

Teoria da Computação

Complexidade de Tempo

parte 3

Leonardo Takuno
{leonardo.takuno@gmail.com}

Centro Universitário Senac

Sumário

- 1 Teorema de Cook-Levin
- 2 Problemas NP-Completo

Sumário

- 1 Teorema de Cook-Levin
- 2 Problemas NP-Completo

Teorema de Cook-Levin

Nos anos de 1970 Stephen Cook e Leonid Levin descobriram, independentemente, que existem certos problemas em NP cuja complexidade está relacionada a todos os outros problemas da classe NP - estes problemas são chamados de NP-Completo.

Teorema de Cook-Levin

Nós vimos anteriormente que os problemas NP-Completo estão relacionados a outros problemas NP via reduções polinomiais.

Problema da satisfabilidade (SAT)

Uma fórmula booleana é uma expressão que envolve variáveis booleanas (x , y , etc) e operações (\wedge , \vee , \neg , onde $\neg x = \bar{x}$).

$$\phi = (\bar{x} \wedge y) \vee (x \wedge \bar{y})$$

Uma fórmula booleana é verdadeira se alguma atribuição de variáveis (*true* ou *false*) torna o valor da fórmula igual a *true*.

Problema da satisfabilidade (SAT)

$$\phi = (\bar{x} \wedge y) \vee (x \wedge \bar{z})$$

É satisfazível pois a atribuição

$x = \text{false},$

$y = \text{true},$

$z = \text{false}.$

faz ϕ valor *true*.

Dizemos que a atribuição satisfaz ϕ .

Problema da satisfabilidade (SAT)

O **problema da satisfabilidade** é testar se uma fórmula booleana é satisfatível.

$$SAT = \{ \langle \phi \rangle \mid \phi \text{ é uma fórmula booleana satisfatível} \}$$

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

Idéia da prova:

- Primeiro mostramos que $SAT \in NP$
- Para qualquer linguagem $A \in NP$ mostramos que $A \leq_p SAT$

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

Prova:

- Primeiro mostramos que $SAT \in NP$
- Uma máquina de tempo polinomial não-determinístico pode adivinhar uma atribuição para uma dada fórmula ϕ e aceitar se a atribuição satisfaz ϕ

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

Prova (continuação):

- Precisamos mostrar que $A \leq_p SAT$ para todo $A \in NP$.
- Isto é feito por simular as computações de MTN decidindo A sobre alguma string w usando fórmulas booleanas tal que

$$w \in A \Leftrightarrow f(w) \in SAT$$

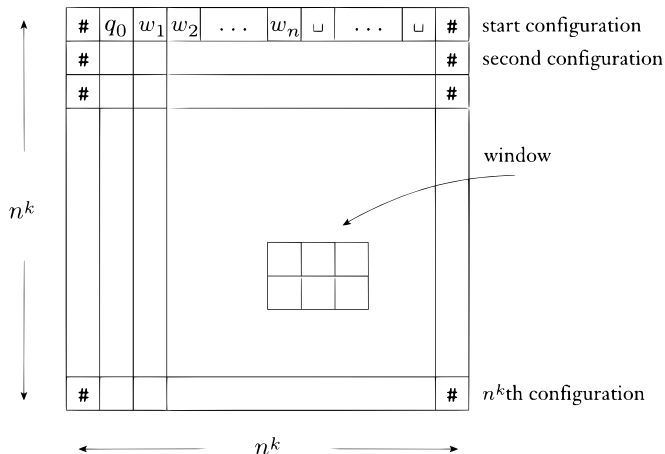
onde f converte a string w na fórmula booleana $f(w)$.

Teorema de Cook-Levin

Definição: Um **tableau** para N sobre w é uma tabela $n^k \times n^k$ cujas linhas são configurações de um ramo de computação de N sobre w .

- 1 C_1 é a configuração inicial de N sobre w
- 2 C_{n^k} (a última linha) é uma configuração de aceitação.
- 3 Para toda linha i , C_i pode produzir C_{i+1} de acordo com as regras de N .

Teorema de Cook-Levin



Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- Todo *tableau* representa um computação de MT sobre a entrada w .
- Desde que N é não-determinístico, podem existir muitas computações.
- Consequentemente, pode haver muitos tableaux. Cada um correspondente a um ramo específico de computação.
- Um tableau de **aceitação** para N sobre w corresponde a um ramo de computação de aceitação de N sobre w .

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- Defina uma redução em tempo polinomial f de A para SAT. Sobre a entrada w , a redução produz a fórmula ϕ . Segue a descrição da redução:
 - A MT N tem um conjunto Q de estados e uma fita de alfabeto Γ
 - O conteúdo da célula é o conjunto $C = Q \cup \Gamma \cup \{\#\}$
 - Para cada i e j entre 1 e n^k e para cada $s \in C$, defina uma variável booleana da seguinte maneira:

$$x_{i,j,s} = \begin{cases} true & \text{se } celula[i,j] = s \\ false & \text{caso contrário} \end{cases}$$

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- Vamos representar um tableau como uma fórmula ϕ .
- Para obter ϕ , nós tomamos a conjunção de quatro sub-fórmulas.

$$\phi = \phi_{\text{celula}} \wedge \phi_{\text{inicio}} \wedge \phi_{\text{movimento}} \wedge \phi_{\text{aceita}}$$

- ϕ_{celula} : cada célula do tableau contém exatamente um símbolo $s \in Q \cup \Gamma \cup \{\#\}$
- ϕ_{inicio} : a primeira linha contém um q_0 seguido por w e seguido por branco.
- $\phi_{\text{movimento}}$: cada passo de computação conforme as regras de N .
- ϕ_{aceita} : C_{n^k} é uma configuração de aceitação.

