office hours change: (this week)

Thursday 1-3 pm → Tuesday 1-3 pm

✗ ✓

Middlebury

CSCI 200: Math Foundations of Computing

Spring 2026

Lecture 9M: Graph Coloring

Goals for today:

1. Describe the concept of graph coloring and why it's useful.
2. Prove that a graph with a maximum degree of k can be colored with $k + 1$ colors.
3. Implement a graph coloring algorithm!

Can you come up with an exam schedule for these 5 courses?
(X denotes an overlap in enrollment for two courses)

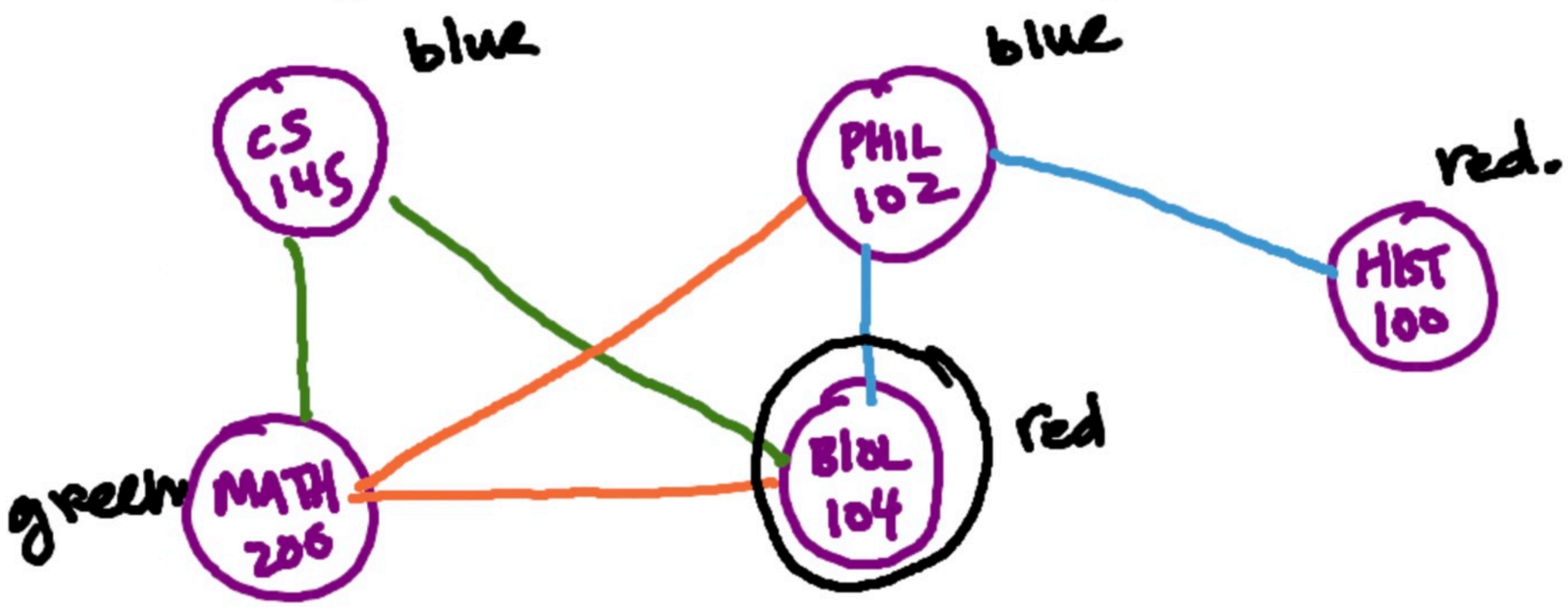
	CSCI 145	MATH 200	PHIL 102	BIOL 104	HIST 100
CSCI 145		X		X	
MATH 200	X		X	X	
PHIL 102		X		X	X
BIOL 104	X	X	X		
HIST 100			X		

Try it out and then vote! **A:** 1 color, **B:** 2 colors, **C:** 3 colors, **D:** 4 colors, **E:** 5 colors.

