Theorem 34.11
The clique problem is NP-complete.

Proof First, we show that CLIQUE $\in$ NP. For a given graph $G = (V, E)$, use the set $V' \subseteq V$ of vertices in the clique as a certificate for G. To check whether V' is a clique in polynomial time, check whether, for each pair $u, v \in V'$, the edge (u, v) belongs to E.

We next prove that 3-CNF-SAT $\leq_P$ CLIQUE, which shows that the clique problem is NP-hard. You might be surprised that the proof reduces an instance of 3-CNF-SAT to an instance of CLIQUE, since on the surface logical formulas seem to have little to do with graphs.

The reduction algorithm begins with an instance of 3-CNF-SAT. Let $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_k$ be a boolean formula in 3-CNF with k clauses. For $r = 1, 2, \ldots, k$, each clause C_r contains exactly three distinct literals: l_1^r, l_2^r, and l_3^r. We will construct a graph G such that ϕ is satisfiable if and only if G contains a clique of size k.

We construct the undirected graph $G = (V, E)$ as follows. For each clause $C_r = (l_1^r \vee l_2^r \vee l_3^r)$ in ϕ, place a triple of vertices v_1^r, v_2^r, and v_3^r into V. Add edge (v_i^r, v_j^s) into E if both of the following hold:

- v_i^r and v_j^s are in different triples, that is, $r \neq s$, and
- their corresponding literals are ***consistent***, that is, l_i^r is not the negation of l_j^s.

We can build this graph from ϕ in polynomial time. As an example of this construction, if

$$\phi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3) ,$$

then G is the graph shown in Figure 34.14.

We must show that this transformation of ϕ into G is a reduction. First, suppose that ϕ has a satisfying assignment. Then each clause C_r contains at least one literal l_i^r that is assigned 1, and each such literal corresponds to a vertex v_i^r. Picking one such "true" literal from each clause yields a set V' of k vertices. We claim that V' is a clique. For any two vertices $v_i^r, v_j^s \in V'$, where $r \neq s$, both corresponding literals l_i^r and l_j^s map to 1 by the given satisfying assignment, and thus the literals cannot be complements. Thus, by the construction of G, the edge (v_i^r, v_j^s) belongs to E.

Conversely, suppose that G contains a clique V' of size k. No edges in G connect vertices in the same triple, and so V' contains exactly one vertex per triple. If $v_i^r \in V'$, then assign 1 to the corresponding literal l_i^r. Since G contains no edges between inconsistent literals, no literal and its complement are both assigned 1. Each clause is satisfied, and so ϕ is satisfied. (Any variables that do not correspond to a vertex in the clique may be set arbitrarily.) ■

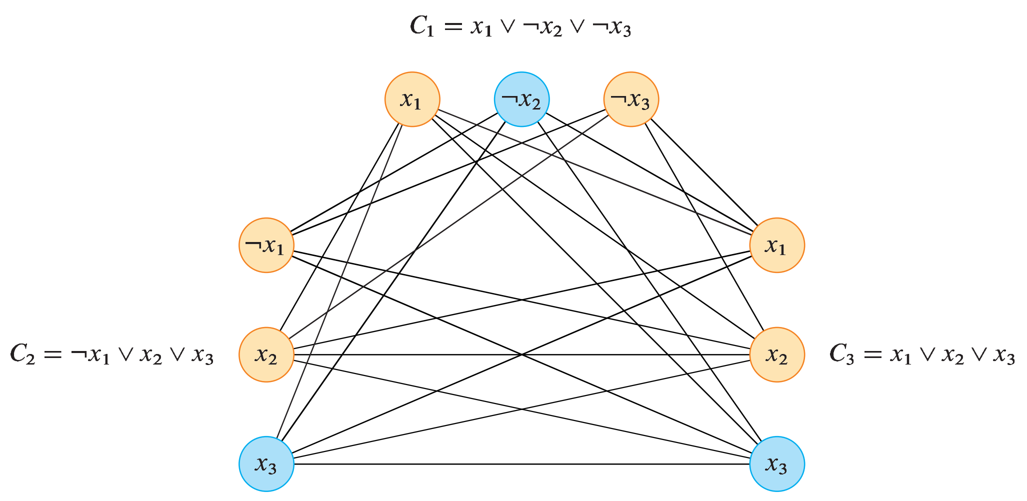
$C_1 = x_1 \vee \neg x_2 \vee \neg x_3$
x_1
$\neg x_2$
$\neg x_3$
$C_2 = \neg x_1 \vee x_2 \vee x_3$
$\neg x_1$
x_2
x_3
$C_3 = x_1 \vee x_2 \vee x_3$
x_1
x_2
x_3

Figure 34.14 The graph G derived from the 3-CNF formula $\phi = C_1 \wedge C_2 \wedge C_3$, where $C_1 = (x_1 \vee \neg x_2 \vee \neg x_3)$, $C_2 = (\neg x_1 \vee x_2 \vee x_3)$, and $C_3 = (x_1 \vee x_2 \vee x_3)$, in reducing 3-CNF-SAT to CLIQUE. A satisfying assignment of the formula has $x_2 = 0$, $x_3 = 1$, and x_1 set to either 0 or 1. This assignment satisfies C_1 with $\neg x_2$, and it satisfies C_2 and C_3 with x_3, corresponding to the clique with blue vertices.

