Temos $w \in A$ sempre que $f(w) \in B$ porque f é uma redução de A para B. Por conseguinte, M aceita $f(w)$ sempre que $w \in A$. Além do mais, N roda em tempo polinomial, pois cada um de seus dois estágios roda em tempo polinomial. Note que o estágio 2 roda em tempo polinomial porque a composição de polinômios é um polinômio.

Antes de exibir uma redução de tempo polinomial, introduzimos *3SAT*, um caso especial do problema da satisfazibilidade no qual todas as fórmulas estão em uma forma especial. Um ***literal*** é uma variável booleana ou uma variável booleana negada, como em x ou $\overline{x}$. Uma ***cláusula*** é uma fórmula composta de vários literais conectados por $\vee$s, como em $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4)$. Uma fórmula booleana está na ***forma normal conjuntiva***, chamada ***fnc-fórmula***, se ela compreende várias cláusulas conectadas por $\wedge$s, como em

$$(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6}).$$

Ela é uma ***3fnc-fórmula*** se todas as cláusulas tiverem três literais, como em

$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4) \wedge (x_4 \vee x_5 \vee x_6).$$

Seja $3SAT = \{\langle\phi\rangle \mid \phi$ é uma 3fnc-fórmula satisfazível$\}$. Em uma fnc-fórmula satisfazível, cada cláusula deve conter pelo menos um literal a que é atribuído o valor 1.

O teorema seguinte apresenta uma redução de tempo polinomial do problema *3SAT* para o problema *CLIQUE*.

TEOREMA 7.32

3SAT é redutível em tempo polinomial a *CLIQUE*.

IDÉIA DA PROVA A redução de tempo polinomial f que mostramos de *3SAT* para *CLIQUE* converte fórmulas para grafos. Nos grafos construídos, os cliques de um dado tamanho correspondem a atribuições que satisfazem à fórmula. Estruturas dentro do grafo são projetadas para imitar o comportamento das variáveis e cláusulas.

PROVA Seja ϕ uma fórmula com k cláusulas tal como

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k).$$

A redução f gera a cadeia $\langle G, k\rangle$, onde G é um grafo não-direcionado definido da seguinte forma.

Os nós em G são organizados em k grupos de três nós cada um chamados de ***triplas***, $t_1, \ldots, t_k$. Cada tripla corresponde a uma das cláusulas em ϕ, e cada nó em uma tripla corresponda a um literal na cláusula associada. Rotule cada nó de G com seu literal correspondente em ϕ.

As arestas de G conectam todos os pares de nós em G, exceto dois tipos de pares. Nenhuma aresta está presente entre nós na mesma tripla e nenhuma aresta está presente entre dois nós com rótulos contraditórios, como x_2 e $\overline{x_2}$. A figura a seguir ilustra essa construção quando $\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$.

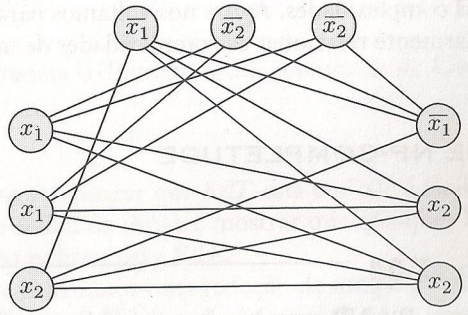

FIGURA 7.33
O grafo que a redução produz a partir de $\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$.

Agora, demonstramos por que essa construção funciona. Mostramos que ϕ é satisfazível sse G tem um k-clique.

Suponha que ϕ tenha uma atribuição que a satisfaz. Nessa atribuição, pelo menos um literal é verdadeiro em cada cláusula. Em cada tripla de G, selecionamos um nó correspondendo a um literal verdadeiro na atribuição que satisfaz a fórmula. Se mais que um literal for verdadeiro em uma cláusula específica, escolhemos um deles arbitrariamente. Os nós que acabam de ser selecionados formam um k-clique. O número de nós selecionado é k, porque escolhemos um para cada uma das k triplas. Cada par de nós selecionados é ligado por uma aresta porque nenhum par se encaixa em uma das exceções descritas anteriormente. Eles não poderiam ser da mesma tripla, porque selecionamos somente um nó por tripla. Eles não poderiam ter rótulos contraditórios porque os literais associados eram ambos verdadeiros na atribuição que satisfaz a fórmula. Por conseguinte, G contém um k-clique.

