Maggie Johnson
CS109

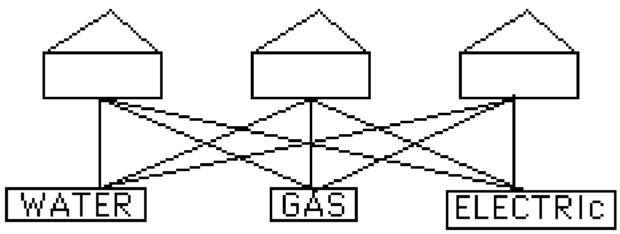
Graph Applications

Key Topics

* Planarity
* Graph Coloring
* The Chinese Postman Problem
* Instant Insanity

Planarity

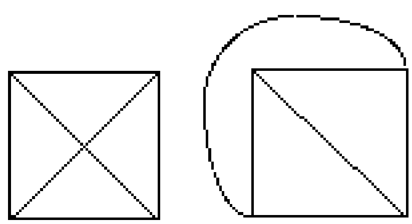
Say you have three houses, each of which must be hooked up to three utilities:



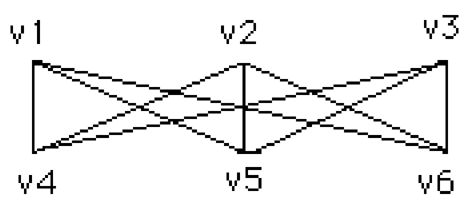
One way to represent this is with a complete bipartite graph $K_{3,3}$. Is it possible to join these houses and these utilities so that none of the connections cross? Or, a more general graph question: Can K3,3 be drawn in a plane so that no two edges cross. There are many ways to draw the same graph. A graph is defined by its vertex and edge set, not by how you choose to draw the edges.

A graph is **planar** if there is a representation of it where none of its edges cross.

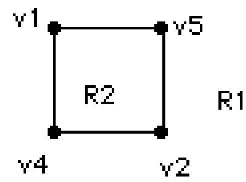
Is K_4 planar? ______________



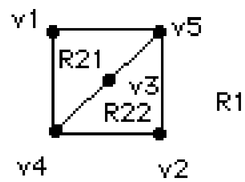
Is $K_{3,3}$ planar? ___________



Any attempt to draw $K_{3,3}$ as a planar graph won't work. In any planar representation of this bipartite graph, the vertices v1 and v2 must be connected to v4 and v5. These four edges create a closed area that divide the plane into two regions (one region inside the area R2, and one on the outside R1).

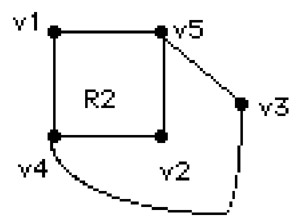


The vertex v3 must be in R1 or R2. When v3 is in R2, it subdivides R2 into two more regions: R21 and R22:



