Melhores momentos

AULA PASSADA

Complex. computacional: P versus NP

CLR 36 ou CLRS 34

Problemas polinomiais

Análise de um algoritmo em um determinado modelo de computação estima o seu consumo de tempo e quantidade de espaço como uma função do tamanho da instância do problema.

Exemplo: o consumo de tempo do algoritmo EUCLIDES (a, b) é expresso como uma função de $\langle a \rangle + \langle b \rangle$.

Um problema é solúvel em tempo polinomial se existe um algoritmo que consome tempo $\mathrm{O}(\langle I \rangle^c)$ para resolver o problema, onde c é uma constante e I é instância do problema.

Classe P

Por um algoritmo eficiente entendemos um algoritmo polinomial.

A classe de todos os problemas de decisão que podem ser resolvidos por algoritmos polinomiais é denotada por P (classe de complexidade).

Exemplo: As versões de decisão dos problemas:

> máximo divisor comum, subseqüência comum máxima e mochila fracionária

estão em P.

Para muitos problemas, não se conhece algoritmo essencialmente melhor que "testar todas as possibilidades". Em geral, isso não está em P.

Verificador polinomial para SIM

Um verificador polinomial para a resposta SIM a um problema Π é uma algoritmo polinomial ALG que recebe

uma instância I de Π e um objeto C, tal que $\langle C \rangle$ é $\mathrm{O}(\langle I \rangle^c)$ para alguma constante c e

$$\mathsf{ALG}(I, C) = \mathsf{SIM} \Leftrightarrow \Pi(I) = \mathsf{SIM}$$

A constante c depende apenas do problema!

Classe NP

Formada pelos problemas de decisão que possuem um verificador polinomial para a resposta SIM.

Em outras palavras, um problemas de decisão Π está em NP se existe um problema Π' em P e uma função polinomial $p(n)$ tais que, para cada instância I de Π, existe um objeto C, $\langle C \rangle = \mathrm{O}(p(\langle I \rangle))$, e

$\Pi(I) =$ SIM se e somente se $\Pi'(I, C) =$ SIM.

O objeto C é dito um certificado polinomial ou certificado curto da resposta SIM de $\Pi(I)$.

Exemplos

Problemas de decisão com certificado polinomial para SIM:

- existe subseqüência crescente $\geq k$?
- existe subcoleção disjunta $\geq k$ de intervalos?
- existe mochila booleana de valor $\geq k$?
- existe mochila de valor $\geq k$?
- existe subseqüência comum $\geq k$?
- grafo tem ciclo de comprimento $\geq k$?
- grafo tem ciclo hamiltoniano?
- grafo tem emparelhamento (casamento) perfeito?

Todos esses problemas estão em NP.

Verificador polinomial para NÃO

Um verificador polinomial para a resposta NÃO a um problema Π é uma algoritmo polinomial ALG que recebe

uma instância I de Π e um objeto C, tal que $\langle C \rangle$ é $\mathrm{O}(\langle I \rangle^c)$ para alguma constante c e

$$\mathsf{ALG}(I, C) = \mathsf{SIM} \Leftrightarrow \Pi(I) = \mathsf{NÃO}$$

A constante c depende apenas do problema!

Classe co-NP

A classe co-NP é definida trocando-se SIM por NÃO na definição de NP.

Um problema de decisão Π está em co-NP se admite um certificado polinomial para a resposta NÃO.

Os problemas em NP $\cap$ co-NP admitem certificados polinomiais para as respostas SIM e NÃO.

Em particular, P $\subseteq$ NP $\cap$ co-NP.

Exemplos

Problemas de decisão com certificado polinomial para NÃO:

