Construção e Análise de Algoritmos

Complexidade Computacional

Livro "Introdução a Teoria da Computação" de Michael **Sipser** (Editora Thomson)

DEFINIÇÃO DE 3-SAT

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*.

REDUÇÃO POLINOMIAL DE 3SAT para CLIQUE
Consequência: CLIQUE é NP-Completo

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 $\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$.

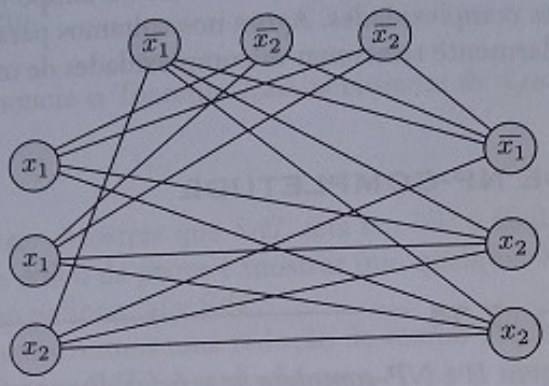

FIGURA 7.33
O grafo que a redução produz a partir de $\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$.

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.

DEFINIÇÃO DE NP-COMPLETO

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.

NP-COMPLETUDE de COBERTURA DE VÉRTICES

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.

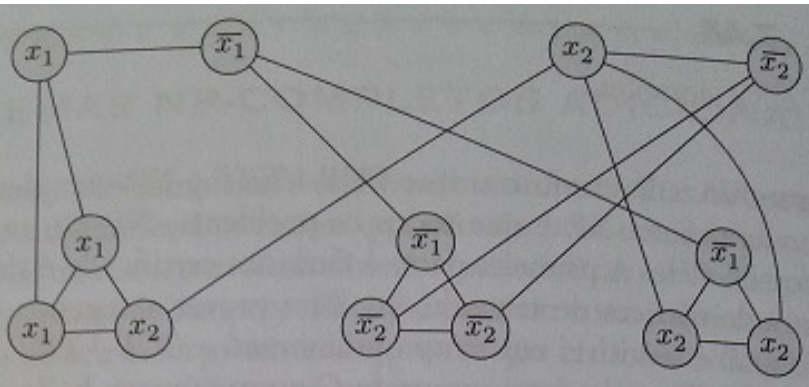

FIGURA 7.45
O grafo que a redução produz a partir de
$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$.

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.

NP-COMPLETUDE de SOMA DE SUBCONJUNTO

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* $\leq_{\mathrm{P}}$ *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 su-

ficiente 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.