In the example of Figure 34.14, a satisfying assignment of ϕ has $x_2 = 0$ and $x_3 = 1$. A corresponding clique of size $k = 3$ consists of the vertices corresponding to $\neg x_2$ from the first clause, x_3 from the second clause, and x_3 from the third clause. Because the clique contains no vertices corresponding to either x_1 or $\neg x_1$, this satisfying assignment can set x_1 to either 0 or 1.

The proof of Theorem 34.11 reduced an arbitrary instance of 3-CNF-SAT to an instance of CLIQUE with a particular structure. You might think that we have shown only that CLIQUE is NP-hard in graphs in which the vertices are restricted to occur in triples and in which there are no edges between vertices in the same triple. Indeed, we have shown that CLIQUE is NP-hard only in this restricted case, but this proof suffices to show that CLIQUE is NP-hard in general graphs. Why? If there were a polynomial-time algorithm that solves CLIQUE on general graphs, it would also solve CLIQUE on restricted graphs.

The opposite approach—reducing instances of 3-CNF-SAT with a special structure to general instances of CLIQUE—does not suffice, however. Why not? Perhaps the instances of 3-CNF-SAT that we choose to reduce from are "easy," and so we would not have reduced an NP-hard problem to CLIQUE.

Moreover, the reduction uses the instance of 3-CNF-SAT, but not the solution. We would have erred if the polynomial-time reduction had relied on knowing

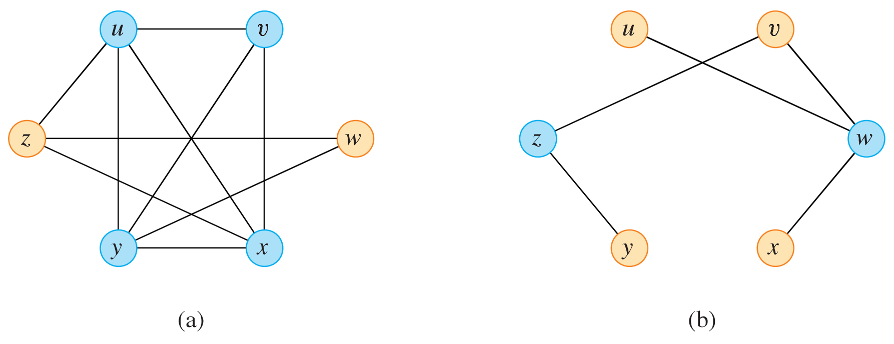

Figure 34.15 Reducing CLIQUE to VERTEX-COVER. **(a)** An undirected graph $G = (V, E)$ with clique $V' = \{u, v, x, y\}$, shown in blue. **(b)** The graph $\overline{G}$ produced by the reduction algorithm that has vertex cover $V - V' = \{w, z\}$, in blue.

whether the formula ϕ is satisfiable, since we do not know how to decide whether ϕ is satisfiable in polynomial time.

34.5.2 The vertex-cover problem

A ***vertex cover*** of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that if $(u, v) \in E$, then $u \in V'$ or $v \in V'$ (or both). That is, each vertex "covers" its incident edges, and a vertex cover for G is a set of vertices that covers all the edges in E. The ***size*** of a vertex cover is the number of vertices in it. For example, the graph in Figure 34.15(b) has a vertex cover $\{w, z\}$ of size 2.

The ***vertex-cover problem*** is to find a vertex cover of minimum size in a given graph. For this optimization problem, the corresponding decision problem asks whether a graph has a vertex cover of a given size k. As a language, we define

$$\text{VERTEX-COVER} = \{\langle G, k \rangle : \text{graph } G \text{ has a vertex cover of size } k\} .$$

The following theorem shows that this problem is NP-complete.

Theorem 34.12
The vertex-cover problem is NP-complete.

Proof We first show that VERTEX-COVER $\in$ NP. Given a graph $G = (V, E)$ and an integer k, the certificate is the vertex cover $V' \subseteq V$ itself. The verification algorithm affirms that $|V'| = k$, and then it checks, for each edge $(u, v) \in E$, that $u \in V'$ or $v \in V'$. It is easy to verify the certificate in polynomial time.

To prove that the vertex-cover problem is NP-hard, we reduce from the clique problem, showing that CLIQUE $\leq_P$ VERTEX-COVER. This reduction relies on the notion of the complement of a graph. Given an undirected graph $G = (V, E)$, we define the ***complement*** of G as a graph $\overline{G} = (V, \overline{E})$, where $\overline{E} = \{(u, v) : u, v \in V, u \neq v, \text{ and } (u, v) \notin E\}$. In other words, $\overline{G}$ is the graph containing exactly those edges that are not in G. Figure 34.15 shows a graph and its complement and illustrates the reduction from CLIQUE to VERTEX-COVER.

The reduction algorithm takes as input an instance $\langle G, k \rangle$ of the clique problem and computes the complement $\overline{G}$ in polynomial time. The output of the reduction algorithm is the instance $\langle \overline{G}, |V| - k \rangle$ of the vertex-cover problem. To complete the proof, we show that this transformation is indeed a reduction: the graph G contains a clique of size k if and only if the graph $\overline{G}$ has a vertex cover of size $|V| - k$.

