

Complexidade computacional

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

CLRS 34

Classes P, NP e co-NP

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

Classes P, NP e co-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, NP e co-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**

Classes P, NP e co-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**

Classe co-NP:

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

Verificador polinomial para SIM

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

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

Verificador polinomial para SIM

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

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

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 Π é um algoritmo polinomial **ALG** que **recebe**

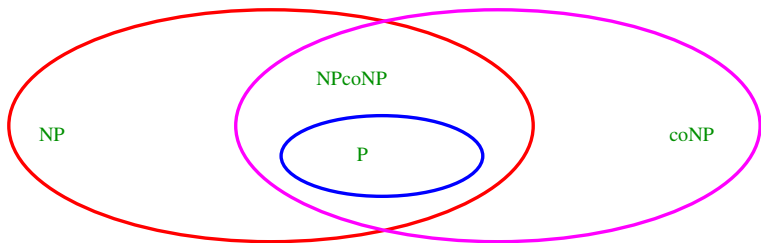
*uma instância I de Π e um objeto C ,
tal que $\langle C \rangle$ é $O(\langle I \rangle^\alpha)$ para alguma constante α*

e **devolve**

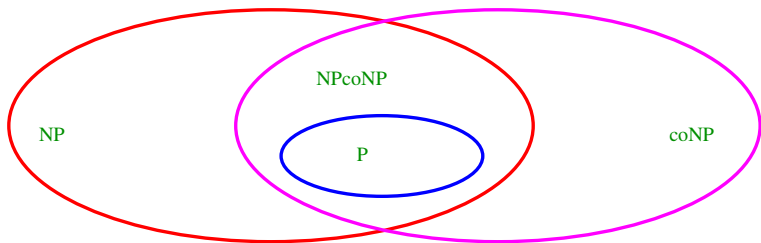
SIM para algum C se a resposta a $\Pi(I)$ é **SIM**;
NÃO para todo C se a resposta a $\Pi(I)$ é **NÃO**.

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

P, NP e co-NP



P, NP e co-NP



$P \neq NP?$

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

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

Redução polinomial

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

Redução polinomial

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

Π , Π' : problemas

Uma **redução** de Π a Π' é um algoritmo **ALG** que resolve Π usando uma subrotina hipotética **ALG'** que resolve Π' , 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.

Π, Π' : problemas

Uma **redução** de Π a Π' é um algoritmo **ALG** que resolve Π usando uma subrotina hipotética **ALG'** que resolve Π' , de 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?

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 $ALG'(G) = \text{NÃO}$
- 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

Π = encontrar um ciclo hamiltoniano

Π' = existe um ciclo hamiltoniano?

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

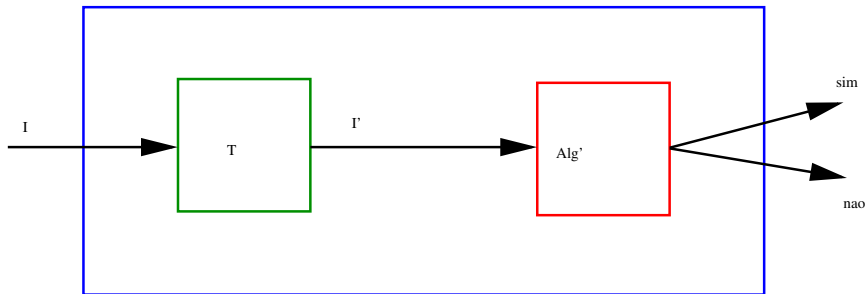
$ALG(G)$

- 1 se $ALG'(G) = \text{NÃO}$
- 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

