A Touch of Color

Graph Coloring

FIGURE 2-1

The map in the figure above is clearly a map of the 48 lower states of the United States, but it lacks a little charm. Maps are supposed to be visually pleasing, and the typical way to make them so is to add a touch of color. The standard approach to coloring a map is to use a single color for a state (we will use "state" as a metaphor for the geographical units in the map, which in reality could be countries, provinces, etc.), and never use the same color for two states that share a common border. On the other hand, two states whose common border is just one point (for example Arizona and Colorado) can be colored, if we so choose, with the same color.

The colorful map shown in Fig. 2-2(a) is a bit too colorful: 48 colors were used, one for each state. First, it's a little jarring to see that many colors—a classic case of too much of a good thing. Second, let's imagine that there is a production cost incurred each time we add another color to the map, so that a map that uses say 48 colors is more expensive than a map that uses 47 colors, which in turn is more expensive than a map that uses 46 colors, and so on. (In the old days this was probably true, but with modern laser and inkjet printers the argument doesn't hold up. On the other hand, the cost argument gives us a convenient way to think about the issues in map coloring, so we will pretend it's still true.) A more typical, "less expensive" colored map is shown in Fig. 2-2(b). Here a total of five different colors are used. Is it possible to cut down the number of colors used to four? How about three? You are encouraged to try answering these questions before you read on. (For a convenient way to experiment, go to *http://www.sailor.lib.md.us/MD_topics/kid/col_applet/color.html*. If you choose "US Map" from the drop down menu that shows "Flag," and then click on "Load Image" you will get a blank map of the lower 48 states and a palette of colors for coloring the map.)

See Exercise 16.

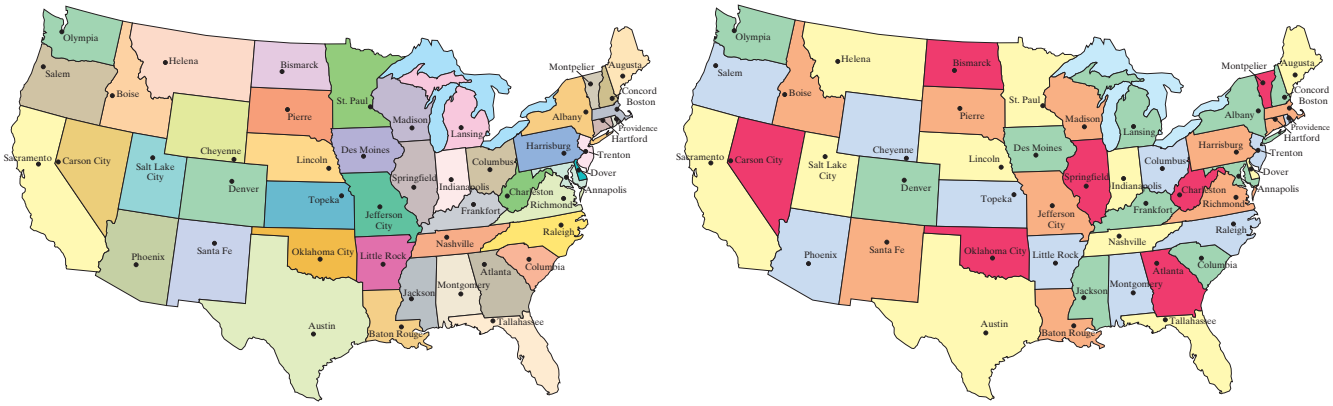


FIGURE 2-2 **(a)** **(b)**

Much has been written about the mathematics of map coloring and its connection with graph theory. In this mini-excursion we will explore some of the ideas behind the fascinating topic of graph coloring and some of its more colorful applications. Some familiarity with the basic terminology and concepts in Chapters 5 and 6 is strongly recommended.

Graph Coloring and Chromatic Numbers

EXAMPLE 2.1 Can't We All Get Along?

Sometimes people just don't get along, and you are caught in the middle. Imagine that you are a wedding planner organizing the rehearsal dinner before a big wedding. There are a total of 16 people attending the rehearsal dinner: $A, B, C, \ldots H$ are relatives of the bride and groom; $I, J, K, \ldots P$ are members of the wedding

party. If things weren't stressful enough, you are told that some of these people have serious issues:

- A doesn't get along with F, G, or H,
- B doesn't get along with C, D, or H,
- C doesn't get along with B, D, E, G, or H,
- D doesn't get along with B, C, or E,
- E doesn't get along with C, D, F, or G,
- F doesn't get along with A, E, or G,
- G doesn't get along with A, C, E, or F,
- H doesn't get along with A, B, or C.

To make the rehearsal dinner go smoothly you are instructed to find a way to seat these people so that people that don't get along must be seated at different tables. (I through P get along with everyone, so they are not a concern.) How are you going to set up the seating arrangements with so many incompatibility issues to worry about? What is the minimum number of tables you will need? You can answer both of these questions using a little graph theory.