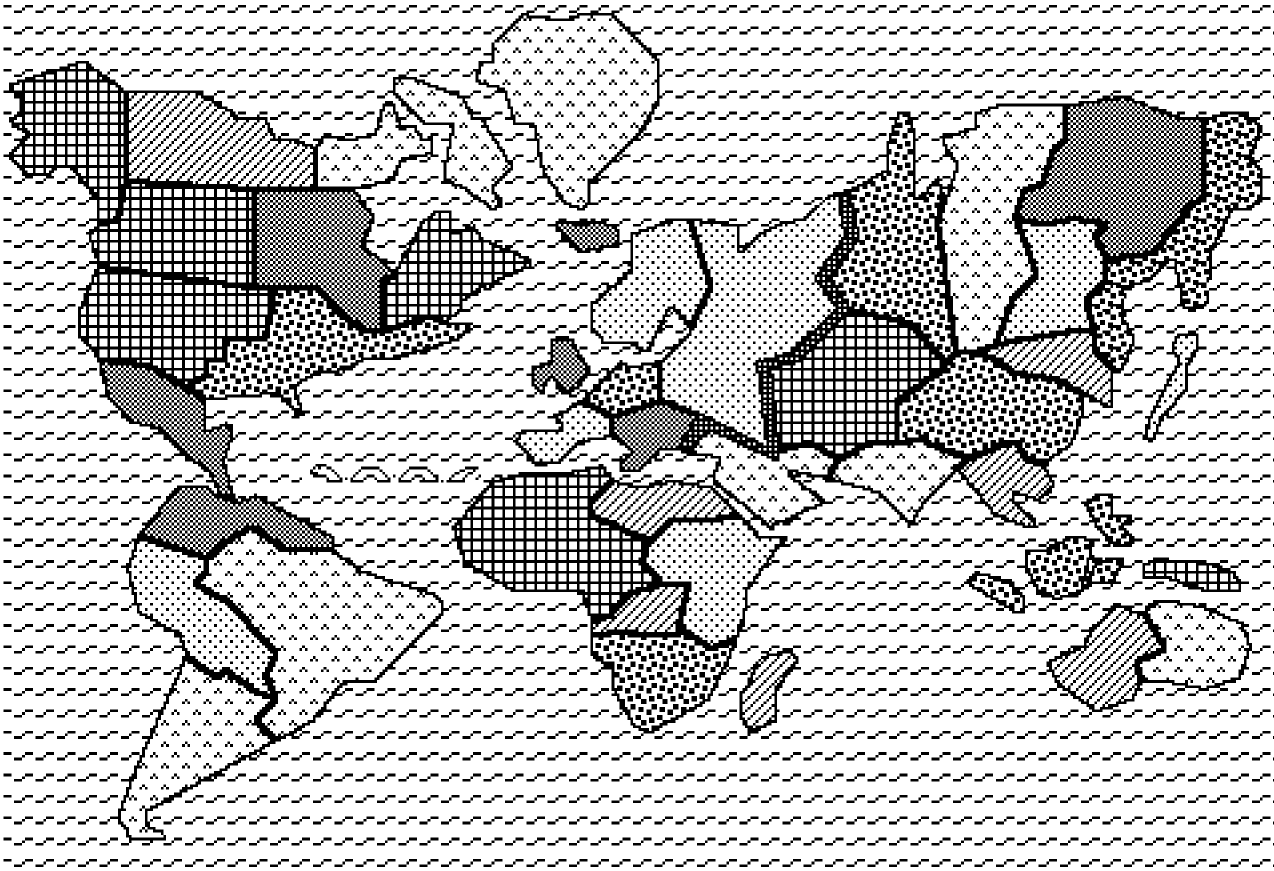
Where can we put v6 without crossing any edges? It must be connected to v1, v2 and v3. If we put it in R21, it crosses the subdivider to get to v2; if we put it in R22, same problem with v1; if we put it outside in R1, we can't get to v3.

That takes care of one case. The other case is if we had put v3 in R1 to begin with. We end up with the same problem - there's no place to put v6:



Consequently, $K_{3,3}$ is not planar.

Graph Coloring



What is the smallest number of colors (patterns) necessary to paint the map above so that no two adjacent countries are the same color? Like virtually every other problem in the universe, this problem reduces to a graph problem, where nodes represent countries and edges represent adjacency.

A **coloring** of a graph *G* assigns colors to the vertices of *G* so that adjacent vertices are given different colors.
The **chromatic number** of a graph *G* is the minimal number of colors required to color *G*. The chromatic number of a graph *G* is denoted c(*G*).

What is the chromatic number of a tree? ___________________________________
What is the chromatic number of a circuit in a graph? _________________________

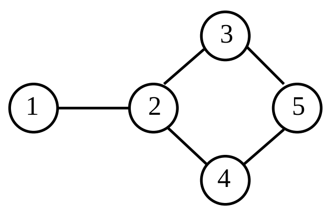
Graph coloring is a useful model for many different types of problems. For example, in scheduling: Consider the problem of attempting to schedule courses so that the number of conflicts is minimized. For example, it is probably a poor idea to schedule CIV, freshman calculus, and freshman English for MWF at 8:00am. (It would be funny, though.) If we picture every course as a node, with courses that are likely to be taken concurrently connected by an edge, then the problem of scheduling reduces to coloring the graph. In this case, each color represents a (disjoint) time to offer a course.

A Greedy Algorithm for Coloring

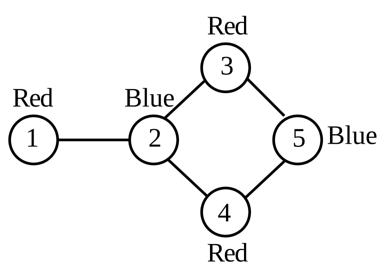
Given that graph coloring is useful for solving a variety of problems, it would be nice if we could find an efficient algorithm. Inspired by the success of greed in the algorithms of Kruskal, Prim, and Dijkstra, we might try a greedy approach to the problem of coloring a graph. The most obvious approach is described below:

A Coloring Algorithm
1) Pick a node and assign it an unused color.
2) Consider every other node, and paint it the same color if possible.
3) Repeat as necessary.