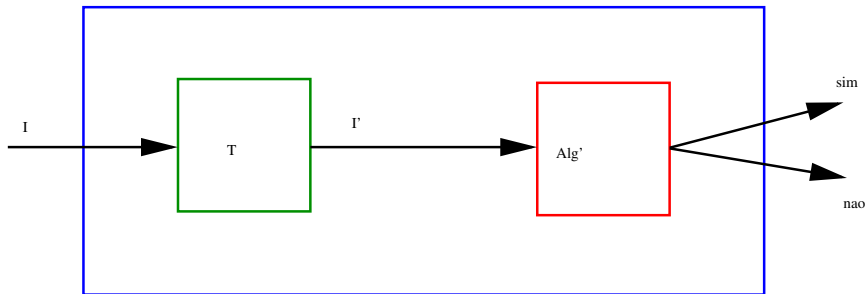
Alg



Faz apenas uma chamada ao algoritmo **ALG'**.

Esquema comum de redução

Alg



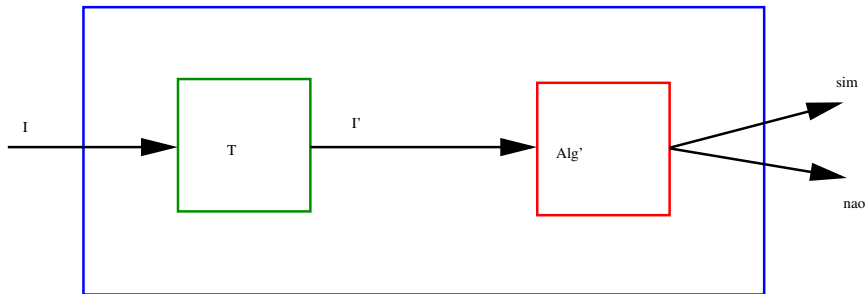
Faz apenas uma chamada ao algoritmo ALG' .

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

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

Esquema comum de redução

Alg



Faz apenas uma chamada ao algoritmo ALG' .

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

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

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

Satisfatibilidade

Problema: Dada uma fórmula booleana ϕ nas variáveis x_1, \dots, x_n , existe uma atribuição

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

que torna ϕ verdadeira?

Satisfatibilidade

Problema: Dada uma fórmula booleana ϕ nas variáveis x_1, \dots, x_n , existe uma atribuição

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

que torna ϕ 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 ϕ nas variáveis x_1, \dots, x_n , existe uma atribuição

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

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 $t(\phi) = \text{VERDADE}$

Satisfatibilidade

Problema: Dada uma fórmula booleana ϕ nas variáveis x_1, \dots, x_n , existe uma atribuição

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

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 ?

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 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 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:

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

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

Exemplo 1

Satisfatibilidade \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:

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

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

Exemplo 2

Verifique que

Ciclo hamiltoniano \leq_P Caminho hamiltoniano entre u e v

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

Exemplo 3

Caminho hamiltoniano \leq_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 \leq_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.

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 \leq_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.

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 claúsulas).

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 $i \neq k$ ($O(n^3)$ claúsulas).

Exemplo 3 (cont.)

Claúsulas de $\phi(G)$:

- ▶ vértice j faz parte do caminho:

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

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 $i \neq k$ ($O(n^3)$ claúsulas).

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

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

para cada i (n claúsulas).

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 $t : \{\text{variáveis}\} \rightarrow \{\text{VERDADE, FALSO}\}$

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 $t : \{\text{variáveis}\} \rightarrow \{\text{VERDADE, FALSO}\}$

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 $\Rightarrow \phi(G)$ satisfável.

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

Exemplo 3 (cont.)

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

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

Então

$t(x_{i,j}) = \text{VERDADE}$ se $\pi(i) = j$ 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 Π em NP é NP-completo se cada problema em NP pode ser reduzido a Π .

Problemas completos em NP

Um problema Π em NP é NP-completo se cada problema em NP pode ser reduzido a Π .

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

Problemas completos em NP

Um problema Π em NP é NP-completo se cada problema em NP pode ser reduzido a Π .

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

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

Problemas completos em NP

Um problema Π em NP é NP-completo se cada problema em NP pode ser reduzido a Π .

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

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

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