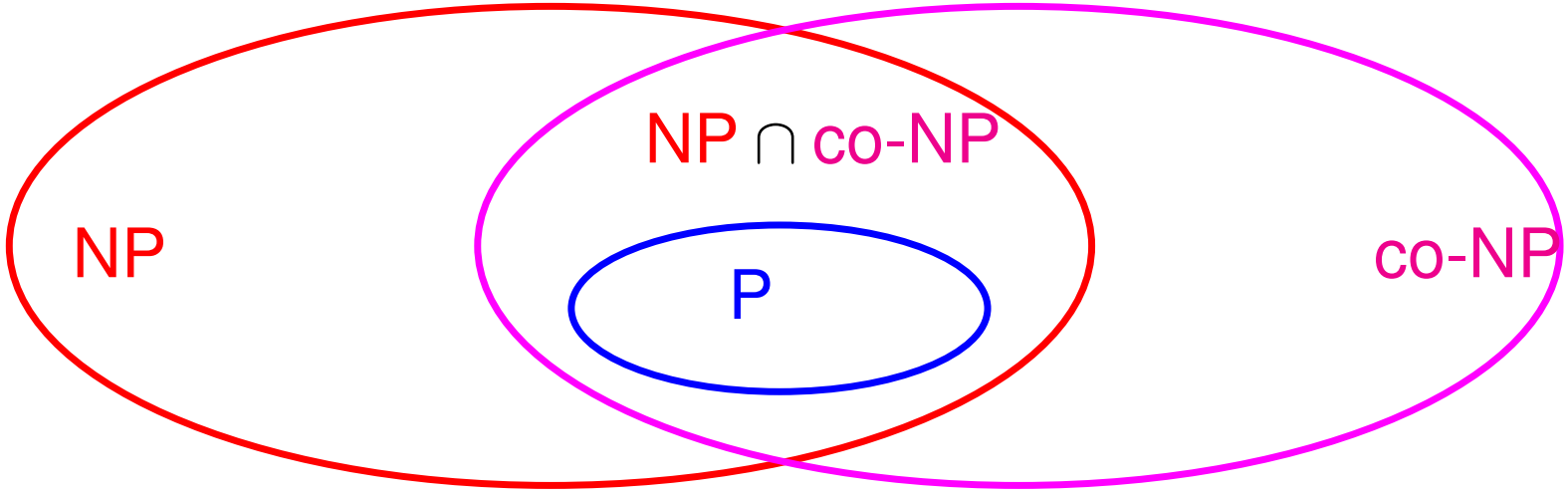
- existe subseqüência crescente $\geq k$?
- um dado número é primo?
- existe subseqüência comum $\geq k$?
- grafo tem emparelhamento (casamento) perfeito?

Todos esses problemas estão em co-NP.

P, NP e co-NP

Classe P	formada por problemas de decisão que podem ser resolvidos em tempo polinomial
Classe NP	formada por problemas de decisão que possuem um verificador polinomial para a resposta SIM
Classe co-NP	formada por problemas de decisão que possuem um verificador polinomial para a resposta NÃO

P, NP e co-NP



$P \neq NP$?

$NP \cap \text{co-NP} \neq P$?

$NP \neq \text{co-NP}$?

AULA 25

Mais complexidade computacional

CLR 36 ou CLRS 34

Redução polinomial

Permite comparar o "grau de complexidade" de problemas diferentes.

Uma redução de um problema Π a um problema Π' é um algoritmo ALG que resolve Π usando uma subrotina hipotética ALG$'$ que resolve Π', de tal forma que, se ALG$'$ é um algoritmo polinomial, então ALG é um algoritmo polinomial.

$\Pi \leq_P \Pi' =$ existe uma redução de Π a Π'.

Se $\Pi \leq_P \Pi'$ e Π' está em P, então Π está em P.

Exemplo

Π = encontrar um ciclo hamiltoniano
Π' = existe um ciclo hamiltoniano?

