Chapter 6
Vertex and edge coloring

Prof. Tesler

Math 154
Winter 2020

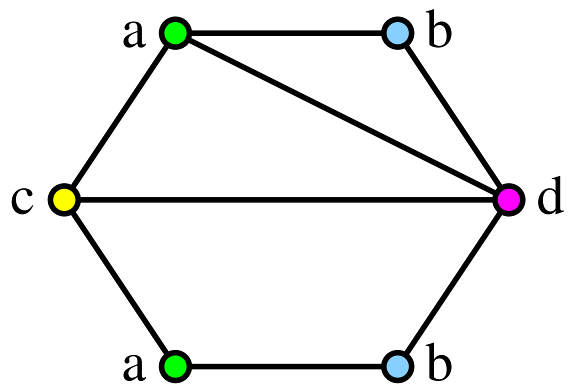
Coloring vertices of a graph

- Let G be a graph and C be a set of *colors*, e.g.,

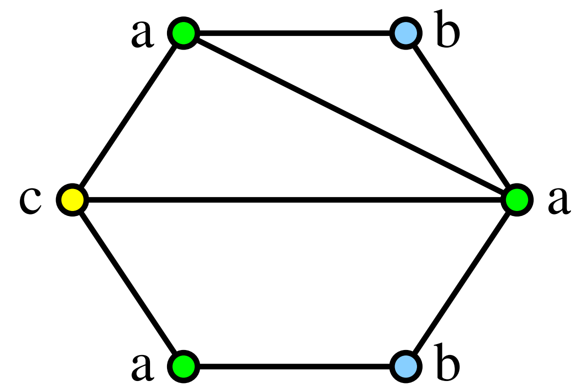
$$C = \{\text{black, white}\} \qquad C = \{a, b\} \qquad C = \{1, 2\}$$

- A *proper coloring* of G by C is to assign a color from C to every vertex, such that in every edge $\{v, w\}$, the vertices v and w have different colors.

Coloring vertices of a graph



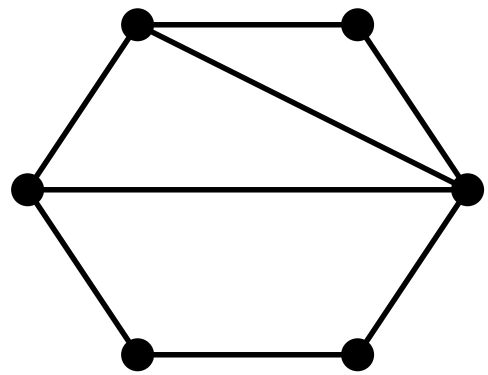
Proper 4-coloring



Not a proper coloring

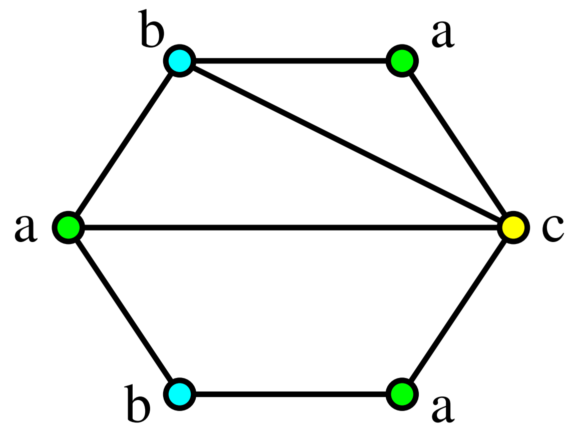
- G is *k-colorable* if it has a proper coloring with k colors (e.g., $C = \{1, 2, \ldots, k\}$). This is also called a *proper k-coloring*.
- In some applications, we literally draw the graph with the vertices in different colors. In proofs and algorithms with a variable number of colors, it's easier to use numbers $1, \ldots, k$.

Color vertices with as few colors $a, b, c, \ldots$ as possible



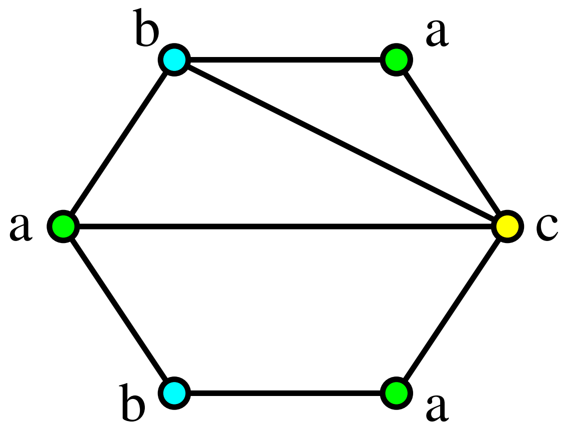
- Color the graph above with as few colors as possible.

Color vertices with as few colors $a, b, c, \ldots$ as possible



- The *chromatic number*, $\chi(G)$, of a graph G is the minimum number of colors needed for a proper coloring of G.
- We also say that G is *k-chromatic* if $\chi(G) = k$.
- Note that if G is k-colorable, then $\chi(G) \leqslant k$.
- This graph is 6-colorable (use a different color on each vertex).
  We also showed it's 4-colorable and it's 3-colorable.
  So far, $\chi(G) \leqslant 3$.

Color vertices with as few colors $a, b, c, \ldots$ as possible

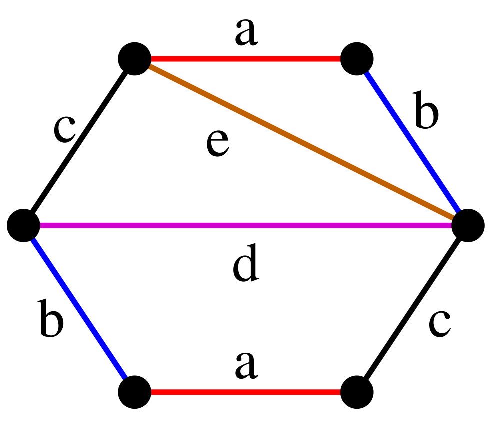


- We've shown it's 3-colorable, so $\chi(G) \leqslant 3$.
- It has a triangle as a subgraph, which requires 3 colors.
  Other vertices may require additional colors, so $\chi(G) \geqslant 3$.
- Combining these gives $\chi(G) = 3$.

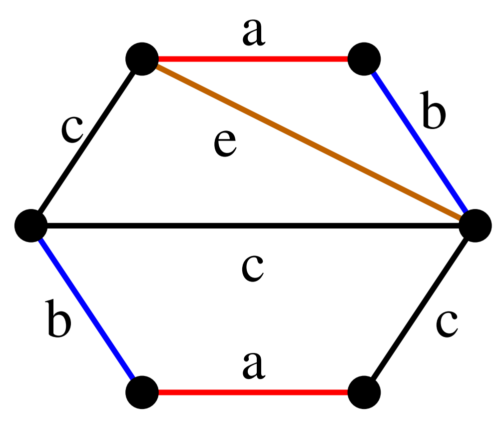
Clique

- A *clique* is a subset X of the vertices s.t. all vertices in X are adjacent to each other. So the induced subgraph $G[X]$ is a complete graph, K_m.
- If G has a clique of size m, its vertices all need different colors, so $\chi(G) \geqslant m$.