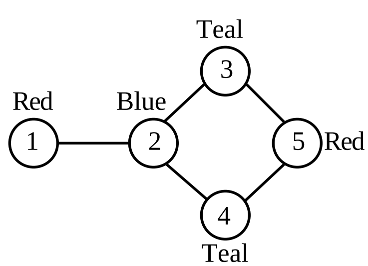
Let's consider applying this algorithm to the following graph:



If we first consider node 1 and paint it red, then we consider the other nodes in numerical order, we discover that 3 and 4 can also be painted red. Then nodes 2 and 5 can be painted blue. This solution is optimal, since it uses only two colors.



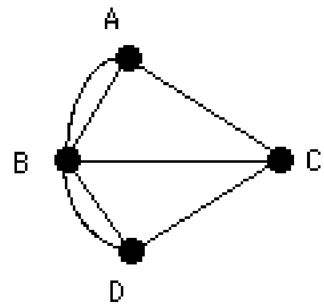
Unfortunately, the very same algorithm can produce a non-optimal solution. If the nodes are considered in the order 1, 5, 2, 3, 4, it now requires three colors:



Various heuristics and tricks can be added to the algorithm, but all (known) attempts have proven futile. The problem of determining if a graph can be colored with k colors (the k-colorability problem) is known to be NP-complete.

The Chinese Postman Problem

Recall the problem of the seven bridges of Königsberg. The graph of this problem is shown below:



Starting and ending at point A, what is the minimum number of bridges that must be crossed in order to cross each bridge at least once? A problem of this sort is called a *Chinese Postman Problem*, named after the Chinese mathematician Mei-Ko Kwan who posed the problem in 1962. We will develop a method for solving this type of problem.

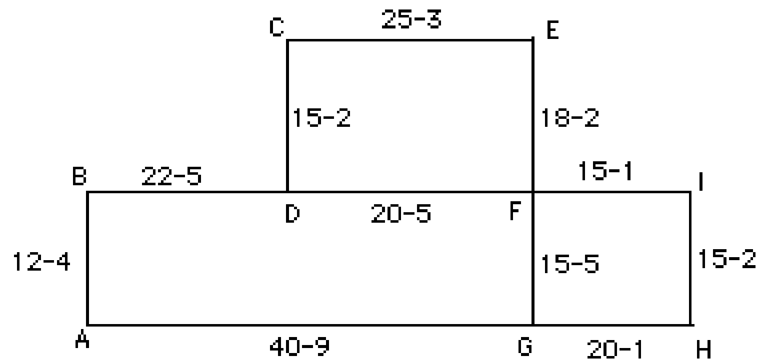
There are several applications of this. Suppose we have a road network which must be traversed by:

- a mail carrier delivering mail to buildings along the streets,
- a snowplow must clear snow from each lane of the streets,
- a police car must make its rounds through all the streets several times a day.

In each case, the person or vehicle must traverse the street at least once. In the best situation, where each vertex (road intersection) has even degree, no retracing is necessary. (Such retracing of edges is referred to as "deadheading"). In such cases, any Euler circuit solves the problem. However, it is very rare that every vertex in a road network is even. Since every road must be traced, some roads will need to be retraced. Thus, we pose a new problem: Plan a route so that the total amount of retracing is as small as possible. This gives us the following definition for the Chinese Postman Problem:

Given a connected weighted graph or digraph G, the *Chinese Postman Problem* is the problem of finding the shortest circuit that uses each edge in G at least once.

The best way to begin is by studying an example in detail. Let's say a mail carrier delivers mail along each street in the weighted graph below, starting and ending at point A. The first number on each edge gives the length of time (in minutes) needed to deliver mail along that block; the second number gives the time for traveling along the block without delivering mail (deadheading time).