Suponha que G tenha um k-clique. Nenhum par de nós do clique ocorre na mesma tripla, porque nós na mesma tripla não são conectados por arestas. Conseqüentemente, cada uma das k triplas contém exatamente um dos k nós do clique. Atribuímos valores-verdade às variáveis de ϕ de modo que cada literal que rotula um nó do clique torne-se verdadeiro. Fazer isso é sempre possível, pois dois nós rotulados de maneira contraditória não são conectados por uma

aresta e, portanto, não podem estar ambos no clique. Essa atribuição às variáveis satisfaz ϕ porque cada tripla contém um nó do clique e, assim, cada cláusula contém um literal ao qual é atribuído VERDADEIRO. Logo, ϕ é satisfazível.

Os Teoremas 7.31 e 7.32 nos dizem que, se *CLIQUE* for solúvel em tempo polinomial, o mesmo acontece com *3SAT*. À primeira vista, essa conexão entre esses dois problemas parece um tanto impressionante porque, superficialmente, eles são bastante diferentes. Mas a redutibilidade de tempo polinomial nos permite relacionar suas complexidades. Agora nos voltamos para uma definição que nos permitirá similarmente relacionar as complexidades de uma classe inteira de problemas.

DEFINIÇÃO DE NP-COMPLETUDE

DEFINIÇÃO 7.34

Uma linguagem B é ***NP-completa*** se satisfaz duas condições:

1. B está em NP, e
2. toda A em NP é redutível em tempo polinomial a B.

TEOREMA 7.35

Se B for NP-completa e $B \in \mathrm{P}$, então $\mathrm{P} = \mathrm{NP}$.

PROVA Esse teorema segue diretamente da definição de redutibilidade de tempo polinomial.

TEOREMA 7.36

Se B for NP-completa e $B \leq_{\mathrm{P}} C$ para C em NP, então C é NP-completa.

PROVA Já sabemos que C está em NP, portanto devemos mostrar que toda A em NP é redutível em tempo polinomial a C. Como B é NP-completa, toda linguagem em NP é redutível em tempo polinomial a B e B, por sua vez, é redutível em tempo polinomial a C. Reduções em tempo polinomial se compõem; ou seja, se A for redutível em tempo polinomial a B e B for redutível em tempo polinomial a C, então A é redutível em tempo polinomial a C. Logo, toda linguagem em NP é redutível em tempo polinomial a C.

O TEOREMA DE COOK-LEVIN

Uma vez que temos um problema NP-completo, podemos obter outros por redução de tempo polinomial a partir dele. Entretanto, estabelecer o primeiro problema NP-completo é mais difícil. A seguir, fazemos isso provando que *SAT* é NP-completo.

TEOREMA 7.37

SAT é NP-completo.[2]

Esse teorema reenuncia o Teorema 7.27, o teorema de Cook-Levin, em outra forma.

IDÉIA DA PROVA Mostrar que *SAT* está em NP é fácil, e vamos fazer isso em breve. A parte difícil da prova é mostrar que qualquer linguagem em NP é redutível em tempo polinomial a *SAT*.

Para fazer isso construímos uma redução de tempo polinomial para cada linguagem A em NP para *SAT*. A redução para A toma uma cadeia w e produz uma fórmula booleana ϕ que simula a máquina NP para A sobre a entrada w. Se a máquina aceita, ϕ tem uma atribuição que a satisfaz que corresponde à computação de aceitação. Se a máquina não aceita, nenhuma atribuição satisfaz ϕ. Conseqüentemente, w está em A se e somente se ϕ é satisfazível.

Construir realmente a redução para funcionar dessa maneira é uma tarefa conceitualmente simples, embora devamos lidar com muitos detalhes. Uma fórmula booleana pode conter as operações booleanas E, OU e NÃO e essas operações formam a base para os circuitos usados em computadores eletrônicos. Logo, o fato de que podemos projetar uma fórmula booleana para simular uma máquina de Turing não é surpreendente. Os detalhes estão na implementação dessa idéia.

PROVA Primeiro, mostramos que *SAT* está em NP. Uma máquina de tempo polinomial não-determinístico pode adivinhar uma atribuição para uma dada fórmula ϕ e aceitar se a atribuição satisfaz ϕ.