Proper edge coloring



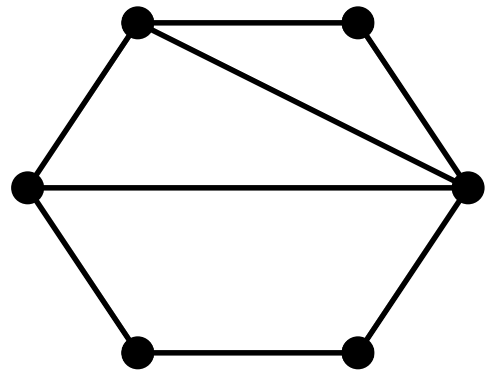
Proper 5-edge-coloring



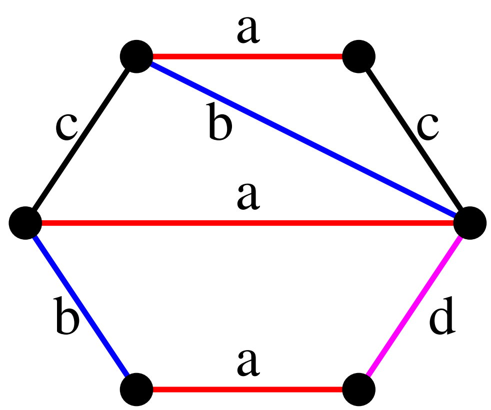
Not a proper edge coloring

- Again, let G be a graph and C be a set of colors.
- A *proper edge coloring* is a function assigning a color from C to every edge, such that if two edges share any vertices, the edges must have different colors.
- A *proper k-edge-coloring* is a proper edge coloring with k colors. A graph is *k-edge-colorable* if this exists. This graph is 5-edge-colorable.

Color edges with as few colors $a, b, c, \ldots$ as possible

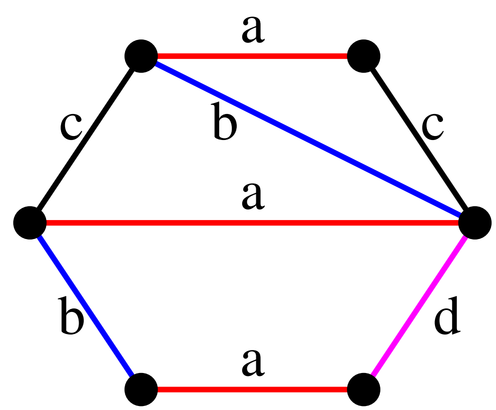


Color edges with as few colors $a, b, c, \ldots$ as possible



- The minimum number of colors needed for a proper edge coloring is denoted $\chi'(G)$. This is called the *chromatic index* or the *edge-chromatic number* of G.

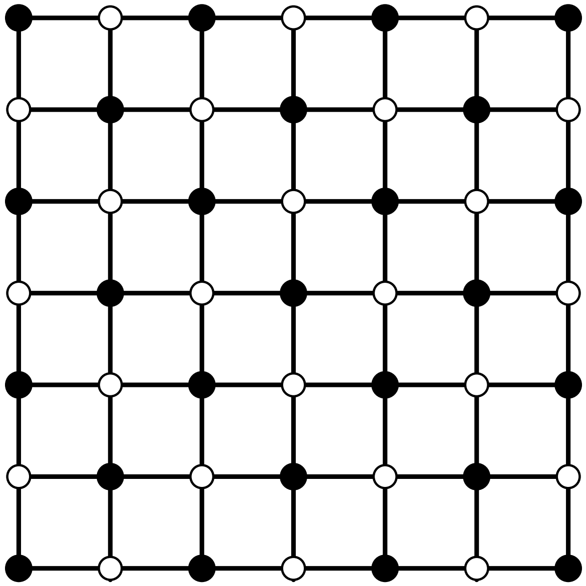
Color edges with as few colors $a, b, c, \ldots$ as possible



- We've shown it's 4-edge-colorable, so $\chi'(G) \leqslant 4$.
- There is a vertex of degree 4.
  All 4 edges on it must have different colors, so $\chi'(G) \geqslant 4$.
- Combining these gives $\chi'(G) = 4$.
- In general, $\chi'(G) \geqslant \Delta(G)$, since all edges on a max degree vertex must have different colors.

Relation of coloring to previous concepts

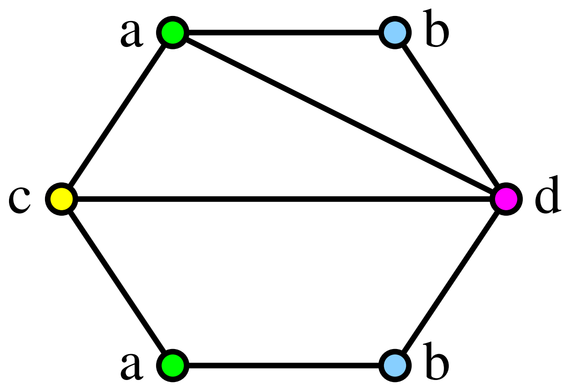
Bipartite graphs



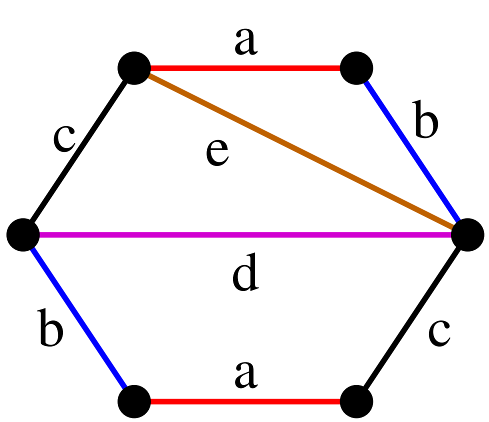
A graph is bipartite if and only if it is 2-colorable

- $A =$ black vertices and $B =$ white vertices.
- **Bipartite:** All edges have one vertex in A and the other in B.
- **2-colorable:** All edges have 1 black vertex and 1 white vertex.
- This graph has $\chi(G) = 2$ and $\chi'(G) = 4$.
- In general, a bipartite graph has $\chi(G) \leqslant 2$
  ($\chi(G) = 1$ for only isolated vertices, and 0 for empty graph).

Independent sets and matchings



- In a proper coloring (vertices), all vertices of the same color form an independent set (since there are no edges between them).



- In a proper edge coloring, all edges of the same color form a matching (since they don't share vertices).

Results for proper edge colorings

Major results about proper colorings

Proper edge colorings:

König's Edge Coloring Theorem

For any bipartite graph, $\chi'(G) = \Delta(G)$.

Vizing's Theorem

For any simple graph, $\chi'(G) = \Delta(G)$ or $\Delta(G) + 1$.

Proper vertex colorings:

Brooks' Theorem

All connected graphs have $\chi(G) \leqslant \Delta(G)$, except for K_n and odd cycles.

König's Edge Coloring Theorem

Don't confuse with König's Theorem on maximum matchings, nor with the König-Ore Formula

König's Edge Coloring Theorem

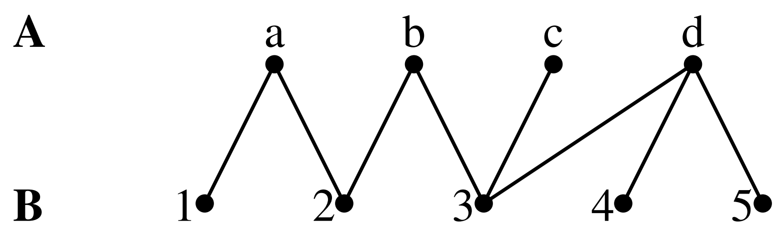
For any bipartite graph, $\chi'(G) = \Delta(G)$.