We will start by creating the "incompatibility" graph shown in Fig. 2-3(a). In this graph the vertices represent the individuals, and two vertices are connected by an edge if the corresponding individuals don't get along (and therefore, should not be seated at the same table). To make seating assignments we assign colors to the vertices of the graph, with each color representing a table. Since we don't want two incompatible individuals seated at the same table, *we don't want to color two vertices that are connected by an edge with the same color*. We will refer to any coloring that satisfies this rule as a *legal coloring* of the graph.

Figure 2-3(b) shows a legal coloring of the vertices of the graph in Fig. 2-3(a) that uses four different colors. This coloring tells us how to seat this obnoxious group using four tables. Can we do better (i.e. use less than four tables)? Yes. Figure 2-3(c) shows a legal coloring of the vertices of the graph that uses just three colors.

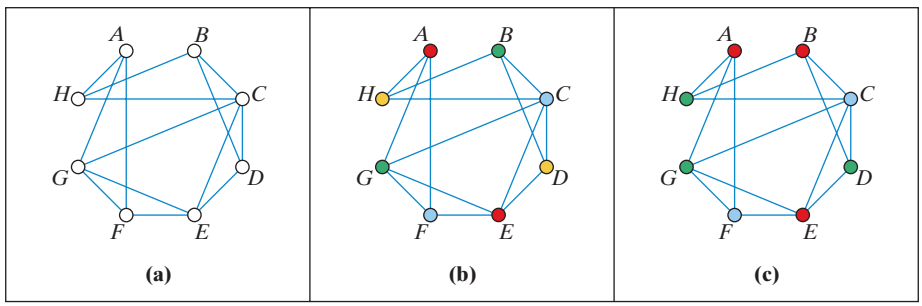


FIGURE 2-3

Could we possibly come up with a legal coloring of the vertices of the graph that uses just two colors? No. The reason is clear if we look at A, F, and G (or any other set of three vertices that form a *triangle*). Since each is adjacent to the other two, A, F, and G will have to be colored with different colors.

The conclusion to our analysis is: (i) the minimum number of tables needed to sit the wedding party is three, and (ii) the seating assignment should put A, B, and E in one table (red), C and F in a second table (blue), and D, G, and H in the third table (green). The remaining members of the wedding party can be arbitrarily assigned to fill up the remaining seats at the three tables.

This is, in fact, the unique solution to the problem—see Exercise 12.

Example 2.1 illustrates the two key ideas of this mini-excursion: (i) the idea of coloring the vertices of a graph so that *adjacent vertices don't get the same color*, and (ii) the concept of trying to do this using as *few colors as possible*. Here are the formal definitions of these concepts.

Note that the actual choice of colors used in coloring the graph is irrelevant. In fact, the colors are just a convenient way to classify the vertices, and we could just as well use numbers or letters to do the same. The use of colors is based primarily on the fact that humans are better able to perceive patterns of colors than patterns of symbols.

k-coloring

A ***k*-coloring** of a graph G is a coloring of the vertices of G using k colors and satisfying the requirement that adjacent vertices are colored with different colors.

Chromatic Number

The **chromatic number** of a graph G is the *smallest* number k for which a k-coloring of the vertices of G is possible. We will use the notation $\chi(G)$ to denote the chromatic number of G.

Using the above notation, for the graph G in Fig. 2-3(a) we have $\chi(G) = 3$. This follows from the observation that a 3-coloring of G is possible, as shown in Fig. 2-3(c), but a 2-coloring of G is not.

EXAMPLE 2.2 Coloring Complete Graphs

The graph in Fig. 2-4(a) is K_5, the complete graph on 5 vertices. In this graph every vertex is adjacent to every other vertex, so no two vertices can have the same color. The only possible way to color K_5 is to use a different color for each vertex, as in Fig. 2-4(b). Thus, we can conclude that $\chi(K_5) = 5$.

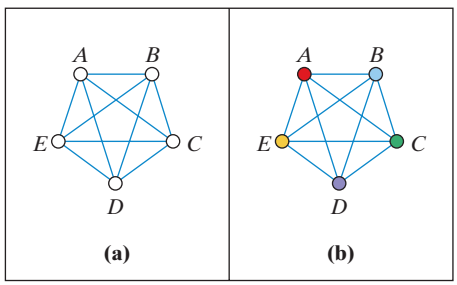


FIGURE 2-4

The argument used in Example 2.2 can be generalized to show that $\chi(K_n) = n$. This illustrates the fact that it is possible to create graphs with arbitrarily large chromatic numbers. You need a graph G with $\chi(G) = 100$? No problem—choose G to be K_{100}.

EXAMPLE 2.3 Coloring Circuits

A graph consisting of a single circuit with n vertices is denoted by C_n. The graph shown in Fig. 2-5(a) is C_6. Figure 2-5(b) shows a 2-coloring of C_6. Since a 1-coloring is clearly out of the question, we can conclude that $\chi(C_6) = 2$.