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

$$\phi = \phi_{\text{celula}} \wedge \phi_{\text{inicio}} \wedge \phi_{\text{movimento}} \wedge \phi_{\text{aceita}}$$

- Se as quatro sub-fórmulas são satisfeitas, então há um tableau válido.
- Agora, a primeira coisa que devemos garantir de modo a obter uma correspondência entre uma atribuição e um tableau é que a atribuição ligue exatamente uma variável para cada célula.

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- A fórmula $\phi_{\text{célula}}$ garante esse requisito:

$$\phi_{\text{célula}} = \bigwedge_{1 \leq i, j \leq n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s, t \in C \\ s \neq t}} \left(\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}} \right) \right) \right]$$

- Na fórmula acima, o primeiro \wedge significa “cada célula do tableau”
- O primeiro parênteses indica “contém pelo menos um símbolo”
- O segundo parênteses estipula que “não mais que uma variável está ligada para cada célula”.
- Isto pode ser produzido em $O(n^{2k})$ passos.

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- A fórmula $\phi_{início}$ garante que a primeira linha da tabela é a configuração inicial:

$$\begin{aligned}\phi_{início} = & x_{1,1,\#} \wedge x_{1,2,q_0} \wedge \\ & x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \cdots \wedge x_{1,n+2,w_n} \wedge \\ & x_{1,n+3,\sqcup} \wedge \cdots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}\end{aligned}$$

- Isto pode ser produzido em tempo polinomial.

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- A fórmula ϕ_{aceita} garante que uma configuração de aceitação ocorre no tableau.

$$\phi_{aceita} = \bigvee_{1 \leq i, j \leq n^k} x_{i,j,q_{aceita}}.$$

- Isto pode ser produzido em tempo polinomial

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- A fórmula $\phi_{movimento}$ assegura que cada linha da tabela corresponde a uma configuração que segue legalmente da configuração da linha precedente conforme as regras de N .
- Uma janela 2×3 é **legal** se ela não viola as ações especificadas pela função de transição de N .

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- Exemplo : $\delta(q_1, a) = \{q_1, b, D\}$ e
 $\delta(q_1, b) = \{(q_2, c, E), (q_2, a, D)\}$

(a)

a	q_1	b
q_2	a	c

(b)

a	q_1	b
a	a	q_2

(c)

a	a	q_1
a	a	b

(d)

#	b	a
#	b	a

(e)

a	b	a
a	b	q_2

(f)

b	b	b
c	b	b

Janelas legais

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- As janelas mostradas abaixo não são legais para a máquina N .

(a)

a	b	a
a	a	a

(b)

a	q_1	b
q_1	a	a

(c)

b	q_1	b
q_2	b	q_2

Teorema de Cook-Levin

Afirmção: Se a linha superior da tabela for a configuração inicial e toda janela na tabela for legal, cada linha da tabela é uma configuração que segue legalmente da precedente.

Prova:

- Considere as configurações C_i e C_{i+1} , chamadas de superior e inferior, respectivamente.
- Na configuração superior, toda célula que não é adjacente a um símbolo de estado e que não contém um símbolo de fronteira $\#$, é a célula central superior em uma janela cuja linha superior não contém nenhum estado.
- Por conseguinte, esse símbolo deve aparecer imutável na posição central inferior da janela. Logo, ele aparece na mesma posição na configuração inferior.

Teorema de Cook-Levin

Afirmção: Se a linha superior da tabela for a configuração inicial e toda janela na tabela for legal, cada linha da tabela é uma configuração que segue legalmente da precedente.

Prova (continuação):

- A janela contendo o símbolo de estado na célula central superior garante que as três posições correspondentes sejam atualizadas consistentemente com a função de transição.
- Consequentemente, se a configuração superior for uma configuração legal, o mesmo acontece com a configuração inferior, e a inferior segue a superior conforme as regras de N.

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

... **voltando à construção de $\phi_{movimento}$**

- $\phi_{movimento}$ estipula que todas as janelas no tableau são legais. Cada janela contém seis células, que podem ser inicializada de um número fixo de maneiras para originar uma janela legal.

$$\phi_{movimento} = \bigwedge_{1 < i \leq n^k, 1 < j < n^k} (\text{a janela } (i, j) \text{ é legal})$$

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- Nessa fórmula, substituímos o texto “a janela (i, j) é legal” pela fórmula a seguir :

Escrevemos o conteúdo de seis células de uma janela como

a_1, \dots, a_6

$$\bigvee_{a_1, \dots, a_6} (x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6})$$

- Requer tempo $O(n^{2k})$.