Suppose that G contains a clique $V' \subseteq V$ with $|V'| = k$. We claim that $V - V'$ is a vertex cover in $\overline{G}$. Let (u, v) be any edge in $\overline{E}$. Then, $(u, v) \notin E$, which implies that at least one of u or v does not belong to V', since every pair of vertices in V' is connected by an edge of E. Equivalently, at least one of u or v belongs to $V - V'$, which means that edge (u, v) is covered by $V - V'$. Since (u, v) was chosen arbitrarily from $\overline{E}$, every edge of $\overline{E}$ is covered by a vertex in $V - V'$. Hence the set $V - V'$, which has size $|V| - k$, forms a vertex cover for $\overline{G}$.

Conversely, suppose that $\overline{G}$ has a vertex cover $V' \subseteq V$, where $|V'| = |V| - k$. Then for all $u, v \in V$, if $(u, v) \in \overline{E}$, then $u \in V'$ or $v \in V'$ or both. The contrapositive of this implication is that for all $u, v \in V$, if $u \notin V'$ and $v \notin V'$, then $(u, v) \in E$. In other words, $V - V'$ is a clique, and it has size $|V| - |V'| = k$. ■

Since VERTEX-COVER is NP-complete, we don't expect to find a polynomial-time algorithm for finding a minimum-size vertex cover. Section 35.1 presents a polynomial-time "approximation algorithm," however, which produces "approximate" solutions for the vertex-cover problem. The size of a vertex cover produced by the algorithm is at most twice the minimum size of a vertex cover.

Thus, you shouldn't give up hope just because a problem is NP-complete. You might be able to design a polynomial-time approximation algorithm that obtains near-optimal solutions, even though finding an optimal solution is NP-complete. Chapter 35 gives several approximation algorithms for NP-complete problems.

34.5.3 The hamiltonian-cycle problem

We now return to the hamiltonian-cycle problem defined in Section 34.2.

Theorem 34.13
The hamiltonian cycle problem is NP-complete.

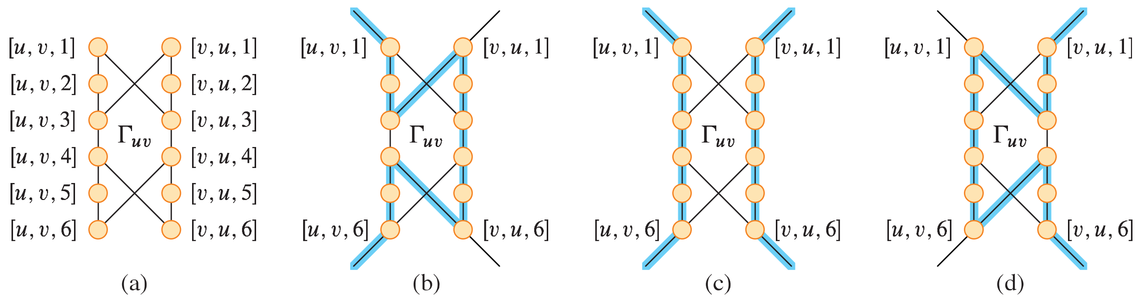

Figure 34.16 The gadget used in reducing the vertex-cover problem to the hamiltonian-cycle problem. An edge (u, v) of graph G corresponds to gadget Γ_{uv} in the graph G' created in the reduction. **(a)** The gadget, with individual vertices labeled. **(b)–(d)** The paths highlighted in blue are the only possible ones through the gadget that include all vertices, assuming that the only connections from the gadget to the remainder of G' are through vertices $[u, v, 1]$, $[u, v, 6]$, $[v, u, 1]$, and $[v, u, 6]$.

Proof We first show that HAM-CYCLE $\in$ NP. Given an undirected graph $G = (V, E)$, the certificate is the sequence of $|V|$ vertices that makes up the hamiltonian cycle. The verification algorithm checks that this sequence contains each vertex in V exactly once and that with the first vertex repeated at the end, it forms a cycle in G. That is, it checks that there is an edge between each pair of consecutive vertices and between the first and last vertices. This certificate can be verified in polynomial time.

We now prove that VERTEX-COVER $\leq_P$ HAM-CYCLE, which shows that HAM-CYCLE is NP-complete. Given an undirected graph $G = (V, E)$ and an integer k, we construct an undirected graph $G' = (V', E')$ that has a hamiltonian cycle if and only if G has a vertex cover of size k. We assume without loss of generality that G contains no isolated vertices (that is, every vertex in V has at least one incident edge) and that $k \leq |V|$. (If an isolated vertex belongs to a vertex cover of size k, then there also exists a vertex cover of size $k-1$, and for any graph, the entire set V is always a vertex cover.)

Our construction uses a ***gadget***, which is a piece of a graph that enforces certain properties. Figure 34.16(a) shows the gadget we use. For each edge $(u, v) \in E$, the constructed graph G' contains one copy of this gadget, which we denote by Γ_{uv}. We denote each vertex in Γ_{uv} by $[u, v, i]$ or $[v, u, i]$, where $1 \leq i \leq 6$, so that each gadget Γ_{uv} contains 12 vertices. Gadget Γ_{uv} also contains the 14 edges shown in Figure 34.16(a).