A seguir, tomamos qualquer linguagem A em NP e mostramos que A é redutível em tempo polinomial a *SAT*. Seja N uma máquina de Turing não-determinística que decide A em tempo n^k para alguma constante k. (Por conveniência, assumimos, na verdade, que N roda em tempo $n^k - 3$, mas apenas aqueles leitores interessados em detalhes deveriam se preocupar com essa questão menor.) A seguinte noção ajuda a descrever a redução.

Um ***tableau*** para N sobre w é uma tabela $n^k \times n^k$ cujas linhas são as configurações de um ramo da computação de N sobre a entrada w, como mostrado na Figura 7.38.

[2] Uma prova alternativa desse teorema aparece na Seção 9.3 na página 373.

7.5 PROBLEMAS NP-COMPLETOS ADICIONAIS

O fenômeno da NP-completude está espalhado. Problemas NP-completos aparecem em muitas áreas. Por razões que não são bem entendidas, a maioria dos problemas NP que ocorrem naturalmente está em P ou é NP-completa. Se você busca um algoritmo de tempo polinomial para um novo problema NP, é sensato gastar parte de seu esforço tentando provar que ele é NP-completo, pois isso pode evitar que você trabalhe para encontrar um algoritmo de tempo polinomial que não existe.

Nesta seção apresentamos teoremas adicionais mostrando que várias linguagens são NP-completas. Esses teoremas provêm exemplos das técnicas que são usadas em provas desse tipo. Nossa estratégia geral é exibir uma redução de tempo polinomial a partir de *3SAT* para a linguagem em questão, embora às vezes reduzamos a partir de outras linguagens NP-completas quando é mais conveniente.

Quando construímos uma redução de tempo polinomial a partir de *3SAT* para uma linguagem, procuramos por estruturas naquela linguagem que possam simular as variáveis e cláusulas nas fórmulas booleanas. Essas estruturas são às vezes chamadas ***engrenagens***. Por exemplo, na redução de *3SAT* para *CLIQUE* apresentada no Teorema 7.32, os nós individuais simulam variáveis e triplas de nós simulam cláusulas. Um nó individual pode ou não ser um membro do clique, o que corresponde a uma variável que pode ou não ser verdadeira em uma atribuição que satisfaz a fórmula. Cada cláusula deve conter um literal a que é atribuído VERDADEIRO e que corresponde à forma pela qual cada tripla deve conter um nó no clique se o tamanho alvo é para ser atingido. O seguinte corolário do Teorema 7.32 afirma que *CLIQUE* é NP-completa.

COROLÁRIO 7.43

CLIQUE é NP-completa.

O PROBLEMA DA COBERTURA DE VÉRTICES

Se G é um grafo não-direcionado, uma ***cobertura de vértices*** de G é um subconjunto dos nós onde toda aresta de G toca um dos nós. O problema da cobertura de vértices pergunta se um grafo contém uma cobertura de vértices de um tamanho especificado:

$$COB\text{-}VERT = \{\langle G, k\rangle \mid G \text{ é um grafo não-direcionado que tem uma cobertura de vértices de } k\text{-nós}\}.$$

TEOREMA 7.44

COB-VERT é NP-completo.

IDÉIA DA PROVA Para mostrar que *COB-VERT* é NP-completo temos de mostrar que ele está em NP e que todos os problemas NP são redutíveis em tempo polinomial a ele. A primeira parte é fácil; um certificado é simplesmente uma cobertura de vértices de tamanho k. Para provar a segunda parte, mostramos que *3SAT* é redutível em tempo polinomial a *COB-VERT*. A redução converte uma 3fnc-fórmula ϕ em um grafo G e um número k, de modo que ϕ seja satisfazível sempre que G tenha uma cobertura de vértices com k nós. A conversão é feita sem saber se ϕ é satisfazível. Com efeito, G simula ϕ. O grafo contém engrenagens que imitam as variáveis e as cláusulas da fórmula. Projetar essas engrenagens requer um pouco de engenhosidade.

Para a engrenagem das variáveis, procuramos por uma estrutura em G que possa participar da cobertura de vértices em uma das duas maneiras possíveis, correspondendo às duas possíveis atribuições de verdade à variável. Dois nós conectados por uma aresta é uma estrutura que funciona, porque um desses nós tem que aparecer na cobertura de vértices. Arbitrariamente, atribuímos VERDADEIRO e FALSO a esses dois nós.