Proof (first case: regular graphs):

- First, suppose G is k-regular. Then $k = \Delta(G)$.
- We showed that if G is a k-regular bipartite graph, its edges can be partitioned into k perfect matchings, $M_1, \ldots, M_k$, with every edge of G in exactly one of the matchings.
  - This also holds for bipartite multigraphs!
- Assign all edges of M_i the color i. This is a proper edge coloring of G, since all edges on each vertex are in different matchings.
- So $\chi'(G) \leqslant k$. We also showed $\chi'(G) \geqslant \Delta(G) = k$, so $\chi'(G) = k$.

König's Edge Coloring Theorem

For any bipartite graph, $\chi'(G) = \Delta(G)$.

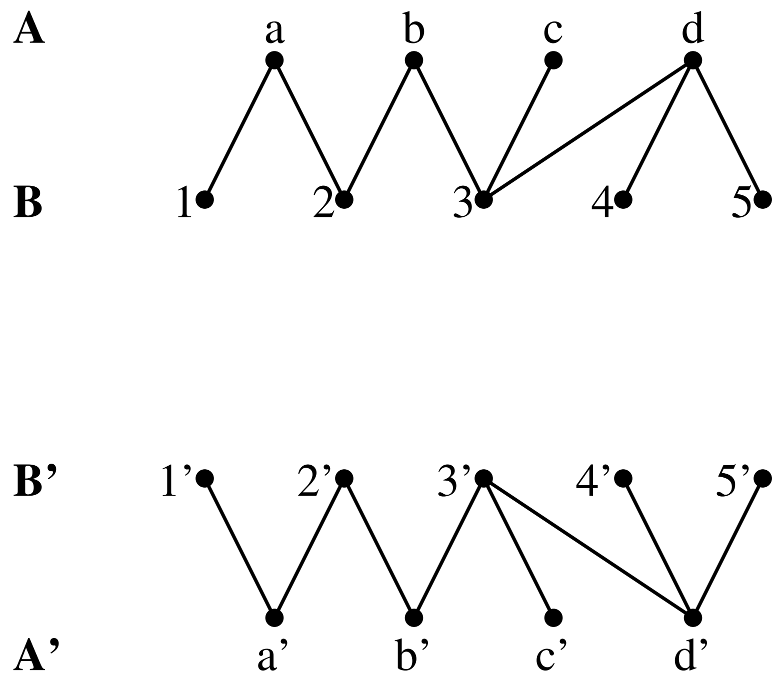


Proof, continued (second case: graphs that aren't regular):

- Now suppose G is not regular (example above).

König's Edge Coloring Theorem

For any bipartite graph, $\chi'(G) = \Delta(G)$.

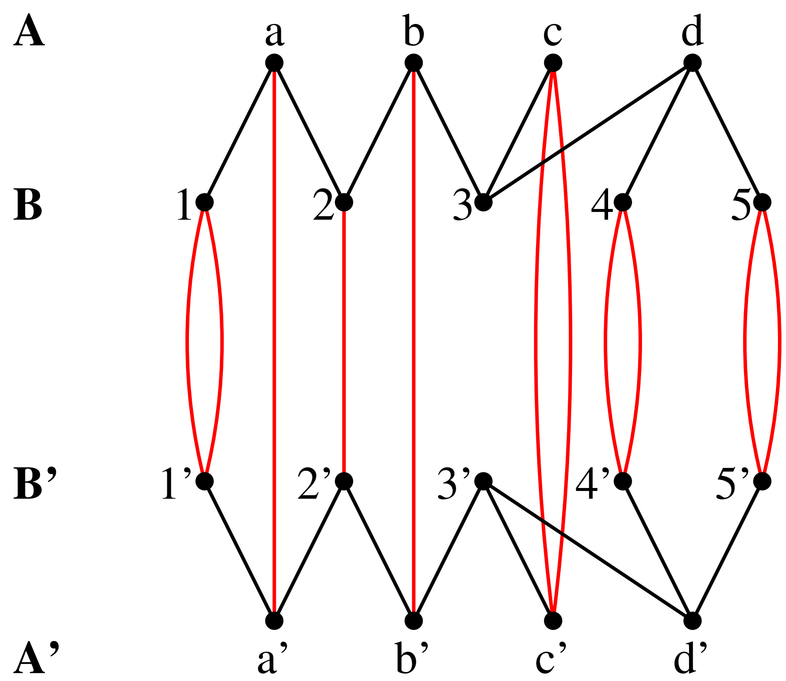


Proof, continued:

- Make a clone G' of G.
- **Vertices:** G' has parts A' and B'. Name the vertices of G' after the vertices of G, but add $'$ symbols to make them different.
- **Edges:** The clone of edge $\{a, b\}$ in G is $\{a', b'\}$ in G'.

König's Edge Coloring Theorem

For any bipartite graph, $\chi'(G) = \Delta(G)$.

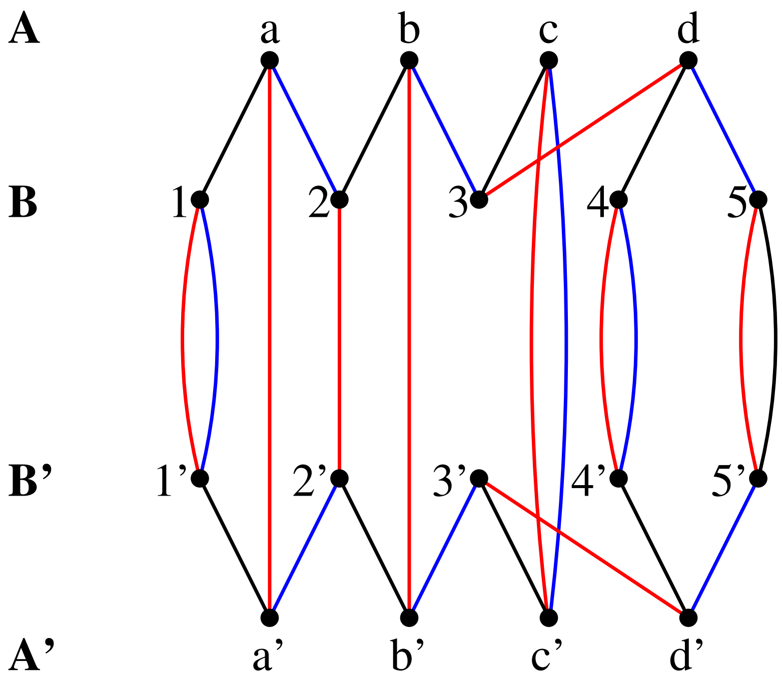


Proof, continued:

- For each vertex $x \in A \cup B$, add $\Delta(G) - d_G(x)$ parallel edges between x and x' (shown in red).
- Now all vertices have degree $\Delta(G)$! (Here, $\Delta(G) = 3$.)
- The new graph, H, is $\Delta(G)$-regular.
- H is bipartite with parts $A \cup B'$ and $A' \cup B$.

König's Edge Coloring Theorem

For any bipartite graph, $\chi'(G) = \Delta(G)$.

