Graph Coloring

Margaret M. Fleck

6 May 2009

This lecture discusses the graph coloring problem (section 9.8 of Rosen).

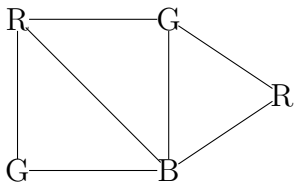
1 The problem

We're only going to talk about simple undirected graphs.

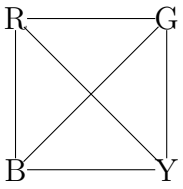
Remember that two vertices are *adjacent* if they are directly connected by an edge.

A *coloring* of a graph G assigns a color to each vertex of G, with the restriction that two adjacent vertices never have the same color. The *chromatic number* of G, written $\chi(G)$, is the smallest number of colors needed to color G.

For example, only three colors are required for this graph:



But K_4 requires 4:



2 Computing colorings

For certain classes of graphs, we can easily compute the chromatic number. For example, the chromatic number of K_n is n, for any n. Notice that we have to argue two separate things to establish that this is its chromatic number:

- K_n can be colored with n colors.
- K_n cannot be colored with less than n colors

For K_n, both of these facts are fairly obvious. Assigning a different color to each vertex will always result in a well-formed coloring (though it may be a waste of colors). Since each vertex in K_n has $n-1$ neighbors, fewer than n colors can't possibly work.

Similar, the chromatic number for $K_{n,m}$ is 2. We color one side of the graph with one color and the other side with a second color.

In general, however, coloring requires exponential time. There's a couple specific versions of the theoretical problem. I could give you a graph and ask you for its chromatic number. Or I could give you a graph and an integer k and ask whether k colors is enough to be able to color G. These problems are all "NP-complete" or "NP-hard", i.e. apparently require exponential time to compute. (That is, assuming that P and NP are actually different, which is one of the big outstanding problems in computer science.) So, except for small examples and special cases, coloring problems can't be solved exactly.

3 Why should I care?

However, graph coloring is required for solving a wide range of practical problems. For example, there is a coloring algorithm embedded in most compilers. Because the general problem can't be solved efficiently, the implemented algorithms use limitations or approximations of various sorts so that they can run in a reasonable amount of time.

For example, suppose that we want to allocate broadcast frequencies to local radio stations. In the corresponding graph problem, each station is a vertex and the frequencies are the "colors." Two stations are connected by an edge if they are geographically too close together, so that they would interfere if they used the same frequency. This graph should not be too bad to color in practice, so long as we have a large enough supply of frequencies compared to the numbers of stations clustered near one another.

We can model a sudoku puzzle by setting up one vertex for each square. The colors are the 9 numbers, and some are pre-assigned to certain vertices. Two vertices are connected if their squares are in the same block or row or column. The puzzle is solvable if we can 9-color this graph, respecting the pre-assigned colors.

Probably the oldest sort of coloring problem is coloring the different countries on a map. In this case, the original map is a graph, in which the regions are the countries. To set up the coloring problem, we need to convert this to the *dual graph*, where each region is a vertex and two regions are connected by an edge exactly when they share a border.[1] In this case, the graph is planar, which makes coloring much easier to do (see below).

We can model exam scheduling as a coloring problem. The exams for two courses should not be put at the same time if there is a student who is in both courses. So we can model this as a graph, in which each course is a vertex and courses are connected by edges if they share students. The question is then whether we can color the graph with k colors, where k is the number of exam times in our schedule.

In the exam scheduling problem, we actually expect the answer to be "no," because eliminating conflicts would require an excessive number of

[1] Two regions touching at a point are not considered to share a border.

exam times. So the real practical problem is: how few students do we have to take out of the picture (i.e. give special conflict exams to) in order to be able to solve the coloring problem with a reasonable value for k. We also have the option of splitting a course (i.e. offering a scheduled conflict exam) to simplify the graph.

A particularly important use of coloring in computer science is register allocation. A large java or C program contains many named variables. But a computer has a smallish number (e.g. 32) fast registers which can feed basic operations such as addition. So variables must be allocated to specific registers.

The vertices in this coloring problem are variables. The colors are registers. Two variables are connected by an edge if they are in use at the same time and, therefore, cannot share a register. As with the exam scheduling problem, we actually expect the raw coloring problem to fail. The compiler then uses so-called "spill" operations to break up the dependencies and create a colorable graph. The goal is to use as few spills as possible.

4 Planar graphs

Planar graphs are an important special case, because they are much easier to color than some other graphs. Way back in 1852, Francis Guthrie hypothesized that any planar graph could be colored with only four colors. This is, in fact, correct, but it took a very long time to prove it. Alfred Kempe thought he had proved it in 1879 and it took 11 years for another mathematician to find an error in his proof.

It was finally proved by Kenneth Appel and Wolfgang Haken at UIUC in 1976. They reduced the problem mathematically, but were left with 1936 specific graphs that needed to be checked exhaustively, using a computer program.

It's much easier to show that all planar graphs can be 6-colored. Recall from Monday's lecture that a planar graph always has a vertex whose degree is no higher than 5. Using this fact, we can do a simple proof by induction.

Base: The graph with just one vertex can clearly be 6-colored.

Induction: Suppose that all graphs with $\leq k$ vertices can be 6-colored. Let G be a graph with $k+1$ vertices. In G, find a vertex v with degree ≤ 5. Remove v (and its edges) to create a smaller graph G'. G' can be 6-colored by the inductive hypothesis. Because v has only 5 neighbors, we can always choose a color for it which will extend this coloring to a coloring of G.

I have it on good authority that 5-colorability can be proved without a computer search, but the proof is difficult.

The proof of 6-colorability can be generalized to other types of graphs. If a graph G has vertices whose degrees are all $\leq D$, then G can be colored with $D+1$ colors. This limits what G's chromatic number could be, but doesn't pin it down exactly. It might be that D can be colored with fewer colors. For example, W_8 (recall: 8 vertices in a circle, one in the center) can be colored with three colors. Alternate two colors around rim of the wheel, then assign a third color to the central vertex.

5 Greedy coloring

This insight motivates a class of good coloring algorithms for practical applications, called "greedy" coloring algorithms. We take our graph G and remove vertices one-by-one, creating a series of smaller and smaller graphs. The goal is to ensure that each vertex has a low degree when removed. So we remove low-degree vertices first in hopes that this will simplify the graph structure around vertices with high degree.

We start by coloring the smallest graph and add the vertices back one-by-one, each time extending the coloring to the new vertex. If each vertex has degree $\leq d$ at the point when it's added to the graph, then we can complete the whole coloring with $d+1$ colors.

This algorithm is quite efficient. Its output might use more colors than the optimal coloring, but it apparently works quite well for problems such as register allocation.

6 Farewell

Thanks for being in the class. It was great to have you! Hope your finals go smoothly. And we hope to see you again in some later course.