


classes c_1 c_2 c_3 c_4 c_5

Can't schedule exam simultaneously for classes taken by s_i

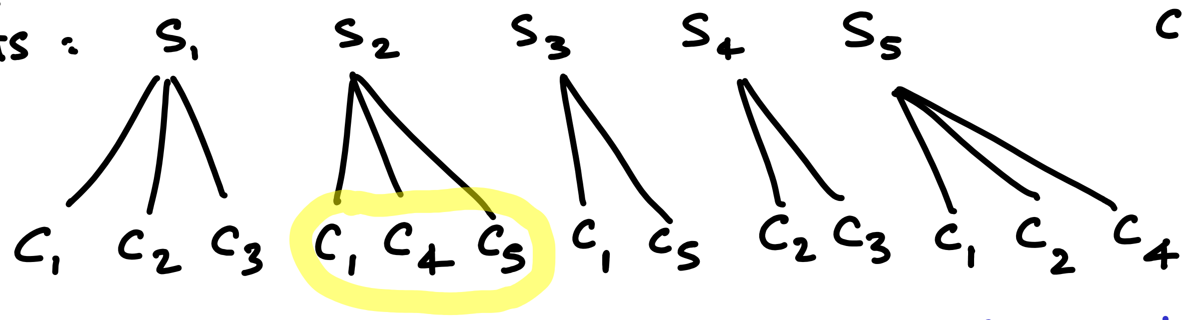
Want to minimize exam slots.

Make G: V = classes E = conflicts



EXAM SCHEDULING

students :

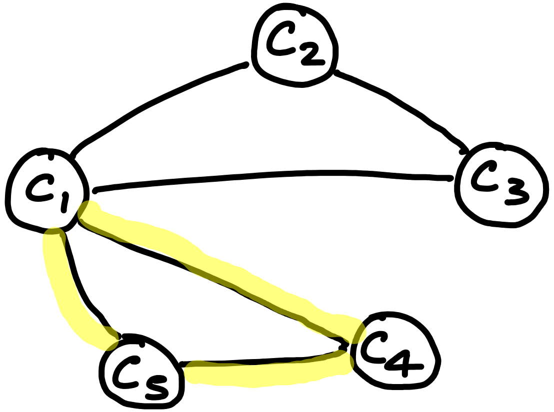


classes c_1 c_2 c_3 c_4 c_5

Can't schedule exam simultaneously for classes taken by s_i

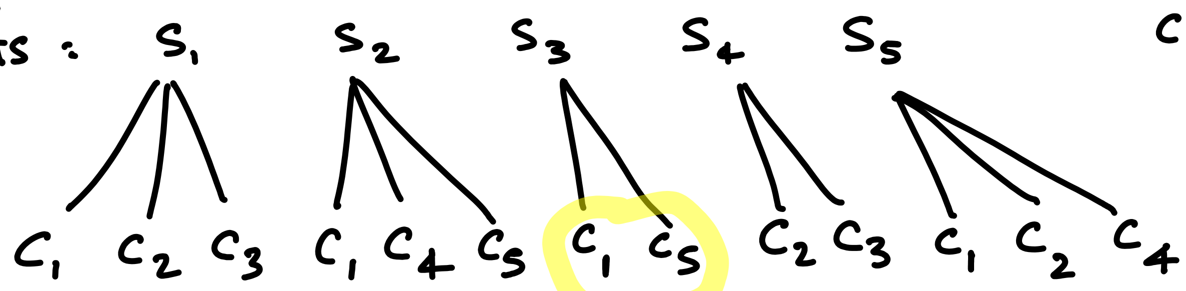
Want to minimize exam slots.

Make G: V = classes E = conflicts



EXAM SCHEDULING

students :

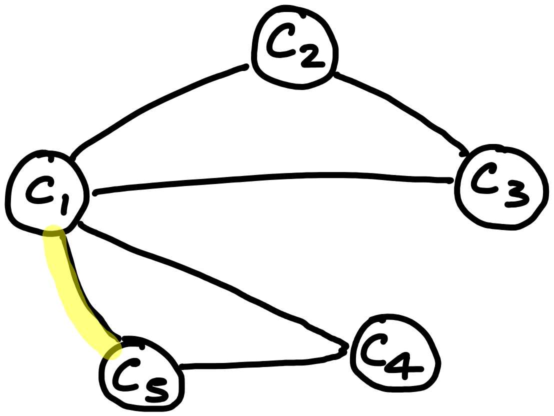


classes c_1 c_2 c_3 c_4 c_5

Can't schedule exam simultaneously for classes taken by s_i

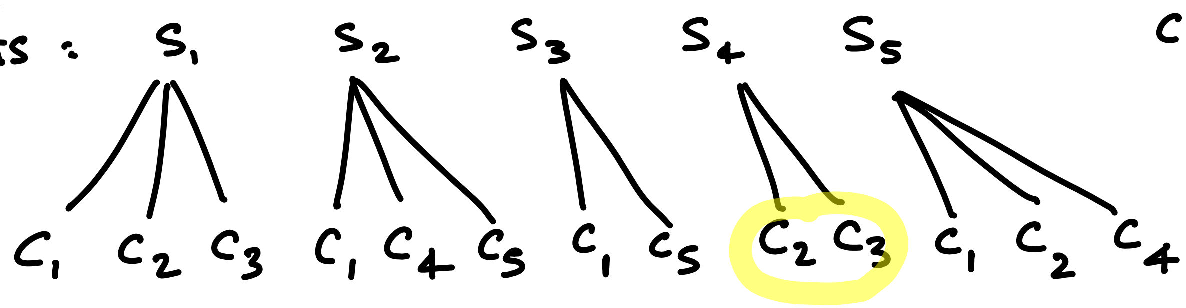
Want to minimize exam slots.

Make G: V = classes E = conflicts



EXAM SCHEDULING

students :

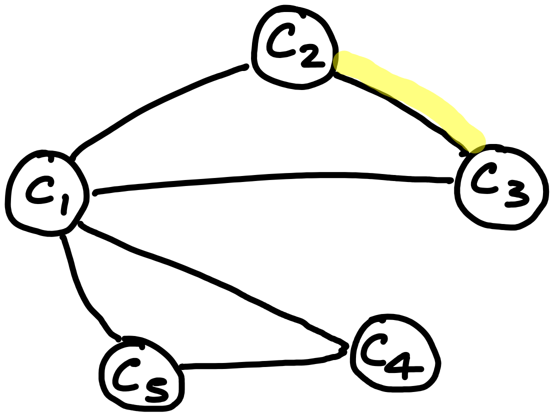


classes c_1 c_2 c_3 c_4 c_5

Can't schedule exam simultaneously for classes taken by s_i

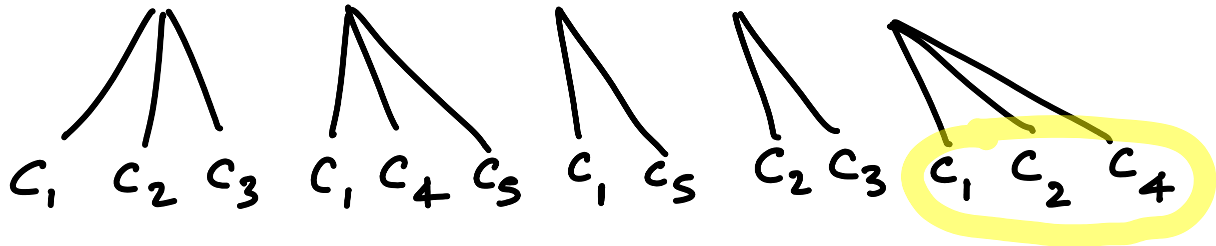
Want to minimize exam slots.

Make G: V = classes E = conflicts



EXAM SCHEDULING

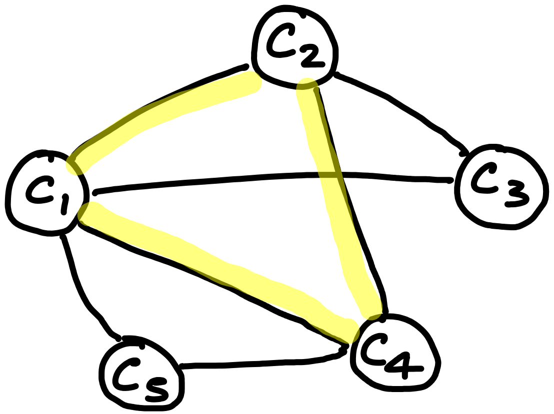
students: s_1 s_2 s_3 s_4 s_5 classes c_1 c_2 c_3 c_4 c_5



Can't schedule exam simultaneously for classes taken by s_i

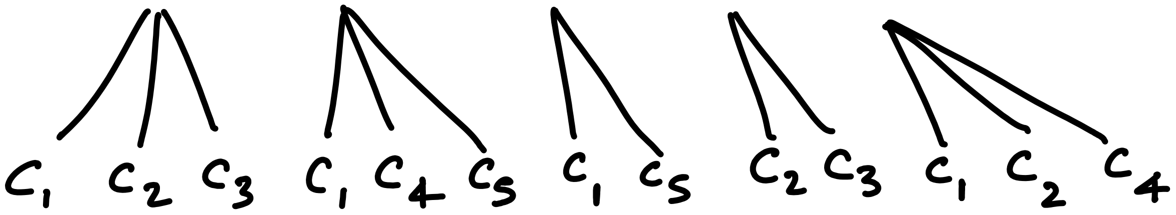
Want to minimize exam slots.