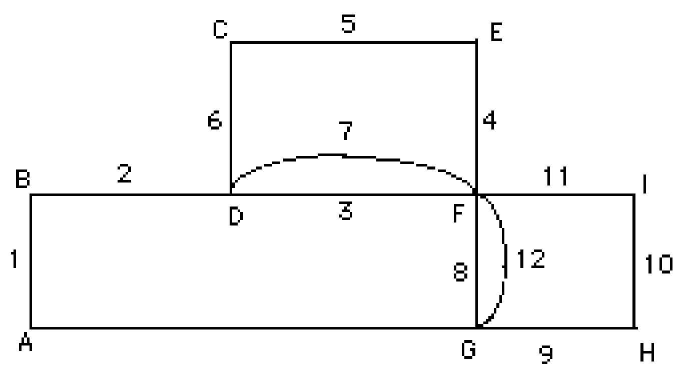


We will assume that the mail is delivered to houses on both sides of the street by traveling once down the street. Our job is to find the shortest route and minimal time required for the mail carrier to deliver the mail.

The total time spent on the route is the sum of the mail delivery times plus any deadheading time. The mail delivery time is simply the sum of the mail delivery weights on the edges = 217 minutes. The problem is now one of determining the minimum deadheading time.

Does this graph have an Euler circuit? ____

Both vertices G and D have odd degree, which means the mail carrier must retrace at least one street in order to cover the entire route. For example, s/he could retrace D-F and F-G. Let's put these retraced streets into the graph:



This new multigraph has an Euler circuit. The edges above are numbered in the order the mail carrier might follow them in the Euler circuit.

Is there a faster route? ____

To answer this question, we need to examine the possible deadheading edges that could be added. We claim that, no matter what route the mail carrier uses, the edges used for deadheading will form a path joining D and G. Because the degree of D was odd in the original graph, a deadheading edge must be incident on D. When this edge is added, it causes another vertex to go odd (F). Since F is now odd, a deadheading edge must be added that is incident on F. This must continue until a vertex that was originally odd is reached (G). Since the deadheading edges form a path joining D to G, to find the minimum weight of deadheading edges, we need to find a path of minimum weight joining D and G, by looking at our original graph.

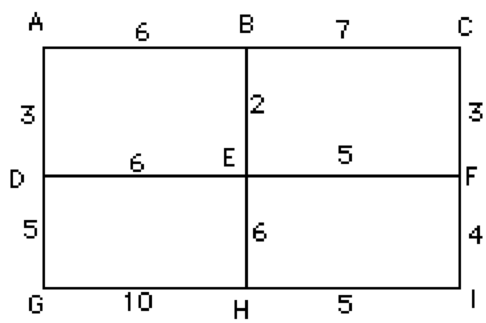
What path from D to G has minimum deadheading weight? _________________________

We can achieve minimal time by planning a route that uses the edges D-F, F-I, I-H, H-G. Any Euler circuit in this graph achieves the minimal time which is 217+9 = 226 minutes.

In this example, it was relatively simple to find the shortest path between the two vertices of odd degree. If there are lots of possibilities to consider, Dijkstra's algorithm could be used. Now the big question is: What if the graph has more than two vertices of odd degree? The following theorem will help.

Suppose G is a graph with 2k odd vertice where k >= 1, and suppose C is a circuit that traces each edge of G at least once. Then the retraced edges in C can be partitioned into k paths joining pairs of the odd vertices, where each odd vertex in G is an endpoint of exactly one of the paths.

A truck is to paint the center strip on each street in the following weighted graph. Each edge is labeled with the time (in minutes) for painting the line on that street. We want to find the minimum time to complete this job and a route that accomplishes this goal. Assume the truck begins and ends at a garage at point A, and that deadheading time in this case is equal to painting time.



There are four odd vertices in this example: B, D, F, H. Traversing each edge on this graph will require deadheading at each of these vertices. Using the theorem above with k = 2, we know that the deadheading edges form two paths joining pairs of the four odd vertices. Therefore, to solve this problem, we need to find the two paths with total weight as small as possible. We first need to list all the ways to put the four odd vertices in two pairs. Then, for each set of two pairs we find the shortest path joining the two vertices in each of the two pairs. Finally, we choose the set of pairs that has the smallest total weight of its two paths.

Here are the pairings, shortest paths and weights for the graph above:

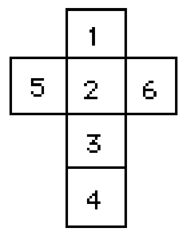
Pairing	Path	Weight	Path	Weight
B-D & F-H	B-E-D	8	F-I-H	9
B-F & D-H	B-E-F	7	D-E-H	12
B-H & D-F	B-E-H	8	D-E-F	11

Of these possible pairings, B-D, F-H has smallest total weight of 17 minutes. The total time for the job then is 62 + 17 = 79 minutes. To find a specific route that achieves this time, take the given graph and add the retraced streets as multiple edges. Then, find an Euler circuit for this multigraph.

