

# Redução polinomial

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

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

**Redução** de  $\Pi$  a  $\Pi'$ :

algoritmo **ALG** que resolve  $\Pi$  usando uma subrotina hipotética **ALG'**  
que resolve  $\Pi'$ , de forma que...

# Redução polinomial

Permite comparar

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

$\Pi, \Pi'$ : problemas

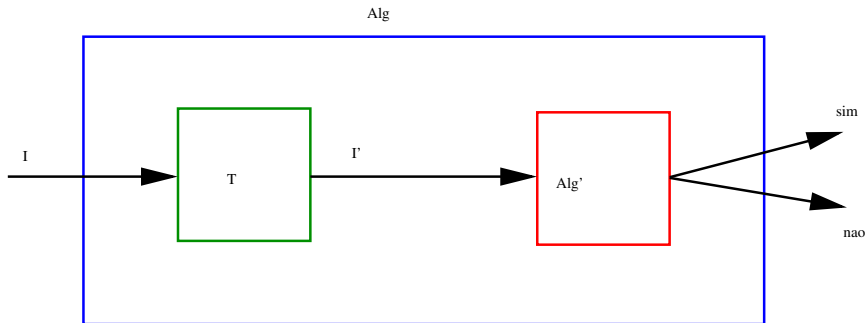
**Redução** de  $\Pi$  a  $\Pi'$ :

algoritmo **ALG** que resolve  $\Pi$  usando uma subrotina hipotética **ALG'** que resolve  $\Pi'$ , de forma que,

se **ALG'** é polinomial, então **ALG** é um algoritmo polinomial.

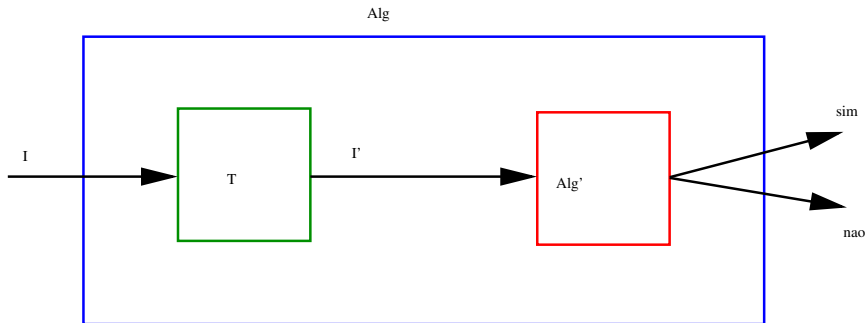
$\Pi \prec_P \Pi' =$  existe uma redução de  $\Pi$  a  $\Pi'$ .

# 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

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

# Satisfatibilidade

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

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

que torna  $\phi$  verdadeira?

# Satisfatibilidade

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

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

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

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

que torna  $\phi$  verdadeira?

**Exemplo:**  $\phi = (x_1) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_3)$ .

Um **literal** é uma variável  $x$  ou sua negação  $\neg x$ .

## 3-Satisfatibilidade

**Problema:** Dada uma fórmula booleana  $\phi$  nas variáveis  $x_1, \dots, x_n$  em que cada cláusula **tem exatamente 3 literais**, existe uma atribuição

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

que torna  $\phi$  verdadeira?



## 3-Satisfatibilidade

**Problema:** Dada uma fórmula booleana  $\phi$  nas variáveis  $x_1, \dots, x_n$  em que cada cláusula **tem exatamente 3 literais**, existe uma atribuição

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

que torna  $\phi$  verdadeira?

Exemplo:

$$\phi = (x_1 \vee \neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

# Exemplo 4

Satisfatibilidade  $<_p$  3-Satisfatibilidade

# Exemplo 4

Satisfatibilidade  $<_p$  3-Satisfatibilidade

Descreveremos um **algoritmo polinomial**  $T$  que recebe uma fórmula booleana  $\phi$  e devolve uma fórmula booleana  $\phi'$  com **exatamente 3 literais** por cláusula tal que

$\phi$  é satisfável  $\Leftrightarrow \phi'$  é satisfável.

# Exemplo 4

Satisfatibilidade  $<_p$  3-Satisfatibilidade

Descreveremos um **algoritmo polinomial**  $T$  que recebe uma fórmula booleana  $\phi$  e devolve uma fórmula booleana  $\phi'$  com **exatamente 3 literais** por cláusula tal que

$\phi$  é satisfável  $\Leftrightarrow \phi'$  é satisfável.