Along with the internal structure of the gadget, we enforce the properties we want by limiting the connections between the gadget and the remainder of the graph G' that we construct. In particular, only vertices $[u, v, 1]$, $[u, v, 6]$, $[v, u, 1]$, and $[v, u, 6]$ will have edges incident from outside Γ_{uv}. Any hamiltonian cycle of G' must traverse the edges of Γ_{uv} in one of the three ways shown in Figures 34.16(b)–(d). If the cycle enters through vertex $[u, v, 1]$, it must exit through vertex $[u, v, 6]$, and it either visits all 12 of the gadget's vertices (Figure 34.16(b)) or the six vertices $[u, v, 1]$ through $[u, v, 6]$ (Figure 34.16(c)). In the latter case, the cycle will have to reenter the gadget to visit vertices $[v, u, 1]$ through $[v, u, 6]$. Similarly, if the cycle enters through vertex $[v, u, 1]$, it must exit through vertex $[v, u, 6]$, and either it visits all 12 of the gadget's vertices (Figure 34.16(d)) or it visits the six vertices $[v, u, 1]$ through $[v, u, 6]$ and reenters to visit $[u, v, 1]$ through $[u, v, 6]$ (Figure 34.16(c)). No other paths through the gadget that visit all 12 vertices are possible. In particular, it is impossible to construct two vertex-disjoint paths, one of which connects $[u, v, 1]$ to $[v, u, 6]$ and the other of which connects $[v, u, 1]$ to $[u, v, 6]$, such that the union of the two paths contains all of the gadget's vertices.

The only other vertices in V' other than those of gadgets are ***selector vertices*** $s_1, s_2, \ldots, s_k$. We'll use edges incident on selector vertices in G' to select the k vertices of the cover in G.

In addition to the edges in gadgets, E' contains two other types of edges, which Figure 34.17 shows. First, for each vertex $u \in V$, edges join pairs of gadgets in order to form a path containing all gadgets corresponding to edges incident on u in G. We arbitrarily order the vertices adjacent to each vertex $u \in V$ as $u^{(1)}, u^{(2)}, \ldots, u^{(\text{degree}(u))}$, where degree$(u)$ is the number of vertices adjacent to u. To create a path in G' through all the gadgets corresponding to edges incident on u, E' contains the edges $\{([u, u^{(i)}, 6], [u, u^{(i+1)}, 1]) : 1 \le i \le \text{degree}(u) - 1\}$. In Figure 34.17, for example, we order the vertices adjacent to w as $\langle x, y, z \rangle$, and so graph G' in part (b) of the figure includes the edges $([w, x, 6], [w, y, 1])$ and $([w, y, 6], [w, z, 1])$. The vertices adjacent to x are ordered as $\langle w, y \rangle$, so that G' includes the edge $([x, w, 6], [x, y, 1])$. For each vertex $u \in V$, these edges in G' fill in a path containing all gadgets corresponding to edges incident on u in G.

The intuition behind these edges is that if vertex $u \in V$ belongs to the vertex cover of G, then G' contains a path from $[u, u^{(1)}, 1]$ to $[u, u^{(\text{degree}(u))}, 6]$ that "covers" all gadgets corresponding to edges incident on u. That is, for each of these gadgets, say $\Gamma_{u,u^{(i)}}$, the path either includes all 12 vertices (if u belongs to the vertex cover but $u^{(i)}$ does not) or just the six vertices $[u, u^{(i)}, 1]$ through $[u, u^{(i)}, 6]$ (if both u and $u^{(i)}$ belong to the vertex cover).

The final type of edge in E' joins the first vertex $[u, u^{(1)}, 1]$ and the last vertex $[u, u^{(\text{degree}(u))}, 6]$ of each of these paths to each of the selector vertices. That is, E' includes the edges

$$\begin{aligned}&\{(s_j, [u, u^{(1)}, 1]) : u \in V \text{ and } 1 \le j \le k\}\\ &\quad \cup \{(s_j, [u, u^{(\text{degree}(u))}, 6]) : u \in V \text{ and } 1 \le j \le k\}\,.\end{aligned}$$