Para a engrenagem das cláusulas, buscamos uma estrutura que induza a cobertura de vértices a incluir nós nas engrenagens de variáveis correspondendo a pelo menos um literal verdadeiro na cláusula. A engrenagem contém três nós e arestas adicionais de modo que qualquer cobertura de vértices tenha que incluir pelo menos dois dos nós ou possivelmente todos os três. Somente dois nós seriam necessários se um dos nós da engrenagem de vértices ajudasse cobrindo uma aresta, como aconteceria se o literal associado satisfaz essa cláusula. Caso contrário, três nós seriam necessários. Finalmente, escolhemos k de modo que a cobertura de vértices procurada tem um nó por engrenagem de variáveis e dois nós por engrenagem de cláusulas.

PROVA Aqui estão os detalhes de uma redução de *3SAT* para *COB-VERT* que opera em tempo polinomial. A redução mapeia uma fórmula booleana ϕ para um grafo G e um valor k. Para cada variável x em ϕ, produzimos uma aresta conectando dois nós. Rotulamos os dois nós nessa engrenagem x e $\overline{x}$. Fazer x VERDADEIRO corresponde a selecionar o nó esquerdo para a cobertura de vértices, enquanto que FALSO corresponde ao nó direito.

As engrenagens para as cláusulas são um pouco mais complexas. Cada engrenagem de cláusulas é uma tripla de três nós que são rotulados com três literais da cláusula. Esses três nós são conectados um ao outro e aos nós nas engrenagens de variáveis que têm os rótulos idênticos. Por conseguinte, o número total de nós que aparecem em G é $2m + 3l$, onde ϕ tem m variáveis e l cláusulas. Faça k igual a $m + 2l$.

Por exemplo, se $\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$, a redução produz $\langle G, k\rangle$ a partir de ϕ, onde $k = 8$ e G toma a forma mostrada na Figura 7.45.

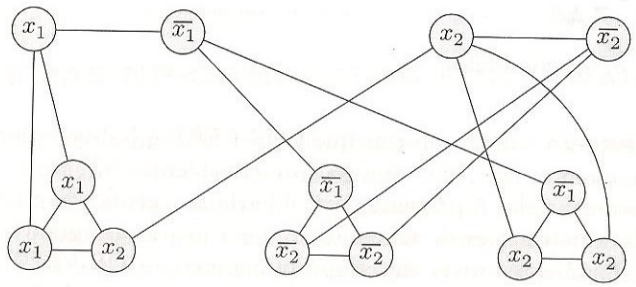

FIGURA 7.45
O grafo que a redução produz a partir de
$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$.

Para provar que essa redução funciona, precisamos mostrar que ϕ é satisfazível se e somente se G tem uma cobertura de vértices com k nós. Começamos com uma atribuição que satisfaz a fórmula. Primeiro colocamos os nós das engrenagens de variáveis que correspondem aos literais verdadeiros na atribuição na cobertura de vértices. Então, selecionamos um literal verdadeiro em toda cláusula e colocamos os dois nós remanescentes de toda engrenagem de cláusulas na cobertura de vértices. Agora, temos um total de k nós. Eles cobrem todas as arestas porque toda engrenagem de variáveis é claramente coberta, todas as três dentro de toda engrenagem de cláusulas são cobertas, e todas as arestas entre as engrenagens de variáveis e de cláusulas são cobertas. Logo, G tem uma cobertura de vértices com k nós.

Segundo, se G tem uma cobertura de vértices com k nós, mostramos que ϕ é satisfazível construindo a atribuição que a satisfaz. A cobertura de vértices tem que conter um nó em cada engrenagem de variáveis e dois em toda engrenagem de cláusulas de forma a cobrir as arestas das engrenagens de variáveis e as três arestas dentro das engrenagens de cláusulas. Isso dá conta de todos os nós, portanto, não sobra nenhum. Tomamos os nós das engrenagens de variáveis que estão na cobertura de vértices e atribuímos VERDADEIRO aos literais correspondentes. Essa atribuição satisfaz ϕ porque cada uma das três arestas conectando as engrenagens de variáveis com cada engrenagem de cláusulas é coberta e somente dois nós da engrenagem de cláusulas estão na cobertura de vértices. Conseqüentemente, uma das arestas tem que ser coberta por um nó de uma engrenagem de variáveis e, portanto, essa atribuição satisfaz a cláusula correspondente.