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- Para resumir:
 - Se $w \in A \Rightarrow N$ aceita $w \Rightarrow \exists$ um tableau válido $\Rightarrow \phi(w)$ é satisfazível
 - Se $w \notin A \Rightarrow N$ rejeita $w \Rightarrow \nexists$ um tableau válido $\Rightarrow \phi(w)$ não é satisfazível

Teorema de Cook-Levin

Teorema (Cook-Levin): SAT é NP-Completo

- Para obter ϕ , nós tomamos a conjunção de quatro sub-fórmulas.

$$\phi = \phi_{celula} \wedge \phi_{inicio} \wedge \phi_{movimento} \wedge \phi_{aceita}$$

- Requer $O(n^{2k})$ de tempo e espaço: complexidade polinomial.
- Conclusões:
 - O mapeamento de w para ϕ é uma redução de tempo polinomial do problema $A \in NP$ para SAT:

$$A \leq_p SAT \text{ para todo } A \in NP$$

Como $SAT \in NP$, então SAT é NP-Completa. \square

Sumário

- 1 Teorema de Cook-Levin
- 2 Problemas NP-Completo

3SAT

- **Literal:** variável booleana (x) ou sua negação (\bar{x})
- **Cláusula:** Um conjunto de literais conectados por OR, ex:
 $(x \vee \bar{y} \vee z)$
- Uma fórmula ϕ está na **forma normal conjuntiva** (FNC), se houver cláusulas conectados por AND.
Ex:

$$\phi(x, y, z) = (x \vee \bar{y}) \wedge (z) \wedge (\bar{z} \vee y \vee \bar{y})$$

Uma **3fnc-fórmula** tem somente cláusulas com 3 literais:

$$3SAT = \{\langle \phi \rangle \mid \phi \text{ é uma 3fnc-fórmula satisfazível}\}$$

3SAT

Teorema: $3SAT \in \text{NP-Completo}$ **Prova:**

- Mostramos isto ao construir uma redução de SAT para $3SAT$
- Primeiro observe que $\phi \in SAT$ pode ser reescrito em FNC $\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ onde cada cláusula c_i é uma disjunção de booleanos digamos $a_1, \cdots a_n$.
- Agora construímos uma redução $f : SAT \rightarrow 3SAT$ tal que $f(\phi) = \phi_{3SAT}$

3SAT

- Substituiremos cada c_i em ϕ por uma coleção de cláusulas com 3 literais. Mais especificamente seja $c_i = a_1 \vee a_2 \vee \dots \vee a_k$, onde cada a_i é uma variável booleana, então
 - $k = 1$: Aqui $c_i = a_1$. Use variáveis adicionais z_1 e z_2 para construir 3cnf
$$(a_1 \vee z_1 \vee z_2) \wedge (a_1 \vee \overline{z_1} \vee z_2) \wedge (a_1 \vee z_1 \vee \overline{z_2}) \wedge (a_1 \vee \overline{z_1} \vee \overline{z_2})$$
 - $k = 2$: Aqui $c_i = (a_1 \vee a_2)$. Use variáveis adicionais z_1 e z_2 para construir 3cnf
$$(a_1 \vee a_2 \vee z_1) \wedge (a_1 \vee a_2 \vee \overline{z_1})$$
 - $k = 3$: Aqui $c_i = (a_1 \vee a_2 \vee a_3)$. Já está em 3cnf, não há nada a fazer.

3SAT

- $k > 3$: Aqui $c_i = (a_1 \vee a_2 \vee \dots \vee a_k)$, use variáveis adicionais z_1, z_2, \dots, z_{k-3} para construir as cláusulas em 3cnf

$$\begin{aligned} & (a_1 \vee a_2 \vee z_1) \wedge \\ & (\overline{z_1} \vee a_3 \vee z_2) \wedge \\ & (\overline{z_2} \vee a_4 \vee z_3) \wedge \\ & (\overline{z_3} \vee a_5 \vee z_4) \wedge \\ & \quad \dots \wedge \\ & (\overline{z_{k-3}} \vee a_{k-1} \vee a_k) \end{aligned}$$

3SAT

- Mostramos que f é uma redução de tempo polinomial.
- Primeiro observe que o número máximo de variáveis que podem ocorrer em uma cláusula ϕ é n .
- Também observe que há m cláusulas.
- Portanto, o número máximo de conversões é limitado a $O(mn)$ que é claramente polinomial.
- Concluimos que f é uma função de tempo polinomial.

3SAT

Agora temos que mostrar que

$$\phi \in SAT \Leftrightarrow f(\phi) \in 3SAT$$

3SAT

Para o caso que $k \leq 3$:

- "se": Note que quando ϕ é satisfeito então $f(\phi)$ também é.
- "somente se": Para o reverso note que se $f(\phi)$ é satisfeito, nós simplesmente restringimos as variáveis que aparecem em ϕ para obter uma atribuição que satisfaz ϕ .