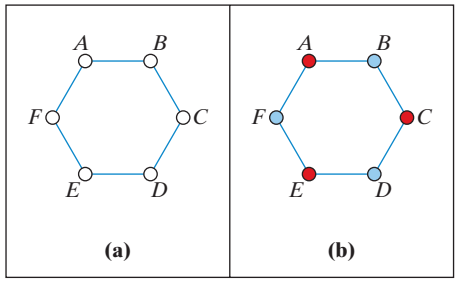


FIGURE 2-5

The graph shown in Fig. 2-6(a) is C_5. Figure 2-6(b) shows a 3-coloring of C_5. A little reflection should be enough to convince ourselves that we will not be able to color C_5 with just two colors, as we did with C_6. The problem is that we can alternate two colors around the circuit until we get to the last vertex, but since the number of vertices is odd, the last vertex will be adjacent to two vertices of different colors, and a third color will be needed. It follows that $\chi(C_5) = 3$.

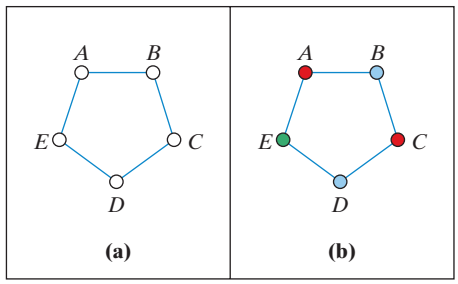


FIGURE 2-6

See Exercise 9.

We can generalize the preceding observations to any $C_n (n \geq 3)$: If n is even, $\chi(C_n) = 2$; if n is odd, $\chi(C_n) = 3$. Thus, we now know that it is possible to have graphs with a large number of vertices and small chromatic number (2 or 3). Graphs with a large number of vertices and chromatic number 1 are also possible, but are extremely uninteresting—they have no edges.

See Exercise 8.

The Greedy Algorithm for Graph Coloring

Like some of the other graph problems discussed in Chapters 5 through 8 (Euler circuit problems, traveling salesman problems, shortest network problems, scheduling problems) graph coloring can be thought of as an *optimization* problem: How can we color a graph using the fewest possible number of colors? We will call such a coloring an *optimal coloring* of the graph, and the general problem of finding optimal colorings is known as the *coloring problem*.

Optimal Coloring

An **optimal coloring** of a graph G is a coloring of the vertices of G using the fewest possible number of colors. To put it in slightly more formal terminology, an optimal coloring of G is a $\chi(G)$-coloring of G. [Too many G's in the last sentence, but all it says is that when we color a graph using $\chi(G)$ colors by definition we have an optimal coloring of the graph.]

The truly interesting question in graph coloring is the following: Given an arbitrary graph G, how do we find an optimal coloring of G? If you are a veteran of Chapters 5 through 8, you may not be entirely surprised to find out that *no efficient general algorithm is known for finding optimal colorings of graphs*. For small graphs we can use trial and error, and for some families of graphs optimal coloring is reasonably easy (for example complete graphs, circuits, etc.) but those are just special cases. In general, the best we can do is to use *approximate algorithms* that hopefully get us close to an optimal coloring.

In this section we will discuss a classic approximate algorithm for graph coloring known as the *greedy algorithm* (for reasons that will become clear soon). To illustrate the ideas behind the greedy algorithm we will start with a couple of simple examples.

EXAMPLE 2.4 Greed Is Good (Sometimes)

Let's try to color the graph in Fig. 2-7(a). The strategy we will use is simple. We will start with vertex A and color it with some color, say blue. We will think of blue as the *first* color in some arbitrary priority list of colors (blue goes first, red second, green third, yellow fourth, and so on.) We now move to the next vertex, B and try to color it with the first color in our list (blue), but we can't do it because B is adjacent to A, which is already blue. OK, fine—we'll just go to the next available color on the list (red) and use it for B. If we go next to vertex C how should we color it? Other than red (C is adjacent to B) we can use any color we want, but why introduce a third color when we can use blue? We are, after all, trying to cut down on the number of colors used. So we color C with blue. Using the same philosophy (try the colors in the designated order) we color D with red, and so on. Everything lines up just right and we can alternate blue and red and get the 2-coloring shown in Fig. 2-7(b). Since we know that the graph cannot be colored with one color, Fig. 2-7(b) gives an optimal 2-coloring of the graph.

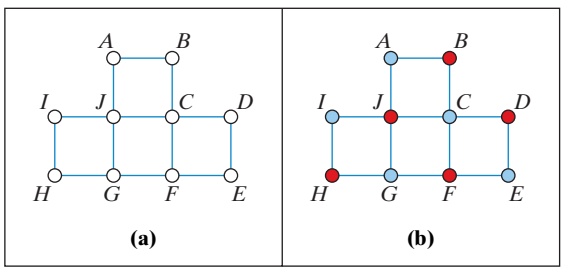


FIGURE 2-7

The strategy we used in Example 2.4 has two key components: (i) we color the vertices of the graph following some designated order (in Example 2.4 we went through the vertices in alphabetical sequence, but it can be done in any order we choose) and (ii) we have a priority list for assigning the colors, and we limit the colors used by always starting at the top of the list and using previously used colors as much as possible. For obvious reasons, we call this a *greedy strategy*.

In Example 2.4 the greedy strategy gave us an optimal coloring, but there was a bit of good karma involved. By pure luck, the order in which we colored the vertices just worked out perfectly. This is not always the case, and our next example shows how bad luck in the order of the vertices can give us a bad coloring of the graph.