Make G: V = classes E = conflicts



EXAM SCHEDULING

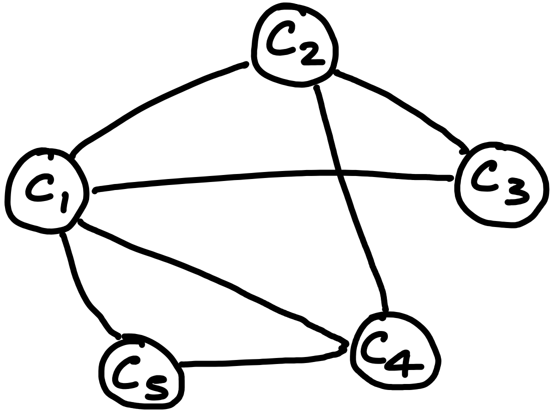
students: s_1 s_2 s_3 s_4 s_5 classes c_1 c_2 c_3 c_4 c_5



Can't schedule exam simultaneously for classes taken by s_i

Want to minimize exam slots.

Make G: V = classes E = conflicts



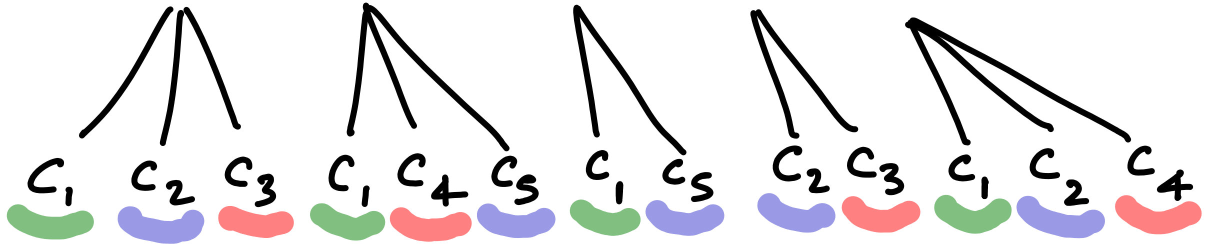
Colors = slots (minimize colors)

If no edge has same color at endpoints,

then no 2 classes are in same slot

EXAM SCHEDULING

students : s_1 s_2 s_3 s_4 s_5 classes c_1 c_2 c_3 c_4 c_5

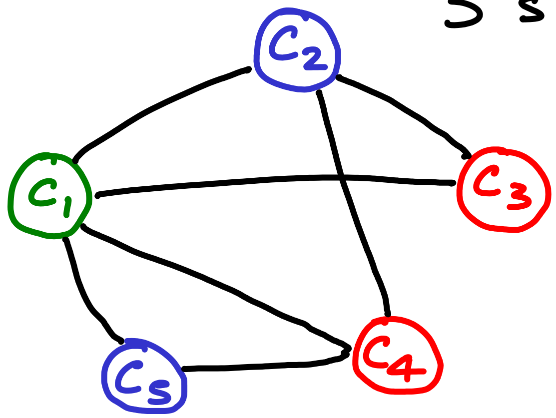


Can't schedule exam simultaneously for classes taken by s_i

Want to minimize exam slots.

Make G: V = classes E = conflicts

3 slots

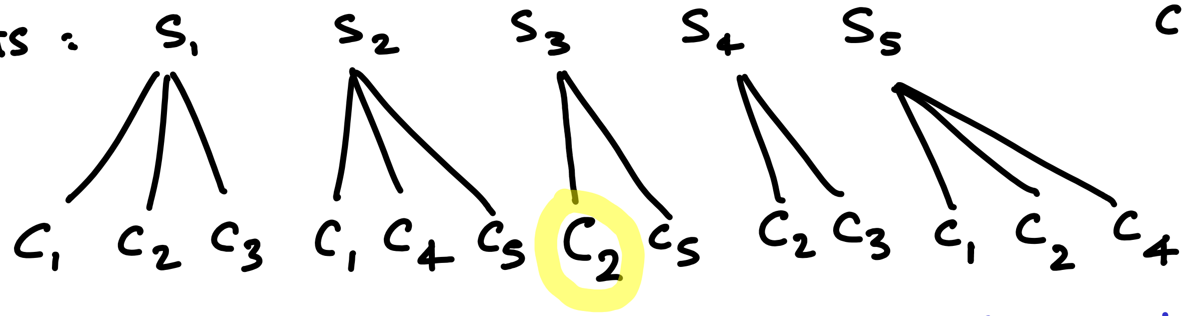


Colors = slots (minimize colors)

If no edge has same color at endpoints,
then no 2 classes are in same slot

EXAM SCHEDULING

students :



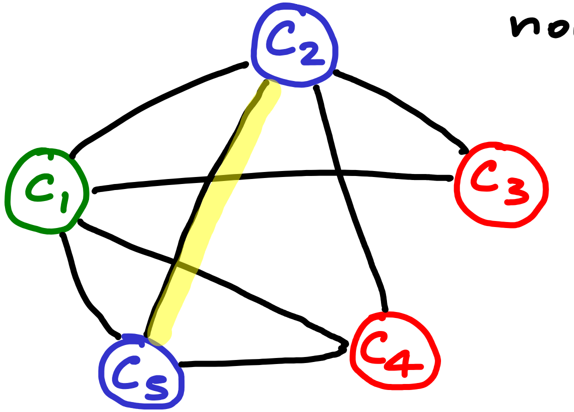
classes c_1 c_2 c_3 c_4 c_5

Can't schedule exam simultaneously for classes taken by s_i

Want to minimize exam slots.

Make G: V = classes E = conflicts

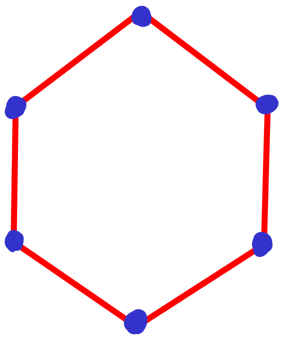
Need 4 now



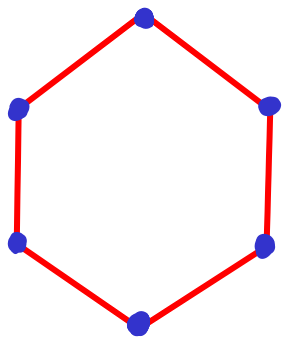
Colors = slots (minimize colors)

If no edge has same color at endpoints,

then no 2 classes are in same slot

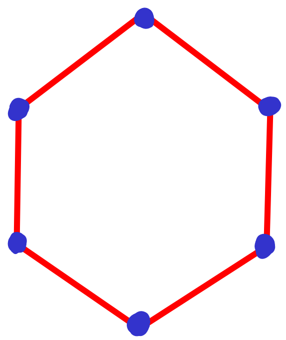


What is χ for cycles?



What is χ for cycles?

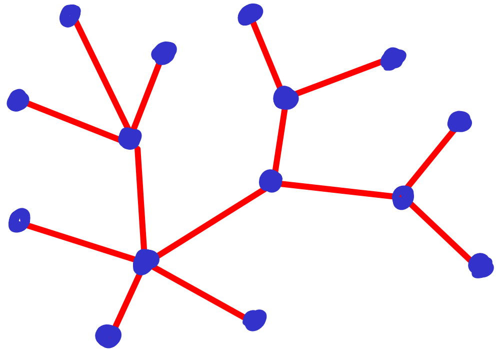
$$\chi = 2 \quad \text{if} \quad V \text{ even}$$
$$= 3 \quad \text{if} \quad V \text{ odd}$$



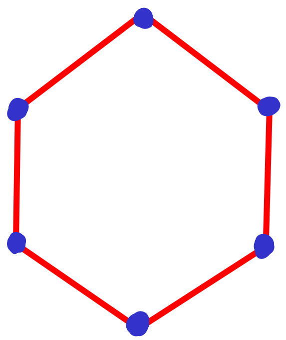
What is χ for cycles?