3SAT

"Se ...": Para o caso que $k > 3$, dado uma atribuição em alguma cláusula c_i em ϕ :

- (a) Se a_1 ou a_2 é true, atribua todas as variáveis adicionais para false. Neste caso o primeiro literal em cada cláusula é true.
- (b) Se a_{k-1} ou a_k é true, atribua todas as variáveis adicionais para true. Neste caso o terceiro literal em cada cláusula é true.
- (c) Caso contrário, se a_l é true, atribua z_j para true quando $1 \leq j \leq l - 2$ e o valor false quando $l - 1 \leq j \leq k - 3$. Neste caso o terceiro literal em cada cláusula precede àquela que inclui a_l é true, enquanto que o primeiro literal em cada cláusula sucessora àquela que inclui a_l é true.

Assim, a satisfabilidade de ϕ implica que a satisfabilidade de $f(\phi)$.

3SAT

"Somente se ...": Para o reverso, simplesmente restrinja a atribuição que satisfaz $f(\phi)$ para variáveis que ocorrem em ϕ . \square

CLIQUE

Teorema:CLIQUE \in NP-Completo

Prova: Provamos isto por uma redução polinomial f de $3SAT$ para $CLIQUE$, tal que

$$\phi_k \in 3SAT \Leftrightarrow f(\phi_k) \in CLIQUE,$$

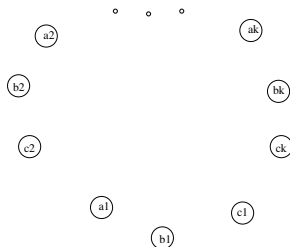
onde ϕ_k é uma $3cnf$ -fórmula com k cláusulas e $f(\phi_k) = \langle G, k \rangle$.

CLIQUE

Dado

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

A redução $f(\phi_k)$ gera a cadeia $\langle G, k \rangle$ onde G é um grafo não-direcionado. Os nós são organizados em triplas que representam os literais das cláusulas.



CLIQUE

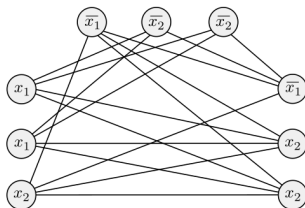
Construímos arestas conectando todos os nós exceto para

- nós que estão na mesma tripla, e
- nós que tem rótulos contraditórios, i.e., x e \bar{x}

Exemplo de construção para

$$\phi_3 = (x_1 \vee \bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_2)$$

produz o grafo



CLIQUE

- É fácil ver que esta é uma construção de tempo polinomial (seja n o número de nós então o algoritmo executa em $O(n^2)$).

CLIQUE

Agora, temos que verificar a condição de redução

$$\phi_k \in 3SAT \Leftrightarrow f(\phi_k) \in CLIQUE$$

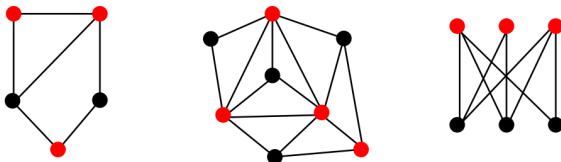
"Se ...": Suponha que ϕ_k tem uma atribuição que a satisfaz, o que significa que cada cláusula tem um literal que é true. Em cada tripla de G escolhemos um nó que corresponde a um literal true. O número de nós selecionados é k , uma em cada tripla. Todos os nós selecionados são conectados por uma aresta. Isto mostra que uma atribuição que satisfaz ϕ_k produz um k -clique

CLIQUE

$$\phi_k \in 3SAT \Leftrightarrow f(\phi_k) \in CLIQUE$$

"Somente se ...": Para o reverso, assumo que G tem um k -clique. Por construção, dois nós não podem estar conectados na mesma tripla. Portanto, cada k tripla contém exatamente um dos nós do k -clique. Cada nó no k -clique denota uma atribuição para true para um literal em ϕ_k . Isto é sempre verdadeiro pois literais opostos não estão conectados. \square

Exemplos:



46 / 86

Cobertura por vértice

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

Teorema:

$COB - VERT \in \text{NP-Completo}$

Cobertura por vértice ($3SAT \leq_p COB - VERT$)

Teorema:

$COB - VERT \in \text{NP-Completo}$

Prova: Descreveremos uma redução polinomial f que recebe uma fórmula ϕ com m variáveis e l cláusulas e exatamente 3 literais por cláusula.

ϕ é satisfatível $\Leftrightarrow G$ possui uma cobertura com k vértices.

Cobertura por vértice ($3SAT \leq_p COB - VERT$)

Teorema:

$COB - VERT \in \text{NP-Completo}$

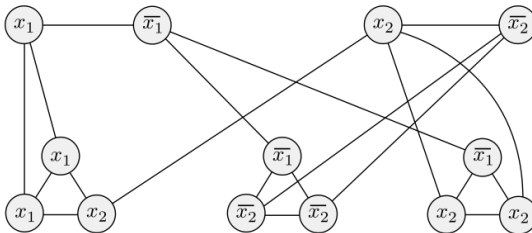
...continuação

- Para cada variável x em ϕ temos em G um vértice x e outro \bar{x} ligados por uma aresta. (engrenagens variáveis)
- Para cada cláusula $(l_1 \vee l_2 \vee l_3)$ temos em G vértices correspondentes a cada literal da cláusula ligados entre si. (engrenagens cláusulas)
- Há ainda uma aresta entre cada vértice correspondente a um literal l_i e o vértice correspondente a uma variável que tem o mesmo rótulo do literal.

