Teoria da Computação
Complexidade de Tempo
parte 3

Leonardo Takuno
{leonardo.takuno@gmail.com}

Centro Universitário Senac

Sumário

Sumário

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-Completos.

Teorema de Cook-Levin

Nós vimos anteriormente que os problemas NP-Completos 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 $\neg x = \overline{x}$).

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

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)

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

É satisfazível pois a atribuição

$$x = \mathit{false},$$
$$y = \mathit{true},$$
$$z = \mathit{false}.$$

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.

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

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

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

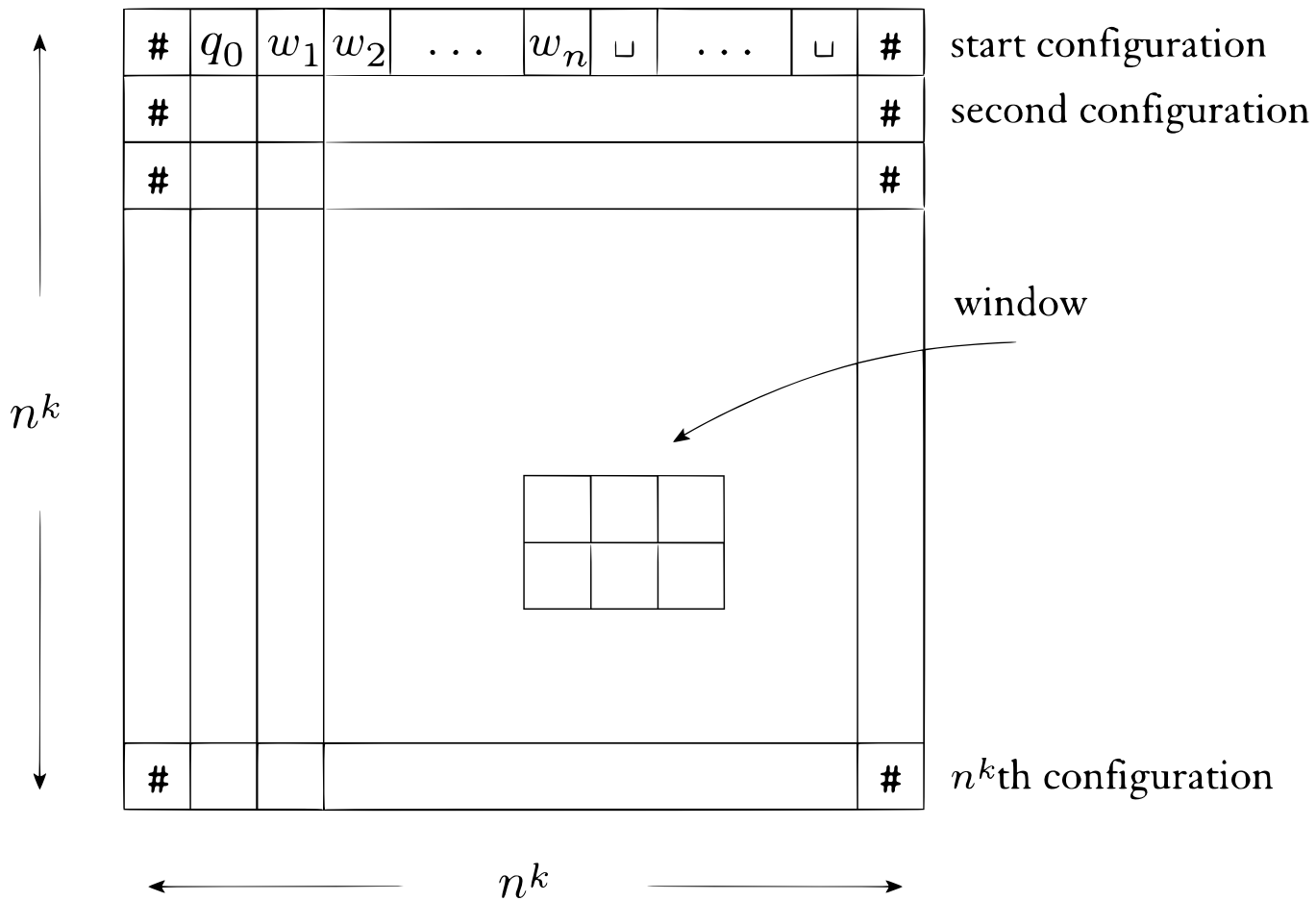
onde f converte a string w na fórmula booleana $f(w)$.

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 tableaus. 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:

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

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.

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

  - 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.
  - $\phi_{movimento}$: 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

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

- 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 ϕ_{celula} garante esse requisito:

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

  - 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 ϕ_{inicio} garante que a primeira linha da tabela é a configuração inicial:

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

- 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

Afirmação: 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

Afirmação: 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

$\cdots$ **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 $a_1, ..., a_6$

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

- 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

3SAT

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

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

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

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

$3SAT$

Teorema:

$3SAT \in$ 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 construimos 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 $c_i = a_1 \vee a_2 \vee \cdots \vee a_k$, 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 $3cfn$
    $(a_1 \vee z_1 \vee z_2) \wedge (a_1 \vee \overline{z_1} \vee z_2) \wedge (a_1 \vee z_1 \vee \overline{z_2}) \wedge (a_1 \vee \overline{z_1} \vee \overline{z_2})$
  - $k = 2$: Aqui $c_i = (a_1 \vee a_2)$. Use variáveis adicionais z_1 e z_2 para construir $3cfn$
    $(a_1 \vee a_2 \vee z_1) \wedge (a_1 \vee a_2 \vee \overline{z_1})$
  - $k = 3$: Aqui $c_i = (a_1 \vee a_2 \vee a_3)$. Já está em $3cfn$, não há nada a fazer.