EXAMPLE 2.5 Greed Can Sometimes Be Bad

The graph in Fig. 2-8(a) is the same graph as the one in Fig. 2-7(a) except for the way the vertices are labeled. If we try the same approach we used in Example 2.4 (color the vertices in alphabetical order and use the same priority order for colors—blue first, red second, green third, yellow fourth) we will get a very different result. We start by coloring vertex A with blue. So far so good. Now B is not adjacent to A, so we will also color it with blue. Ditto for C and D [Fig. 2-8(b)]. So far we have used only one color, so we are doing well. When we get to vertex E we have to go down to our second color because E is adjacent to D, which has already been colored blue. Thus, E is colored red. Likewise, F is adjacent to a blue vertex, so we color it red. We are now looking at Fig. 2-8(c). Since G is now adjacent to a blue and a red vertex, we have to pull out a third color (green) for G [Fig. 2-8(d)]. Now you can clearly see our bad luck. The next vertex (H) is adjacent to a blue, a red, and a green vertex so we need a fourth color (yellow) to color it [Fig. 2-8(e)]. The last two vertices (I and J) are adjacent to blue vertices only, so we will color them red. This gives us the final coloring of the graph [Fig. 2-8(f)].

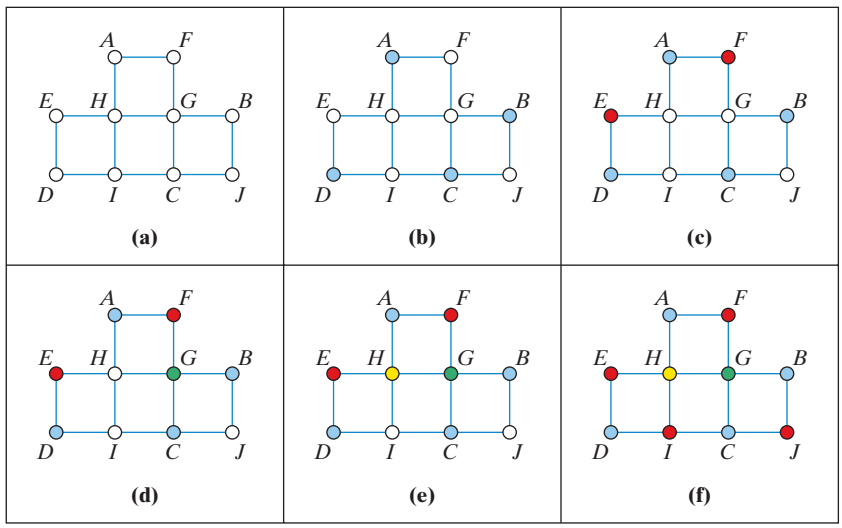


FIGURE 2-8

Examples 2.4 and 2.5 illustrate an important point: The order in which the vertices of the graph are colored can make a big difference in the kind of coloring we get. There is always some way to order the vertices so that we get an optimal coloring, but there is no easy way to figure out what that order is. Moreover, there are $n!$ different ways to order the n vertices, so going through the different orderings to find the best one is out of the question. (Take a quick look at Table 6-4 in Chapter 6 if you need to jog your memory on how quickly factorials grow.)

See Exercise 21.

We will now formalize the ideas introduced in Examples 2.4 and 2.5 into what is known as the *greedy algorithm* for graph coloring. To implement the algorithm we assume that the vertices are ordered in some arbitrary order $v_1, v_2, \ldots, v_n$, and that $c_1, c_2, c_3, c_4, \ldots$ represents a priority order for the colors.

Greedy Algorithm for Graph Coloring

- **Step 1.** Assign the first color (c_1) to the first vertex (v_1).
- **Step 2.** Vertex v_2 is assigned color c_1 if it is not adjacent to v_1; otherwise it gets assigned color c_2.
- **Steps 3, 4, . . . , *n*.** Vertex v_i is assigned the first possible color in the priority list of colors (i.e. the first color that has not been assigned to one of the already colored neighbors of v_i).

In spite of its limitations, the greedy algorithm is a reasonable approach for graph coloring, especially when we use some ingenuity in the way we order the vertices. Since there are more restrictions in coloring vertices of higher degree than there are in coloring vertices of lower degree, one obvious refinement of the greedy algorithm is to order the vertices in *decreasing* order of degrees: color vertices of highest degrees first (if there are ties choose among them at random), color vertices of next highest degree next, and so on. If we were to apply this approach to the graph in Fig. 2-8 one possible ordering of the vertices would be: *G*, *H*, *C*, *I*, *A*, *B*, *D*, *E*, *F*, *J*. If we were to color the vertices in this order using the greedy algorithm we would get an optimal coloring of the graph in Fig. 2-8.

See Exercise 20.

The greedy algorithm can also be used to produce upper bounds on the chromatic number of the graph. The best known of these upper bounds is given by the following fact, known as Brook's Theorem.

Brook's Theorem

Let $d_1 \geq d_2 \geq \cdots \geq d_n$ be the degrees of the vertices of the graph G listed in decreasing order. Then $\chi(G) \leq d_1 + 1$.