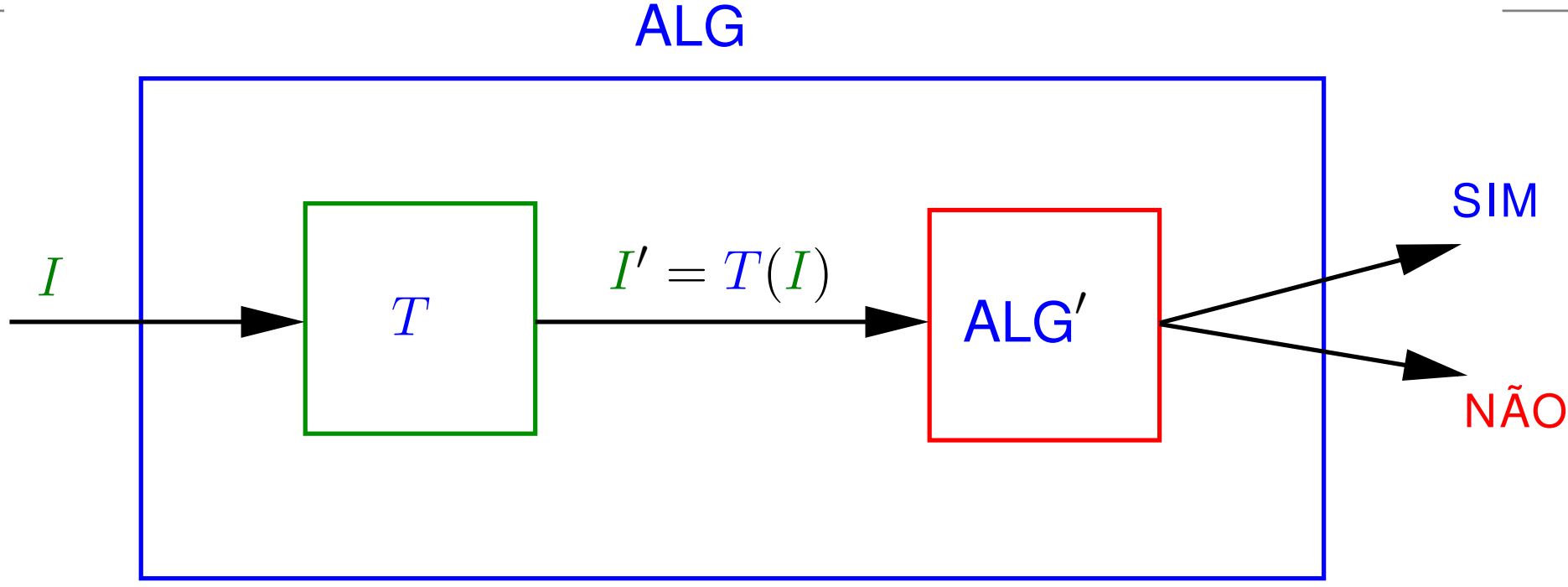
Redução de Π a Π': ALG$'$ é um algoritmo que resolve Π'

```
ALG (G)

1   se ALG'(G) = NÃO
2       então devolva "G não é hamiltoniano"
3   para cada aresta uv de G faça
4       H ← G − uv
5       se ALG'(H) = SIM
6           então G ← G − uv
7   devolva G
```

Se ALG$'$ consome tempo $\mathrm{O}(p(n))$, então ALG consome tempo $\mathrm{O}(m\, p(\langle G \rangle))$, onde m = número de arestas de G.

Esquema comum de reduções



Faz apenas uma chamada ao algoritmo ALG′.

T transforma uma instância I de Π em uma instância $I' = T(I)$ de Π' tal que

$$\Pi(I) = \text{SIM} \text{ se e somente se } \Pi'(I') = \text{SIM}$$

T é uma espécie de “filtro” ou “compilador”.

Satisfatibilidade

Problema: Dada um fórmula booleana ϕ nas variáveis $x_1, \ldots, x_n$, existe uma atribuição

$$t : \{x_1, \ldots, x_n\} \rightarrow \{\text{VERDADE}, \text{FALSO}\}$$

que torna ϕ verdadeira?

Exemplo:

$$\phi = (x_1) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3)$$

Se $t(x_1) = \text{VERDADE}, t(x_2) = \text{FALSO}, t(x_3) = \text{FALSO}$,
então $t(\phi) = \text{VERDADE}$

Se $t(x_1) = \text{VERDADE}, t(x_2) = \text{VERDADE}, t(x_3) = \text{FALSO}$,
então $t(\phi) = \text{FALSO}$

Sistemas lineares 0-1

Problema: Dadas uma matriz A e um vetor b,

$$Ax \geq b$$

possui uma solução tal que $x_i = 0$ ou $x_i = 1$ para todo i?

Exemplo:

$$\begin{array}{rcrcrcr} & x_1 & & & & \geq & 1 \\ - & x_1 & - & x_2 & + & x_3 & \geq & -1 \\ & & & & - & x_3 & \geq & 0 \end{array}$$

tem uma solução 0-1?

Sim! $x_1 = 1, x_2 = 0$ e $x_3 = 0$ é solução.

Exemplo 1

Satisfatibilidade $\leq_P$ Sistemas lineares 0-1

A transformação T recebe uma fórmula booleana ϕ
e devolve um sistema linear $Ax \geq b$
tal que ϕ é satisfatível se e somente se o sistema $Ax \geq b$ admite uma solução 0-1.

