THEOREM 7.56

SUBSET-SUM is NP-complete.

PROOF IDEA We have already shown that *SUBSET-SUM* is in NP in Theorem 7.25. We prove that all languages in NP are polynomial time reducible to *SUBSET-SUM* by reducing the NP-complete language *3SAT* to it. Given a 3cnf-formula ϕ we construct an instance of the *SUBSET-SUM* problem that contains a subcollection summing to the target t if and only if ϕ is satisfiable. Call this subcollection T.

To achieve this reduction we find structures of the *SUBSET-SUM* problem that represent variables and clauses. The *SUBSET-SUM* problem instance that we construct contains numbers of large magnitude presented in decimal notation. We represent variables by pairs of numbers and clauses by certain positions in the decimal representations of the numbers.

We represent variable x_i by two numbers, y_i and z_i. We prove that either y_i or z_i must be in T for each i, which establishes the encoding for the truth value of x_i in the satisfying assignment.

Each clause position contains a certain value in the target t, which imposes a requirement on the subset T. We prove that this requirement is the same as the one in the corresponding clause—namely, that one of the literals in that clause is assigned TRUE.

PROOF We already know that *SUBSET-SUM* $\in$ NP, so we now show that *3SAT* $\leq_{\text{P}}$ *SUBSET-SUM*.

Let ϕ be a Boolean formula with variables $x_1, \ldots, x_l$ and clauses $c_1, \ldots, c_k$. The reduction converts ϕ to an instance of the *SUBSET-SUM* problem $\langle S, t \rangle$, wherein the elements of S and the number t are the rows in the table in Figure 7.57, expressed in ordinary decimal notation. The rows above the double line are labeled

$$y_1, z_1, y_2, z_2, \ldots, y_l, z_l \quad \text{and} \quad g_1, h_1, g_2, h_2, \ldots, g_k, h_k$$

and comprise the elements of S. The row below the double line is t.

Thus S contains one pair of numbers, y_i, z_i, for each variable x_i in ϕ. The decimal representation of these numbers is in two parts, as indicated in the table. The left-hand part comprises a 1 followed by $l - i$ 0s. The right-hand part contains one digit for each clause, where the jth digit of y_i is 1 if clause c_j contains literal x_i and the jth digit of z_i is 1 if clause c_j contains literal $\overline{x_i}$. Digits not specified to be 1 are 0.

The table is partially filled in to illustrate sample clauses, c_1, c_2, and c_k:

$$(x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee \cdots) \wedge \cdots \wedge (\overline{x_3} \vee \cdots \vee \cdots).$$

Additionally, S contains one pair of numbers, g_j, h_j, for each clause c_j. These two numbers are equal and consist of a 1 followed by $k - j$ 0s.

Finally, the target number t, the bottom row of the table, consists of l 1s followed by k 3s.

	1	2	3	4	$\cdots$	l	c_1	c_2	$\cdots$	c_k
y_1	1	0	0	0	$\cdots$	0	1	0	$\cdots$	0
z_1	1	0	0	0	$\cdots$	0	0	0	$\cdots$	0
y_2		1	0	0	$\cdots$	0	0	1	$\cdots$	0
z_2		1	0	0	$\cdots$	0	1	0	$\cdots$	0
y_3			1	0	$\cdots$	0	1	1	$\cdots$	0
z_3			1	0	$\cdots$	0	0	0	$\cdots$	1
$\vdots$					$\ddots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
y_l						1	0	0	$\cdots$	0
z_l						1	0	0	$\cdots$	0
g_1							1	0	$\cdots$	0
h_1							1	0	$\cdots$	0
g_2								1	$\cdots$	0
h_2								1	$\cdots$	0
$\vdots$									$\ddots$	$\vdots$
g_k										1
h_k										1
t	1	1	1	1	$\cdots$	1	3	3	$\cdots$	3

FIGURE 7.57
Reducing *3SAT* to *SUBSET-SUM*

Now we show why this construction works. We demonstrate that ϕ is satisfiable iff some subset of S sums to t.

Suppose that ϕ is satisfiable. We construct a subset of S as follows. We select y_i if x_i is assigned TRUE in the satisfying assignment and z_i if x_i is assigned FALSE. If we add up what we have selected so far, we obtain a 1 in each of the first l digits because we have selected either y_i or z_i for each i. Furthermore, each of the last k digits is a number between 1 and 3 because each clause is satisfied and so contains between 1 and 3 true literals. Now we further select enough of the g and h numbers to bring each of the last k digits up to 3, thus hitting the target.

Suppose that a subset of S sums to t. We construct a satisfying assignment to ϕ after making several observations. First, all the digits in members of S are either 0 or 1. Furthermore, each column in the table describing S contains at most five 1s. Hence a "carry" into the next column never occurs when a subset of S is added. To get a 1 in each of the first l columns the subset must have either

y_i or z_i for each i, but not both.

Now we make the satisfying assignment. If the subset contains y_i, we assign x_i TRUE; otherwise, we assign it FALSE. This assignment must satisfy ϕ because in each of the final k columns the sum is always 3. In column c_j, at most 2 can come from g_j and h_j, so at least 1 in this column must come from some y_i or z_i in the subset. If it is y_i, then x_i appears in c_j and is assigned TRUE, so c_j is satisfied. If it is z_i, then $\overline{x_i}$ appears in c_j and x_i is assigned FALSE, so c_j is satisfied. Therefore ϕ is satisfied.

Finally, we must be sure that the reduction can be carried out in polynomial time. The table has a size of roughly $(k + l)^2$, and each entry can be easily calculated for any ϕ. So the total time is $O(n^2)$ easy stages.