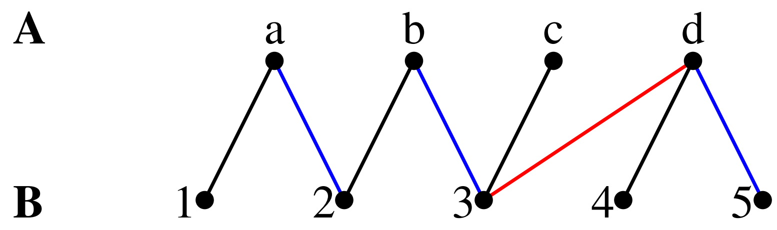


Proof, continued:

- Let $k = \Delta(G)$. Here, $k = 3$.
- Since H is bipartite and k-regular, it has a proper k-edge-coloring (shown here in black, red, and blue).

König's Edge Coloring Theorem

For any bipartite graph, $\chi'(G) = \Delta(G)$.



Proof, continued:

- Remove G' and the edges that were added between G and G'.
- This gives a proper edge coloring of G with $\leqslant \Delta(G)$ colors, so $\chi'(G) \leqslant \Delta(G)$.
- Since $\chi'(G) \geqslant \Delta(G)$ as well, we conclude $\chi'(G) = \Delta(G)$.

Vizing's Theorem

Vizing's Theorem

For any simple graph, $\chi'(G) = \Delta(G)$ or $\Delta(G) + 1$.

Proof outline:

- We showed $\chi'(G) \geqslant \Delta(G)$ for any graph.
- We can construct a proper edge coloring with $\Delta(G) + 1$ colors. It's rather detailed, so we'll skip it; see the text book.
- Then $\chi'(G) \leqslant \Delta(G) + 1$.
- Combining the two inequalities gives $\chi'(G) = \Delta(G)$ or $\Delta(G) + 1$.

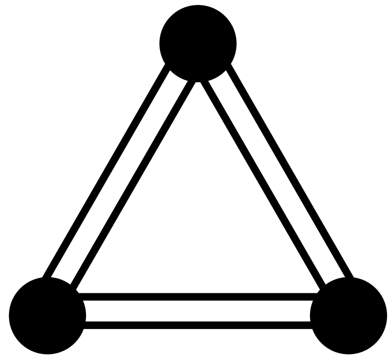
Vizing's Theorem

For any simple graph, $\chi'(G) = \Delta(G)$ or $\Delta(G) + 1$.

- The graphs with $\chi'(G) = \Delta(G)$ are called *class 1*
  $\chi'(G) = \Delta(G) + 1$ are called *class 2*.
- Determining whether a graph is class 1 or class 2 is NP-complete.
- But it turns out "almost all" graphs are class 1!
  - Recall there are $2^{\binom{n}{2}}$ simple graphs on vertices $\{1, \ldots, n\}$.
  - Erdös and Wilson (1975) proved:

$$\lim_{n\to\infty} \left(\frac{\text{\# class 1 graphs on } n \text{ vertices}}{\text{\# simple graphs on } n \text{ vertices}} \right) = 1$$

Vizing's Theorem — Multigraphs



- Consider this multigraph.
- All 6 edges touch, so in a proper edge coloring, they must all be different colors. Thus, $\chi'(G) = 6$.
- $\Delta(G) = 4$, so $\chi'(G)$ doesn't equal $\Delta(G)$ or $\Delta(G) + 1$.
- Let $\mu(G)$ be the maximum edge multiplicity.
  For a simple graph, it's 1, but here, it's 2.

Vizing's Theorem for Multigraphs

For any multigraph, $\chi'(G) = \Delta(G) + d$ for some $0 \leqslant d \leqslant \mu(G)$.

Results for proper vertex colorings

Proper colorings of certain graphs

Proper coloring of K_n

- $\chi(K_n) = n$: All vertices are adjacent, so their colors are all distinct.
- $\Delta(K_n) = n - 1$.

Proper coloring of a cycle C_n ($n \geqslant 3$)

- Any even length cycle has $\chi(C_n) = 2$.
- Any odd length cycle has $\chi(C_n) = 3$.
- All cycles (whether odd or even) have $\Delta(C_n) = 2$.

Brooks' Theorem

All connected graphs have $\chi(G) \leqslant \Delta(G)$, **except** K_n and odd length cycles have $\chi(G) = \Delta(G) + 1$.

- We'll do a zillion special cases, building up to a complete proof.

Brooks' Theorem

Special case: Small values of $\Delta(G)$

$\Delta(G) = 0$ or 1, with G connected

- $\Delta(G) = 0$ gives an isolated vertex, $G = K_1$.
- $\Delta(G) = 1$ gives just one edge, $G = K_2$.
- Complete graphs are one of the exceptions in Brooks' Theorem.

$\Delta(G) = 2$, with G connected

Then G is a path or a cycle, and $n \geqslant 3$.

- If G is a path, $\chi(G) = \Delta(G) = 2$.
- If G is a even length cycle, $\chi(G) = \Delta(G) = 2$.
- If G is an odd length cycle, $\chi(G) = 3$ but $\Delta(G) = 2$.
  This is the other exception in Brooks' Theorem.

For the rest of the cases, assume $\Delta(G) \geqslant 3$.

Brooks' Theorem

Lemma

Every graph has a proper coloring with $\Delta(G) + 1$ colors.
Thus, $\chi(G) \leqslant \Delta(G) + 1$.

- Notation:

Max degree	$\Delta = \Delta(G)$
Vertices	$v_1, \ldots, v_n$ (ordered arbitrarily)
Colors	$1, 2, \ldots, \Delta + 1$

- Assign a color to v_i as follows (going in order $i = 1, 2, \ldots, n$):
  - v_i has at most Δ neighbors among $v_1, \ldots, v_{i-1}$.
  - At most Δ different colors are used by those neighbors.
  - With $\Delta + 1$ colors, at least one color different from those is available.
  - Assign the smallest available color to v_i.
- We'll do several special cases where carefully choosing the vertex order reduces the number of colors needed.

Brooks' Theorem

Special case: Vertex of smaller degree than maximum

Lemma

If connected graph G has a vertex v with $d(v) < \Delta(G)$, then $\chi(G) \leqslant \Delta(G)$.

- Again let $\Delta = \Delta(G)$. We will color the vertices with Δ colors.
- Do a breadth first search starting at v.
  The vertices in order of discovery are $v_1, \ldots, v_n$, with $v_1 = v$.
- Color vertices in reverse order, $v_n, \ldots, v_2$, as follows:
  - Each v_i ($i \neq 1$) has at least one neighbor v_j with $j < i$, and
    at most $\Delta - 1$ neighbors with $j > i$.
  - So at most $\Delta - 1$ colors have been assigned so far to its neighbors.
  - At least one of the Δ colors is available to assign to v_i.
- Finally, color $v_1 = v$.
  Since $d(v) < \Delta$, at least $\Delta - d(v) \geqslant 1$ colors are available.