A transformação consiste em substituir **cada cláusula** de  $\phi$  por uma **coleção de cláusulas** com **exatamente 3 literais** cada, e **equivalente** a  $\phi$ .

## Exemplo 4 (cont.)

Seja  $(l_1 \vee l_2 \vee \cdots \vee l_k)$  uma cláusula de  $\phi$ .

## Exemplo 4 (cont.)

Seja  $(l_1 \vee l_2 \vee \dots \vee l_k)$  uma cláusula de  $\phi$ .

Caso 1.  $k = 1$

Troque  $(l_1)$  por

$$(l_1 \vee y_1 \vee y_2)(l_1 \vee \neg y_1 \vee y_2)(l_1 \vee y_1 \vee \neg y_2)(l_1 \vee \neg y_1 \vee \neg y_2)$$

onde  $y_1$  e  $y_2$  são **variáveis novas**.

## Exemplo 4 (cont.)

Seja  $(l_1 \vee l_2 \vee \dots \vee l_k)$  uma cláusula de  $\phi$ .

Caso 1.  $k = 1$

Troque  $(l_1)$  por

$$(l_1 \vee y_1 \vee y_2)(l_1 \vee \neg y_1 \vee y_2)(l_1 \vee y_1 \vee \neg y_2)(l_1 \vee \neg y_1 \vee \neg y_2)$$

onde  $y_1$  e  $y_2$  são **variáveis novas**.

Caso 2.  $k = 2$

Troque  $(l_1 \vee l_2)$  por  $(l_1 \vee l_2 \vee y)(l_1 \vee l_2 \vee \neg y)$ ,

onde  $y$  é uma **variável nova**.

## Exemplo 4 (cont.)

Seja  $(l_1 \vee l_2 \vee \dots \vee l_k)$  uma cláusula de  $\phi$ .

Caso 1.  $k = 1$

Troque  $(l_1)$  por

$$(l_1 \vee y_1 \vee y_2)(l_1 \vee \neg y_1 \vee y_2)(l_1 \vee y_1 \vee \neg y_2)(l_1 \vee \neg y_1 \vee \neg y_2)$$

onde  $y_1$  e  $y_2$  são **variáveis novas**.

Caso 2.  $k = 2$

Troque  $(l_1 \vee l_2)$  por  $(l_1 \vee l_2 \vee y)(l_1 \vee l_2 \vee \neg y)$ ,

onde  $y$  é uma **variável nova**.

Caso 3.  $k = 3$

Mantenha  $(l_1 \vee l_2 \vee l_3)$ .



## Exemplo 4 (cont.)

Caso 4.  $k > 3$

Troque  $(l_1 \vee l_2 \vee \dots \vee l_k)$  por

$(l_1 \vee l_2 \vee y_1)$

$(\neg y_1 \vee l_3 \vee y_2)(\neg y_2 \vee l_4 \vee y_3)(\neg y_3 \vee l_5 \vee y_4) \dots$

$(\neg y_{k-3} \vee l_{k-1} \vee l_k)$

onde  $y_1, y_2, \dots, y_{k-3}$  são **variáveis novas**.

## Exemplo 4 (cont.)

Caso 4.  $k > 3$

Troque  $(l_1 \vee l_2 \vee \dots \vee l_k)$  por

$(l_1 \vee l_2 \vee y_1)$

$(\neg y_1 \vee l_3 \vee y_2)(\neg y_2 \vee l_4 \vee y_3)(\neg y_3 \vee l_5 \vee y_4) \dots$

$(\neg y_{k-3} \vee l_{k-1} \vee l_k)$

onde  $y_1, y_2, \dots, y_{k-3}$  são **variáveis novas**.

Verifique que  $\phi$  é satisfátivel  $\Leftrightarrow$  nova fórmula é satisfátivel.

## Exemplo 4 (cont.)

Caso 4.  $k > 3$

Troque  $(l_1 \vee l_2 \vee \dots \vee l_k)$  por

$(l_1 \vee l_2 \vee y_1)$

$(\neg y_1 \vee l_3 \vee y_2)(\neg y_2 \vee l_4 \vee y_3)(\neg y_3 \vee l_5 \vee y_4) \dots$

$(\neg y_{k-3} \vee l_{k-1} \vee l_k)$

onde  $y_1, y_2, \dots, y_{k-3}$  são **variáveis novas**.