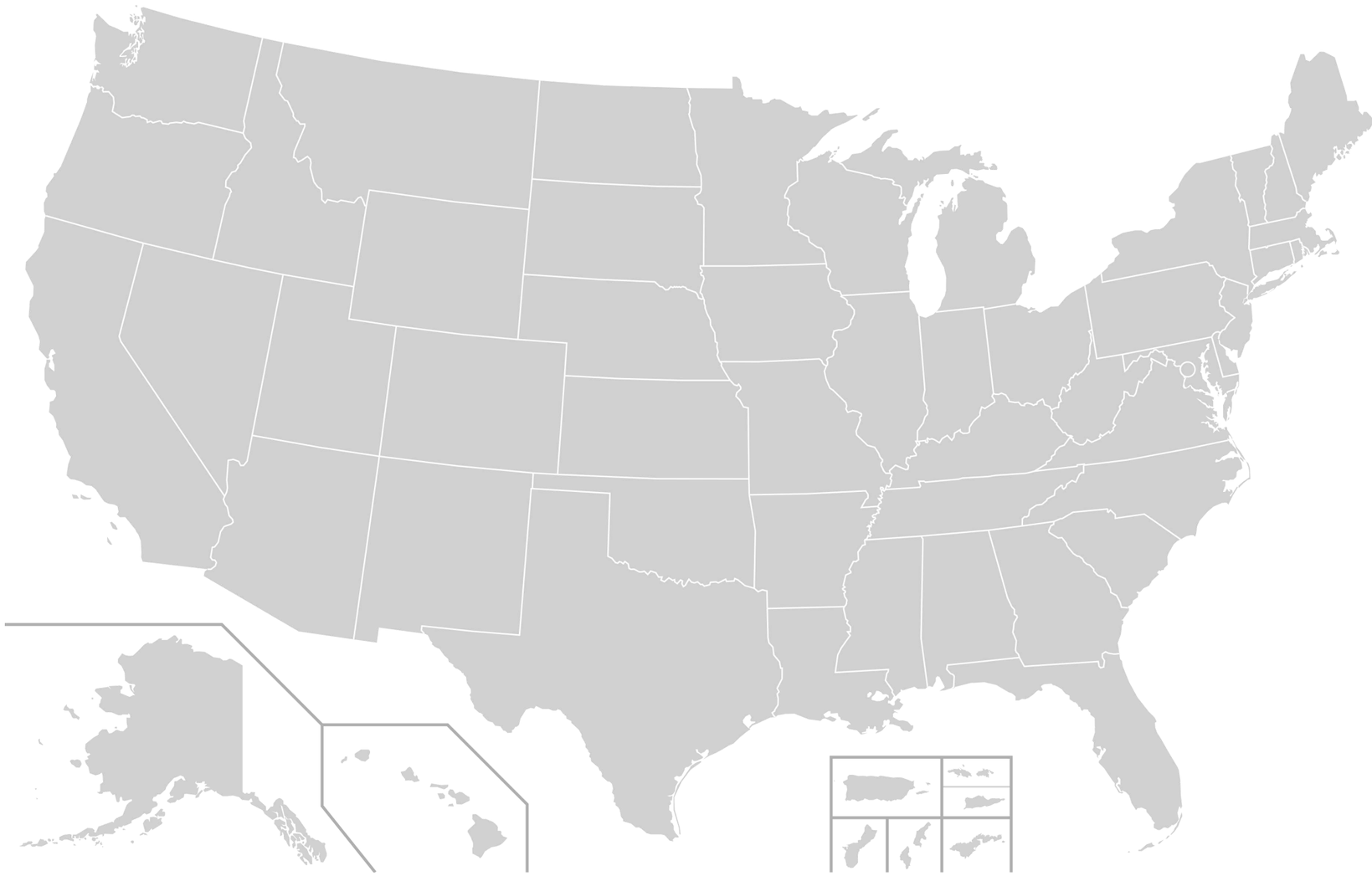
Let's interpret this as a graph:

	CSCI 145	MATH 200	PHIL 102	BIOL 104	HIST 100
CSCI 145		X		X	
MATH 200	X		X	X	
PHIL 102		X		X	X
BIOL 104	X	X	X		
HIST 100			X		

Goal: assign colors to nodes so that two adjacent nodes have a different color.