$\chi = 2$ if V even

$= 3$ if V odd



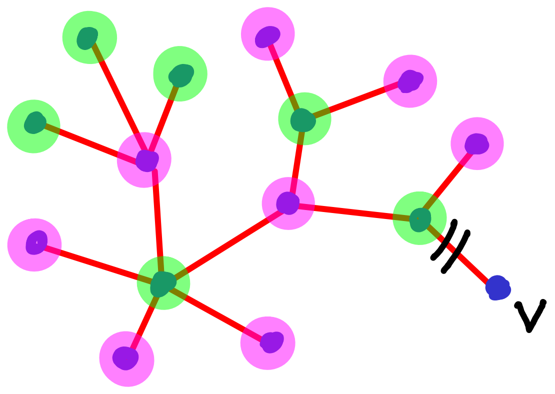
For trees?



What is χ for cycles?

$\chi = 2$ if V even

$= 3$ if V odd

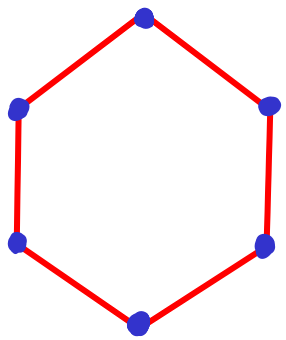


For trees?

Remove a leaf, v.

2-color the rest...

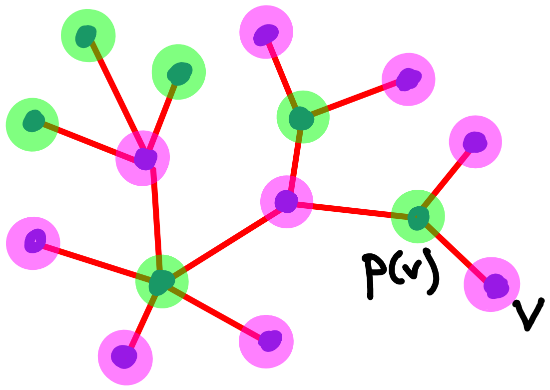
...



What is χ for cycles?

$\chi = 2$ if V even

$= 3$ if V odd



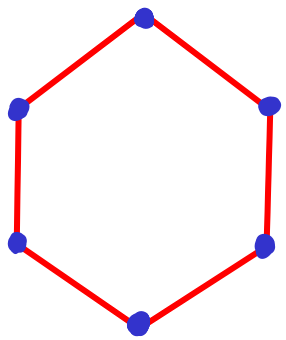
For trees?

Remove a leaf, v.

2-color the rest.

Color v opposite of p(v)

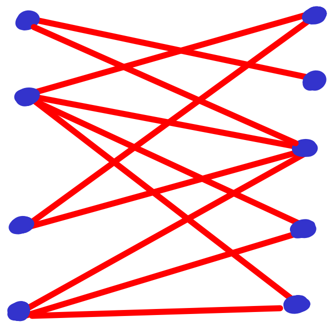
$\chi = 2$



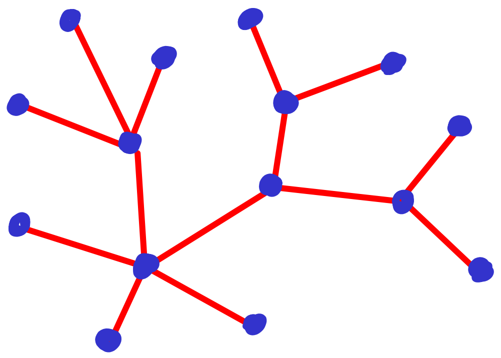
What is χ for cycles?

$\chi = 2$ if V even

$= 3$ if V odd



For bipartite graphs?

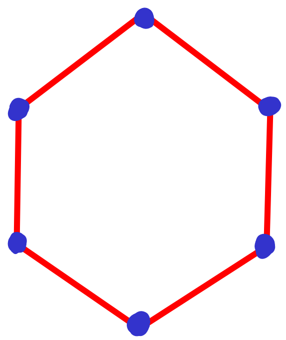


For trees?

Remove a leaf, v. 2-color the rest.

Color v opposite of p(v)

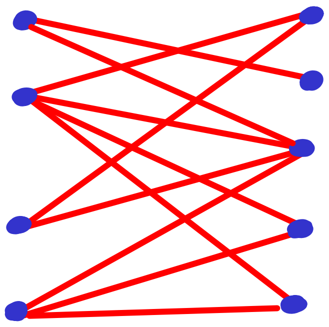
$\chi = 2$



What is χ for cycles?

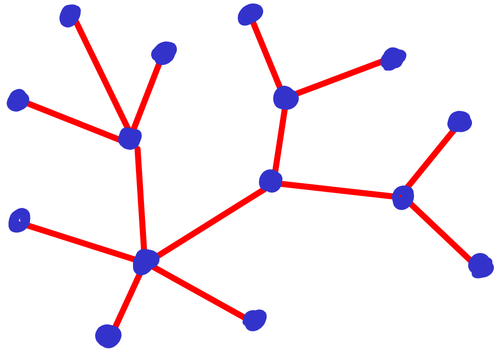
$\chi = 2$ if V even

$= 3$ if V odd



For bipartite graphs?

$\chi = 2$



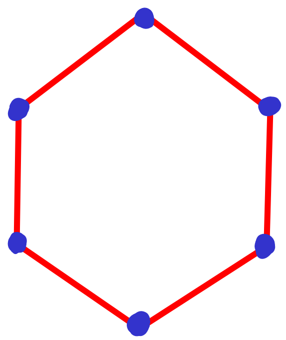
For trees?

Remove a leaf, v.

2-color the rest.

Color v opposite of $p(v)$

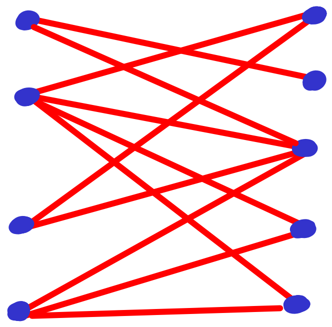
$\chi = 2$



What is χ for cycles?

$\chi = 2$ if V even

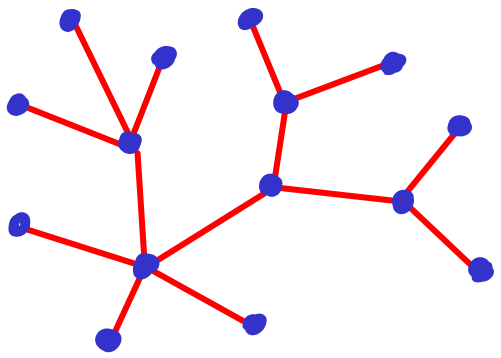
$= 3$ if V odd



For bipartite graphs?

$\chi = 2$

In fact if $\chi(G) = 2$ then G is bipartite by definition



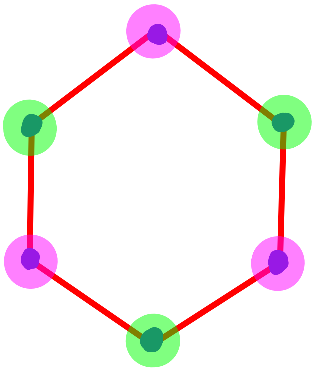
For trees?

Remove a leaf, v. 2-color the rest.

Color v opposite of $p(v)$

$\chi = 2$

(trees are bipartite)



What is χ for cycles?

$\chi = 2$ if V even

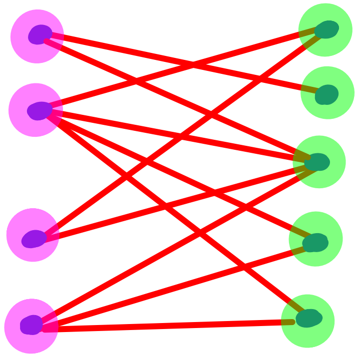
$= 3$ if V odd

Claim:

G is bipartite

$\Longleftrightarrow$

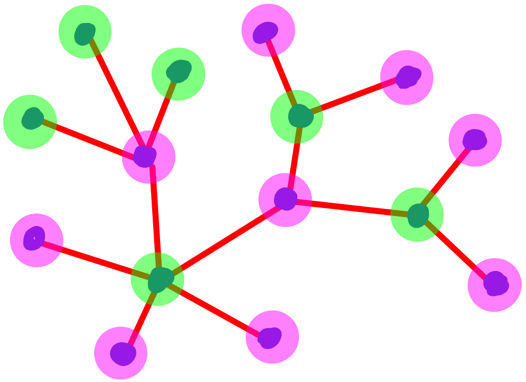
G contains no odd cycle



For bipartite graphs?