Brook's Theorem essentially says that *the chromatic number of a graph cannot be more than the largest degree in the graph plus one*. To see why this is true, think about the worst case scenario we can run into when we are coloring a vertex using the greedy algorithm: the vertex is adjacent to many other vertices all of which have been colored with different colors. Since the largest number of adjacent vertices a vertex can have is d_1, the worst thing that could happen is that the first d_1 colors have been used and we would need one more to color our vertex (Fig. 2-9).

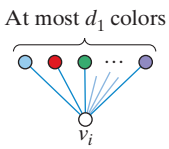


FIGURE 2-9

It turns out that for *connected graphs*, the only two cases where we actually have to use the maximum $d_1 + 1$ colors to color the graph are when the graph is the complete graph K_n (all vertices have degree $n - 1$, the chromatic number is n), or the circuit C_n where n is odd (all vertices have degree 2, the chromatic number is 3). If we rule these two cases out, we no longer need the $+1$ in Brook's Theorem. We will call this the *strong version* of Brook's Theorem.

Brook's Theorem (Strong Version)

Let $d_1 \geq d_2 \geq \cdots \geq d_n$ be the degrees of the vertices of a *connected* graph G listed in decreasing order. If G is not C_n (n odd) or K_n, then $\chi(G) \leq d_1$.

Graph Coloring and Sudoku

The game of Sudoku has become, in the last few years, the rage among puzzle and game enthusiasts looking for a more intellectual (and cheaper) challenge than the one provided by an X-Box. Sudoku is addictive, and even ordinary people that are not drawn to video games are hooked on it. These days practically every major newspaper carries a daily Sudoku puzzle.

If you haven't played Sudoku yet, the rules are quite simple: You start with a 9×9 grid of 81 squares called *cells*. The grid is also subdivided into nine 3×3 subgrids called *boxes*. Some of the cells are already filled with the numbers 1 through 9. These are called the *givens*. The challenge of the game is to complete the grid by filling the remaining cells with the numbers 1 through 9. The requirements are: (i) every row and every column of the grid must have the numbers 1 through 9 appear once; (ii) each of the nine boxes must have the numbers 1 through 9 appear once.

A typical Sudoku puzzle may have somewhere between 25 and 40 givens, depending on the level of difficulty. Figure 2-10 is an example of a moderately easy Sudoku puzzle. (*Source*: *http://www.geometer.org/mathcircles*). The labels 1 through 9 on the columns and *a* through *i* on the rows are not part of the puzzle, but they provide a convenient way to refer to the cells. Just for fun, you may want to try this one out before you read on. (*Hint*: Try to figure out what number should go in cell *f*2. Once you have that one figured, go to cell *d*3. That's enough help for now. The solution is shown after the References and Further Readings.)

	1	2	3	4	5	6	7	8	9
a			4	8					
b		9		4	6			7	
c		5					6	1	4
d	2	1		6			5		
e	5	8		7		9		4	1
f			7			8		6	9
g	3	4	5					9	
h		6			3	7		2	
i						4	1		

FIGURE 2-10

The Sudoku Graph

To see the connection between Sudoku and graph coloring, we will first describe the **Sudoku graph**, which for convenience we will refer to as *S*. The graph *S* has 81 vertices, with each vertex representing a cell. When two cells *cannot* have the same number (either because they are in the same row, in the same column, or in the same box) we put an edge connecting the corresponding vertices of the Sudoku graph *S*. For example, since cells *a*3 and *a*7 are in the same row, there is an edge joining their corresponding vertices; there is also an edge connecting *a*1 and *b*3 (they are in the same box), and so on. When everything is said and done, each vertex of the Sudoku graph has degree 20, and the graph has a total of 810 edges. *S* is too large to draw, but we can get a sense of the structure of *S* by looking at a partial drawing such as the one in Fig. 2-11. The drawing shows all 81 vertices of *S*, but only two (*a*1 and *e*5) have their full set of incident edges showing.

See Exercise 22.

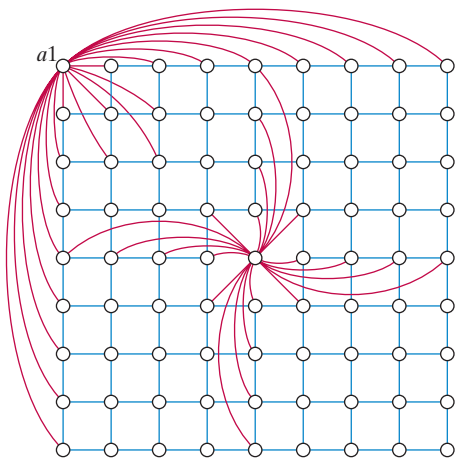


FIGURE 2-11 A partial drawing of the Sudoku graph

The second step in converting a Sudoku puzzle into a graph coloring problem is to assign colors to the numbers 1 through 9. This assignment is arbitrary, and is not a priority ordering of the colors as in the greedy algorithm —it's just a simple correspondence between numbers and colors. Figure 2-12 shows one such assignment.