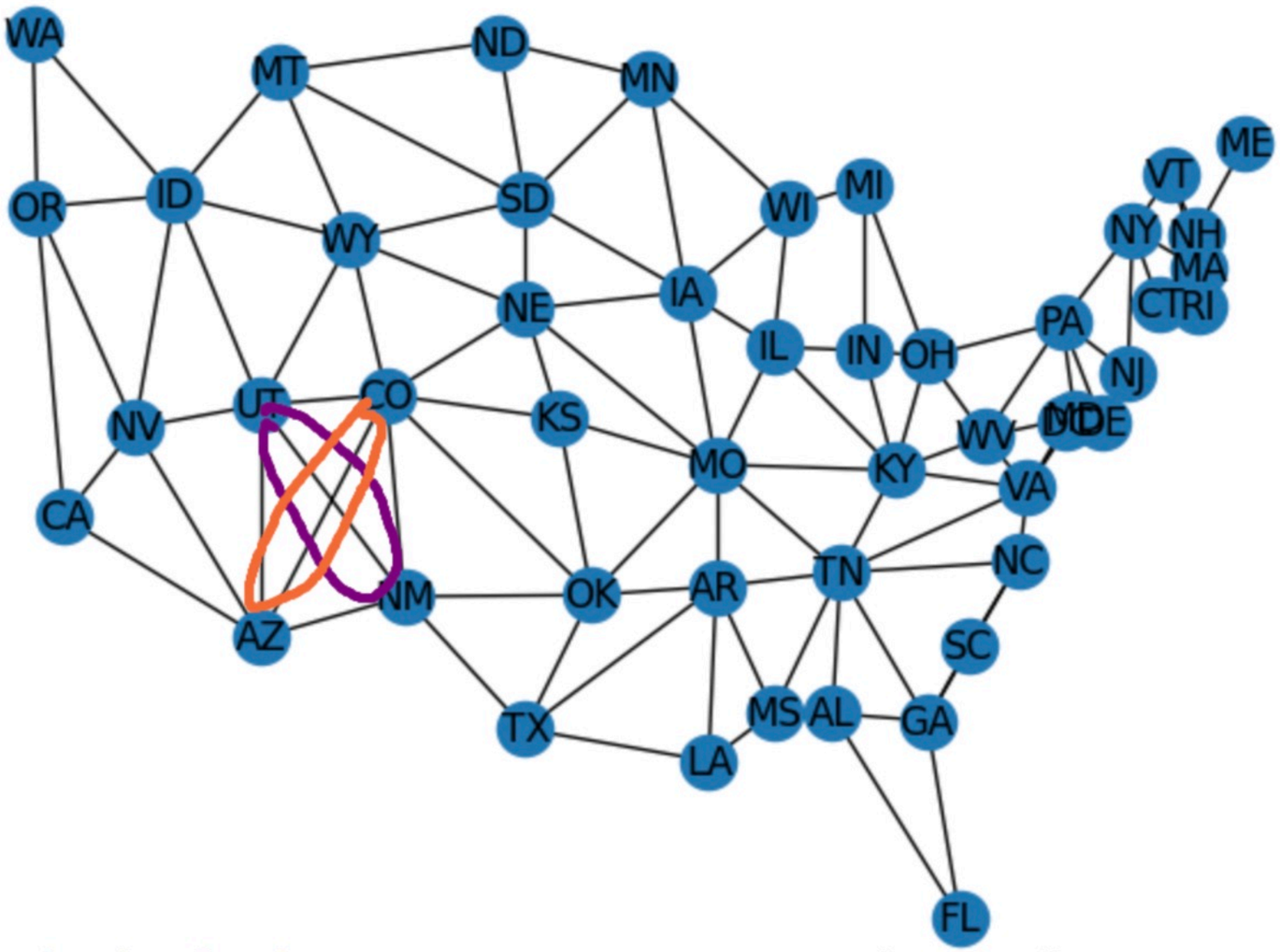
Also useful for coloring maps!