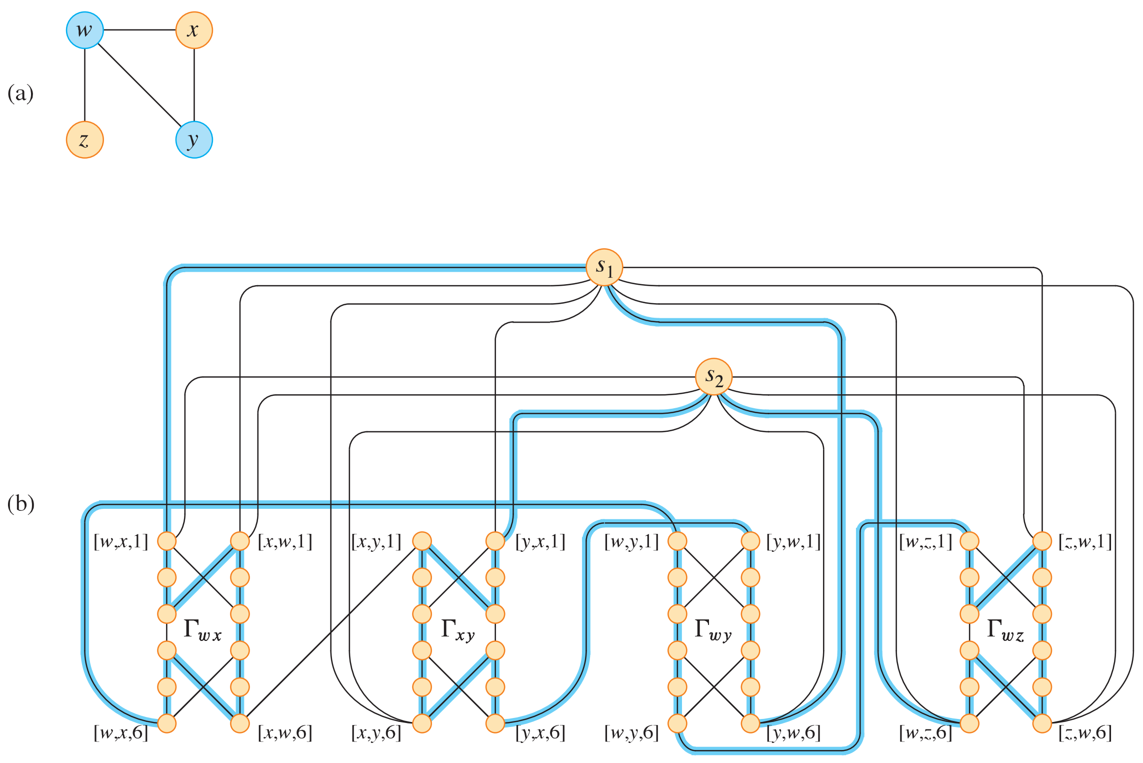

Figure 34.17 Reducing an instance of the vertex-cover problem to an instance of the hamiltonian-cycle problem. **(a)** An undirected graph G with a vertex cover of size 2, consisting of the blue vertices w and y. **(b)** The undirected graph G' produced by the reduction, with the hamiltonian cycle corresponding to the vertex cover highlighted in blue. The vertex cover $\{w, y\}$ corresponds to edges $(s_1, [w, x, 1])$ and $(s_2, [y, x, 1])$ appearing in the hamiltonian cycle.

Next we show that the size of G' is polynomial in the size of G, and hence it takes time polynomial in the size of G to construct G'. The vertices of G' are those in the gadgets, plus the selector vertices. With 12 vertices per gadget, plus $k \le |V|$ selector vertices, G' contains a total of

$$\begin{aligned} |V'| &= 12\,|E| + k \\ &\le 12\,|E| + |V| \end{aligned}$$

vertices. The edges of G' are those in the gadgets, those that go between gadgets, and those connecting selector vertices to gadgets. Each gadget contains 14 edges, totaling $14\,|E|$ in all gadgets. For each vertex $u \in V$, graph G' has degree$(u) - 1$ edges going between gadgets, so that summed over all vertices in V,

$$\sum_{u \in V} (\text{degree}(u) - 1) = 2\,|E| - |V|$$

edges go between gadgets. Finally, G' has two edges for each pair consisting of a selector vertex and a vertex of V, totaling $2k\,|V|$ such edges. The total number of edges of G' is therefore

$$\begin{aligned} |E'| &= (14\,|E|) + (2\,|E| - |V|) + (2k\,|V|) \\ &= 16\,|E| + (2k-1)\,|V| \\ &\le 16\,|E| + (2\,|V| - 1)\,|V| \;. \end{aligned}$$

Now we show that the transformation from graph G to G' is a reduction. That is, we must show that G has a vertex cover of size k if and only if G' has a hamiltonian cycle.

Suppose that $G = (V, E)$ has a vertex cover $V^* \subseteq V$, where $|V^*| = k$. Let $V^* = \{u_1, u_2, \ldots, u_k\}$. As Figure 34.17 shows, we can construct a hamiltonian cycle in G' by including the following edges[11] for each vertex $u_j \in V^*$. Start by including edges $\left\{([u_j, u_j^{(i)}, 6], [u_j, u_j^{(i+1)}, 1]) : 1 \le i \le \text{degree}(u_j) - 1\right\}$, which connect all gadgets corresponding to edges incident on u_j. Also include the edges within these gadgets as Figures 34.16(b)–(d) show, depending on whether the edge is covered by one or two vertices in V^*. The hamiltonian cycle also includes the edges

$$\begin{aligned} &\{(s_j, [u_j, u_j^{(1)}, 1]) : 1 \le j \le k\} \\ &\quad \cup \{(s_{j+1}, [u_j, u_j^{(\text{degree}(u_j))}, 6]) : 1 \le j \le k-1\} \\ &\quad \cup \{(s_1, [u_k, u_k^{(\text{degree}(u_k))}, 6])\} \;. \end{aligned}$$

By inspecting Figure 34.17, you can verify that these edges form a cycle, where $u_1 = w$ and $u_2 = y$. The cycle starts at s_1, visits all gadgets corresponding to edges incident on u_1, then visits s_2, visits all gadgets corresponding to edges incident on u_2, and so on, until it returns to s_1. The cycle visits each gadget either once or twice, depending on whether one or two vertices of V^* cover its corresponding edge. Because V^* is a vertex cover for G, each edge in E is incident on some vertex in V^*, and so the cycle visits each vertex in each gadget of G'. Because the cycle also visits every selector vertex, it is hamiltonian.

Conversely, suppose that $G' = (V', E')$ contains a hamiltonian cycle $C \subseteq E'$. We claim that the set

$$V^* = \{u \in V : (s_j, [u, u^{(1)}, 1]) \in C \text{ for some } 1 \le j \le k\} \tag{34.4}$$

[11] Technically, a cycle is defined as a sequence of vertices rather than edges (see Section B.4). In the interest of clarity, we abuse notation here and define the hamiltonian cycle by its edges.

is a vertex cover for G.