Verifique que  $\phi$  é satisfátivel  $\Leftrightarrow$  nova fórmula é satisfátivel.

O tamanho da nova cláusula é  $O(m)$ , onde  $m$  é o número de literais que ocorrem em  $\phi$  (contando-se as repetições).

# 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  $\Pi <_p \Pi'$  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  $\Pi <_p \Pi'$  e  $\Pi$  é NP-completo, então  $\Pi'$  é NP-completo.

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

# Demonstração de NP-completude

Para demonstrar que um problema  $\Pi'$  é NP-completo podemos utilizar o Teorema de Cook e Levin.



# Demonstração de NP-completude

Para demonstrar que um problema  $\Pi'$  é NP-completo podemos utilizar o Teorema de Cook e Levin.

Para isto devemos:

- ▶ Demonstrar que  $\Pi'$  está em NP.
- ▶ Escolher um problema  $\Pi$  sabidamente NP-completo.
- ▶ Demonstrar que  $\Pi <_P \Pi'$ .

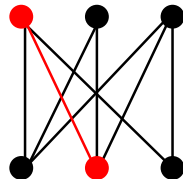
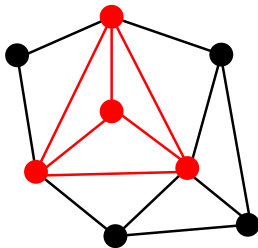
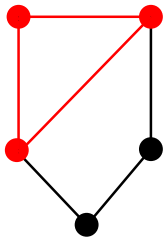
# Clique

**Problema:** Dado um grafo  $G$  e um inteiro  $k$ ,  
 $G$  possui um clique com  $\geq k$  vértices?

# Clique

**Problema:** Dado um grafo  $G$  e um inteiro  $k$ ,  $G$  possui um clique com  $\geq k$  vértices?

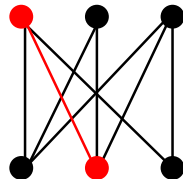
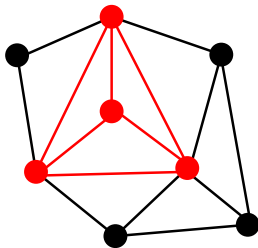
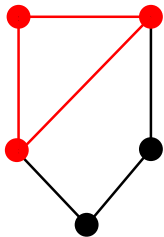
Exemplos:



# Clique

**Problema:** Dado um grafo  $G$  e um inteiro  $k$ ,  $G$  possui um clique com  $\geq k$  vértices?

Exemplos:



clique com  $k$  vértices = subgrafo completo com  $k$  vértices

# Clique é NP-completo

Clique está em NP e 3-Satisfatibilidade  $\leq_p$  Clique.

# Clique é NP-completo

Clique está em NP e 3-Satisfatibilidade  $\leq_p$  Clique.

Descreveremos um algoritmo polinomial  $T$  que recebe uma fórmula booleana  $\phi$  com  $k$  cláusulas e exatamente 3 literais por cláusula e devolve um grafo  $G$  tais que

$\phi$  é satisfatível  $\Leftrightarrow G$  possui um clique  $\geq k$ .

# Clique é NP-completo

Clique está em NP e 3-Satisfatibilidade  $\leq_p$  Clique.

Descreveremos um algoritmo polinomial  $T$  que recebe uma fórmula booleana  $\phi$  com  $k$  cláusulas e exatamente 3 literais por cláusula e devolve um grafo  $G$  tais que

$\phi$  é satisfável  $\Leftrightarrow G$  possui um clique  $\geq k$ .

Para cada cláusula, o grafo  $G$  terá três vértices, um correspondente a cada literal da cláusula. Logo,  $G$  terá  $3k$  vértices.

# Clique é NP-completo

Clique está em NP e 3-Satisfatibilidade  $\leq_p$  Clique.

Descreveremos um algoritmo polinomial  $T$  que recebe uma fórmula booleana  $\phi$  com  $k$  cláusulas e exatamente 3 literais por cláusula e devolve um grafo  $G$  tais que

$\phi$  é satisfável  $\Leftrightarrow G$  possui um clique  $\geq k$ .