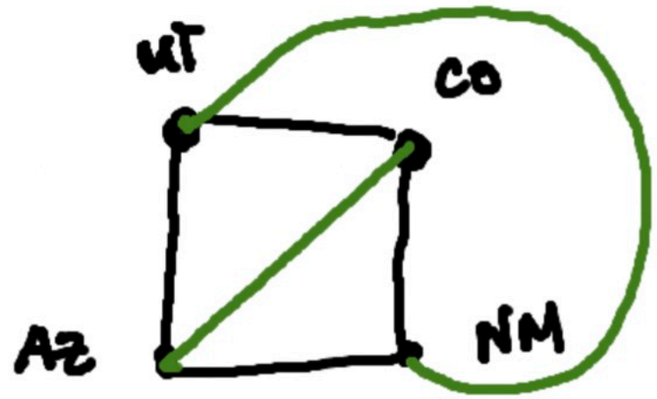


The Four Color Theorem: A *planar* graph can be colored with 4 colors.
A **planar graph** is a graph in which the vertices can be *embedded* in the plane such that no edges cross (edges can be curved though).

What if we include the diagonal adjacencies at the Four Corners?

- **Nodes** are states and there is an **edge** between states that share a border (adjacent).





An upper bound on the number of colors needed.

Theorem: A graph $G = (V, E)$ with $n = |V|$ vertices and a maximum degree of k can be colored with $k + 1$ colors.

Proof: We use a proof by induction on the # of vertices n. Let the induction hypothesis be the predicate p(n):

"*A graph with n vertices and maximum degree k can be colored with $k + 1$ colors.*"

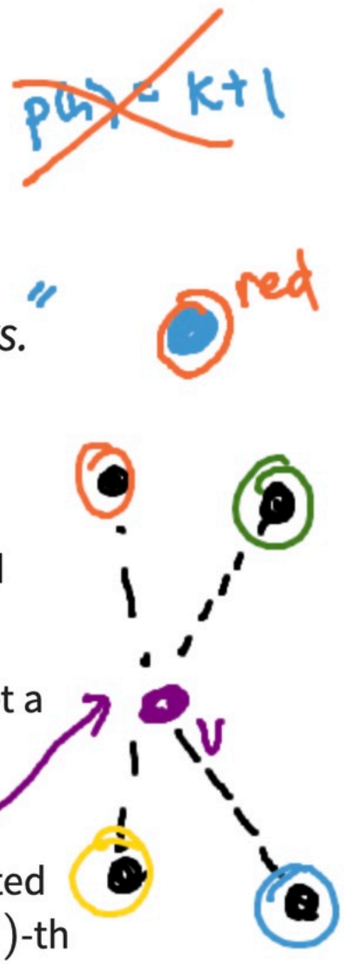
- **Base case:** For n=1, we have 0 edges, so the maximum degree is 0. This single node/vertex can be assigned a single color, which is the maximum degree (0) + 1.

- **Inductive step:** Assume p(n) is true. We need to show a graph with n+1 vertices and maximum degree k can be colored with k+1 colors.

  Consider a graph G with k or k+1? n+1 vertices and maximum degree k. Remove a vertex v from G to get a n-vertex graph G'. The maximum degree of G' is also k which can be colored with $k + 1$ colors by our assumption.

  Let's add v back in. Since the degree of v will be at most k, suppose (in the worst case), all vertices connected to v have a different color, which means there are k colors we cannot pick from. Therefore, pick the $(k + 1)$-th color for v, which shows that G' can be colored with $k + 1$ colors.