O PROBLEMA DO CAMINHO HAMILTONIANO

Lembre-se de que o problema do caminho hamiltoniano pergunta se o grafo de entrada contém um caminho de s para t que passa por todo nó exatamente uma vez.

TEOREMA 7.46

CAMHAM é NP-completo.

IDÉIA DA PROVA Mostramos que *CAMHAM* está em NP na Seção 7.3. Para mostrar que todo problema NP é redutível em tempo polinomial a *CAMHAM*, mostramos que *3SAT* é redutível em tempo polinomial a *CAMHAM*. Damos uma maneira de converter 3fnc-fórmulas para grafos na qual caminhos hamiltonianos correspondem a atribuições que satisfazem às fórmulas. Os grafos contêm engrenagens que imitam variáveis e cláusulas. A engrenagem de variáveis é uma estrutura em formato de diamante que pode ser percorrida em uma de duas maneiras, correspondendo a duas atribuições de valores-verdade. A engrenagem de cláusulas é um nó. Assegurar que o caminho passa por cada engrenagem de cláusulas corresponde a assegurar que cada cláusula seja satisfeita.

PROVA Anteriormente, demonstramos que *CAMHAM* está em NP, portanto tudo o que resta a ser feito é mostrar que *3SAT* $\leq_P$ *CAMHAM*. Para cada 3fnc-fórmula ϕ, mostramos como construir um grafo direcionado G com dois nós, s e t, tal que existe um caminho hamiltoniano entre s e t sse ϕ é satisfazível.

Começamos a construção com uma 3fnc-fórmula ϕ contendo k cláusulas:

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k),$$

onde cada a, b e c é um literal x_i ou $\overline{x_i}$. Sejam $x_1, \ldots, x_l$ as l variáveis de ϕ.

Agora mostramos como converter ϕ em um grafo G. O grafo G que construímos tem várias partes para representar as variáveis e cláusulas que aparecem em ϕ.

Represente cada variável x_i como uma estrutura em formato de diamante que contém uma linha horizontal de nós, como mostrado na Figura 7.47. Adiante, especificaremos o número de nós que aparecem na linha horizontal.

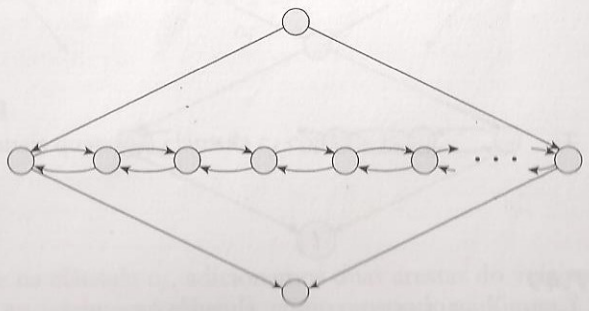

FIGURA 7.47
Representando a variável x_i como uma estrutura no formato de diamante.

Representamos cada cláusula de ϕ como um único nó, como segue.

nó a_2. O caminho não pode entrar em a_2 de c ou a_1, porque o caminho vai para outros lugares a partir desses nós. O caminho não pode entrar em a_2 a partir de a_3, porque a_3 é o único nó disponível para o qual a_2 aponta, assim, o caminho deve deixar a_2 via a_3. Logo, um caminho hamiltoniano tem de ser normal. Essa redução obviamente opera em tempo polinomial, e a prova está completa.

A seguir, consideramos uma versão não-direcionada do problema do caminho hamiltoniano, chamado *CAMHAMN*. Para mostrar que *CAMHAMN* é NP-completo, damos uma redução de tempo polinomial a partir da versão direcionada do problema.

TEOREMA 7.55

CAMHAMN é NP-completo.

PROVA A redução toma um grafo direcionado G com nós s e t e constrói um grafo não-direcionado G' com nós s' e t'. O grafo G tem um caminho hamiltoniano de s para t sse G' tem um caminho hamiltoniano de s' para t'. Descrevemos G' da seguinte forma.