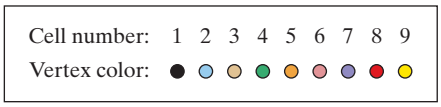


FIGURE 2-12

Once we have the Sudoku graph and an assignment of colors to the numbers 1 through 9, any Sudoku puzzle can be described by a Sudoku graph where some of the vertices are already colored (the ones corresponding to the givens). For example, the Sudoku puzzle shown in Fig. 2-10 is equivalent to the partial coloring shown in Fig. 2-13. To solve the Sudoku puzzle all we have to do is color the rest of the vertices using the nine colors in Fig. 2-12.

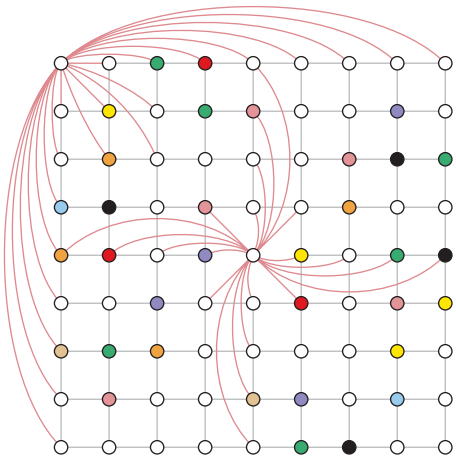


FIGURE 2-13 A Sudoku puzzle as a graph coloring problem

Map Coloring

Map drawing and coloring is an ancient art, but the connection between map coloring and mathematics originated in 1852 when a University of London student by the name of Francis Guthrie mentioned to his mathematics professor (the well known mathematician Augustus De Morgan) that he had been coloring many maps of English counties (don't ask why) and noticed that every map he had tried, no matter how complicated, could be colored with four colors (where districts with a common border had to be colored with different colors). He inquired if this was a known mathematical fact, and De Morgan (who didn't know the answer) checked with some of the most famous mathematicians of the time, including his friend William Rowan Hamilton.

A short biographical profile of Hamilton is given in Chapter 6.

Guthrie's notion that any map could be colored with just four colors, sounded so simple that everyone assumed it could be easily proved mathematically. After 100 years and many failed attempts at a proof, the *Four-Color Conjecture*, as the problem was famously known, was finally solved in 1976 by Kenneth Appel and Wolfgang Haken of the University of Illinois. Yes, indeed, Guthrie was right: Every map can be colored with four colors or less. The solution to this simple question took up 500 pages and about 1000 hours of computer time.

Any map coloring problem can be converted into a graph coloring problem by first finding the **dual graph** of the map. In the dual graph, each vertex represents a "state" (remember that this is a metaphor for the political units used in the map), and two vertices are connected by an edge if the corresponding states have a common border. If the common border happens to be just a point (as is the case with Arizona and Colorado), then we do not connect the vertices. The problem of coloring the map using the fewest number of colors now becomes simply the problem of *finding an optimal coloring of the vertices of the dual graph*. We can now take all the tools and techniques we learned for graph coloring and use them for map coloring.

Our final example is a very small example whose only purpose is to illustrate the idea behind coloring a map by means of its dual graph.

For a slightly more meaningful map coloring problem, see Exercise 15.

EXAMPLE 2.6 Map Coloring with Dual Graphs

Figure 2-14(a) shows a small map with six states, and Fig. 12-14(b) shows its dual graph. Figure 2-14(c) shows an optimal 3-coloring of the dual graph. We know that the 3-coloring is optimal because the graph has triangles, and thus cannot be colored with 2 colors. The corresponding optimal coloring of the map is shown in Fig. 2-14(d).

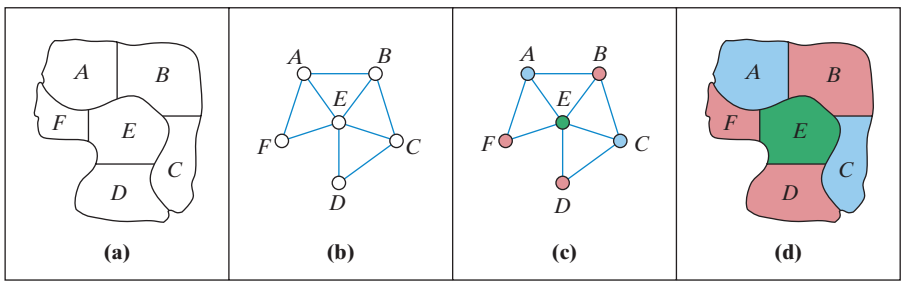


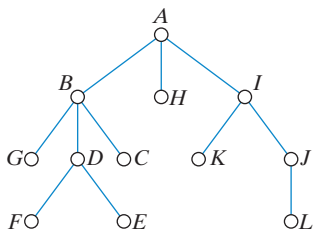
FIGURE 2-14

Exercises

A. Graph Colorings and Chromatic Numbers

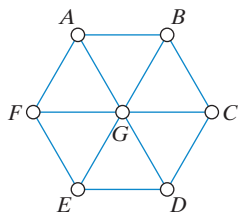
For Exercises 1 through 6, you can experiment with coloring the graphs using a Java applet available at: http://www.cut-the-knot.org/Curriculum/Combinatorics/ColorGraph.shtml. The applet was designed specifically to match these exercises.