$\chi = 2$

In fact if $\chi(G) = 2$ then G is bipartite by definition



For trees?

Remove a leaf, v.

2-color the rest.

Color v opposite of p(v)

$\chi = 2$

(trees are bipartite)

If G has $n > 1$ vertices, trivial bounds: $2 \leq \chi \leq n$

(K_n)

If G has $n > 1$ vertices, trivial bounds: $2 \leq \chi \leq n$

(K_n)

What can χ be if max degree of $G = \Delta$?

If G has $n > 1$ vertices, trivial bounds : $2 \leq \chi \leq n$
(K_n)

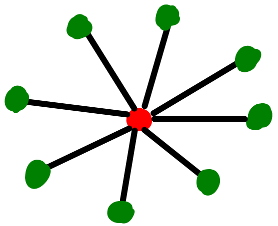
What can χ be if max degree of $G = \Delta$?

$K_n \rightarrow$

If G has $n > 1$ vertices, trivial bounds : $2 \leq \chi \leq n$ (K_n)

What can χ be if max degree of $G = \Delta$?

$K_n \rightarrow \Delta = n-1$: $\chi = \Delta + 1$



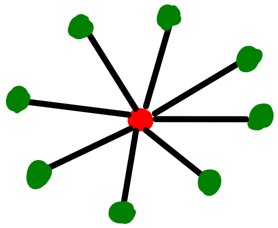
$\Delta = n-1$: $\chi = 2$

If G has $n > 1$ vertices, trivial bounds: $2 \leq \chi \leq n$

(K_n)

What can χ be if max degree of $G = \Delta$?

$K_n \rightarrow \Delta = n-1 \; : \; \chi = \Delta + 1$



$\Delta = n-1 \; : \; \chi = 2$

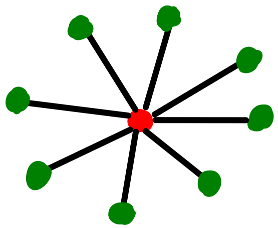
Can we have $\chi \gg \Delta$? (need $n \gg \Delta$)

If G has $n > 1$ vertices, trivial bounds : $2 \leq \chi \leq n$

(K_n)

What can χ be if max degree of $G = \Delta$?

$K_n \to \Delta = n-1$: $\chi = \Delta + 1$



$\Delta = n-1$: $\chi = 2$

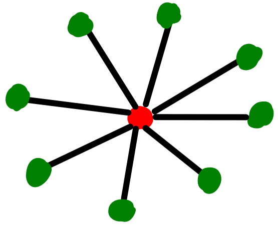
Can we have $\chi \gg \Delta$? (need $n \gg \Delta$)

Claim $\chi \leq \Delta + 1$

If G has $n > 1$ vertices, trivial bounds : $2 \leq \chi \leq n$
(K_n)

What can χ be if max degree of $G = \Delta$?

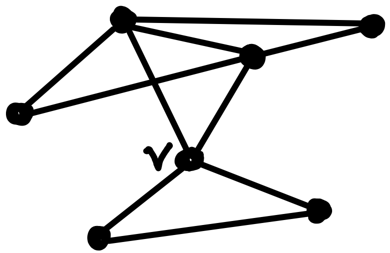
$K_n \to \Delta = n-1 \; : \; \chi = \Delta + 1$



$\Delta = n-1 \; : \; \chi = 2$

Can we have $\chi \gg \Delta$? (need $n \gg \Delta$)

Claim $\chi \leq \Delta + 1$

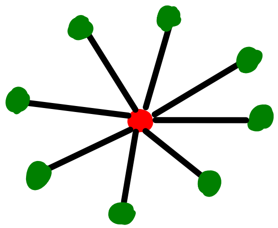


$\Delta = 4$

If G has $n > 1$ vertices, trivial bounds : $2 \leq \chi \leq n$ (K_n)

What can χ be if max degree of $G = \Delta$?

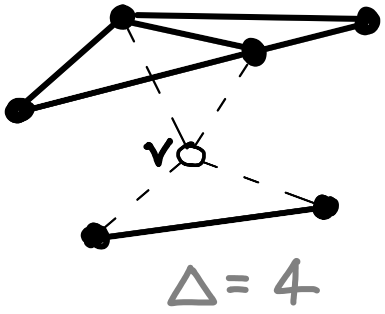
$K_n \rightarrow \Delta = n-1$: $\chi = \Delta + 1$



$\Delta = n-1$: $\chi = 2$

Can we have $\chi \gg \Delta$? (need $n \gg \Delta$)

Claim $\chi \leq \Delta + 1$

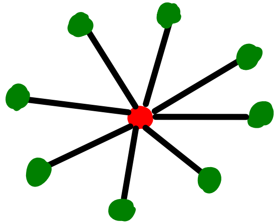


- Remove any vertex v.

If G has $n > 1$ vertices, trivial bounds : $2 \leq \chi \leq n$
(K_n)

What can χ be if max degree of $G = \Delta$?

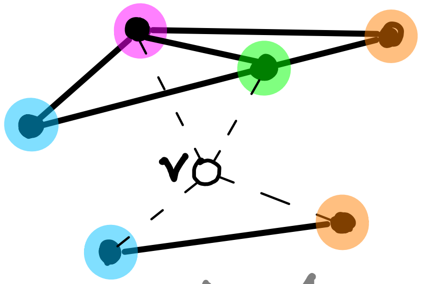
$K_n \rightarrow \Delta = n-1 \; : \; \chi = \Delta + 1$



$\Delta = n-1 \; : \; \chi = 2$

Can we have $\chi \gg \Delta$? (need $n \gg \Delta$)

Claim $\chi \leq \Delta + 1$



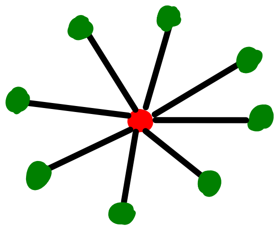
$\Delta = 4$

- Remove any vertex v.
- Color G-v by induction.

If G has $n > 1$ vertices, trivial bounds: $2 \le \chi \le n$ (K_n)

What can χ be if max degree of $G = \Delta$?

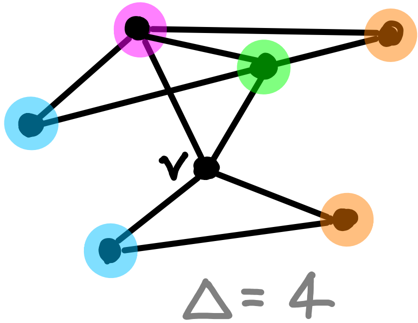
$K_n \to \Delta = n-1$: $\chi = \Delta + 1$



$\Delta = n-1$: $\chi = 2$

Can we have $\chi \gg \Delta$? (need $n \gg \Delta$)

Claim $\chi \le \Delta + 1$

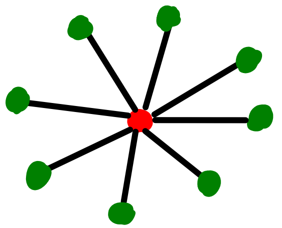


- Remove any vertex v.
- Color $G-v$ by induction.
- Re-insert v.

If G has $n > 1$ vertices, trivial bounds: $2 \le \chi \le n$ (K_n)

What can χ be if max degree of $G = \Delta$?

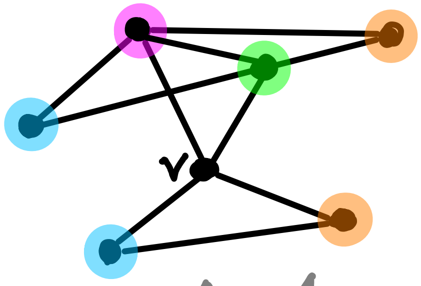
$K_n \to \Delta = n-1 : \chi = \Delta + 1$



$\Delta = n-1 : \chi = 2$

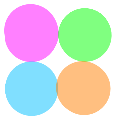
Can we have $\chi \gg \Delta$? (need $n \gg \Delta$)

Claim $\chi \le \Delta + 1$



$\Delta = 4$

- Remove any vertex v.
- Color $G-v$ by induction.
- Re-insert v.
- v has $\le \Delta$ neighbors.

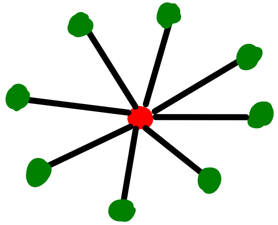


If G has $n > 1$ vertices, trivial bounds : $2 \le \chi \le n$

(K_n)

What can χ be if max degree of $G = \Delta$?