Brooks' Theorem

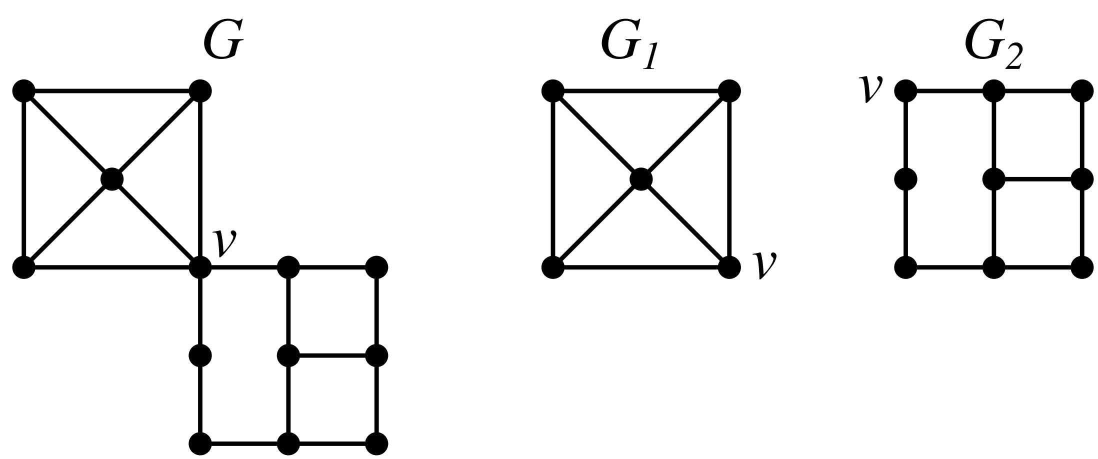
Special case: G has a cut vertex

Lemma

If G is connected and has a cut vertex, then $\chi(G) \leqslant \Delta(G)$.

Proof:

- Let v be a cut vertex.
- $G - \{v\}$ has $r \geqslant 2$ components. Let $G_1, \ldots, G_r$ be those components but with v and its edges to vertices of G_i included.



- We'll show each G_i can be colored with $\leqslant \Delta(G)$ colors.

Brooks' Theorem

Special case: G has a cut vertex — proof continued

Lemma

If G is connected and has a cut vertex, then $\chi(G) \leqslant \Delta(G)$.

Proof, continued: In G_i,

- All vertices still have degree $\leqslant \Delta(G)$.
- Additionally, $d_{G_i}(v) \leqslant \Delta(G) - (r-1) \leqslant \Delta(G) - 1$.
  So if $\Delta(G_i) = \Delta(G)$, then G_i can be $\Delta(G)$-colored.
- If $\Delta(G_i) < \Delta(G)$, it can be colored with $\Delta(G_i) + 1 \leqslant \Delta(G)$ colors.

Recall previous lemmas

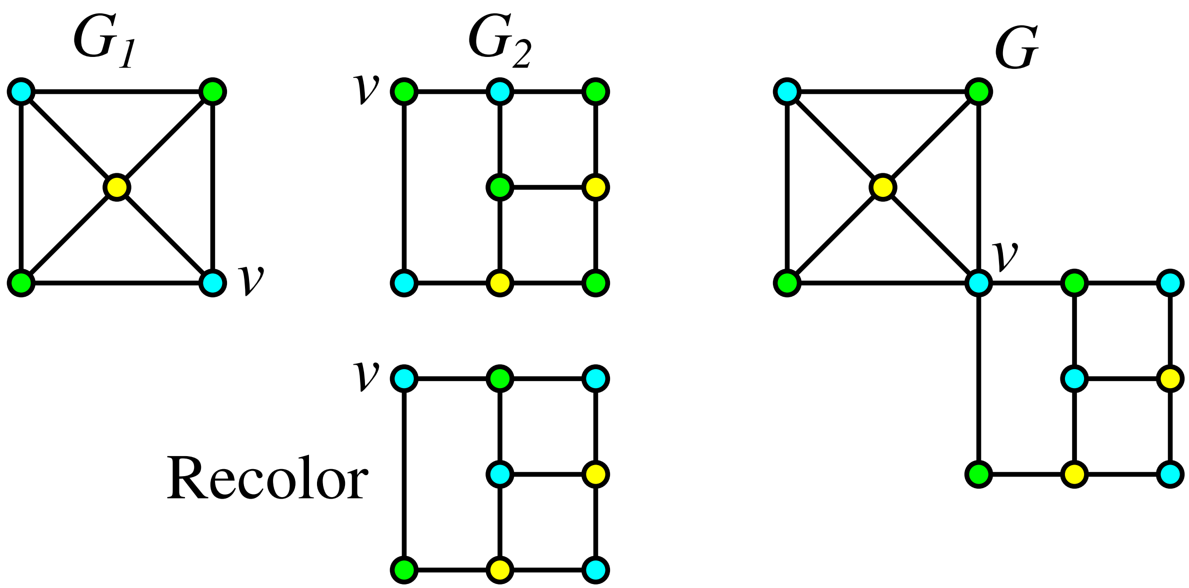
- If conn. graph G has vertex v with $d(v) < \Delta(G)$, then $\chi(G) \leqslant \Delta(G)$.
- Every graph has $\chi(G) \leqslant \Delta(G) + 1$.

Lemma

If G is connected and has a cut vertex, then $\chi(G) \leqslant \Delta(G)$.

Proof, continued:

- Rename colors in $G_1, \ldots, G_r$ so v has the same color in all of them.
- Combine proper colorings of $G_1, \ldots, G_r$ to get a proper coloring of G with $\Delta(G)$ colors.



Brooks' Theorem

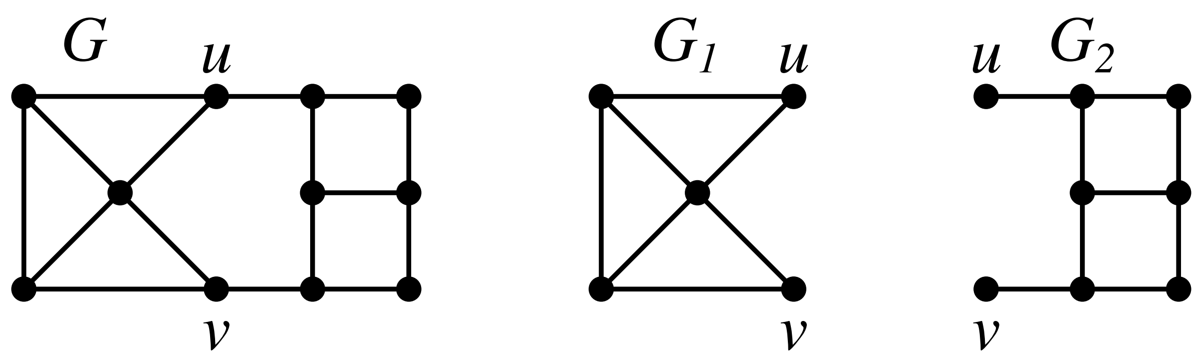
Special case: G has a vertex cut of size 2

Lemma

If G is connected, has $\Delta(G) \geqslant 3$, and has a vertex cut $\{u, v\}$ with $uv \notin E(G)$, then $\chi(G) \leqslant \Delta(G)$.

Proof:

- Now $G - \{u, v\}$ has two or more components.
- Split G into G_1 (one component) and G_2 (all others), each including u, v and the edges to the other vertices of that component.
- In each of G_1 & G_2, both u & v have degrees between 1 and $\Delta(G) - 1$.