1. For the graph G shown in the figure,



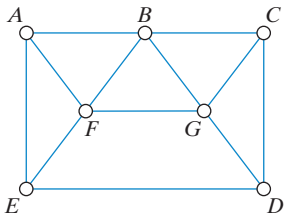
(a) find a 3-coloring of G.
(b) find a 2-coloring of G.
(c) find $\chi(G)$. Explain your answer.

2. For the graph G shown in the figure,



(a) find a 4-coloring of G.
(b) find a 3-coloring of G.
(c) find $\chi(G)$. Explain your answer.

3. For the graph G shown in the figure,

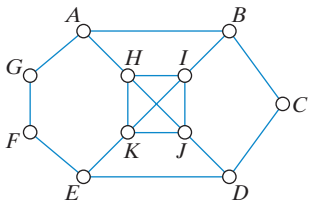


(a) find a 5-coloring of G.

(b) find a 4-coloring of G.

(c) find $\chi(G)$. Explain your answer.

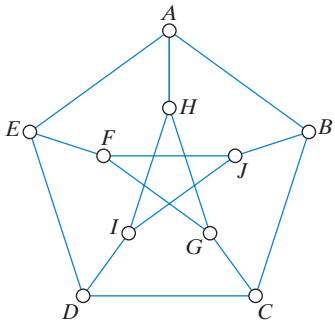
4. For the graph G shown in the figure,



(a) find a 4-coloring of G.

(b) find $\chi(G)$. Explain your answer.

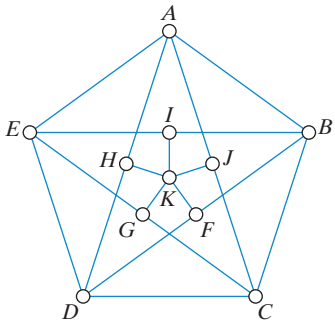
5. For the graph G shown in the figure,



(a) find a 3-coloring of G.

(b) find $\chi(G)$. Explain your answer.

6. For the graph G shown in the figure below,



(a) find a 4-coloring of G.

(b) find $\chi(G)$. Explain your answer.

7. Let K_n denote the complete graph on n vertices.

(a) Explain why $\chi(K_n) = n$ (see Example 2.2).

(b) Let G be a graph obtained by removing just one edge from K_n. Find $\chi(G)$. Explain your answer.

8. Explain why if $\chi(G) = 1$, then G consists of just isolated vertices.

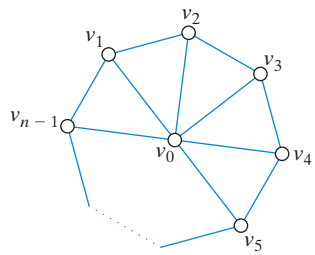
9. Let C_n denote the circuit with n vertices (see Example 2.3).

(a) When n is even, $\chi(C_n) = 2$. Describe a 2-coloring of C_n.

(b) When n is odd, $\chi(C_n) = 3$. Describe a 3-coloring of C_n. Explain why a 2-coloring is impossible.

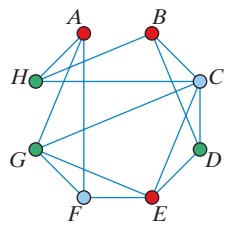
10. Let W_n denote the "wheel" with n vertices, as shown in the figure. Find $\chi(W_n)$. Explain your answer.

(***Hint:*** *You should do Exercise 9 first.*)



11. If G is a tree (i.e., a connected graph with no circuits), find $\chi(G)$. Explain your answer.

12. The graph shown below is the graph discussed in Example 2.1. Explain why, except for a change of colors, the 3-coloring shown in the figure below is the only possible 3-coloring of the graph.



B. Map Coloring

13. Give an example of a map with 10 states that can be colored with just two colors.

14. Give an example of a map with 4 states that cannot be colored using less than four colors.

15. Find a map of South America and then:

(a) Find the dual graph for the map.

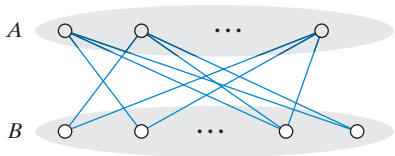
(b) Use the greedy algorithm with the vertices ordered by decreasing order of degree to color the dual graph found in (a).

(c) Find the chromatic number of the dual graph found in (a). What does this imply about the coloring of the original map of South America?

16. Color a map of the lower 48 states in the United States using 4 colors. Explain why it is impossible to color the map using less than four colors. (You might find it convenient to do the coloring online at *http://www.sailor.lib.md.us/MD_topics/kid/col_applet/color.html*. If you choose "U.S. Map" from the drop-down menu that shows "Flag," and then click on "Load Image" you will get a blank map of the lower 48 states and a palette of colors for coloring the map.)

