

# Complexidade computacional

Classifica os problemas em relação à dificuldade de resolvê-los algoritmicamente.

CLRS 34

# Classes P e NP

Por **algoritmo eficiente** entende-se um **algoritmo polinomial**.

# Classes P e NP

Por **algoritmo eficiente** entende-se um **algoritmo polinomial**.

**Classe P:**

classe de todos os **problemas de decisão** que podem ser resolvidos por **algoritmos polinomiais**.

# Classes P e NP

Por **algoritmo eficiente** entende-se um **algoritmo polinomial**.

## Classe P:

classe de todos os **problemas de decisão** que podem ser resolvidos por **algoritmos polinomiais**.

## Classe NP:

classe de todos os **problemas de decisão** que possuem um **verificador polinomial para a resposta SIM**

# Verificador polinomial para sim

Um **verificador polinomial para a resposta SIM** a um problema  $\Pi$  é um algoritmo polinomial **ALG** que **recebe** *uma instância  $I$  de  $\Pi$  e um objeto  $C$ , tal que  $\langle C \rangle$  é  $O(\langle I \rangle^\alpha)$  para alguma constante  $\alpha$*

# Verificador polinomial para sim

Um **verificador polinomial para a resposta SIM** a um problema  $\Pi$  é um algoritmo polinomial **ALG** que **recebe**

*uma instância  $I$  de  $\Pi$  e um objeto  $C$ ,  
tal que  $\langle C \rangle$  é  $O(\langle I \rangle^\alpha)$  para alguma constante  $\alpha$*

e **devolve**

**SIM** para algum  $C$  se a resposta a  $\Pi(I)$  é **SIM**;  
**NÃO** para todo  $C$  se a resposta a  $\Pi(I)$  é **NÃO**.

# Verificador polinomial para sim

Um **verificador polinomial para a resposta SIM** a um problema  $\Pi$  é um algoritmo polinomial **ALG** que **recebe**

*uma instância  $I$  de  $\Pi$  e um objeto  $C$ ,  
tal que  $\langle C \rangle$  é  $O(\langle I \rangle^\alpha)$  para alguma constante  $\alpha$*

e **devolve**

**SIM** para algum  $C$  se a resposta a  $\Pi(I)$  é **SIM**;  
**NÃO** para todo  $C$  se a resposta a  $\Pi(I)$  é **NÃO**.

No caso de resposta **SIM**, o objeto  $C$  é dito um **certificado polinomial** ou **certificado curto** da resposta **SIM** a  $\Pi(I)$ .

# P ≠ NP?

É crença de muitos que a classe NP é maior que a classe P, ainda que isso não tenha sido provado até agora.

Este é o intrigante problema matemático conhecido pelo rótulo  
“P ≠ NP?”

Não confunda NP com “não-polinomial”.



# Classe co-NP

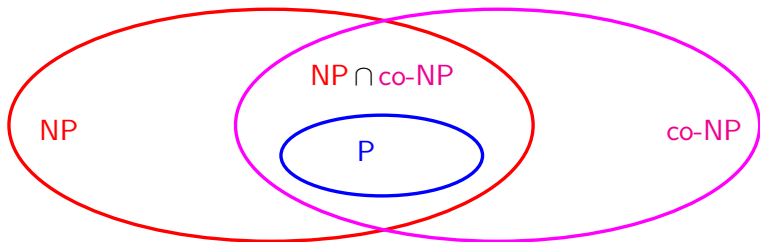
A classe co-NP é definida trocando-se SIM por NÃO na definição de NP.

Um problema de decisão  $\Pi$  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,  $P \subseteq NP \cap co-NP$ .

# P, NP e co-NP



$P \neq NP?$

$NP \cap co-NP \neq P?$

$NP \neq co-NP?$

# Classes P, NP e co-NP

## Classe P:

classe de todos os problemas de decisão que podem ser resolvidos por algoritmos polinomiais.

## Classe NP:

classe de todos os problemas de decisão que possuem um verificador polinomial para a resposta SIM

## Classe co-NP:

classe de todos os problemas de decisão que possuem um verificador polinomial para a resposta NÃO

# Redução polinomial

Permite comparar

o “**grau de complexidade**” de problemas diferentes.

# Redução polinomial

Permite comparar  
o “**grau de complexidade**” de problemas diferentes.

$\Pi$ ,  $\Pi'$ : problemas

Uma **redução** de  $\Pi$  a  $\Pi'$  é um algoritmo **ALG** que resolve  $\Pi$  usando uma subrotina hipotética **ALG'** que resolve  $\Pi'$ ,

# Redução polinomial

Permite comparar  
o “**grau de complexidade**” de problemas diferentes.