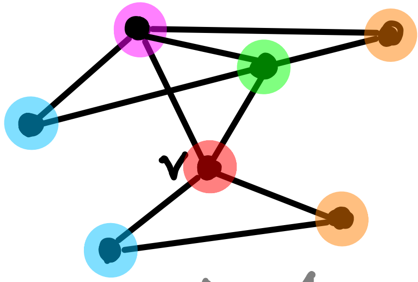
$K_n \rightarrow \Delta = n-1 \; : \; \chi = \Delta + 1$



$\Delta = n-1 \; : \; \chi = 2$

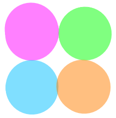
Can we have $\chi \gg \Delta$? (need $n \gg \Delta$)

Claim $\chi \le \Delta + 1$



$\Delta = 4$

- Remove any vertex v.
- Color $G - v$ by induction.
- Re-insert v.
- v has $\le \Delta$ neighbors.

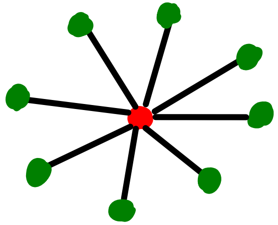


- Use color $\underline{\Delta + 1}$ for v.

If G has $n > 1$ vertices, trivial bounds : $2 \le \chi \le n$
(K_n)

What can χ be if max degree of $G = \Delta$?

$K_n \to \Delta = n-1 \ : \ \chi = \Delta + 1$

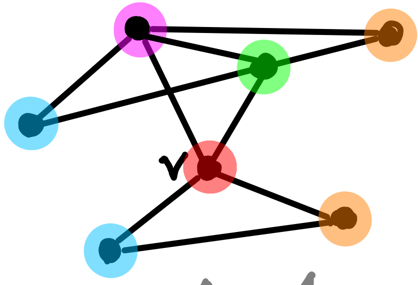


$\Delta = n-1 \ : \ \chi = 2$

Can we have $\chi \gg \Delta$? (need $n \gg \Delta$)

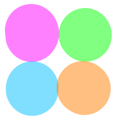
Claim $\chi \le \Delta + 1$

- Incrementally "add" vertices.
- When adding vertex v, look at all neighboring colors.
- Always have ≥ 1 color available.

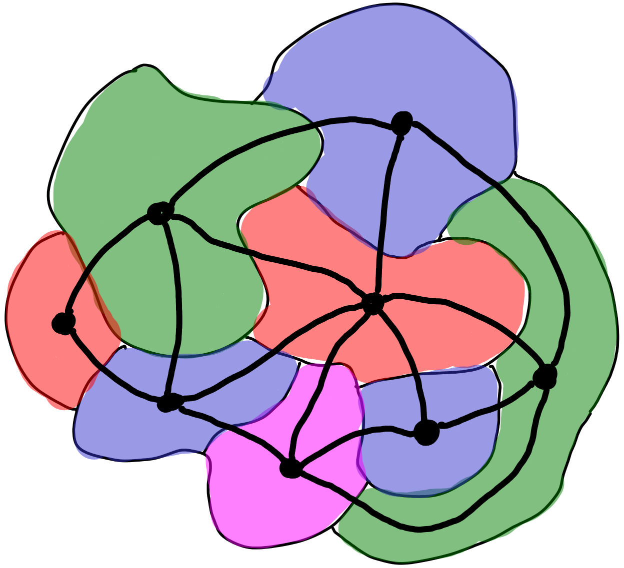


$\Delta = 4$

- Remove any vertex v.
- Color $G - v$ by induction.
- Re-insert v.
- v has $\le \Delta$ neighbors.
- Use color $\Delta + 1$ for v.



COLORING PLANAR GRAPHS (like map duals)



COLORING PLANAR GRAPHS (like map duals)

Claim: $\chi \leq 6$ ... trivial if $V \leq 6$

COLORING PLANAR GRAPHS (like map duals)

Claim: $\chi \leq 6$ ... trivial if $V \leq 6$

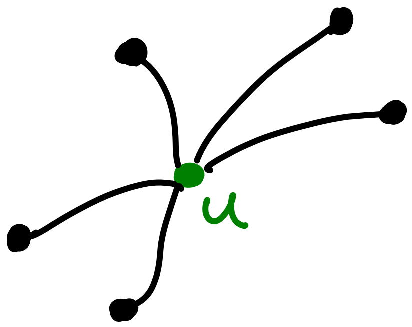
We know planar graphs have a vertex w/ degree ≤ 5 (Euler)

COLORING PLANAR GRAPHS (like map duals)

Claim: $\chi \leq 6$ ... trivial if $V \leq 6$

We know planar graphs have a vertex w/ degree ≤ 5

Given planar G s.t. $V > 6$ & $u \in G$, $d(u) \leq 5$



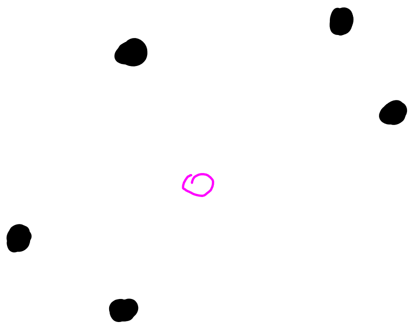
COLORING PLANAR GRAPHS (like map duals)

Claim: $\chi \leq 6$ ... trivial if $V \leq 6$

We know planar graphs have a vertex w/ degree ≤ 5

Given planar G s.t. $V > 6$ & $u \in G$, $d(u) \leq 5$: look at $G - u$

still planar



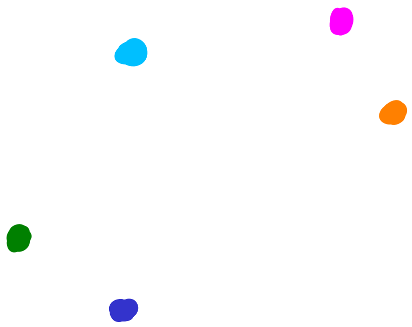
COLORING PLANAR GRAPHS (like map duals)

Claim: $\chi \leq 6$ ... trivial if $V \leq 6$

We know planar graphs have a vertex w/ degree ≤ 5

Given planar G s.t. $V > 6$ & $u \in G$, $d(u) \leq 5$: look at $G - u$

Assume by induction that $G - u$ is 6-colorable



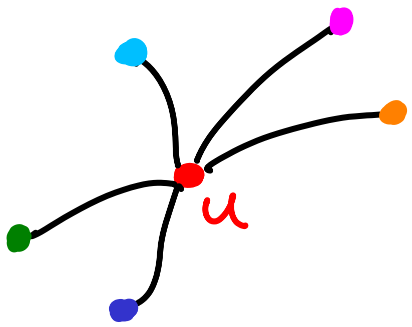
COLORING PLANAR GRAPHS (like map duals)

Claim: $\chi \leq 6$ ... trivial if $V \leq 6$

We know planar graphs have a vertex w/ degree ≤ 5

Given planar G s.t. $V > 6$ & $u \in G$, $d(u) \leq 5$: look at $G-u$

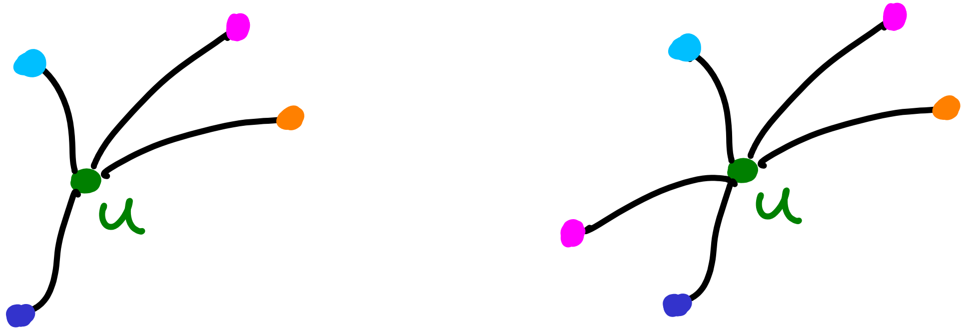
Assume by induction that $G-u$ is 6-colorable



re-insert u : give it a color not used by neighbors $\square$

Claim: $\chi \leq 5$ ... trivial if ???