Desta forma G tem $2m + 3l$ vértices

Cobertura por vértice ($3SAT \leq_p COB - VERT$)

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



Cobertura por vértice ($3SAT \leq_p COB - VERT$)

ϕ é satisfatível \Rightarrow existe cobertura com $m + 2l$ vértices.

Prova:

- Escolha m vértices das variáveis ou negação das variáveis que correspondem a literais verdadeiros e coloque na cobertura (candidata).
- Em cada cláusula selecione um literal verdadeiro e coloque os outros dois vértices na cobertura (candidata)
- Verifique que a candidata é uma cobertura.

Cobertura por vértice ($3SAT \leq_p COB - VERT$)

ϕ é satisfatível \Rightarrow existe cobertura com $m + 2l$ vértices.

Prova:

- A cobertura por vértices precisa conter pelo menos um vértice da engrenagem de variáveis.
- A cobertura por vértices precisa conter pelo menos dois vértices da engrenagem de cláusulas.
- Tome a atribuição de forma a tornar verdadeira os literais correspondentes a variáveis da cobertura que estão na engrenagens das variáveis. \square

Caminho Hamiltoniano

Teorema:

$CAMHAM = \{ \langle G, s, t \rangle \mid G \text{ é um grafo direcionado com um caminho hamiltoniano de } s \text{ para } t \} \in NP\text{-Completo}$

Um caminho Hamiltoniano é um caminho no grafo que passa em cada nó exatamente uma vez.

Caminho Hamiltoniano

Teorema:

$CAMHAM = \{ \langle G, s, t \rangle \mid G \text{ é um grafo direcionado com um caminho hamiltoniano de } s \text{ para } t \} \in NP\text{-Completo}$

Idéia da prova: Sabemos que $CAMHAM \in NP$. Temos que mostrar que $A \leq_p CAMHAM$, para todo $A \in NP$. Mostramos isso por uma redução de tempo polinomial f de $3SAT$ para $CAMHAM$

$$\phi \in 3SAT \text{ sse } f(\phi) \in CAMHAM$$

onde $f(\phi) = \langle G, s, t \rangle$

Caminho Hamiltoniano

Teorema:
$$CAMHAM = \{ \langle G, s, t \rangle \mid G \text{ é um grafo direcionado com um caminho hamiltoniano de } s \text{ para } t \} \in NP\text{-Completo}$$

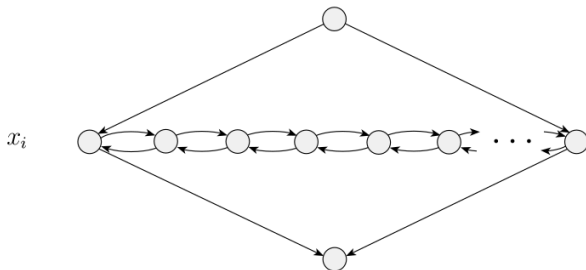
Prova: Assumimos que temos uma fórmula 3SAT ϕ que contenha k cláusulas.

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge (a_1 \vee b_1 \vee c_1)$$

Precisamos construir (em tempo polinomial) um grafo G tal que um caminho Hamiltoniano existe sse ϕ é satisfativo.

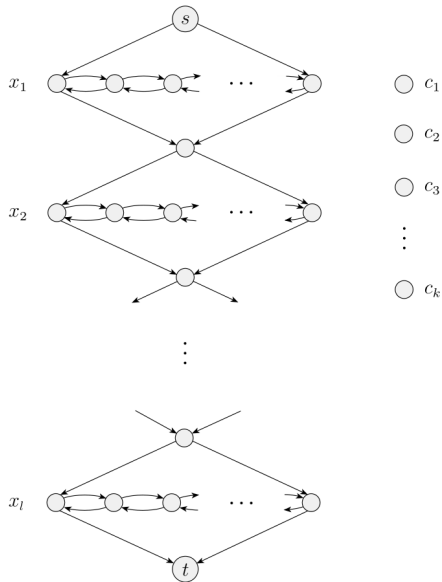
Caminho Hamiltoniano

Para cada variável produzimos um diamante como este.



Representamos cada cláusula de ϕ como um único nó, e dispomos ao lado do diamante.

Caminho Hamiltoniano

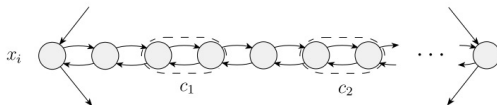


Caminho Hamiltoniano

Agora, mostramos como conectar os diamantes das variáveis aos nós representando as cláusulas.

- Cada estrutura em diamante contém uma linha horizontal de nós conectados por arestas em ambas as direções.
- A linha horizontal tem $3k + 1$ nós além de dois nós nas extremidades pertencente ao diamante.
- Esses nós são agrupados em pares adjacentes, um para cada cláusula, com nós separadores extras em seguida aos pares.