$\Pi$ ,  $\Pi'$ : problemas

Uma **redução** de  $\Pi$  a  $\Pi'$  é um algoritmo **ALG** que resolve  $\Pi$  usando uma subrotina hipotética **ALG'** que resolve  $\Pi'$ , de forma que, se **ALG'** é um algoritmo polinomial, então **ALG** é um algoritmo polinomial.

# Redução polinomial

Permite comparar  
o “**grau de complexidade**” de problemas diferentes.

$\Pi$ ,  $\Pi'$ : problemas

Uma **redução** de  $\Pi$  a  $\Pi'$  é um algoritmo **ALG** que resolve  $\Pi$  usando uma subrotina hipotética **ALG'** que resolve  $\Pi'$ , de forma que, se **ALG'** é um algoritmo polinomial, então **ALG** é um algoritmo polinomial.

$\Pi \prec_P \Pi' =$  existe uma redução de  $\Pi$  a  $\Pi'$ .

Se  $\Pi \prec_P \Pi'$  e  $\Pi'$  está em **P**, então  $\Pi$  está em **P**.

## Exemplo

$\Pi$  = encontrar um ciclo hamiltoniano

$\Pi'$  = existe um ciclo hamiltoniano?



## Exemplo

$\Pi$  = encontrar um ciclo hamiltoniano

$\Pi'$  = existe um ciclo hamiltoniano?

Redução de  $\Pi$  a  $\Pi'$ :

Suponha que  $ALG'$  é um algoritmo que resolve  $\Pi'$ .

## Exemplo

$\Pi$  = encontrar um ciclo hamiltoniano

$\Pi'$  = existe um ciclo hamiltoniano?

Redução de  $\Pi$  a  $\Pi'$ :

Suponha que  $ALG'$  é um algoritmo que resolve  $\Pi'$ .

Como você poderia resolver  $\Pi$  usando  $ALG'$ ?

## Exemplo

$\Pi$  = encontrar um ciclo hamiltoniano

$\Pi'$  = existe um ciclo hamiltoniano?

Redução de  $\Pi$  a  $\Pi'$ :

Suponha que  $ALG'$  é um algoritmo que resolve  $\Pi'$ .

Como você poderia resolver  $\Pi$  usando  $ALG'$ ?

Ideias?

## Exemplo

$\Pi$  = encontrar um ciclo hamiltoniano

$\Pi'$  = existe um ciclo hamiltoniano?

Redução de  $\Pi$  a  $\Pi'$ :

Suponha que  $ALG'$  é um algoritmo que resolve  $\Pi'$ .

Como você poderia resolver  $\Pi$  usando  $ALG'$ ?

Ideias?

Se posso decidir se um grafo tem ou não um ciclo hamiltoniano, o que posso fazer para, dado um grafo  $G$ , encontrar um ciclo hamiltoniano em  $G$ ?

## Exemplo

$\Pi$  = encontrar um ciclo hamiltoniano

$\Pi'$  = existe um ciclo hamiltoniano?

Redução de  $\Pi$  a  $\Pi'$ :

Suponha que  $ALG'$  é um algoritmo que resolve  $\Pi'$ .

Como você poderia resolver  $\Pi$  usando  $ALG'$ ?

Ideias?

Se posso decidir se um grafo tem ou não um ciclo hamiltoniano, o que posso fazer para, dado um grafo  $G$ , encontrar um ciclo hamiltoniano em  $G$ ?

Como posso descobrir as arestas de algum ciclo hamiltoniano de  $G$ ?

## Exemplo

$\Pi$  = encontrar um ciclo hamiltoniano

$\Pi'$  = existe um ciclo hamiltoniano?

Redução de  $\Pi$  a  $\Pi'$ :  $ALG'$  é um algoritmo que resolve  $\Pi'$ .

$ALG(G)$

- 1 se  $ALG'(G) = \text{NÃO}$
- 2     então devolva “ $G$  não é hamiltoniano”
- 3 para cada aresta  $uv$  de  $G$  faça
- 4      $H \leftarrow G - uv$
- 5     se  $ALG'(H) = \text{SIM}$
- 6         então  $G \leftarrow G - uv$
- 7 devolva  $G$

## Exemplo

$\Pi$  = encontrar um ciclo hamiltoniano

$\Pi'$  = existe um ciclo hamiltoniano?