We first argue that the set V^* is well defined, that is, for each selector vertex s_j, exactly one of the incident edges in the hamiltonian cycle C is of the form $(s_j, [u, u^{(1)}, 1])$ for some vertex $u \in V$. To see why, partition the hamiltonian cycle C into maximal paths that start at some selector vertex s_i, visit one or more gadgets, and end at some selector vertex s_j without passing through any other selector vertex. Let's call each of these maximal paths a "cover path." Let P be one such cover path, and orient it going from s_i to s_j. If P contains the edge $(s_i, [u, u^{(1)}, 1])$ for some vertex $u \in V$, then we have shown that one edge incident on s_i has the required form. Assume, then, that P contains the edge $(s_i, [v, v^{(\text{degree}(v))}, 6])$ for some vertex $v \in V$. This path enters a gadget from the bottom, as drawn in Figures 34.16 and 34.17, and it leaves from the top. It might go through several gadgets, but it always enters from the bottom of a gadget and leaves from the top. The only edges incident on vertices at the top of a gadget either go to the bottoms of other gadgets or to selector vertices. Therefore, after the last gadget in the series of gadgets visited by P, the edge taken must go to a selector vertex s_j, so that P contains an edge of the form $(s_j, [u, u^{(1)}, 1])$, where $[u, u^{(1)}, 1]$ is a vertex at the top of some gadget. To see that not both edges incident on s_j have this form, simply reverse the direction of traversing P in the above argument.

Having established that the set V^* is well defined, let's see why it is a vertex cover for G. We have already established that each cover path starts at some s_i, takes the edge $(s_i, [u, u^{(1)}, 1])$ for some vertex $u \in V$, passes through all the gadgets corresponding to edges in E incident on u, and then ends at some selector vertex s_j. (This orientation is the reverse of the orientation in the paragraph above.) Let's call this cover path P_u, and by equation (34.4), the vertex cover V^* includes u. Each gadget visited by P_u must be Γ_{uv} or Γ_{vu} for some $v \in V$. For each gadget visited by P_u, its vertices are visited by either one or two cover paths. If they are visited by one cover path, then edge $(u, v) \in E$ is covered in G by vertex u. If two cover paths visit the gadget, then the other cover path must be P_v, which implies that $v \in V^*$, and edge $(u, v) \in E$ is covered by both u and v. Because each vertex in each gadget is visited by some cover path, we see that each edge in E is covered by some vertex in V^*. ■

34.5.4 The traveling-salesperson problem

In the ***traveling-salesperson problem***, which is closely related to the hamiltonian-cycle problem, a salesperson must visit n cities. Let's model the problem as a complete graph with n vertices, so that the salesperson wishes to make a ***tour***, or hamiltonian cycle, visiting each city exactly once and finishing at the starting city. The salesperson incurs a nonnegative integer cost $c(i, j)$ to travel from city i

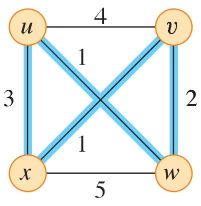

Figure 34.18 An instance of the traveling-salesperson problem. Edges highlighted in blue represent a minimum-cost tour, with cost 7.

to city j. In the optimization version of the problem, the salesperson wishes to make the tour whose total cost is minimum, where the total cost is the sum of the individual costs along the edges of the tour. For example, in Figure 34.18, a minimum-cost tour is $\langle u, w, v, x, u \rangle$, with cost 7. The formal language for the corresponding decision problem is

$$\begin{aligned}\text{TSP} = \{\langle G, c, k \rangle : \ & G = (V, E) \text{ is a complete graph,} \\ & c \text{ is a function from } V \times V \to \mathbb{N}, \\ & k \in \mathbb{N}, \text{ and} \\ & G \text{ has a traveling-salesperson tour with cost at most } k\}\ .\end{aligned}$$

The following theorem shows that a fast algorithm for the traveling-salesperson problem is unlikely to exist.

Theorem 34.14
The traveling-salesperson problem is NP-complete.

Proof We first show that TSP $\in$ NP. Given an instance of the problem, the certificate is the sequence of n vertices in the tour. The verification algorithm checks that this sequence contains each vertex exactly once, sums up the edge costs, and checks that the sum is at most k. This process can certainly be done in polynomial time.

To prove that TSP is NP-hard, we show that HAM-CYCLE $\le_P$ TSP. Given an instance $G = (V, E)$ of HAM-CYCLE, construct an instance of TSP by forming the complete graph $G' = (V, E')$, where $E' = \{(i, j) : i, j \in V \text{ and } i \neq j\}$, with the cost function c defined as

$$c(i, j) = \begin{cases} 0 & \text{if } (i, j) \in E\ , \\ 1 & \text{if } (i, j) \notin E\ . \end{cases}$$

(Because G is undirected, it contains no self-loops, and so $c(v, v) = 1$ for all vertices $v \in V$.) The instance of TSP is then $\langle G', c, 0 \rangle$, which can be created in polynomial time.