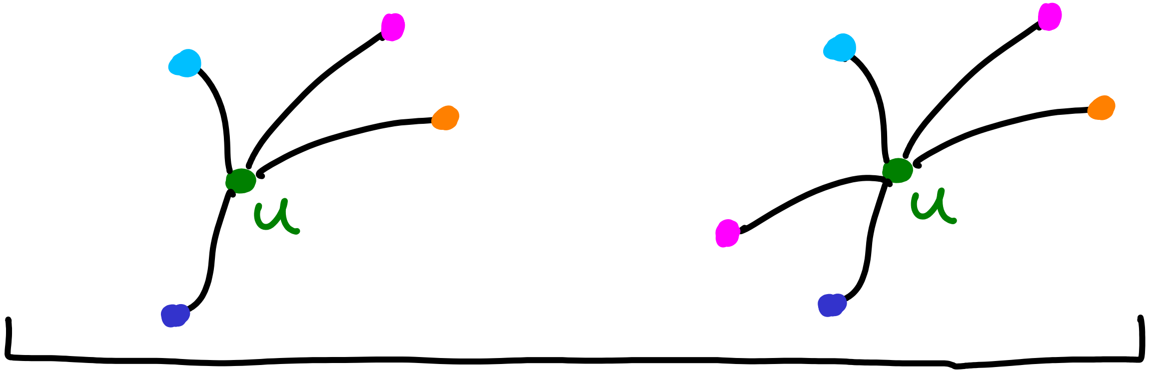
Claim: $\chi \leq 5$ ... trivial if $V \leq 5$



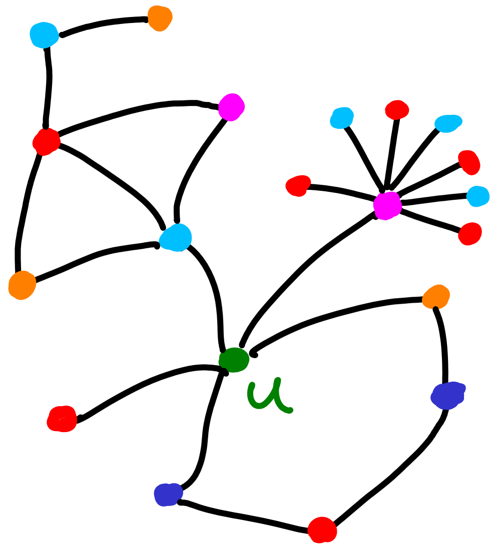
Also trivial if neighbors use < 5 colors

Claim: $\chi \leq 5$ ... trivial if $V \leq 5$

Use induction & $d(u) \leq 5$ again

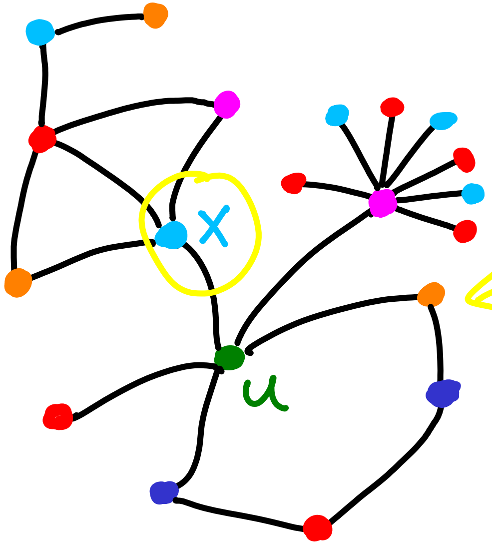


Also trivial if neighbors use < 5 colors



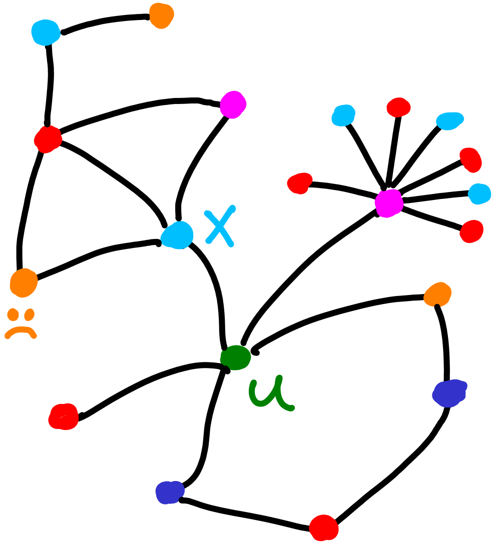
Consider any embedding of G

We need a neighbor of u to change color



Try to change x from ● to ●

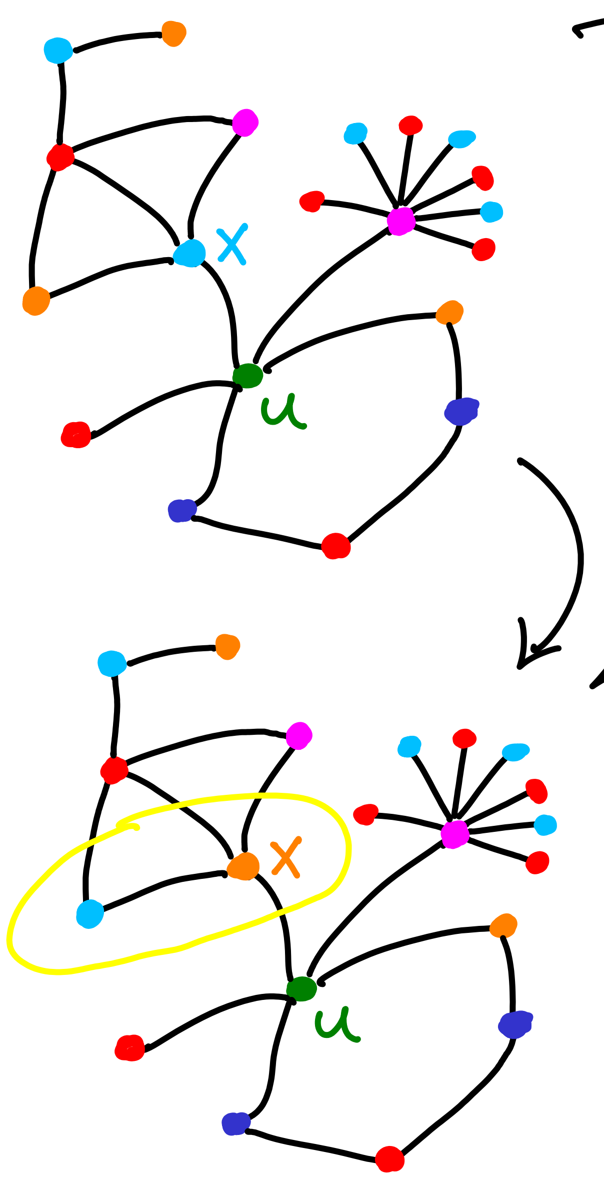
[specifically skipping 2 over in adj(u)]



Try to change x from ● to ●

[specifically skipping 2 over in adj(u)]

↳ This works if x has no ● neighbors

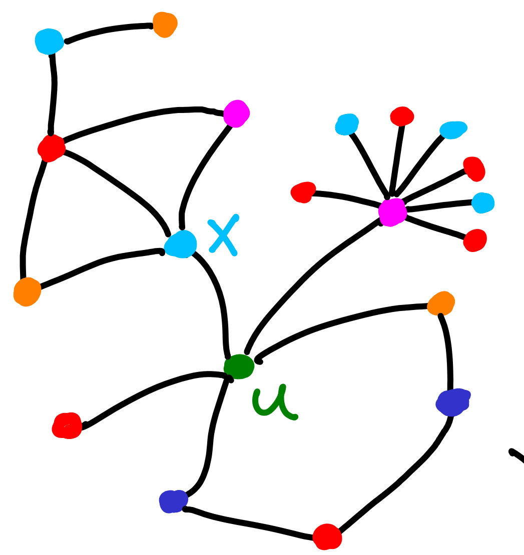


Try to change x from ● to ●

[specifically skipping 2 over in adj(u)]

↳ This works if x has no ● neighbors

↳ else, swap colors on the connected component of the subgraph of G that contains only colors ●● and x.