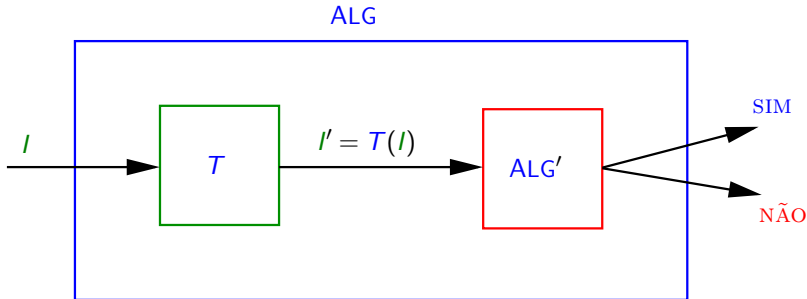
Redução de  $\Pi$  a  $\Pi'$ :  $ALG'$  é um algoritmo que resolve  $\Pi'$ .

$ALG(G)$

- 1 se  $ALG'(G) = \text{NÃO}$
- 2     então devolva “ $G$  não é hamiltoniano”
- 3 para cada aresta  $uv$  de  $G$  faça
- 4      $H \leftarrow G - uv$
- 5     se  $ALG'(H) = \text{SIM}$
- 6         então  $G \leftarrow G - uv$
- 7 devolva  $G$

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

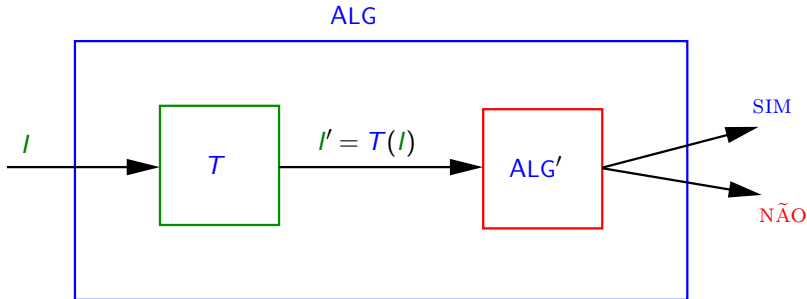
# Esquema comum de redução



Faz apenas uma chamada ao algoritmo  $ALG'$ .



# Esquema comum de redução

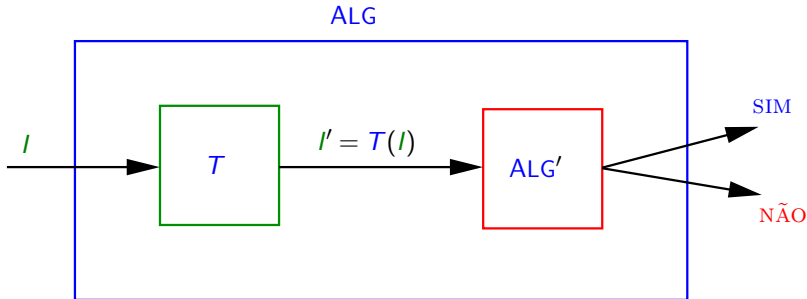


Faz apenas uma chamada ao algoritmo  $ALG'$ .

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

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

# Esquema comum de redução



Faz apenas uma chamada ao algoritmo  $ALG'$ .

$T$  transforma uma instância  $I$  de  $\Pi$   
em uma instância  $I' = T(I)$  de  $\Pi'$  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 uma fórmula booleana  $\phi$  nas variáveis  $x_1, \dots, x_n$ , existe uma atribuição

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

que torna  $\phi$  verdadeira?

# Satisfatibilidade

**Problema:** Dada uma fórmula booleana  $\phi$  nas variáveis  $x_1, \dots, x_n$ , existe uma atribuição

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

que torna  $\phi$  verdadeira?

**Exemplo:**

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

# Satisfatibilidade

**Problema:** Dada uma fórmula booleana  $\phi$  nas variáveis  $x_1, \dots, x_n$ , existe uma atribuição

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

que torna  $\phi$  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  $t(\phi) = \text{VERDADE}$ .

# Satisfatibilidade

**Problema:** Dada uma fórmula booleana  $\phi$  nas variáveis  $x_1, \dots, x_n$ , existe uma atribuição

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

que torna  $\phi$  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  $i$ ?

# 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:**

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

tem uma solução 0-1?



# 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:**

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

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

# Exemplo 1

Satisfatibilidade  $\leq_P$  Sistemas lineares 0-1

A transformação deve **receber** uma fórmula booleana  $\phi$   
e **devolver** um sistema linear  $Ax \geq b$   
**tal que**  $\phi$  é satisfatível se e somente se  
o sistema  $Ax \geq b$  admite uma solução 0-1.