C. Miscellaneous

17. A **bipartite graph** is a graph whose vertices can be separated into two sets A and B such that every edge of the graph joins a vertex in A to a vertex in B. A generic bipartite graph is illustrated in the figure.



(a) Explain why if G is a bipartite graph, $\chi(G) = 2$.

(b) Explain why if $\chi(G) = 2$, then G must be a bipartite graph.

(c) If G is a bipartite graph, then every circuit in G must have an even number of vertices. True or False? Explain.

18. Suppose G is a graph with n vertices and no loops or multiple edges such that every vertex has degree 3. (These graphs are called *3-regular* graphs.)

(a) Explain why n must be even and $n \geq 4$.

(b) Explain why $\chi(G) \leq 4$.

(c) Describe the 3-regular graphs for which $\chi(G) = 4$. (***Hint****: What types of 3-regular graphs can fail to meet the conditions of the strong version of Brook's theorem?)*

19. Suppose G is a connected graph with n vertices and no loops or multiple edges such that $n - 1$ vertices have degree 3, and one vertex has degree 2.

(a) Explain why n must be odd and $n \geq 5$.

(b) Explain why $\chi(G) = 3$.

*(**Hint**: Use the strong version of Brook's theorem together with Exercises 8 and 17.)*

20. This exercise refers to the graph in Fig. 2-8(a) (see Example 2.5). Show that if the vertices of the graph are ordered in decreasing order, the greedy algorithm will always give an optimal coloring of the graph.

21. Given a graph G with n vertices, there is a way to order the vertices in the order $v_1, v_2, \ldots, v_n$ so that when the greedy algorithm is applied to the vertices in that order, the resulting coloring of the graph uses $\chi(G)$ colors.

(***Hint:*** *Assume a coloring of the graph with $\chi(G)$ colors and work your way backwards to find an ordering of the vertices that produces that particular coloring.*)

22. Let S denote the Sudoku graph discussed in this mini-excursion.

(a) Explain why every vertex of S has degree 20.

(b) Explain why S has 810 edges.

(***Hint:*** *See Euler's Theorems in Chapter 5.*)

23. If you haven't done so yet, try the Sudoku puzzle given in Fig. 2-10.

Projects

Scheduling Committee Meetings of the U. S. Senate

This project is a surprising application of the graph coloring techniques we developed in this mini-excursion, and involves an important real-life problem: scheduling meetings for the standing committees of the United States Senate.

The U.S. Senate has 16 different standing committees that meet on a regular basis. The business of these committees represents a very important part of the Senate's work, since legislation typically originates in a standing committee and only if it gets approved there moves on to the full Senate. Scheduling meetings of the 16 standing committees is complicated because many of these committees have members in common, and in such cases the committee meetings cannot be scheduled at the same time. An easy solution would be to schedule the meetings of the committees all at different time slots that do not overlap, but this would eat up the lion's share of the Senate's schedule, leaving little time for the many other activities that the Senate has to take on. The optimal solution would be to schedule committees that do not have a member in common for the same time slot and committees that do have a member in common for different time slots. This is where graph coloring comes in.

Imagine a graph where the vertices of the graph represent the 16 committees, and two vertices are adjacent if the corresponding committees have one or more members in common. (For convenience, call this graph the *Senate Committees graph*.) If we think of the possible time slots as colors, a k-coloring of the Senate Committees graph gives a way to schedule the committee meetings using k different time

slots. In this project you are to find a meetings schedule for the 16 standing committees of the U.S. Senate using the fewest possible number of time slots.

[***Hints***: (1) Find the membership lists for each of the 16 standing committees of the U.S. Senate. You can find the necessary information at *http://www.senate.gov/reference/ resources/pdf/committeelist.pdf*. For the most up to date information, go to *http://www.senate.gov* and click on the Committees tab. (2) Create the Senate Committees graph. (You can use a spreadsheet instead of drawing the graph—it is quite a large graph—and get all your work done through the spreadsheet.) (3) List the vertices of the graph in decreasing order of degrees and use the greedy algorithm to color the graph.]

References and Further Readings

1. Delahaye, Jean-Paul, "The Science of Sudoku," *Scientific American*, June 2006, 81–87.
2. *http://www.cut-the-knot.org/Curriculum/Combinatorics/ColorGraph.shtml* (graph coloring applet created by Alex Bogomolny).
3. *http://www.sailor.lib.md.us/MD_topics/kid/col_applet/color.html* (U.S. map coloring applet provided by Sailor, Maryland's Public Information Network).
4. West, Douglas, *Introduction to Graph Theory*. Upper Saddle River, N.J.: Prentice-Hall, Inc., 1996.
5. Wilson, Robin, "The Sudoku Epidemic," *MAA Focus*, January 2006, 5-7.

6	2	4	8	7	1	9	5	3
1	9	3	4	6	5	8	7	2
7	5	8	3	9	2	6	1	4
2	1	9	6	4	3	5	8	7
5	8	6	7	2	9	3	4	1
4	3	7	1	5	8	2	6	9
3	4	5	2	1	6	7	9	8
8	6	1	9	3	7	4	2	5
9	7	2	5	8	4	1	3	6

Solution to Sudoku puzzle