Exemplo:

$$\phi = (x_1) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3)$$

$$\begin{array}{rcrcrcl} x_1 & & & & & \geq & 1 \\ 1 - x_1 & + & 1 - x_2 & + & x_3 & \geq & 1 \\ & & & & 1 - x_3 & \geq & 1 \end{array}$$

Exemplo 2

Verifique que

Ciclo hamiltoniano $\leq_P$ Caminho hamiltoniano entre u e v

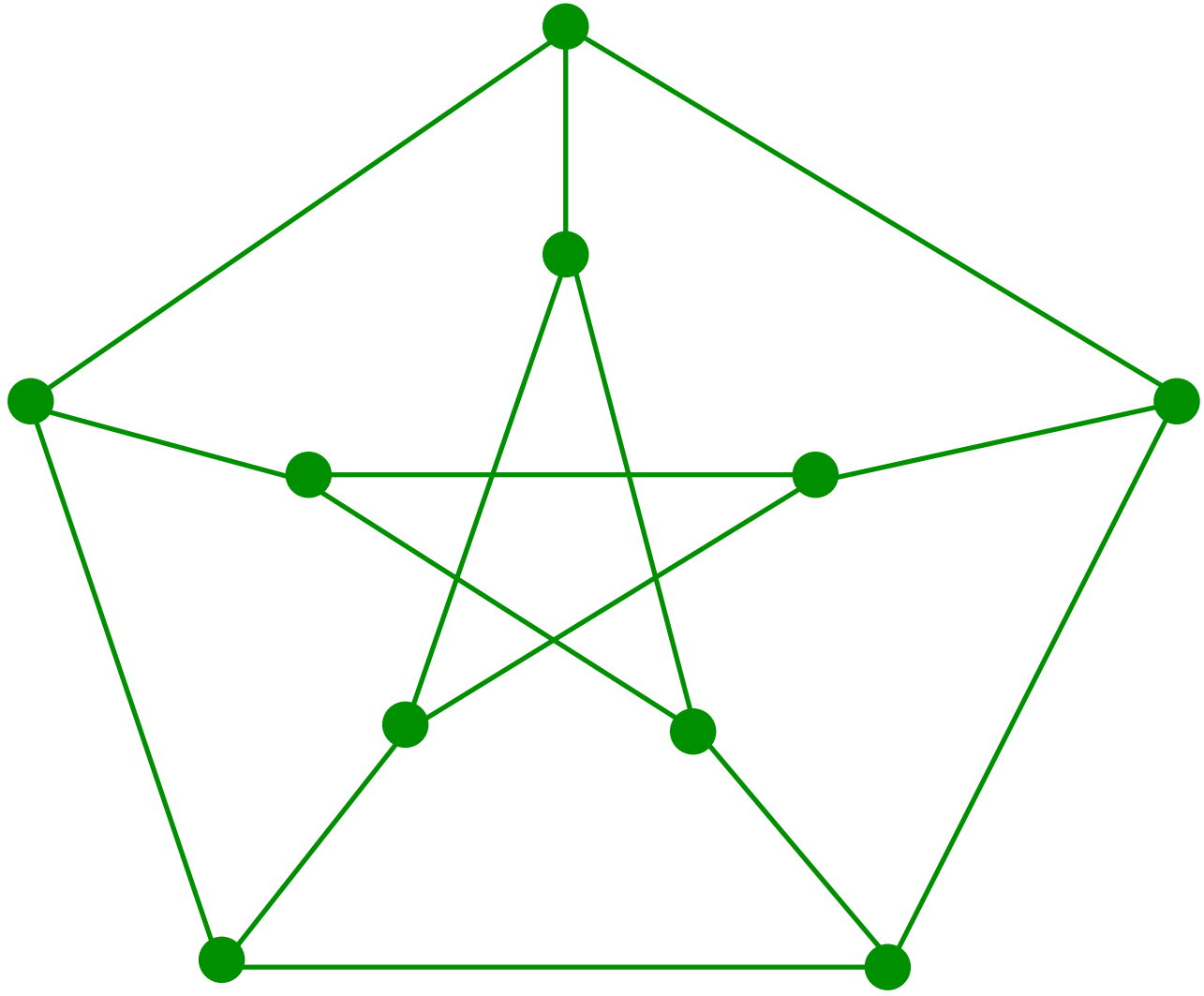
Verifique que

Caminho hamiltoniano entre u e v $\leq_P$ Caminho hamiltoniano

Caminhos hamiltonianos

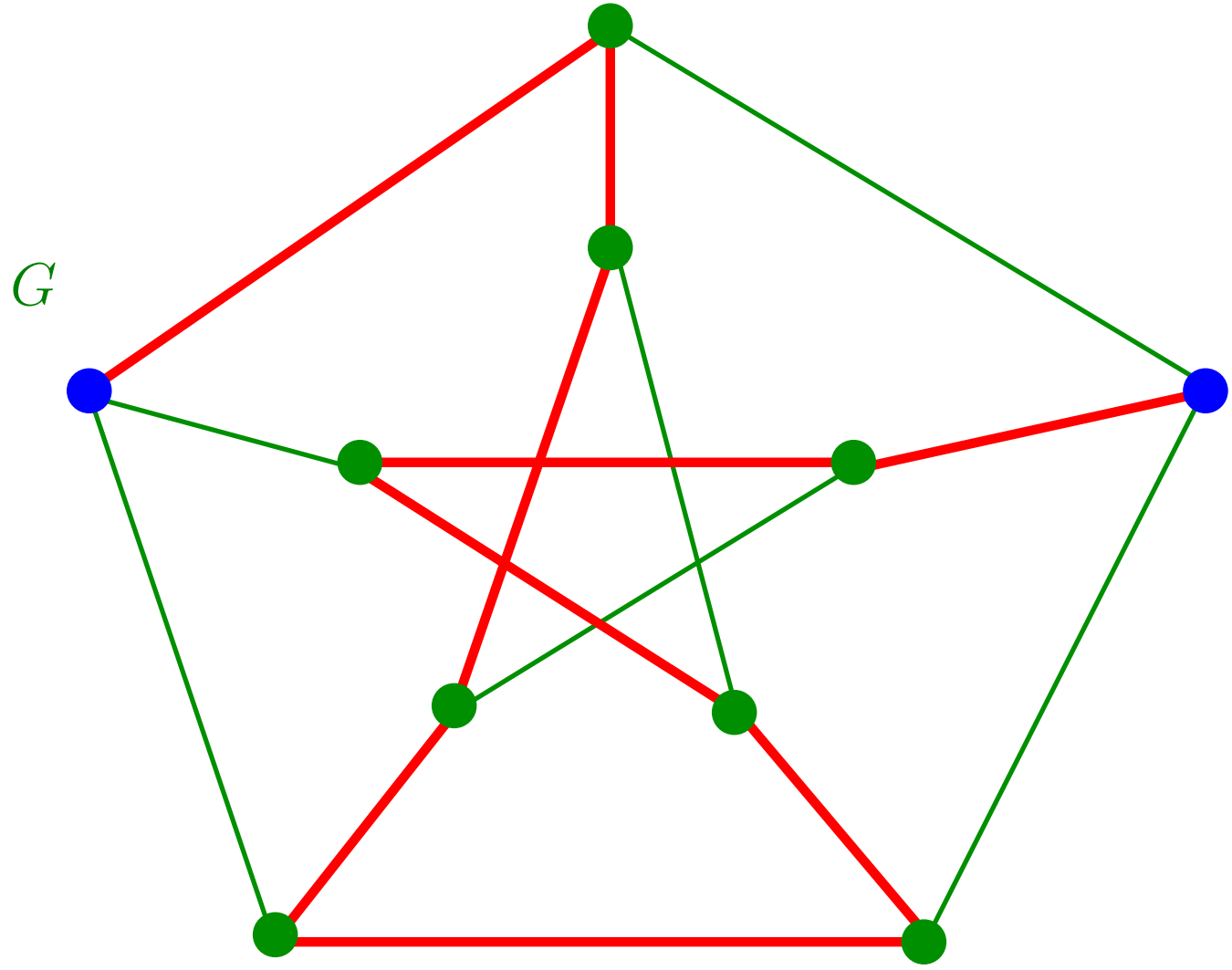
Problema: um dado grafo possui um caminho hamiltoniano?

G



Caminhos hamiltonianos

Problema: um dado grafo possui um caminho hamiltoniano?