Brooks' Theorem

Special case: G has a vertex cut of size 2 — proof continued

Lemma

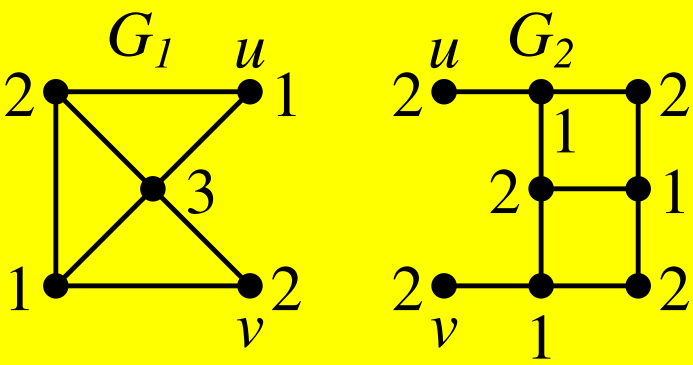
If G is connected, has $\Delta(G) \geqslant 3$, and has a vertex cut $\{u, v\}$ with $uv \notin E(G)$, then $\chi(G) \leqslant \Delta(G)$.

Proof, continued:

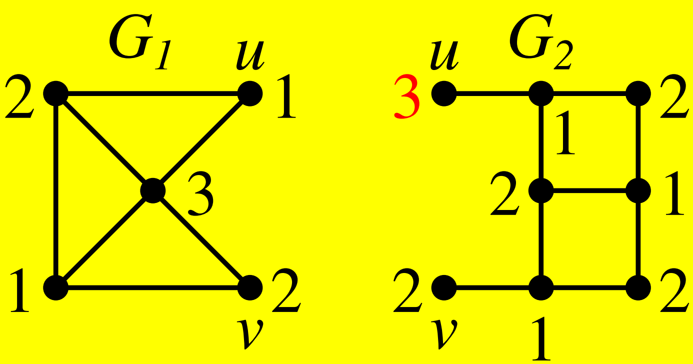
Case 1: In both G_1 and G_2, either u or v has degree $\leqslant \Delta(G) - 2$.

- G_1 and G_2 can each be Δ-colored with different colors for u & v.
- For example, say in G_1: $\quad d(u) \leqslant \Delta(G) - 2$
  - By previous cases, we can color G_1 with Δ colors.
  - If u and v have the same color in G_1 on on our first try, then u and its neighbors in G_1 use at most $(\Delta - 2) + 1 = \Delta - 1$ colors, so there's still a color remaining (out of Δ colors) to change u's color.
- Rename colors in G_1 and G_2 so that u and v match in each.
- Combine the Δ-colorings of G_1 and G_2 into a Δ-coloring of G.

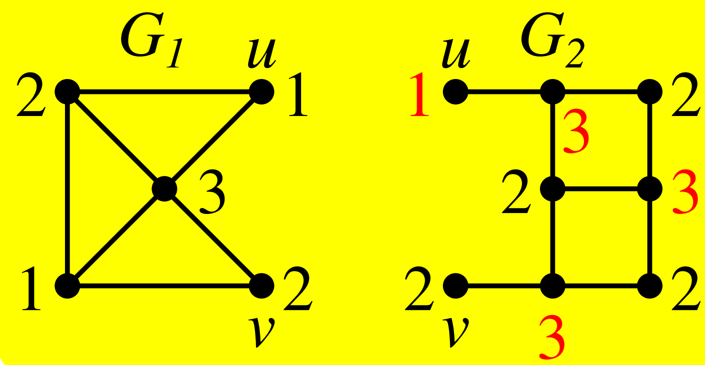
1. Initial colorings:



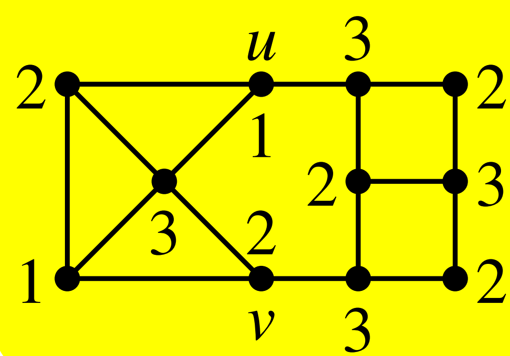
2. Make u,v different in each part



3. Permute colors to match u's & v's



4. Combine



Brooks' Theorem

Special case: G has a vertex cut of size 2 — proof continued

Lemma

If G is connected, has $\Delta(G) \geqslant 3$, and has a vertex cut $\{u, v\}$ with $uv \notin E(G)$, then $\chi(G) \leqslant \Delta(G)$.

Proof, continued:

Case 2: In G_1 or G_2, both u and v have degree $> \Delta(G) - 2$.

- Assume it's G_1 (G_2 works similarly). Then
$$d_{G_1}(u) = d_{G_1}(v) = \Delta(G) - 1 \qquad d_{G_2}(u) = d_{G_2}(v) = 1$$
- So in G_2, both u and v are in one edge each: ua and vb. Note it can't be uv since we assumed uv is not an edge.
- $\{a, v\}$ is also a vertex cut, and gives Case 1.

Brooks' Theorem

All connected graphs have $\chi(G) \leqslant \Delta(G)$, except for K_n and odd cycles.

Proof: If any special case applies, we're done. But if none apply, then:

- $\Delta \geqslant 3$.
- It's not a complete graph or odd cycle.
- There are no cut vertices.
- There are no vertex cuts $\{u, v\}$ with uv not an edge.
- There is no vertex with $d(v) < \Delta(G)$; thus, G is Δ-regular.

This is the "case" we're in: ALL of the above at once.

Brooks' Theorem

All connected graphs have $\chi(G) \leqslant \Delta(G)$, except for K_n and odd cycles.

Proof of Brooks' Theorem, continued:

- Let x be any vertex in G.
- x must have neighbors y, z where xy and xz are edges but yz isn't:
  - If all of x's neighbors are adjacent to each other, then x and its neighbors form a clique of size $\Delta + 1$.
  - This accounts for Δ neighbors of each of those vertices.
    G is Δ-regular, so that's all of their neighbors, making this clique a connected component of G.
  - G is connected, so that's the whole graph.
  - Thus, $G = K_{\Delta+1}$, contradicting that it's not a complete graph.

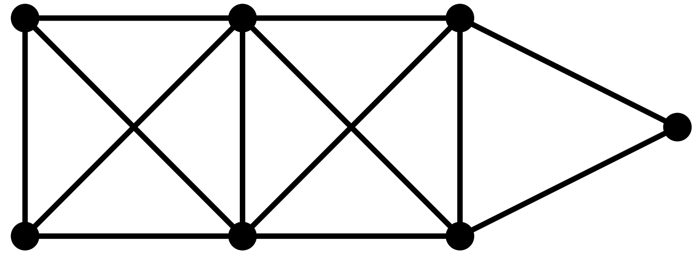
Brooks' Theorem

All connected graphs have $\chi(G) \leqslant \Delta(G)$, except for K_n and odd cycles.

Proof of Brooks' Theorem, continued:

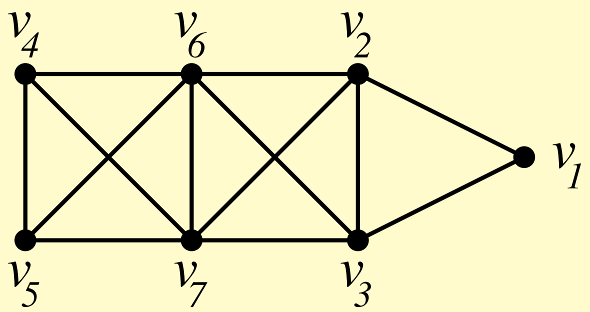
- We have vertices x, y, z where xy and xz are edges but yz isn't.
- $G - \{y, z\}$ is connected (since that's the case we're in).
  - Do BFS in $G - \{y, z\}$ starting at x.
  - List vertices in order of discovery $v_1, \ldots, v_{n-2}$, with $v_1 = x$.
  - Then set $v_{n-1} = y$ and $v_n = z$.
- Color the vertices in reverse order $v_n, v_{n-1}, \ldots, v_1$:
  - $v_n = z$ and $v_{n-1} = y$ both get color 1.
  - Each v_i (for $i = n-2, \ldots, 2$) has $\leqslant \Delta - 1$ neighbors already colored (v_j with $j > i$), so at least one of the Δ colors is available for each.
  - When we reach v_1, all Δ of its neighbors were already colored.
    But y and z both got color 1!
    So at most $\Delta - 1$ colors were used on v_1's neighbors.
    So at least one of the Δ colors is available for v_1.