Para cada cláusula, o grafo  $G$  terá três vértices, um correspondente a cada literal da cláusula. Logo,  $G$  terá  $3k$  vértices. Teremos uma aresta ligando vértices  $u$  e  $v$  se

- ▶  $u$  e  $v$  são vértices que correspondem a literais em diferentes cláusulas; e
- ▶ se  $u$  corresponde a um literal  $x$  então  $v$  não corresponde ao literal  $\neg x$ .

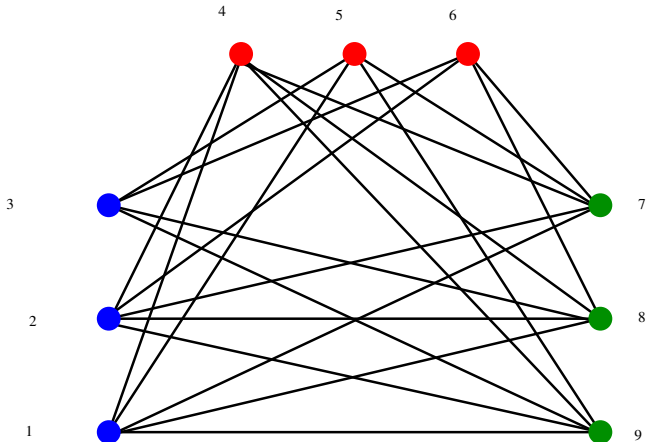


## Clique é NP-completo (cont.)

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

# Clique é NP-completo (cont.)

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



# Cobertura por vértices

Um conjunto  $S$  de vértices de um grafo  $G$  é uma **cobertura** se toda aresta de  $G$  tem uma ponta em  $S$ .

# Cobertura por vértices

Um conjunto  $S$  de vértices de um grafo  $G$  é uma **cobertura** se toda aresta de  $G$  tem uma ponta em  $S$ .

**Problema:** Dado um grafo  $G$  e um inteiro  $k$ ,  $G$  possui uma cobertura com  $\leq k$  vértices?

# Cobertura por vértices

Um conjunto  $S$  de vértices de um grafo  $G$  é uma **cobertura** se toda aresta de  $G$  tem uma ponta em  $S$ .

**Problema:** Dado um grafo  $G$  e um inteiro  $k$ ,  $G$  possui uma cobertura com  $\leq k$  vértices?

Você consegue provar que este problema é NP-completo?

# Problemas NP-difíceis

Um problema  $\Pi$ , não necessariamente em NP, é NP-difícil se a existência de um algoritmo polinomial para  $\Pi$  implica que  $P = NP$ .

# Problemas NP-difíceis

Um problema  $\Pi$ , não necessariamente em NP, é NP-difícil se a existência de um algoritmo polinomial para  $\Pi$  implica que  $P = NP$ .

Todo problema NP-completo é NP-difícil.

# Problemas NP-difíceis

Um problema  $\Pi$ , não necessariamente em NP, é NP-difícil se a existência de um algoritmo polinomial para  $\Pi$  implica que  $P = NP$ .

Todo problema NP-completo é NP-difícil.

Exemplos:

- ▶ Encontrar um ciclo hamiltoniano é NP-difícil, mas não é NP-completo, pois não é um problema de decisão e portanto não está em NP.
- ▶ Satisfabilidade é NP-completo e NP-difícil.



# Mais problemas NP-difíceis

Os seguintes problema são NP-difíceis:

- ▶ mochila booleana
- ▶ caminho máximo
- ▶ caminho hamiltoniano
- ▶ escalonamento de tarefas
- ▶ subset-sum
- ▶ clique máximo
- ▶ cobertura por vértices
- ▶ sistemas 0-1

e mais um montão deles ...

# Um problema limítrofe

# Um problema limítrofe

GRAPH ISOMORPHISM: *Dados dois grafos, eles são isomorfos?*

# Um problema limítrofe

GRAPH ISOMORPHISM: *Dados dois grafos, eles são isomorfos?*

Na prática, bem resolvido: o programa **NAUTY** (B. McKay).

# Um problema limítrofe

GRAPH ISOMORPHISM: *Dados dois grafos, eles são isomorfos?*

Na prática, bem resolvido: o programa **NAUTY** (B. McKay).

Note que SUBGRAPH ISOMORPHISM (dados  $G$  e  $H$ ,  $G$  tem subgrafo isomorfo a  $H$ ?) é **NP**-completo.