34-3 Graph coloring

Mapmakers try to use as few colors as possible when coloring countries on a map, as long as no two countries that share a border have the same color. We can model this problem with an undirected graph $G = (V, E)$ in which each vertex represents a country and vertices whose respective countries share a border are adjacent. Then, a ***k-coloring*** is a function $c : V \to \{1, 2, \ldots, k\}$ such that $c(u) \neq c(v)$ for every edge $(u, v) \in E$. In other words, the numbers $1, 2, \ldots, k$ represent the k colors, and adjacent vertices must have different colors. The ***graph-coloring problem*** is to determine the minimum number of colors needed to color a given graph.

a. Give an efficient algorithm to determine a 2-coloring of a graph, if one exists.

b. Cast the graph-coloring problem as a decision problem. Show that your decision problem is solvable in polynomial time if and only if the graph-coloring problem is solvable in polynomial time.

c. Let the language 3-COLOR be the set of graphs that can be 3-colored. Show that if 3-COLOR is NP-complete, then your decision problem from part (b) is NP-complete.

To prove that 3-COLOR is NP-complete, we use a reduction from 3-CNF-SAT. Given a formula ϕ of m clauses on n variables $x_1, x_2, \ldots, x_n$, we construct a graph $G = (V, E)$ as follows. The set V consists of a vertex for each variable, a vertex for the negation of each variable, 5 vertices for each clause, and 3 special vertices: TRUE, FALSE, and RED. The edges of the graph are of two types: "literal" edges that are independent of the clauses and "clause" edges that depend on the clauses. The literal edges form a triangle on the special vertices and also form a triangle on x_i, $\neg x_i$, and RED for $i = 1, 2, \ldots, n$.

d. Argue that in any 3-coloring c of a graph containing the literal edges, exactly one of a variable and its negation is colored $c(\text{TRUE})$ and the other is colored $c(\text{FALSE})$. Argue that for any truth assignment for ϕ, there exists a 3-coloring of the graph containing just the literal edges.

The widget shown in Figure 34.20 helps to enforce the condition corresponding to a clause $(x \vee y \vee z)$. Each clause requires a unique copy of the 5 vertices that are heavily shaded in the figure; they connect as shown to the literals of the clause and the special vertex TRUE.

e. Argue that if each of x, y, and z is colored $c(\text{TRUE})$ or $c(\text{FALSE})$, then the widget is 3-colorable if and only if at least one of x, y, or z is colored $c(\text{TRUE})$.

f. Complete the proof that 3-COLOR is NP-complete.

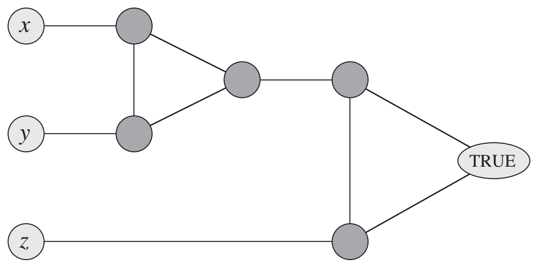

Figure 34.20 The widget corresponding to a clause $(x \vee y \vee z)$, used in Problem 34-3.

34-4 Scheduling with profits and deadlines

Suppose that we have one machine and a set of n tasks $a_1, a_2, \ldots, a_n$, each of which requires time on the machine. Each task a_j requires t_j time units on the machine (its processing time), yields a profit of p_j, and has a deadline d_j. The machine can process only one task at a time, and task a_j must run without interruption for t_j consecutive time units. If we complete task a_j by its deadline d_j, we receive a profit p_j, but if we complete it after its deadline, we receive no profit. As an optimization problem, we are given the processing times, profits, and deadlines for a set of n tasks, and we wish to find a schedule that completes all the tasks and returns the greatest amount of profit. The processing times, profits, and deadlines are all nonnegative numbers.

a. State this problem as a decision problem.

b. Show that the decision problem is NP-complete.

c. Give a polynomial-time algorithm for the decision problem, assuming that all processing times are integers from 1 to n. (*Hint:* Use dynamic programming.)

d. Give a polynomial-time algorithm for the optimization problem, assuming that all processing times are integers from 1 to n.

Chapter notes

The book by Garey and Johnson [129] provides a wonderful guide to NP-completeness, discussing the theory at length and providing a catalogue of many problems that were known to be NP-complete in 1979. The proof of Theorem 34.13 is adapted from their book, and the list of NP-complete problem domains at the beginning of Section 34.5 is drawn from their table of contents. Johnson wrote a series