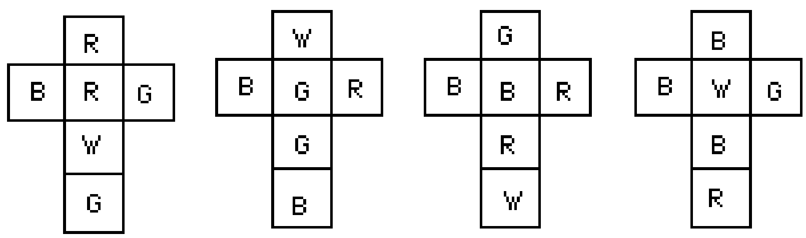
Instant Insanity

The game of Instant Insanity is played with four cubes, each of whose sides is colored one of four colors. The object of the game is to place the cubes next to each other in such a way that each of the long sides of the rectangular solid obtained has all four colors on it. The game has been in existence since 1940 when it was sold commercially under the name Tantalizer, and more recently as the popular puzzle Instant Insanity.

For simplicity, we will exhibit each cube unfolded as follows. We will want to arrange the cubes so each of the four colors appears on each of the top, front, bottom, and back sides. The colors appearing on the left and right are not relevant to the solution.



1: top; 2: front; 3: bottom; 4: back; 5: left side; 6: right side. Here is a possible solution to the puzzle given the following four cubes:

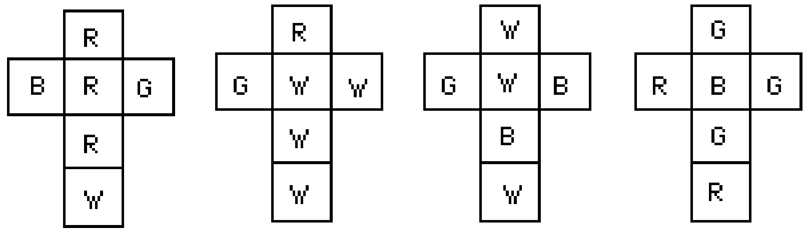


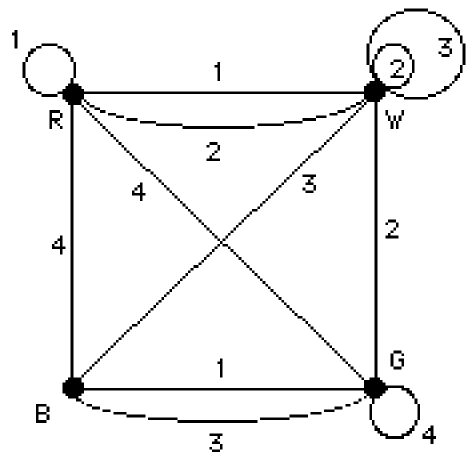
It should be fairly obvious that the difficulty in finding a solution depends greatly on the color pattern of each cube. One extreme case would be to have each cube colored a single different solid color. In this case, every arrangement would provide a solution. On the other hand, if all four cubes were colored solid red or green, no solution would exist.

Our goal is to use graph theory to simplify the search for a solution.

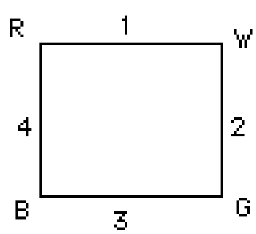
Picking an arrangement which is a solution can require quite a bit of search time, but graph theory can facilitate finding a solution by providing an analysis of what a solution looks like. Given a set of cubes with faces colored R, W, B, G, we define a graph with four vertices labeled R, W, B, G. Edges are defined as follows: if cube i has a pair of opposite faces with colors x and y, draw an edge with label i between x and y.

What is the graph for the following puzzle?