Degenerate graphs



- A graph is *k-degenerate* if *all* subgraphs have min. degree $\leqslant k$.
- This graph has minimum degree $\delta(G) = 2$, but subgraphs ⊠ have higher minimum degree, so it's not 2-degenerate.
- All subgraphs have min degree $\leqslant \Delta(G) = 5$, so it's 5-degenerate.
- What's the smallest k for which it's k-degenerate? $\boxed{\mathbf{3}}$
- The *degeneracy* (or *degeneracy number*) of a graph is the smallest k for which it's k-degenerate. Here, it's 3.
- **Theorem:** *If G is k-degenerate, then $\chi(G) \leqslant k+1$.*
  This is often an improvement over $\chi(G) \leqslant \Delta(G)$.

Degenerate graphs

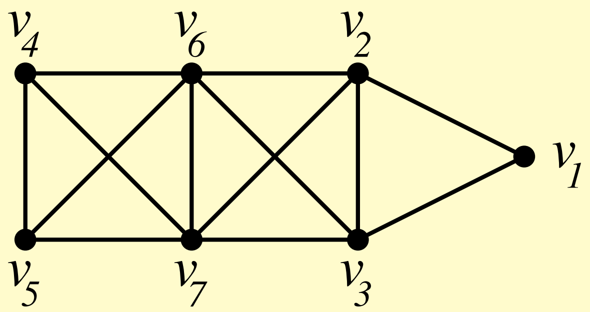


Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7
$\boldsymbol{d_i}$	2	3	2	3	2	1	0

- Repeatedly choose a vertex of minimum degree (in the remaining graph) and remove it, getting a sequence of vertices $v_1, \ldots, v_n$.
- Let d_i be the degree of v_i just before it's removed (so it's the degree in $G - \{v_1, \ldots, v_{i-1}\} = G[v_i, \ldots, v_n]$).
- Every edge is accounted for in exactly one d_i (whichever of it's vertices is removed first), so $\sum_i d_i = |E(G)|$ (here it equals 13).
- If G is k-degenerate, then every v_i has $\leqslant k$ neighbors in $v_{i+1}, \ldots, v_n$ (since v_i has degree $\leqslant k$ in every subgraph, including $G[v_i, \ldots, v_n]$).

Degenerate graphs

Computing degeneracy number

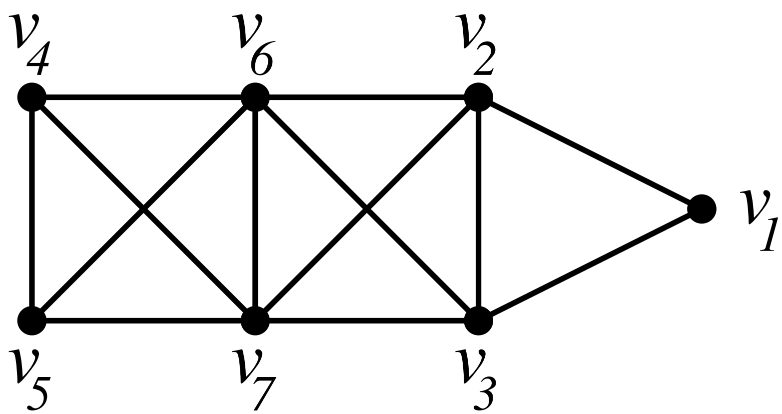


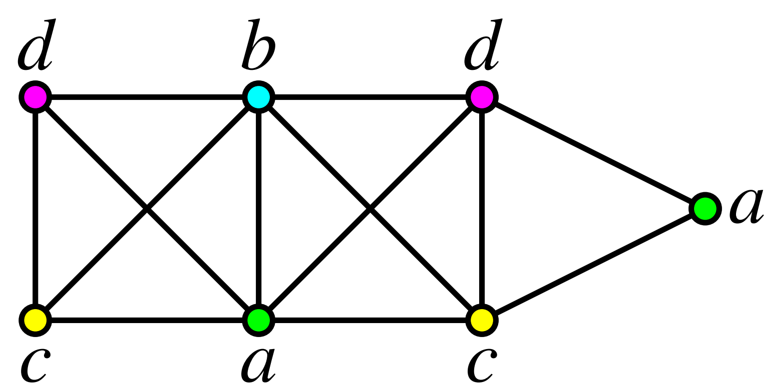
Vertex	v_1	v_2	v_3	v_4	v_5	v_6	v_7
d_i	2	3	2	3	2	1	0

- Sometimes we'll use that a graph is k-degenerate for a particular value of k, even if it's not the smallest number possible.
- But you can also compute the degeneracy number by this algorithm! It's
$$\max\{\, d_i \;:\; i = 1, \ldots, n \,\}.$$

Degenerate graphs

Theorem: Every k-degenerate graph has $\chi(G) \leqslant k+1$.





Proof: We'll show G can be colored with $k+1$ colors.

- Form the order $v_1, \ldots, v_n$ just described.
- Color vertices in reverse order $v_n, \ldots, v_1$:
  - When considering v_i, at most k of its neighbors (among $v_{i+1}, \ldots, v_n$) have been colored, so at least one color remains out of $k+1$ colors.
  - Assign the smallest available color to v_i.
  - This gives a proper $(k+1)$-coloring of G.

Complexity of chromatic number

While we have bounds on $\chi(G)$ and can compute it in special cases, computing it for an arbitrary graph is NP-hard.

Scheduling Problem

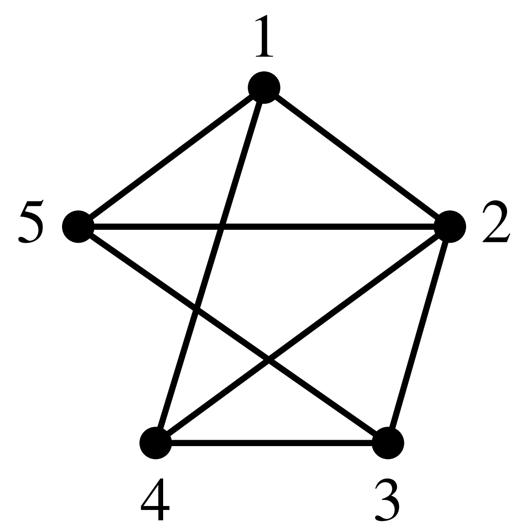
Scheduling Problem
a.k.a. Timetable Problem or Storage Problem

Student	Classes
a	1,2,4
b	2,3,5
c	3,4
d	1,5

- Students want to take certain classes, shown in the table above.
- How can we schedule the classes in so that students can take all the classes on their wishlist without any conflicts?
- We could schedule them at 5 different times. How about fewer?