# Exemplo 1

Satisfatibilidade  $\prec_P$  Sistemas lineares 0-1

A transformação deve **receber** uma fórmula booleana  $\phi$   
e **devolver** um sistema linear  $Ax \geq b$   
**tal que**  $\phi$  é satisfatível se e somente se  
o sistema  $Ax \geq b$  admite uma solução 0-1.

Exemplo:

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

Como fazer um sistema que tem solução sse  $\phi$  é satisfatível?

# Exemplo 1

Satisfatibilidade  $\prec_P$  Sistemas lineares 0-1

A transformação deve **receber** uma fórmula booleana  $\phi$   
e **devolver** um sistema linear  $Ax \geq b$   
**tal que**  $\phi$  é satisfatível se e somente se  
o sistema  $Ax \geq b$  admite uma solução 0-1.

Exemplo:

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

Como fazer um sistema que tem solução sse  $\phi$  é satisfatível?

Quem podem ser as variáveis do sistema?

# Exemplo 1

Satisfatibilidade  $\prec_P$  Sistemas lineares 0-1

A transformação deve **receber** uma fórmula booleana  $\phi$   
e **devolver** um sistema linear  $Ax \geq b$   
**tal que**  $\phi$  é satisfatível se e somente se  
o sistema  $Ax \geq b$  admite uma solução 0-1.

Exemplo:

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

Como fazer um sistema que tem solução sse  $\phi$  é satisfatível?

Quem podem ser as variáveis do sistema?

E quais seriam as restrições?

# Exemplo 1

Satisfatibilidade  $\prec_P$  Sistemas lineares 0-1

A transformação deve **receber** uma fórmula booleana  $\phi$   
e **devolver** um sistema linear  $Ax \geq b$   
**tal que**  $\phi$  é satisfatível se e somente se  
o sistema  $Ax \geq b$  admite uma solução 0-1.

Exemplo:

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

Como fazer um sistema que tem solução sse  $\phi$  é satisfatível?

Quem podem ser as variáveis do sistema?

E quais seriam as restrições?

Ideias?

# Exemplo 1

Satisfatibilidade  $\prec_P$  Sistemas lineares 0-1

A transformação  $T$  recebe uma fórmula booleana  $\phi$  e devolve um sistema linear  $Ax \geq b$

tal que  $\phi$  é satisfatível se e somente se o sistema  $Ax \geq b$  admite uma solução 0-1.

Exemplo:

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



# Exemplo 1

Satisfatibilidade  $\prec_P$  Sistemas lineares 0-1

A transformação  $T$  recebe uma fórmula booleana  $\phi$  e devolve um sistema linear  $Ax \geq b$

tal que  $\phi$  é satisfatível se e somente se o sistema  $Ax \geq b$  admite uma solução 0-1.

Exemplo:

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

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

# Exemplo 1

Satisfatibilidade  $\prec_P$  Sistemas lineares 0-1

A transformação  $T$  recebe uma fórmula booleana  $\phi$  e devolve um sistema linear  $Ax \geq b$

tal que  $\phi$  é satisfatível se e somente se o sistema  $Ax \geq b$  admite uma solução 0-1.

Exemplo:

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

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

## Exemplo 2

Verifique que

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

## Exemplo 2

Verifique que

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

Verifique que

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

## Exemplo 3

Caminho hamiltoniano  $\prec_P$  Satisfatibilidade

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

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

## Exemplo 3

Caminho hamiltoniano  $\prec_P$  Satisfatibilidade

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

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

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

## Exemplo 3

Caminho hamiltoniano  $\prec_P$  Satisfatibilidade

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

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

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

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

## Exemplo 3

Caminho hamiltoniano  $\prec_P$  Satisfatibilidade

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

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

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

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

Interpretação:  $x_{i,j} = \text{VERDADE}$   $\Leftrightarrow$  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  $j$  ( $n$  cláúsulas).

## 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  $j$  ( $n$  cláusulas).

- ▶ vértice  $j$  não está em duas posições do caminho:

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

para cada  $j$  e  $i \neq k$  ( $O(n^3)$  cláusulas).

## 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  $j$  ( $n$  cláúsulas).

- ▶ vértice  $j$  não está em duas posições do caminho:

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

para cada  $j$  e  $i \neq k$  ( $O(n^3)$  cláúsulas).