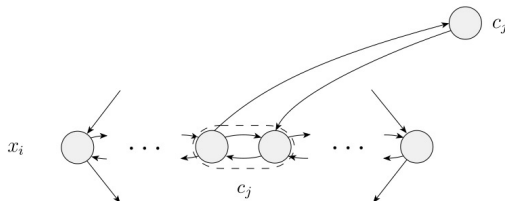
Caminho Hamiltoniano



Os nós horizontais em uma estrutura no formato diamante

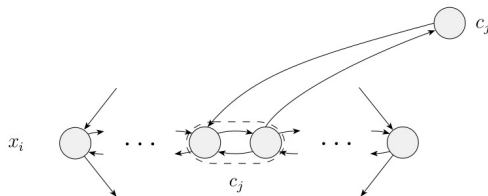
Caminho Hamiltoniano

Se a variável x_i aparece na cláusula c_j , adicionamos duas arestas seguintes do j -ésimo par no i -ésimo diamante ao j -ésimo nó da cláusula.



Caminho Hamiltoniano

Se a variável \bar{x}_i aparece na cláusula c_j , adicionamos duas arestas no i -ésimo diamante ao j -ésimo nó da cláusula, como abaixo

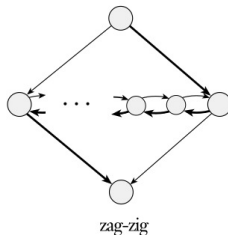
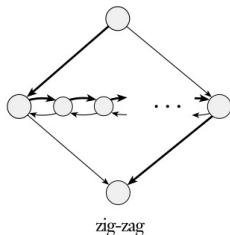


Caminho Hamiltoniano

Dado uma atribuição verdadeira para ϕ , existe um caminho através do Grafo.

- Para exibir um caminho hamiltoniano de s para t , primeiro ignoramos os nós cláusulas.
- O caminho começa em s e passa por cada diamante e termina em t .

Caminho Hamiltoniano



- Se x_i for atribuída VERDADEIRO, o caminho zigue-zagueia pelo diamante.
- Se x_i for atribuída FALSO, o caminho zague-zigueia pelo diamante.

Caminho Hamiltoniano

- Para alcançar os nós cláusulas, incluímos os desvios nos nós horizontais dos diamantes.
- Em cada cláusula selecionamos um literal atribuído VERDADEIRO pela atribuição satisfatória.
- Se selecionar x_i na cláusula c_j , podemos desviar no j -ésimo par no i -ésimo diamante. Como x_i é VERDADEIRO, o caminho é um zigue-zague da esquerda para a direita pelo diamante correspondente a x_i
- Logo, as arestas de c_j estão na ordem correta para permitir um desvio e um retorno.

Caminho Hamiltoniano

- Similarmente, se seleccionar $\overline{x_i}$ na cláusula c_j , podemos desviar no j -ésimo par no i -ésimo diamante. Como x_i é FALSO, o caminho é um zague-zigue da direita para a esquerda pelo diamante correspondente a x_i
- Logo, as arestas de c_j estão novamente na ordem correta para permitir um desvio e um retorno.

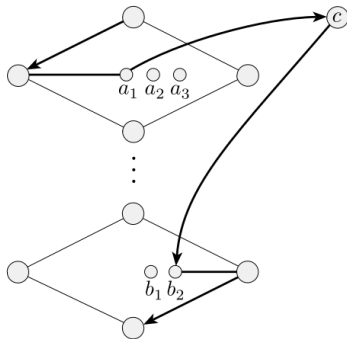
Caminho Hamiltoniano

Para a direção reversa, G tem um caminho hamiltoniano de s para t , exibimos uma atribuição satisfetora para ϕ .

- Se o caminho hamiltoniano é normal - ele passa pelos diamantes da ordem superior para o inferior.
- Se o caminho zigue-zagueia pelo diamante, atribuímos à variável correspondente VERDADEIRO
- Se o caminho zague-zigueia pelo diamante, atribuímos à variável correspondente FALSO
- Em razão do fato de que cada cláusula aparece no caminho, observando como o desvio para ele é tomado, determinamos qual literal é VERDADEIRO

Caminho Hamiltoniano

Um caminho hamiltoniano tem que ser normal. Normalidade pode falhar se o caminho entra numa cláusula de um diamante mas retorna em outro.

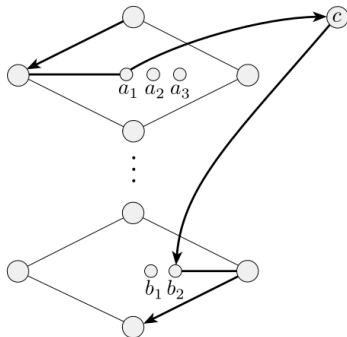


Se isso ocorre, ou a_2 ou a_3 é um separador.