Scheduling Problem

Student	Classes
a	1,2,4
b	2,3,5
c	3,4
d	1,5

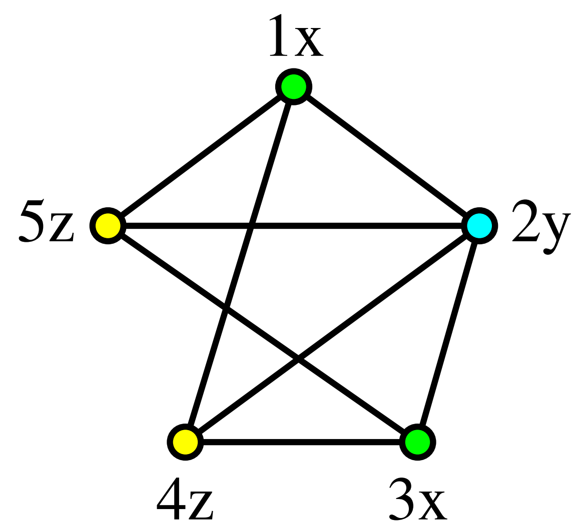


Make an *interference graph*:

- **Vertices:** One vertex for each class.
- **Edges:** Add edge uv if classes u and v *interfere* (a student wants to take both of them).
- Any proper coloring of the graph gives a schedule w/o anyone having a conflict (colors correspond to time slots).
- Find a solution with a minimum number of colors (to minimize the number of time slots).

Scheduling Problem

Student	Classes
a	1,2,4
b	2,3,5
c	3,4
d	1,5



- Above is a proper coloring with the minimum number of colors (denoted x, y, z).
- 9am (color x): Classes 1 and 3
- 10am (color y): Class 2
- 11am (color z): Classes 4 and 5

Scheduling Problem:

Register allocation in compilers

Register allocation in compilers

- A *compiler* translates a high level programming language (C, C++, . . .) to assembly language for a particular CPU instruction set architecture (like x86, AMD, etc.).
- C/C++ instruction n++ compiled for an x86_64 processor:

```
movl   -20(%rbp), %eax    # copy n from RAM to register %eax
addl   $1, %eax           # add 1 to register %eax
movl   %eax, -20(%rbp)    # copy result back to n in RAM
```

- A C/C++ program may have 1000s of variables, stored in memory (RAM), and you choose their names.
- A CPU has a very small number of *registers*: special variables stored in the CPU with fixed names.
  - x86_64 CPUs (on many laptops in the last decade) have 8 general purpose registers in 32-bit mode / 16 in 64-bit mode.
- C/C++ variables are copied from RAM to a CPU register for arithmetic, comparisons, . . . and back to RAM if needed.

Register allocation in compilers

Code

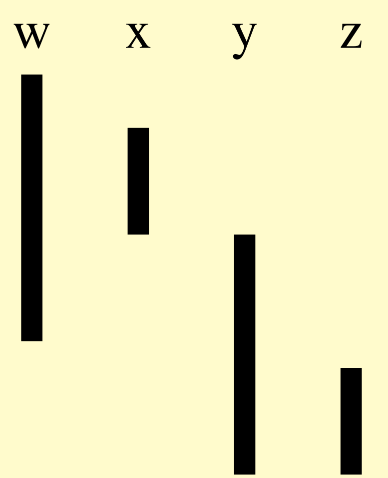
```
w = ...
x = ...
FOO(x)
y = ...
BAR(w,y)
z = ...
BAZ(y,z)
```

- The code above has four variables, w, x, y, z.

Register allocation in compilers

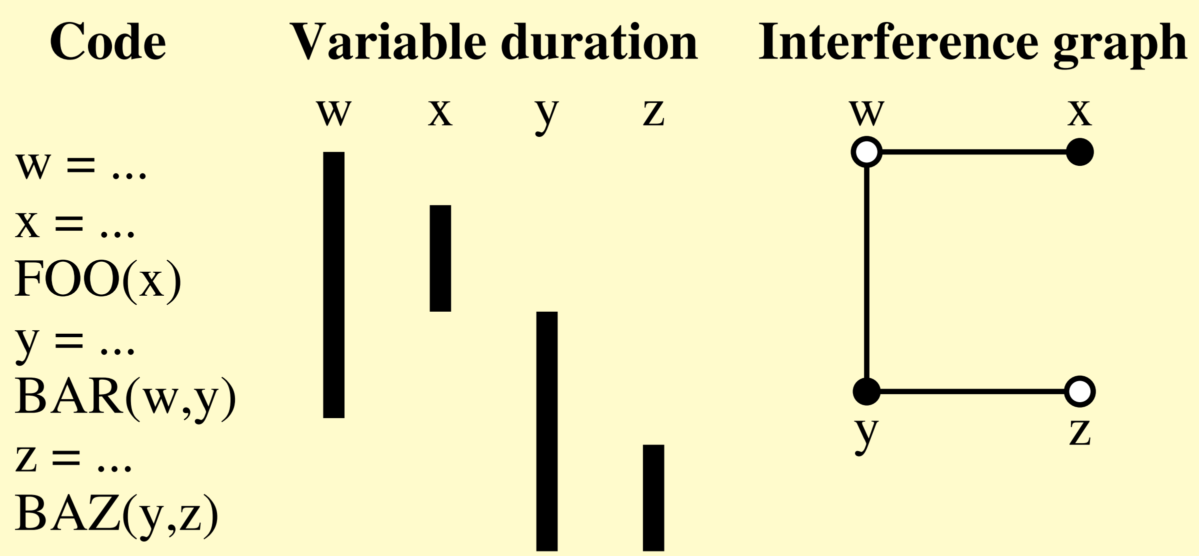
Code **Variable duration**

```
w = ...
x = ...
FOO(x)
y = ...
BAR(w,y)
z = ...
BAZ(y,z)
```



- Determine duration of each variable's use.

Register allocation in compilers



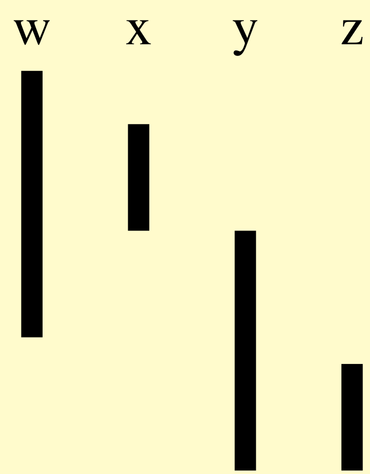
- Make an interference graph with vertices = variables, and an edge between variables in use at the same time.
- Find a proper coloring of the graph (ideally with a min # colors).

Register allocation in compilers

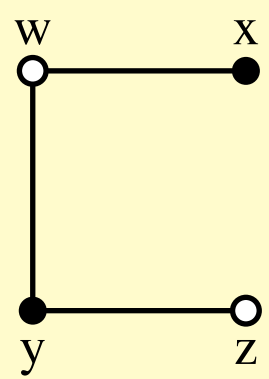
Code

```
w = ...
x = ...
FOO(x)
y = ...
BAR(w,y)
z = ...
BAZ(y,z)
```

Variable duration



Interference graph



Registers

```
R1 = ...
R2 = ...
FOO(R2)
R2 = ...
BAR(R1,R2)
R1 = ...
BAZ(R2,R1)
```

- Assign variables to registers based on the coloring; here, R1 (white) and R2 (black).
- R1 and R2 represent different variables at different times.