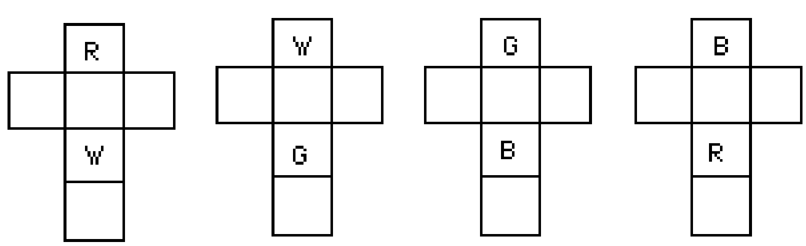




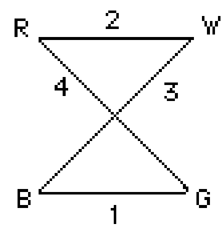
Now suppose there is a solution for the puzzle defined above. Since the set of top faces and bottom faces must each have all four colors represented, the graph can be used to trace the colors on both faces simulataneously. Consider the following subgraph of the above graph:



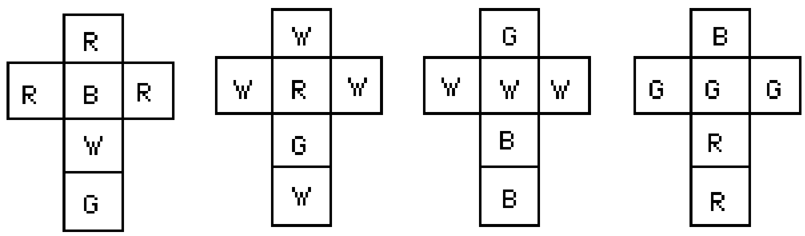
This subgraph can be used to place the top and bottom faces. Since cube 1 has a pair of opposite faces labeled R-W, place cube 1 with R as top face and W as bottom face. Cube 2 has a W-G pair, so place cube 2 with W as the top face and G as the bottom. Continue as the graph indicates which gives us the following:



The key fact about the subgraph that made this work is there are four edges, one with each of the labels 1, 2, 3, 4 and each vertex has degree 2, so that each color is represented exactly twice. Now the task is to rotate the remaining four faces to obtain a solution. This will be possible only if there is another subgraph using 4 different edges, since each edge represents a different pair of opposite faces and the top and bottom ones have already been used. Here is the other subgraph:



Using this subgraph, we get the following solution:



Bibliography

* For more on planarity, refer to the following. Much of the above on planar graphs was adapted from this source:

K. Rosen, *Discrete Mathematics and its Applications 2nd Ed.*, New York: McGraw-Hill, 1991.

* For more on graph coloring:

K. Appel, W. Haken, "Every Planar Graph is 4-Colorable," *Bulletin of the AMS*, 82 (1976), 711-712.

D. Barnette, *Map Coloring, Polyhedra, and the Four Color Problem*, Washington: Mathematical Association of America, 1983.

F.S. Roberts, *Applied Combinatorics*, Englewood Cliffs, NJ: Prentice Hall, 1984.

* For more on the Chinese Postman Problem:

J. Edmonds, "The Chinese Postman Problem," *Operations Research*, Vol. 13, Supplement 1, 1965, B73.

M. Kwan, "Graphics Programming Using Odd and Even Points," *Chinese Math*, Vol. 1, 1962, pp. 237-77.

* For more on Instant Insanity:

G. Chartrand, *Graphs as Mathematical Models,* Boston: PWS, 1977.

J. Van Deventer, "Graph Theory and 'Instant Insanity'", in *The Many Facets of Graph Theory*, G. Chartrand and S. Kapoor, editors, Berlin: Springer-Verlag, 1969.

Historical Notes

The Four Color Theorem is one of the most famous theorems in mathematics. It asserts that the chromatic number of any planar graph is no larger than four. This theorem was originally posed as a conjecture in 1852 in a letter from Augustus DeMorgan to Sir William Rowan Hamilton. It was finally proved by the American mathematicians K. Appel and W. Haken in 1976. The proof relies on a careful case-by-case analysis done by computer. They showed that if this theorem is false, there would have to be a counterexample of one of around 2000 different types. Each of these types was analyzed (using over 1000 hours of CPU time) and they concluded that the theorem is correct. (By the way, this proof is very controversial because of the use of computers: what if there is a bug in their program? Is this really a "proof"?)

Prior to 1976, many incorrect proofs of the Four Color Theorem were published often with hard-to-find errors. Perhaps the most notorious fallacious proof in all mathematics is an incorrect proof published in 1879 by a London barrister and amateur mathematician, Alfred Kempe. This was accepted as true until 1890 when the error was found.