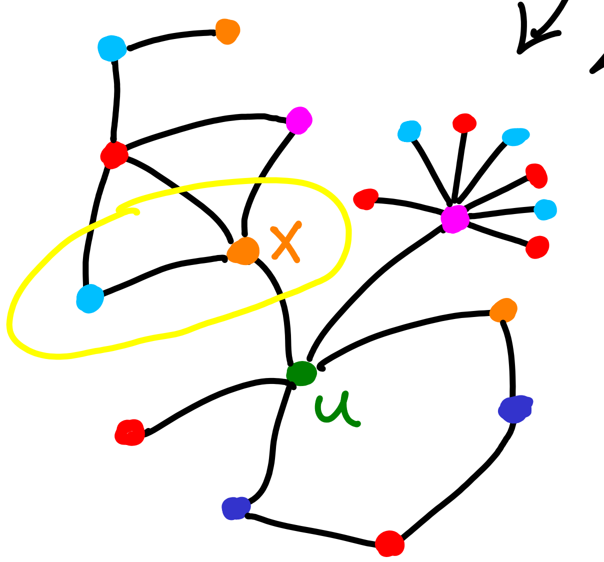


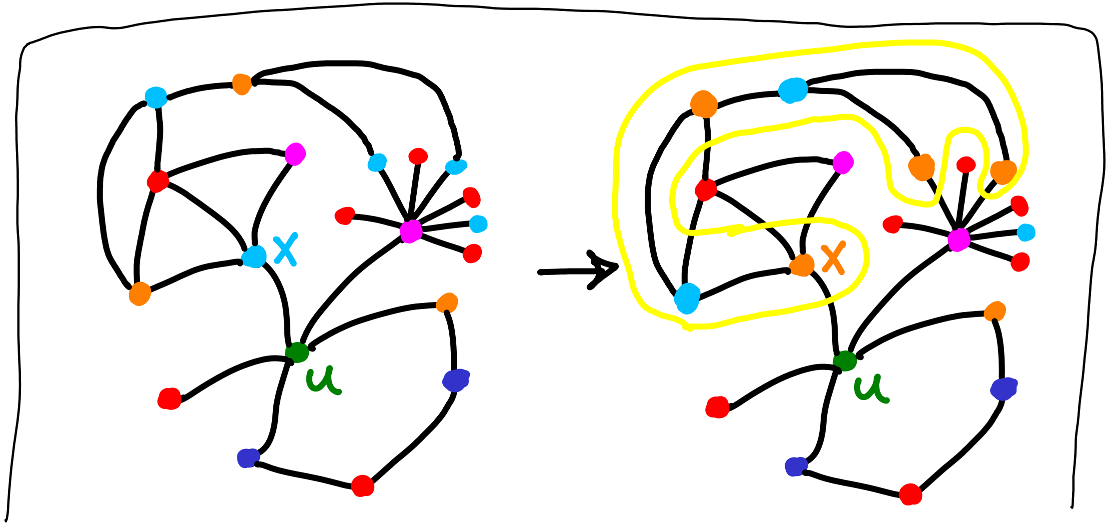
Try to change x from ● to ●

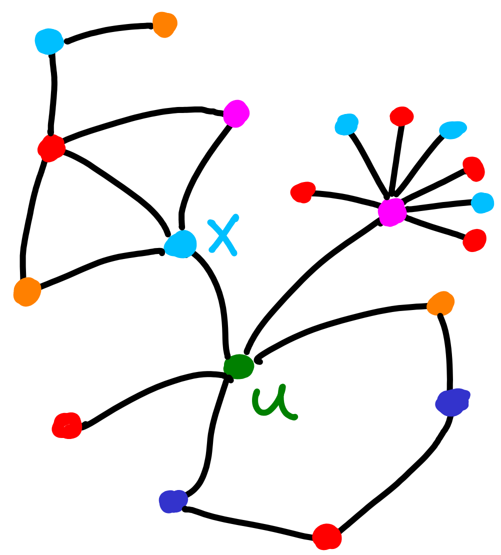
[specifically skipping 2 over in adj(u)]

↳ This works if x has no ● neighbors

↳ else, swap colors on the connected component of the subgraph of G that contains only colors ●● and x.







Try to change x from ● to ●

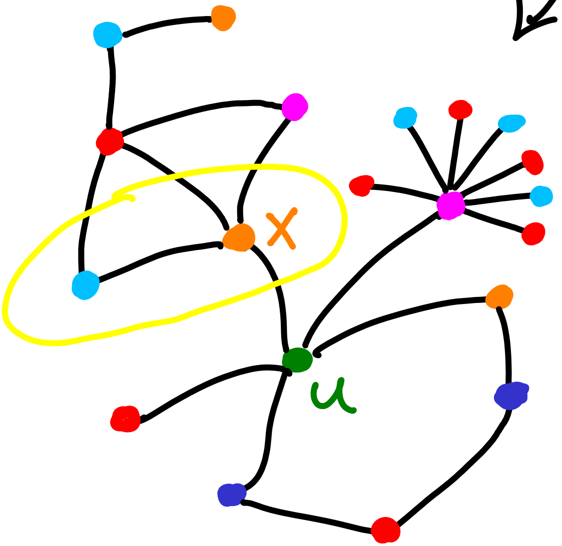
[specifically skipping 2 over in adj(u)]

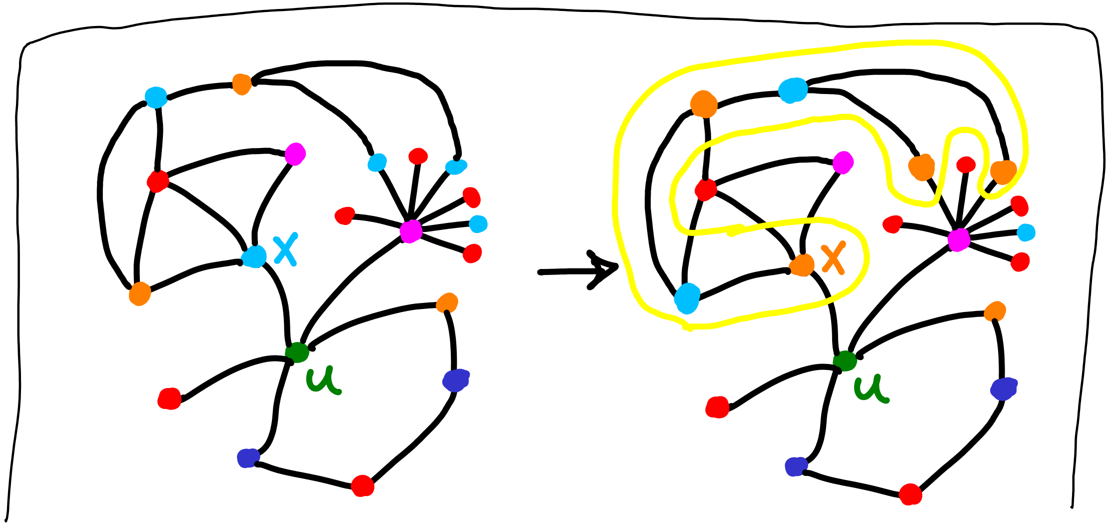
↳ This works if x has no ● neighbors

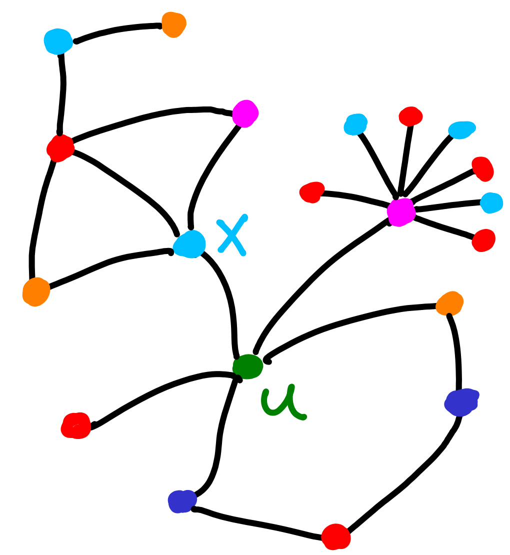
↳ else, swap colors on the connected component of the subgraph of G that contains only colors ●● and x.

ONE PROBLEM

?