Satisfatibilidade

Problema: Dada um fórmula booleana ϕ nas variáveis $x_1, \ldots, x_n$, existe uma atribuição

$$t : \{x_1, \ldots, x_n\} \rightarrow \{\text{VERDADE}, \text{FALSO}\}$$

que torna ϕ verdadeira?

Exemplo:

$$\phi = (x_1) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3)$$

Se $t(x_1) = \text{VERDADE}, t(x_2) = \text{FALSO}, t(x_3) = \text{FALSO}$,
então $t(\phi) = \text{VERDADE}$

Se $t(x_1) = \text{VERDADE}, t(x_2) = \text{VERDADE}, t(x_3) = \text{FALSO}$,
então $t(\phi) = \text{FALSO}$

Exemplo 3

Caminho hamiltoniano $\leq_P$ Satisfatibilidade

Descreveremos um algoritmo polinomial T que recebe um grafo G e devolve uma fórmula booleana $\phi(G)$ tais que

G tem caminho hamiltoniano $\Leftrightarrow$ $\phi(G)$ é satisfatível.

Suponha que G tem vértices $1, \ldots, n$.

$\phi(G)$ tem n^2 variáveis $x_{i,j}$, $1 \leq i, j \leq n$.

Interpretação: $x_{i,j} =$ VERDADE $\Leftrightarrow$ vértice j é o i-ésimo vértice do caminho.

Exemplo 3 (cont.)

Claúsulas de $\phi(G)$:

- "vértice j faz parte do caminho:

$$(x_{1,j} \vee x_{2,j} \vee \cdots \vee x_{n,j})$$

  para cada j (n claúsulas).
- "vértice j não está em duas posições do caminho:

$$(\neg x_{i,j} \vee \neg x_{k,j})$$

  para cada j e $i \neq k$ ($\mathrm{O}(n^3)$ claúsulas).
- "algum vértice é o i-ésimo do caminho":

$$(x_{i,1} \vee x_{i,2} \vee \cdots \vee x_{i,n})$$

  para cada i (n claúsulas).

Exemplo 3 (cont.)

Mais claúsulas de $\phi(G)$:

- "dois vértices não podem ser o i-ésimo":

$$(\neg x_{i,j} \vee \neg x_{i,k})$$

para cada i e $j \neq k$ ($\mathrm{O}(n^3)$ claúsulas).

- "se ij não é aresta, j não pode seguir i no caminho":

$$(\neg x_{k,i} \vee \neg x_{k+1,j})$$

para cada ij que não é aresta ($\mathrm{O}(n^3)$ claúsulas).

A fórmula $\phi(G)$ tem $\mathrm{O}(n^3)$ claúsulas e cada claúsula tem $\leq n$ literais. Logo, $\langle \phi(G) \rangle$ é $\mathrm{O}(n^4)$.

Não é difícil construir o algoritmo polinomial T.

Exemplo 3 (cont.)

$\phi(G)$ satisfatível $\Rightarrow$ G tem caminho hamiltoniano.

Prova: Seja $t : \{\text{variáveis}\} \rightarrow \{\text{VERDADE}, \text{FALSO}\}$ tal que $t(\phi(G)) = \text{VERDADE}$.

Para cada i existe um único j tal que $t(x_{i,j}) = \text{VERDADE}$.
Para cada j existe um único i tal que $t(x_{i,j}) = \text{VERDADE}$.
Logo, t é a codificadação de uma permutação

$$\pi(1), \pi(2), \ldots, \pi(n)$$

dos vértices de G, onde

$$\pi(i) = j \Leftrightarrow t(x_{i,j}) = \text{VERDADE}.$$

Para cada k, $(\pi(k), \pi(k+1))$ é uma aresta de G.

Logo, $(\pi(1), \pi(2), \ldots, \pi(n))$ é um caminho hamiltoniano.

Exemplo 3 (cont.)

G tem caminho hamiltoniano $\Rightarrow$ $\phi(G)$ satisfatível.

Suponha que $(\pi(1), \pi(2), \ldots, \pi(n))$ é um caminho hamiltoniano, onde π é uma pernutação dos vértices de G. Então

$t(x_{i,j}) =$ VERDADE se $\pi(i) = j$ e

$t(x_{i,j}) =$ FALSO se $\pi(i) \neq j$,

é uma atribuição de valores que satisfaz todas as claúsulas de $\phi(G)$.