3SAT

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

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

3SAT

- Mostramos que f é uma redução de tempo polinomial.
- Primeiro observe que o número máximo de variáveis que podem ocorrer em uma clásula ϕ é n.
- Também observe que há m clásulas.
- Portanto, o número máximo de conversões é limitado a $O(mn)$ que é claramente polinomial.
- Concluímos 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 $1 \leq j \leq j-2$ 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 7.43:
CLIQUE $\in$ NP-Completo

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

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

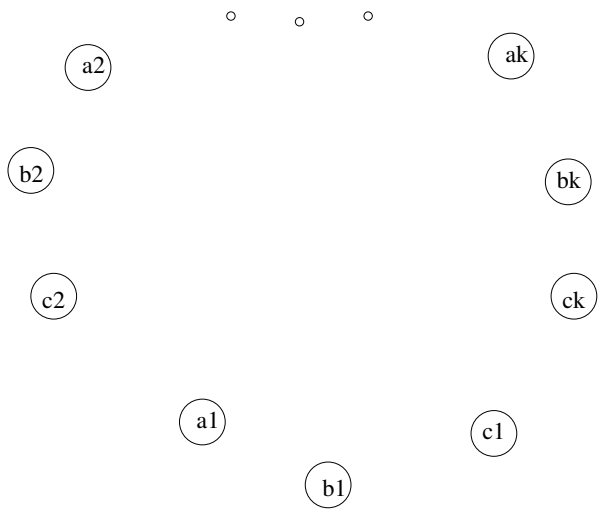
onde ϕ_k é uma $3cnf$-fórmula com k cláusulas e $f(\phi_k) = \langle G, k \rangle$.

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

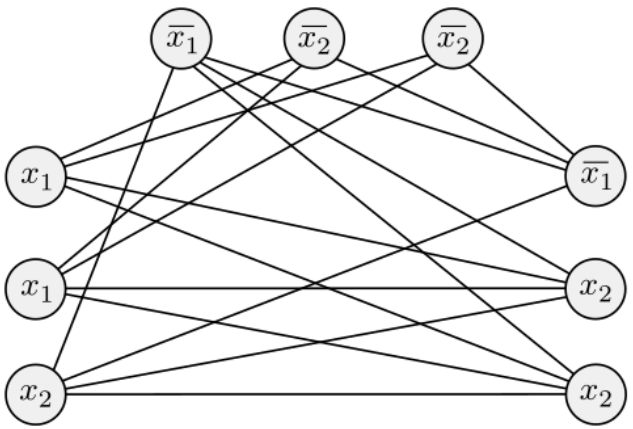
Construímos arestas conectando todos os nós exceto para

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

Exemplo de construção para

$$\phi_3 = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$

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, assuma 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. $\square$

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.

Teorema 7.43:
$COB - VERT = \{\langle G, k\rangle | G$ é um grafo não-direcionado que tem uma cobertura de vértices de k-nós $\}$

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

Teorema 7.43:
$COB - VERT = \{\langle G, k\rangle | G$ é um grafo não-direcionado que tem uma cobertura de vértices de k-nós $\}$

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

Idéia da prova:

- $COB - VERT$ pertence a NP
  - Certificado: uma cobertura de tamanho k.

Teorema 7.43:
$COB - VERT = \{\langle G, k\rangle | G$ é um grafo não-direcionado que tem uma cobertura de vértices de k-nós $\}$

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

Idéia da prova:

- $3SAT \leq_m COB - VERT$
  - Um grafo que simule 3fnc-fórmula ϕ
  - ϕ é satisfatível sse o grafo tiver uma cobertura de tamanho k

Teorema 7.43:
$COB - VERT = \{\langle G, k\rangle | G$ é um grafo não-direcionado que tem uma cobertura de vértices de k-nós $\}$

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

Idéia da prova:

- $3SAT \leq_m COB - VERT$, mapeamento:
  - ϕ: variáveis que assumem valor verdadeiro ou falso.
    - G: dois nós de cada aresta (um deles ao menos tem que aparecer na cobertura - verdadeiro ou falso).
  - ϕ: cada cláusula com 3 literais, pelo menos um verdadeiro
    - G: grupos de 3 nós conectados por arestas, pelo menos 2 nós inclusos na cobertura.

Teorema 7.43:
$COB - VERT = \{\langle G, k\rangle | G$ é um grafo não-direcionado que tem uma cobertura de vértices de k-nós $\}$

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

Para cada variável x em ϕ, produzimos uma aresta conectando dois nós. Rotulamos os dois nós por 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.

Teorema 7.43:
$COB - VERT = \{\langle G, k\rangle | G$ é um grafo não-direcionado que tem uma cobertura de vértices de k-nós $\}$

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

Cada cláusula corresponde a uma tripla de nós conectados e que são rotulados com os três literais da cláusula.