Caminho Hamiltoniano

Se a_2 é um separador

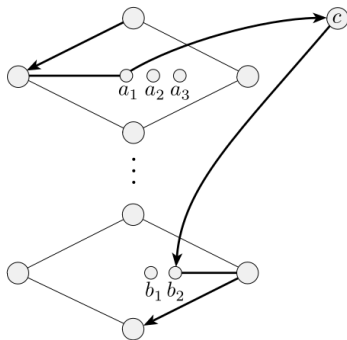
- As únicas arestas entrando em a_2 seriam a_1 e a_3



Caminho Hamiltoniano

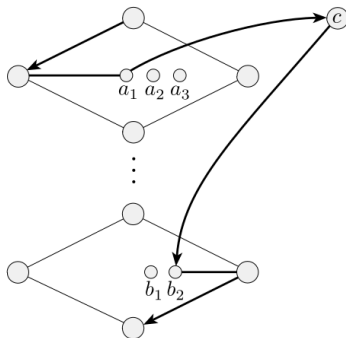
Se a_3 é um separador

- a_1 e a_2 estariam no mesmo par, e portanto
- As únicas arestas entrando em a_2 seriam a_1 e a_3 e c



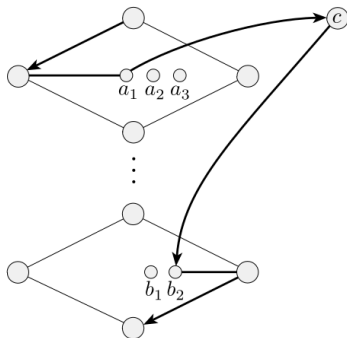
Caminho Hamiltoniano

O caminho não pode entrar em a_2 a partir c ou a_1 porque o caminho vai para outros lugares a partir desses nós.

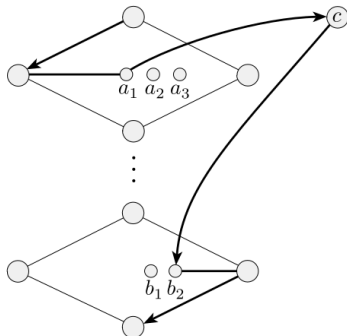


Caminho Hamiltoniano

O caminho não pode entrar em a_2 a partir a_3 , pois o a_3 é o único nó disponível para o qual a_2 aponta, portanto a_2 tem que deixar a_2 via a_3 .



Caminho Hamiltoniano



Logo, o caminho tem que ser normal.

Essa redução opera em tempo polinomial e a prova está completa.



Caminho Hamiltoniano para grafo não-direcionado

Teorema:

$CAMHAMN = \{ \langle G, s, t \rangle \mid G \text{ é um grafo não-direcionado com um caminho hamiltoniano de } s \text{ para } t \} \in NP\text{-Completo}$

Idéia da prova:

Construir uma redução $CAMHAM \leq_p CAMHAMN$ tal que

$$G \in CAMHAM \Leftrightarrow f(G) = G' \in CAMHAMN$$

Todos os nós, exceto s e t no grafo direcionado G , mapeiam para 3 nós no grafo não direcionado G' .

Caminho Hamiltoniano para grafo não-direcionado

Teorema:

$CAMHAMN = \{ \langle G, s, t \rangle \mid G \text{ é um grafo não-direcionado com um caminho hamiltoniano de } s \text{ para } t \} \in NP\text{-Completo}$

Prova: Para todos os nós de $G \in CAMHAM$, substitua por nós u^{entra} , u^{meio} , u^{sai} em G' , para s e t em G substitua por s^{sai} , t^{entra} em G' .

Introduza arestas que conectam u^{entra} com u^{meio} e este com u^{sai} . Acrescente uma conexão entre u^{sai} e v^{entra} se existe uma aresta de u para v .

Caminho Hamiltoniano para grafo não-direcionado

Para mostrar que a construção funciona:

(\Rightarrow) Observamos que um caminho hamiltoniano P em G

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

tem um caminho hamiltoniano P' em G' ,

$$s^{sai}, u_1^{entra}, u_1^{meio}, u_1^{sai}, u_2^{entra}, u_2^{meio}, u_2^{sai}, \dots, t,$$

(\Leftarrow) Qualquer caminho hamiltoniano em G' de s^{sai} para t^{entra} 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.

SOMA-SUBC

Teorema:
$$SOMA - SUBC = \{ \langle S, t \rangle \mid S = \{x_1, \dots, x_m\} \text{ e para algum } \{y_1, \dots, y_n\} \subseteq S, \sum y_i = t \} \in NP\text{-Completo}$$

Idéia da prova: Reduzimos uma instância de 3SAT para uma instância de *SOMA – SUBC* com um conjunto *S* e um valor *t*, para que

- se ϕ tem uma atribuição que a satisfaz,
- então *S* tem um subconjunto *T* que soma um valor *t*.

Nós já sabemos que *SOMA – SUBC* está em NP.

SOMA-SUBC

Teorema:
$$SOMA - SUBC = \{ \langle S, t \rangle \mid S = \{x_1, \dots, x_m\} \text{ e para algum } \{y_1, \dots, y_n\} \subseteq S, \sum y_i = t \} \in NP\text{-Completo}$$
Prova:

- Seja ϕ uma fórmula booleana com l variáveis x_1, \dots, x_l , e k cláusulas c_1, \dots, c_k
- A redução converte ϕ para uma instância de *SOMA – SUBC* $\langle S, t \rangle$, na qual os elementos de S e o número de linhas t definem uma tabela (mostraremos posteriormente), expressos na notação decimal ordinária.