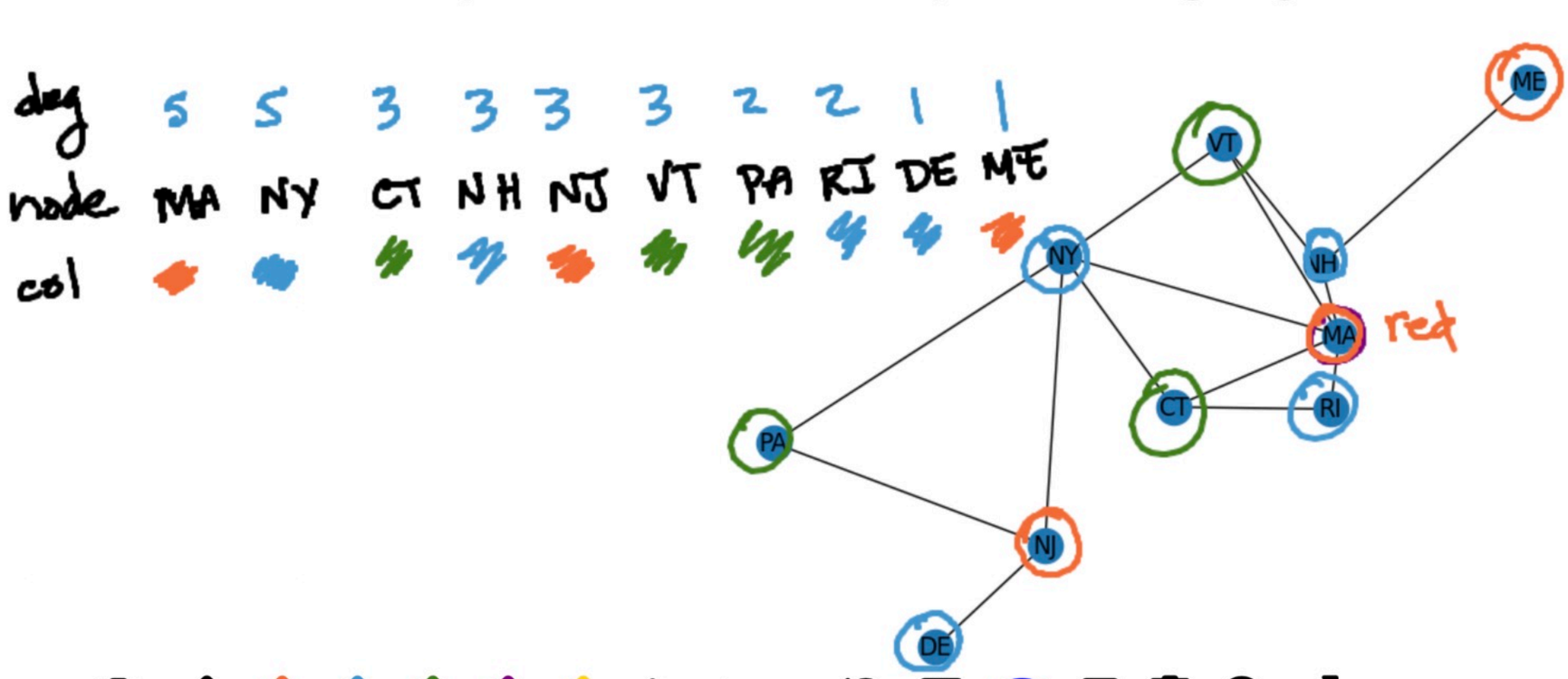
Therefore, by induction on n, a graph with maximum degree k can be colored with $k + 1$ colors. □