We now show that graph G has a hamiltonian cycle if and only if graph G' has a tour of cost at most 0. Suppose that graph G has a hamiltonian cycle H. Each edge in H belongs to E and thus has cost 0 in G'. Thus, H is a tour in G' with cost 0. Conversely, suppose that graph G' has a tour H' of cost at most 0. Since the costs of the edges in E' are 0 and 1, the cost of tour H' is exactly 0 and each edge on the tour must have cost 0. Therefore, H' contains only edges in E. We conclude that H' is a hamiltonian cycle in graph G. ∎

34.5.5 The subset-sum problem

We next consider an arithmetic NP-complete problem. The ***subset-sum problem*** takes as inputs a finite set S of positive integers and an integer ***target*** $t > 0$. It asks whether there exists a subset $S' \subseteq S$ whose elements sum to exactly t. For example, if $S = \{1, 2, 7, 14, 49, 98, 343, 686, 2409, 2793, 16808, 17206, 117705, 117993\}$ and $t = 138457$, then the subset $S' = \{1, 2, 7, 98, 343, 686, 2409, 17206, 117705\}$ is a solution.

As usual, we express the problem as a language:

$$\text{SUBSET-SUM} = \left\{\langle S, t\rangle : \text{there exists a subset } S' \subseteq S \text{ such that } t = \textstyle\sum_{s \in S'} s\right\} .$$

As with any arithmetic problem, it is important to recall that our standard encoding assumes that the input integers are coded in binary. With this assumption in mind, we can show that the subset-sum problem is unlikely to have a fast algorithm.

Theorem 34.15
The subset-sum problem is NP-complete.

Proof To show that SUBSET-SUM $\in$ NP, for an instance $\langle S, t\rangle$ of the problem, let the subset S' be the certificate. A verification algorithm can check whether $t = \sum_{s \in S'} s$ in polynomial time.

We now show that 3-CNF-SAT $\le_P$ SUBSET-SUM. Given a 3-CNF formula ϕ over variables $x_1, x_2, \ldots, x_n$ with clauses $C_1, C_2, \ldots, C_k$, each containing exactly three distinct literals, the reduction algorithm constructs an instance $\langle S, t\rangle$ of the subset-sum problem such that ϕ is satisfiable if and only if there exists a subset of S whose sum is exactly t. Without loss of generality, we make two simplifying assumptions about the formula ϕ. First, no clause contains both a variable and its negation, for such a clause is automatically satisfied by any assignment of values to the variables. Second, each variable appears in at least one clause, because it does not matter what value is assigned to a variable that appears in no clauses.

The reduction creates two numbers in set S for each variable x_i and two numbers in S for each clause C_j. The numbers will be represented in base 10, with each number containing $n + k$ digits and each digit corresponding to either one variable

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
v_1	=	1	0	0	1	0	0	1
v'_1	=	1	0	0	0	1	1	0
v_2	=	0	1	0	0	0	0	1
v'_2	=	0	1	0	1	1	1	0
v_3	=	0	0	1	0	0	1	1
v'_3	=	0	0	1	1	1	0	0
s_1	=	0	0	0	1	0	0	0
s'_1	=	0	0	0	2	0	0	0
s_2	=	0	0	0	0	1	0	0
s'_2	=	0	0	0	0	2	0	0
s_3	=	0	0	0	0	0	1	0
s'_3	=	0	0	0	0	0	2	0
s_4	=	0	0	0	0	0	0	1
s'_4	=	0	0	0	0	0	0	2
t	=	1	1	1	4	4	4	4

Figure 34.19 The reduction of 3-CNF-SAT to SUBSET-SUM. The formula in 3-CNF is $\phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$, where $C_1 = (x_1 \vee \neg x_2 \vee \neg x_3)$, $C_2 = (\neg x_1 \vee \neg x_2 \vee \neg x_3)$, $C_3 = (\neg x_1 \vee \neg x_2 \vee x_3)$, and $C_4 = (x_1 \vee x_2 \vee x_3)$. A satisfying assignment of ϕ is $\langle x_1 = 0, x_2 = 0, x_3 = 1 \rangle$. The set S produced by the reduction consists of the base-10 numbers shown: reading from top to bottom, $S = \{1001001, 1000110, 100001, 101110, 10011, 11100, 1000, 2000, 100, 200, 10, 20, 1, 2\}$. The target t is 1114444. The subset $S' \subseteq S$ is shaded blue, and it contains v'_1, v'_2, and v_3, corresponding to the satisfying assignment. Subset S' also contains slack variables $s_1, s'_1, s'_2, s_3, s_4$, and s'_4 to achieve the target value of 4 in the digits labeled by C_1 through C_4.

or one clause. Base 10 (and other bases, as we shall see) has the property we need of preventing carries from lower digits to higher digits.

As Figure 34.19 shows, we construct set S and target t as follows. Label each digit position by either a variable or a clause. The least significant k digits are labeled by the clauses, and the most significant n digits are labeled by variables.

- The target t has a 1 in each digit labeled by a variable and a 4 in each digit labeled by a clause.
- For each variable x_i, set S contains two integers v_i and v'_i. Each of v_i and v'_i has a 1 in the digit labeled by x_i and 0s in the other variable digits. If literal x_i appears in clause C_j, then the digit labeled by C_j in v_i contains a 1. If literal $\neg x_i$ appears in clause C_j, then the digit labeled by C_j in v'_i contains a 1. All other digits labeled by clauses in v_i and v'_i are 0.