SOMA-SUBC

A redução $3SAT \leq_p SOMA - SUBC$, constrói uma tabela da seguinte maneira:

- Como ϕ tem l variáveis x_i , $1 \leq i \leq l$, e k cláusulas c_j , $1 \leq j \leq k$
- Para cada variável x_i , construa números y_i , z_i de $l + k$ dígitos.
 - O i -ésimo dígito de y_i e z_i é igual a 1
 - Para a parte direita contém um dígito para cada cláusula:
 - O j -ésimo dígito de y_i é 1 se a cláusula c_j contém o literal x_i .
 - O j -ésimo dígito de z_i é 1 se a cláusula c_j contém o literal \bar{x}_i .
- Os dígitos não especificados como sendo 1 são 0.

SOMA-SUBC

- Adicionalmente, S contém um par de números, g_j, h_j , para cada cláusula c_j , $1 \leq j \leq k$.
 - Esses dois números são iguais e consistem de 1, os demais $k - j$ dígitos valem 0.
- Finalmente, o número alvo t , na linha inferior da tabela, consiste de l números 1s, onde l é o número de variáveis, seguidos por k números 3s, k é o número de cláusulas.

SOMA-SUBC

Para

$$\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_2 \vee x_3 \vee \dots) \wedge \dots \wedge (\overline{x_3} \vee \dots \vee \dots)$$

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

SOMA-SUBC

- Temos que mostrar a redução funciona demonstrando que ϕ é satisfatível sse existe um $T \subseteq S$ que soma t .

(\Rightarrow)

- Selecionar y_i se x_i é VERDADEIRO
- Selecionar z_i se x_i é FALSO
- Selecionar uma quantidade suficiente de número g e h para trazer cada um dos k últimos dígitos para 3, portanto é atingido o alvo.

SOMA-SUBC

Exemplo: Todas as variáveis são VERDADEIRAS.

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)$$

	1	2	3	1	2	3	4
y1	1	0	0	1	0	0	1
z1	1	0	0	0	1	1	0
y2	0	1	0	1	0	1	0
z2	0	1	0	0	1	0	1
y3	0	0	1	1	1	0	1
z3	0	0	1	0	0	1	0
g1	0	0	0	1	0	0	0
h1	0	0	0	1	0	0	0
g2	0	0	0	0	1	0	0
h2	0	0	0	0	1	0	0
g3	0	0	0	0	0	1	0
h3	0	0	0	0	0	1	0
g4	0	0	0	0	0	0	1
h4	0	0	0	0	0	0	1
t	1	1	1	3	3	3	3

Primeiro montamos a tabela

SOMA-SUBC

Exemplo: Todas as variáveis são VERDADEIRAS.

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)$$

	1	2	3	1	2	3	4
*y1	1	0	0	1	0	0	1
z1	1	0	0	0	1	1	0
*y2	0	1	0	1	0	1	0
z2	0	1	0	0	1	0	1
*y3	0	0	1	1	1	0	1
z3	0	0	1	0	0	1	0
g1	0	0	0	1	0	0	0
h1	0	0	0	1	0	0	0
g2	0	0	0	0	1	0	0
h2	0	0	0	0	1	0	0
g3	0	0	0	0	0	1	0
h3	0	0	0	0	0	1	0
g4	0	0	0	0	0	0	1
h4	0	0	0	0	0	0	1
t	1	1	1	3	3	3	3

Depois, selecionamos somente as variáveis

SOMA-SUBC

Exemplo: Todas as variáveis são VERDADEIRAS.

$$\phi = (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee x_3)$$

	1	2	3	1	2	3	4
*y1	1	0	0	1	0	0	1
z1	1	0	0	0	1	1	0
*y2	0	1	0	1	0	1	0
z2	0	1	0	0	1	0	1
*y3	0	0	1	1	1	0	1
z3	0	0	1	0	0	1	0
g1	0	0	0	1	0	0	0
h1	0	0	0	1	0	0	0
*g2	0	0	0	0	1	0	0
*h2	0	0	0	0	1	0	0
*g3	0	0	0	0	0	1	0
*h3	0	0	0	0	0	1	0
*g4	0	0	0	0	0	0	1
h4	0	0	0	0	0	0	1
t	1	1	1	3	3	3	3

Para completar os últimos dígitos para 3 selecione as linhas g e h.

SOMA-SUBC

 (\Leftarrow)

- Mostrar que se existe um subconjunto S então ϕ é satisfatível
 - Atribua VERDADEIRO a x_i se y_i está no subconjunto
 - Atribua FALSO a x_i se z_i está no subconjunto
 - Exatamente um número por variável deve estar no subconjunto.
 - Caso contrário, os primeiros dígitos da soma é maior do que 1
 - Pelo menos uma variável que corresponde ao literal na cláusula deve estar dentro do conjunto.
 - Caso contrário, os k últimos dígitos da tabela será menor do que 3.
- Cada cláusula está satisfeita. \square

Exercícios

- 1) Seja $\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$ construa o grafo G como descrito na redução de 3SAT para o Caminho Hamiltoniano.
- 2) O que você acha do problema 2 – SAT (Ele é NP-Completo?)