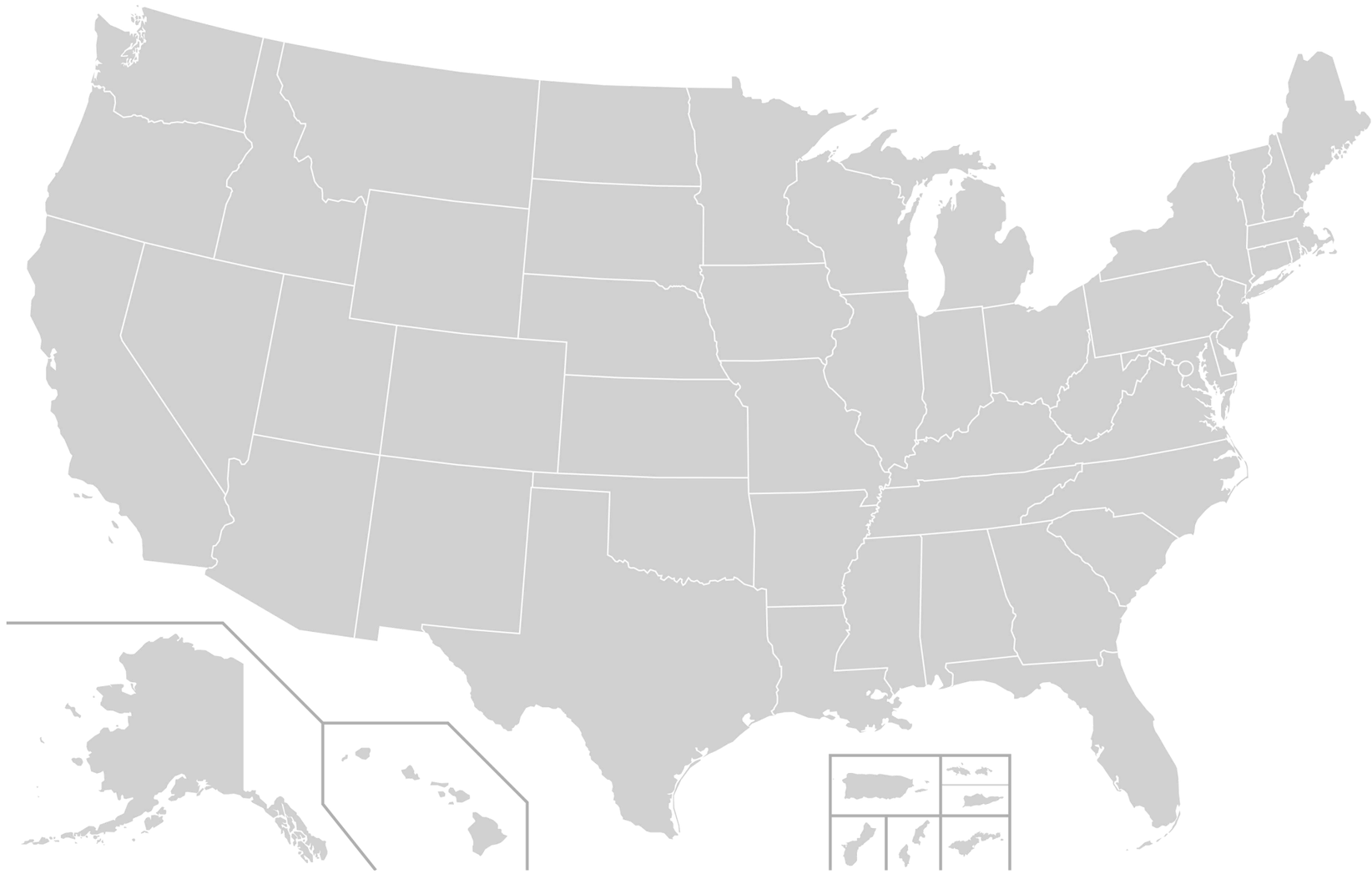
We might be able to use fewer colors, but this bound helps us develop an algorithm for coloring the nodes of a graph.

1. Determine the maximum degree k of the graph. We know we need at most $k + 1$ colors.
2. Iterate through all possible colors i ($0 \leq i < k + 1$).
    - Iterate through all nodes with an *unassigned color*.
        - Check if color i can be used for unassigned node u: if any node v adjacent to u is assigned color i, then we cannot use color i for node u.

It also helps to sort the nodes by decreasing degree beforehand. Try it out!



Exercise: complete the code in the notebook (see "exercise" link in the row for today's class) to assign a color to each state.