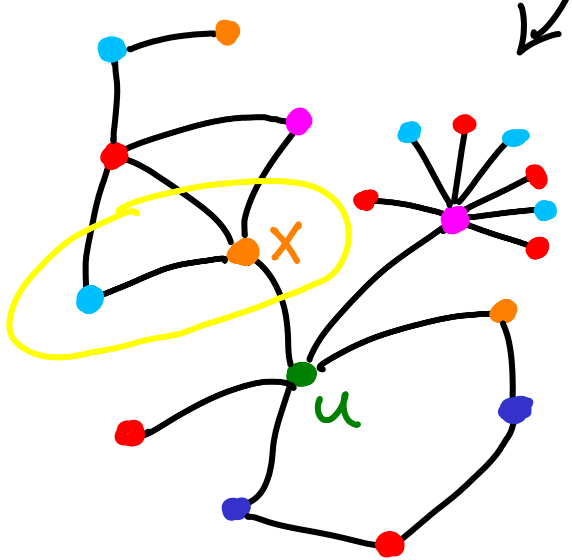
Try to change x from ● to ●

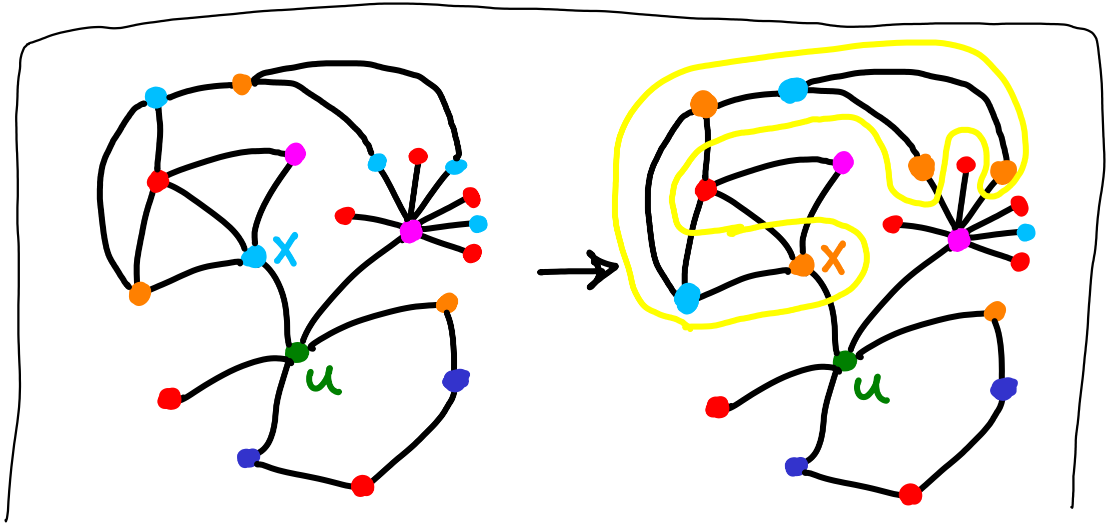
[specifically skipping 2 over in adj(u)]

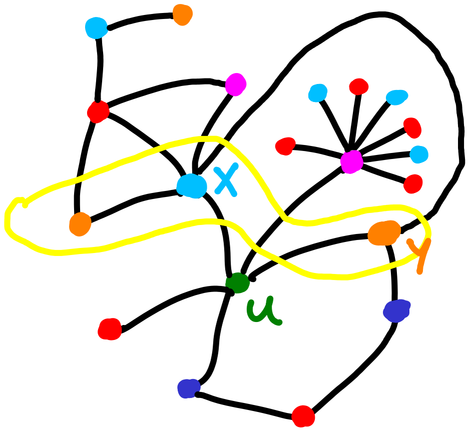
↳ This works if x has no ● neighbors

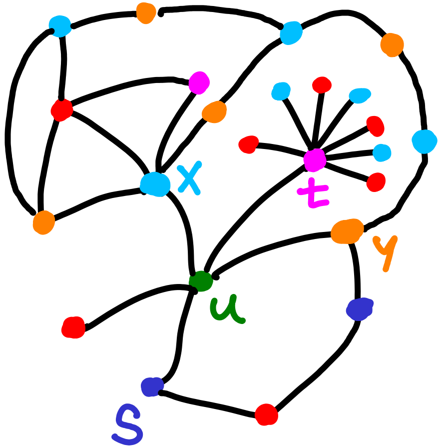
↳ else, swap colors on the connected component of the subgraph of G that contains only colors ●● and x.

ONE PROBLEM

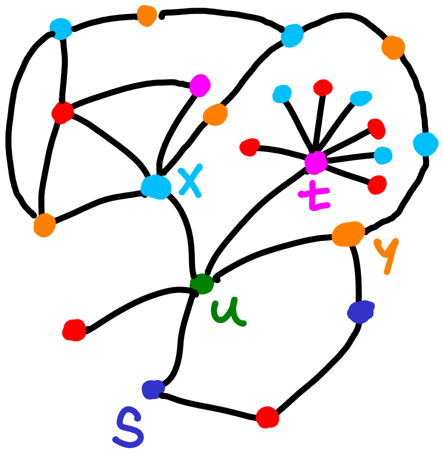








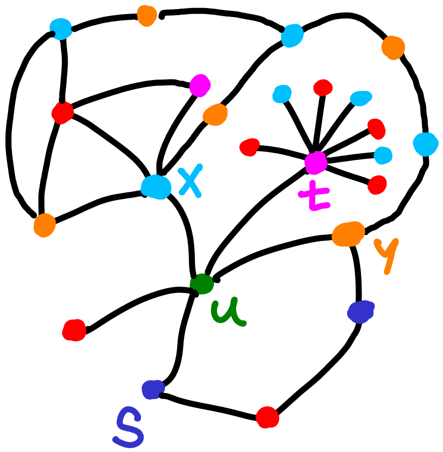
The only bad case involves a path from x to y that alternates x ● ● ● ● ... y



The only bad case involves a path from x to y that alternates x ● ● ● ● … y

Together with u the path forms a cycle surrounding the ● t neighbor of u.

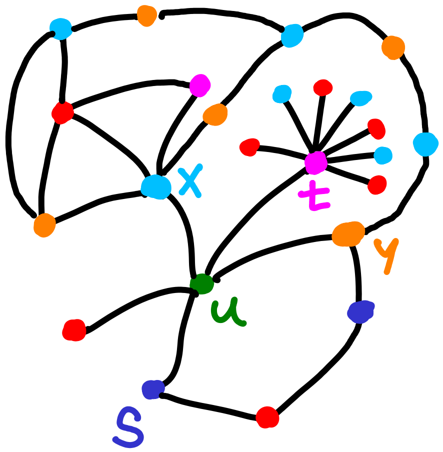
So, … ?



The only bad case involves a path from x to y that alternates x ● ● ● ●... y

Together with u the path forms a cycle surrounding the ● neighbor of u.

Restart the entire procedure using s & t instead of x & y.

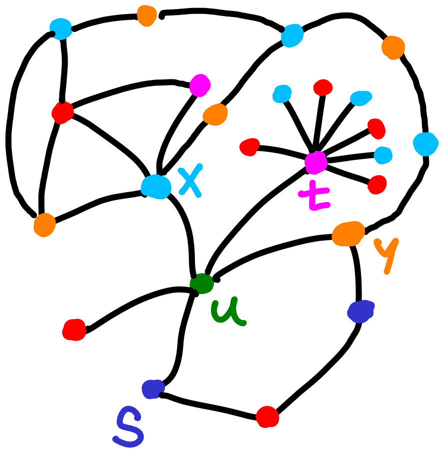


The only bad case involves a path from x to y that alternates x●●●●... y

Together with u the path forms a cycle surrounding the ● neighbor of u.

Restart the entire procedure using s & t instead of x & y.

The only way to fail is if there is a path s●●●●t

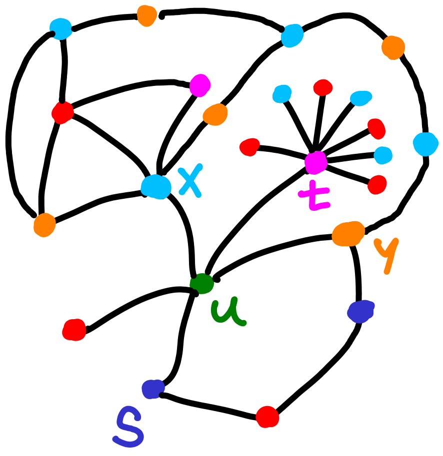


The only bad case involves a path from x to y that alternates x ● ● ● ●... y

Together with u the path forms a cycle surrounding the ● neighbor of u.

Restart the entire procedure using s & t instead of x & y.

The only way to fail is if there is a path s ● ● ● ● t but this would have to cross x ● ● ● ●... y



The only bad case involves a path from x to y
that alternates x ● ● ● ●… y

Together with u the path forms a cycle
surrounding the ● neighbor of u.

Restart the entire procedure using s & t instead of x & y.

The only way to fail is if there is a path s ● ● ● ● t
but this would have to cross x ● ● ● ●… y

Impossible: This is a plane drawing

∎

Planar graphs:

6-coloring: $\sim$ trivial

5-coloring: short elegant proof

Planar graphs:

6-coloring: ~ trivial

5-coloring: short elegant proof

4-coloring:
- unsolved from ≤ 1850 until 1977
- proof involved ~2000 cases solved by computer

Planar graphs:

6-coloring: ~ trivial

5-coloring: short elegant proof

4-coloring:
- unsolved from ≤ 1850 until 1977
- proof involved ~2000 cases solved by computer

3-coloring:
- clearly not always possible
- if triangle-free then 3-colorable
  (in fact if ≤ 3 triangles)