Cada nó u de G, excetuados s e t, é substituído por uma tripla de nós u^{entra}, u^{meio} e u^{sai} em G'. Os nós s e t em G são substituídos por nós s^{sai} e t^{entra} em G'. Aparecem arestas de dois tipos em G'. Primeiro, arestas conectam u^{meio} com u^{entra} e este com u^{sai}. Segundo, uma aresta conecta u^{sai} com v^{entra} se uma aresta vai de u para v em G. Isso completa a construção de G'.

Podemos demonstrar que essa construção funciona mostrando que G tem um caminho hamiltoniano de s para t sse G' tem um caminho hamiltoniano de s^{sai} para t^{entra}. Para mostrar uma direção, observamos que um caminho hamiltoniano P em G,

$$s, u_1, u_2, \ldots, u_k, t,$$

tem um caminho hamiltoniano P' em G',

$$s^{\text{sai}}, u_1^{\text{entra}}, u_1^{\text{meio}}, u_1^{\text{sai}}, u_2^{\text{entra}}, u_2^{\text{meio}}, u_2^{\text{sai}}, \ldots, t^{\text{entra}}.$$

Para mostrar a outra direção, afirmamos que qualquer caminho hamiltoniano em G' de s^{sai} para t^{entra} em G' deve ir de uma tripla de nós para uma tripla de nós, exceto pelo início e o fim, como faz o caminho P' que acabamos de descrever. Isso completaria a prova porque qualquer desses caminhos tem um caminho hamiltoniano correspondente em G. Provamos a afirmação seguindo o caminho começando no nó s^{sai}. Observe que o nó seguinte no caminho deve ser u_i^{entra} para algum i, pois somente aqueles nós são conectados a s^{sai}. O nó seguinte deve ser u_i^{meio}, porque não há outra maneira disponível para incluir u_i^{meio} no caminho hamiltoniano. Após u_i^{meio}, vem u_i^{sai}, porque esse é o único outro ao qual u_i^{meio} está conectado. O nó seguinte deve ser u_j^{entra}, para algum j, pois nenhum outro nó disponível está conectado a u_i^{sai}. O argumento então se repete até que t^{entra} seja atingido.

O PROBLEMA DA SOMA DE SUBCONJUNTOS

Retomemos o problema *SOMA-SUBC* definido na página 284. Naquele problema, nos era dada uma coleção de números $x_1, \ldots, x_k$ juntamente com um número alvo t, e tínhamos que determinar se a coleção contém uma subcoleção cuja soma é t. Agora mostramos que esse problema é NP-completo.

TEOREMA 7.56

SOMA-SUBC é NP-completo.

IDÉIA DA PROVA Já mostramos que *SOMA-SUBC* está em NP no Teorema 7.25. Provamos que todas as linguagens em NP são redutíveis em tempo polinomial a *SOMA-SUBC* reduzindo a linguagem NP-completa *3SAT* a ela. Dada uma 3fnc-fórmula ϕ construímos uma instância do problema *SOMA-SUBC* que contém uma subcoleção cuja soma é o alvo t se e somente se ϕ é satisfazível. Chame essa subcoleção T.

Para conseguir essa redução, encontramos estruturas do problema *SOMA-SUBC* que representem variáveis e cláusulas. A instância do problema *SOMA-SUBC* que construímos contém números de grande magnitude apresentados em notação decimal. Representamos variáveis por pares de números e cláusulas por certas posições nas representações decimais dos números.

Representamos a variável x_i por dois números, y_i e z_i. Provamos que ou y_i ou z_i deve estar em T para cada i, o que estabelece a codificação para o valor-verdade de x_i na atribuição que satisfaz a fórmula.

Cada posição de cláusula contém um certo valor no alvo t, o que impõe um requisito no subconjunto T. Provamos que esse requisito é o mesmo que aquele na cláusula correspondente — a saber, que a um dos literais nessa cláusula é atribuído VERDADEIRO.

PROVA Já sabemos que *SOMA-SUBC* $\in$ NP, portanto, agora mostramos que *3SAT* $\leq_{\text{P}}$ *SOMA-SUBC*.

Seja ϕ uma fórmula booleana com as variáveis $x_1, \ldots, x_l$ e as cláusulas $c_1, \ldots, c_k$. A redução converte ϕ para uma instância do problema *SOMA-SUBC* $\langle S, t \rangle$, na qual os elementos de S e o número t são as linhas na tabela na Figura 7.57, expressos na notação decimal ordinária. As linhas acima da linha dupla são rotuladas

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

e compreende os elementos de S. A linha abaixo da linha dupla é t.