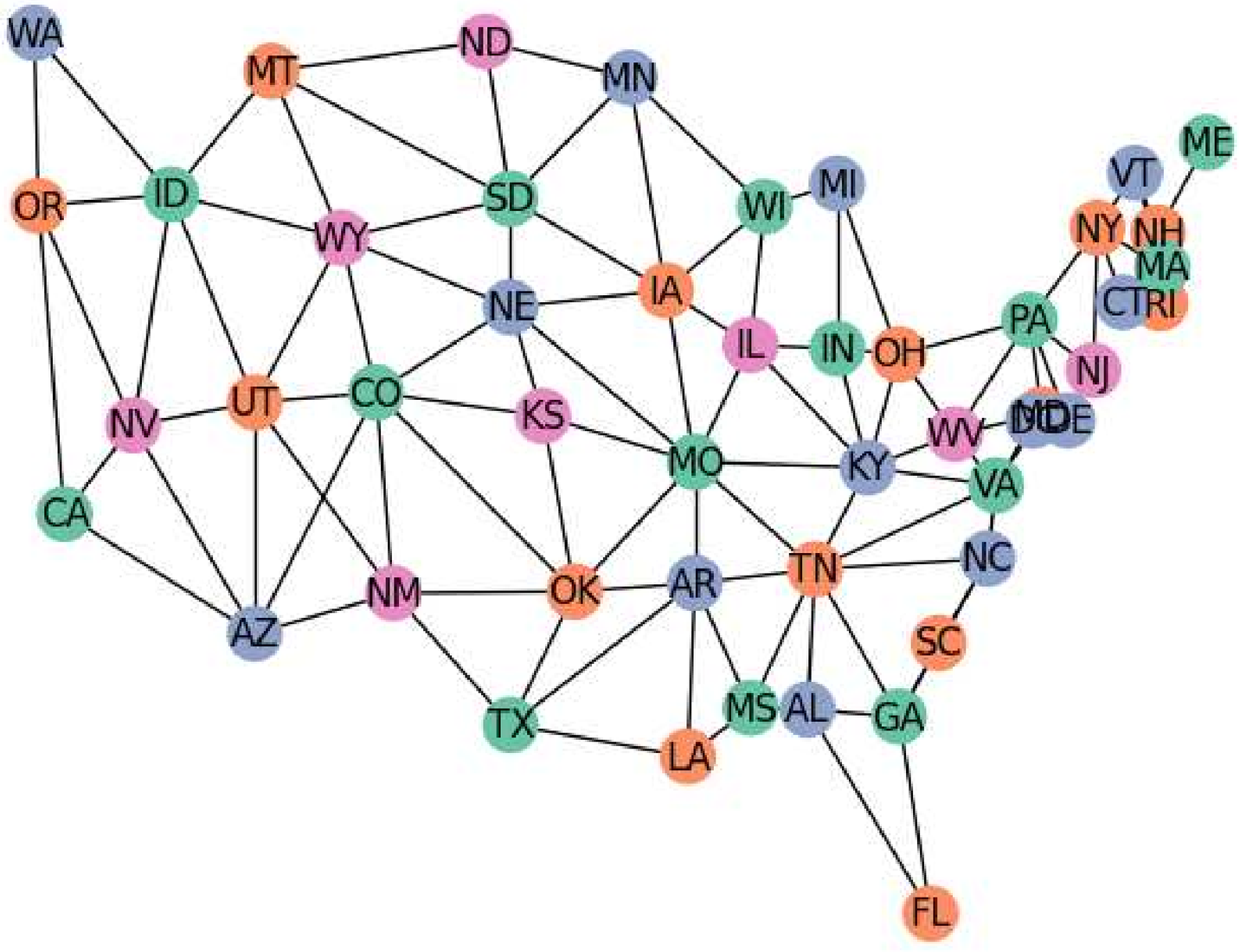


Possible **Python** code for our graph coloring algorithm.

```python
def color_graph(adj):
  nodes = list(adj.keys())
  colors = {}
  degrees = {node: len(adj[node]) for node in adj}
  nodes = sorted(nodes, key=lambda u: degrees[u], reverse=True)

  k = len(adj[max(adj, key=lambda u: len(adj[u]))])
  for i in range(k + 1):
    if len(colors) == len(nodes):
      print(f"Completed using {i} colors (of max = {k + 1}).")
      break
    for u in nodes:
      if u in colors:
        continue
      color_ok = True
      for v in adj[u]:
        if v in colors and colors[v] == i:
          color_ok = False
          break
      if color_ok:
        colors[u] = i
  return colors
```

The result!


WA
MT
ND
MN
OR
ID
SD
WI
MI
VT
ME
NY
NH
MA
CT
RI
WY
IA
NE
IL
IN
OH
PA
NJ
NV
UT
CO
KS
MO
KY
WV
MD
DE
VA
CA
NC
NM
OK
AR
TN
SC
AZ
TX
MS
AL
GA
LA
FL