All v_i and v'_i values in set S are unique. Why? For $\ell \neq i$, no v_ℓ or v'_ℓ values can equal v_i and v'_i in the most significant n digits. Furthermore, by our simplifying assumptions above, no v_i and v'_i can be equal in all k least significant digits. If v_i and v'_i were equal, then x_i and $\neg x_i$ would have to appear in exactly the same set of clauses. But we assume that no clause contains both x_i and $\neg x_i$ and that either x_i or $\neg x_i$ appears in some clause, and so there must be some clause C_j for which v_i and v'_i differ.

- For each clause C_j, set S contains two integers s_j and s'_j. Each of s_j and s'_j has 0s in all digits other than the one labeled by C_j. For s_j, there is a 1 in the C_j digit, and s'_j has a 2 in this digit. These integers are "slack variables," which we use to get each clause-labeled digit position to add to the target value of 4.

  Simple inspection of Figure 34.19 demonstrates that all s_j and s'_j values in S are unique in set S.

The greatest sum of digits in any one digit position is 6, which occurs in the digits labeled by clauses (three 1s from the v_i and v'_i values, plus 1 and 2 from the s_j and s'_j values). Interpreting these numbers in base 10, therefore, no carries can occur from lower digits to higher digits.[12]

The reduction can be performed in polynomial time. The set S consists of $2n + 2k$ values, each of which has $n + k$ digits, and the time to produce each digit is polynomial in $n + k$. The target t has $n + k$ digits, and the reduction produces each in constant time.

Let's now show that the 3-CNF formula ϕ is satisfiable if and only if there exists a subset $S' \subseteq S$ whose sum is t. First, suppose that ϕ has a satisfying assignment. For $i = 1, 2, \ldots, n$, if $x_i = 1$ in this assignment, then include v_i in S'. Otherwise, include v'_i. In other words, S' includes exactly the v_i and v'_i values that correspond to literals with the value 1 in the satisfying assignment. Having included either v_i or v'_i, but not both, for all i, and having put 0 in the digits labeled by variables in all s_j and s'_j, we see that for each variable-labeled digit, the sum of the values of S' must be 1, which matches those digits of the target t. Because each clause is satisfied, the clause contains some literal with the value 1. Therefore, each digit labeled by a clause has at least one 1 contributed to its sum by a v_i or v'_i value in S'. In fact, one, two, or three literals may be 1 in each clause, and so each clause-labeled digit has a sum of 1, 2, or 3 from the v_i and v'_i values in S'. In Figure 34.19 for example, literals $\neg x_1$, $\neg x_2$, and x_3 have the value 1 in a satisfying assignment. Each of clauses C_1 and C_4 contains exactly one of these literals, and so together v'_1, v'_2, and v_3 contribute 1 to the sum in the digits for C_1 and C_4.

[12] In fact, any base $b \geq 7$ works. The instance at the beginning of this subsection is the set S and target t in Figure 34.19 interpreted in base 7, with S listed in sorted order.

Clause C_2 contains two of these literals, and v'_1, v'_2, and v_3 contribute 2 to the sum in the digit for C_2. Clause C_3 contains all three of these literals, and v'_1, v'_2, and v_3 contribute 3 to the sum in the digit for C_3. To achieve the target of 4 in each digit labeled by clause C_j, include in S' the appropriate nonempty subset of slack variables $\{s_j, s'_j\}$. In Figure 34.19, S' includes s_1, s'_1, s'_2, s_3, s_4, and s'_4. Since S' matches the target in all digits of the sum, and no carries can occur, the values of S' sum to t.

Now suppose that some subset $S' \subseteq S$ sums to t. The subset S' must include exactly one of v_i and v'_i for each $i = 1, 2, \ldots, n$, for otherwise the digits labeled by variables would not sum to 1. If $v_i \in S'$, then set $x_i = 1$. Otherwise, $v'_i \in S'$, and set $x_i = 0$. We claim that every clause C_j, for $j = 1, 2, \ldots, k$, is satisfied by this assignment. To prove this claim, note that to achieve a sum of 4 in the digit labeled by C_j, the subset S' must include at least one v_i or v'_i value that has a 1 in the digit labeled by C_j, since the contributions of the slack variables s_j and s'_j together sum to at most 3. If S' includes a v_i that has a 1 in C_j's position, then the literal x_i appears in clause C_j. Since $x_i = 1$ when $v_i \in S'$, clause C_j is satisfied. If S' includes a v'_i that has a 1 in that position, then the literal $\neg x_i$ appears in C_j. Since $x_i = 0$ when $v'_i \in S'$, clause C_j is again satisfied. Thus, all clauses of ϕ are satisfied, which completes the proof. ■

34.5.6 Reduction strategies

From the reductions in this section, you can see that no single strategy applies to all NP-complete problems. Some reductions are straightforward, such as reducing the hamiltonian-cycle problem to the traveling-salesperson problem. Others are considerably more complicated. Here are a few things to keep in mind and some strategies that you can often bring to bear.

Pitfalls

Make sure that you don't get the reduction backward. That is, in trying to show that problem Y is NP-complete, you might take a known NP-complete problem X and give a polynomial-time reduction from Y to X. That is the wrong direction. The reduction should be from X to Y, so that a solution to Y gives a solution to X.

Remember also that reducing a known NP-complete problem X to a problem Y does not in itself prove that Y is NP-complete. It proves that Y is NP-hard. In order to show that Y is NP-complete, you additionally need to prove that it's in NP by showing how to verify a certificate for Y in polynomial time.