Assim, S contém um par de números, y_i, z_i, para cada variável x_i em ϕ. A representação decimal desses números está dada em duas partes, como indicado na tabela. A parte da esquerda compreende um 1 seguido de $l - i$ 0s. A parte da direita contém um dígito para cada cláusula, onde o j-ésimo dígito de y_i é 1 se a cláusula c_j contém o literal x_i e o j-ésimo dígito de z_i é 1 se a cláusula c_j contém o literal $\overline{x_i}$. Os dígitos não especificados como sendo 1 são 0.

A tabela está parcialmente preenchida para ilustrar as cláusulas amostra, c_1, c_2 e c_k:

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

Adicionalmente, S contém um par de números, g_j, h_j, para cada cláusula c_j. Esses dois números são iguais e consistem de um 1 seguido por $k - j$ 0s.

Finalmente, o número alvo t, na linha inferior da tabela, consiste de l 1s seguidos por 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

FIGURA 7.57
Reduzindo *3SAT* a *SOMA-SUBC*.

Agora mostramos por que essa construção funciona. Demonstramos que ϕ é satisfazível sse algum subconjunto de S soma t.

Suponha que ϕ seja satisfazível. Construímos um subconjunto de S da seguinte forma. Selecionamos y_i se a x_i é atribuído VERDADEIRO na atribuição que satisfaz a fórmula ou z_i se a x_i é atribuído FALSO. Se somarmos o que selecionamos até então, obtemos um 1 em cada um dos primeiros l dígitos, porque selecionamos ou y_i ou z_i para cada i. Além disso, cada um dos últimos k dígitos é um número entre 1 e 3, porque cada cláusula é satisfeita e, portanto, contém entre 1 e 3 literais verdadeiros. Agora selecionamos ainda uma quantidade suficiente dos números g e h para trazer cada um dos últimos k dígitos para 3, portanto, atingindo o alvo.

Suponha que um subconjunto de S tenha t como soma. Construímos uma atribuição que satisfaz ϕ após fazer várias observações. Primeiro, todos os dígitos de membros de S são 0 ou 1. Além disso, cada coluna da tabela que descreve S contém no máximo cinco 1s. Logo, nunca ocorre um "vai-um" para a próxima coluna quando um subconjunto de S é somado. Para obter um 1 em cada uma das l primeiras colunas, o subconjunto deve ter y_i ou z_i para cada i, mas não ambos.

Agora, construímos a atribuição que satisfaz a fórmula. Se o subconjunto contém y_i, atribuímos VERDADEIRO a x_i; caso contrário, atribuímos FALSO. Essa atribuição deve satisfazer ϕ, porque em cada uma das k colunas finais a soma é sempre 3. Na coluna c_j, pode vir no máximo 2 de g_j e h_j; logo, pelo menos 1 nesta coluna deve vir de algum y_i ou z_i do subconjunto. Se for y_i, então aparece x_i em c_j e é atribuído o valor VERDADEIRO, de forma que c_j é satisfeita. Se for z_i, então $\overline{x_i}$ ocorre in c_j e é atribuído FALSO a x_i, e assim c_j é satisfeita. Portanto, ϕ é satisfeita.

Finalmente, devemos estar certos de que a redução possa ser feita em tempo polinomial. A tabela tem um tamanho em torno de $(k+l)^2$, e cada entrada pode ser facilmente calculada para qualquer ϕ. Assim, o tempo total é $O(n^2)$ estágios fáceis.

EXERCÍCIOS

7.1 Responda VERDADEIRO ou FALSO para cada parte.

a. $2n = O(n)$.
b. $n^2 = O(n)$.
[R]c. $n^2 = O(n \log^2 n)$.
[R]d. $n \log n = O(n^2)$.
e. $3^n = 2^{O(n)}$.
f. $2^{2^n} = O(2^{2^n})$.

7.2 Responda VERDADEIRO ou FALSO para cada parte.

a. $n = o(2n)$.
b. $2n = o(n^2)$.
[R]c. $2^n = o(3^n)$.
[R]d. $1 = o(n)$.
e. $n = o(\log n)$.
f. $1 = o(1/n)$.

7.3 Quais dos seguintes pares de números são primos entre si? Mostre os cálculos que levaram às suas conclusões.

a. 1274 e 10505
b. 7289 e 8029