- ▶ algum vértice é o  $i$ -ésimo do caminho:

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

para cada  $i$  ( $n$  cláúsulas).

## Exemplo 3 (cont.)

Mais cláusulas de  $\phi(G)$ :

- ▶ dois vértices não podem ser o  $i$ -ésimo:

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

para cada  $i$  e  $j \neq k$  ( $O(n^3)$  cláusulas).

## Exemplo 3 (cont.)

Mais cláusulas de  $\phi(G)$ :

- ▶ dois vértices não podem ser o  $i$ -ésimo:

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

para cada  $i$  e  $j \neq k$  ( $O(n^3)$  cláusulas).

- ▶ 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 ( $O(n^3)$  cláusulas).

## Exemplo 3 (cont.)

Mais cláusulas de  $\phi(G)$ :

- ▶ dois vértices não podem ser o  $i$ -ésimo:

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

para cada  $i$  e  $j \neq k$  ( $O(n^3)$  cláusulas).

- ▶ 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 ( $O(n^3)$  cláusulas).

A fórmula  $\phi(G)$  tem  $O(n^3)$  cláusulas e cada cláusula tem  $\leq n$  literais. Logo,  $\langle \phi(G) \rangle$  é  $O(n^4)$ .

## Exemplo 3 (cont.)

Mais cláusulas de  $\phi(G)$ :

- ▶ dois vértices não podem ser o  $i$ -ésimo:

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

para cada  $i$  e  $j \neq k$  ( $O(n^3)$  cláusulas).

- ▶ 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 ( $O(n^3)$  cláusulas).

A fórmula  $\phi(G)$  tem  $O(n^3)$  cláusulas e cada cláusula tem  $\leq n$  literais. Logo,  $\langle \phi(G) \rangle$  é  $O(n^4)$ .

Não é difícil projetar o **algoritmo polinomial**  $T$ .

## Exemplo 3 (cont.)

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

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



## Exemplo 3 (cont.)

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

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

Para cada  $i$ , existe um único  $j$  tal que  $t(x_{i,j}) = \text{VERDADE}$ .

## Exemplo 3 (cont.)

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

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

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

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

dos vértices de  $G$ , onde

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

## 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  $t(x_{i,j}) = \text{VERDADE}$ .  
Logo,  $t$  é a codificação de uma permutação

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

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  $G$ .

## 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  $t(x_{i,j}) = \text{VERDADE}$ .  
Logo,  $t$  é a codificação de uma permutação

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

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

## Exemplo 3 (cont.)

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

Suponha que  $(\pi(1), \pi(2), \dots, \pi(n))$  é um caminho hamiltoniano, onde  $\pi$  é uma permutação dos vértices de  $G$ .

## Exemplo 3 (cont.)

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

Suponha que  $(\pi(1), \pi(2), \dots, \pi(n))$  é um caminho hamiltoniano, onde  $\pi$  é uma permutação dos vértices de  $G$ .

Então

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

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

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

# Problemas completos em NP

Um problema  $\Pi$  em NP é NP-completo se cada problema em NP pode ser reduzido a  $\Pi$ .

# Problemas completos em NP

Um problema  $\Pi$  em NP é NP-completo se cada problema em NP pode ser reduzido a  $\Pi$ .

Teorema de S. Cook e L.A. Levin:  
Satisfatibilidade é NP-completo.



# Problemas completos em NP

Um problema  $\Pi$  em NP é NP-completo se cada problema em NP pode ser reduzido a  $\Pi$ .

Teorema de S. Cook e L.A. Levin:  
Satisfatibilidade é NP-completo.

Se  $\Pi \prec_P \Pi'$  e  $\Pi$  é NP-completo, então

# Problemas completos em NP

Um problema  $\Pi$  em NP é NP-completo se cada problema em NP pode ser reduzido a  $\Pi$ .

Teorema de S. Cook e L.A. Levin:  
Satisfatibilidade é NP-completo.

Se  $\Pi \prec_P \Pi'$  e  $\Pi$  é NP-completo, então  $\Pi'$  é NP-completo.

# Problemas completos em NP

Um problema  $\Pi$  em NP é NP-completo se cada problema em NP pode ser reduzido a  $\Pi$ .

Teorema de S. Cook e L.A. Levin:  
Satisfatibilidade é NP-completo.

Se  $\Pi \prec_P \Pi'$  e  $\Pi$  é NP-completo, então  $\Pi'$  é NP-completo.

Existe um algoritmo polinomial para um problema NP-completo se e somente se  $P = NP$ .