

Análise de Algoritmos

**Parte destes slides são adaptações de slides
do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.**

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

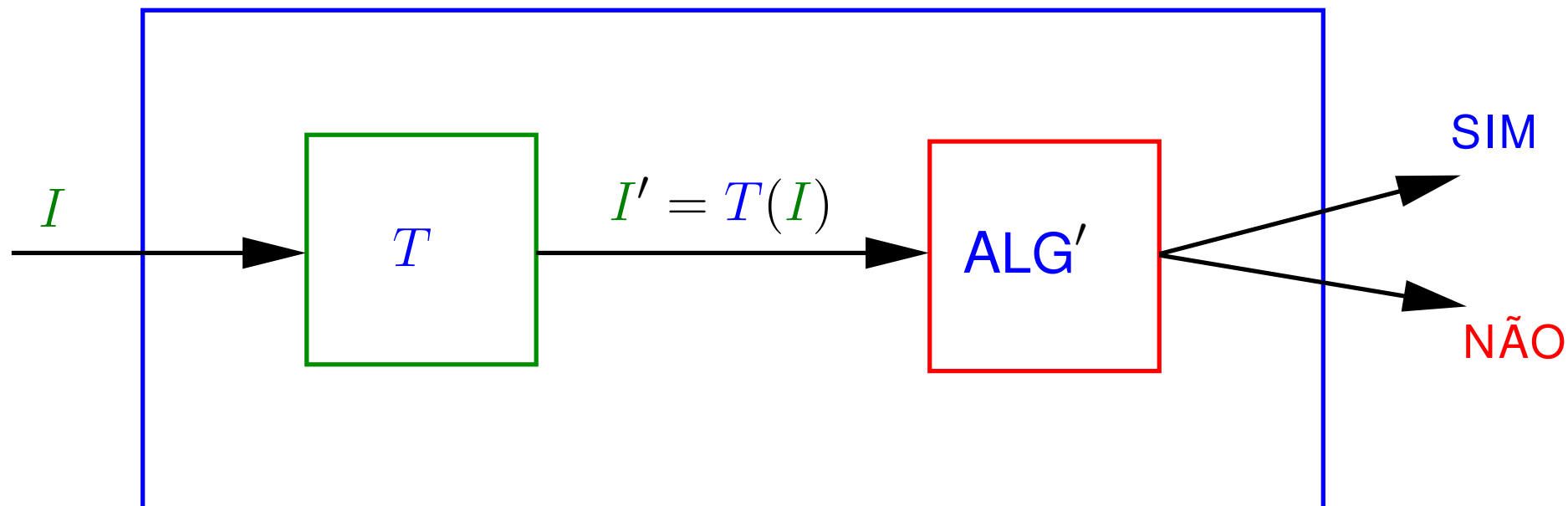
Uma **redução** de Π a Π' é um algoritmo **ALG** que resolve Π usando uma subrotina hipotética **ALG'** que resolve Π' , de forma que, se **ALG'** é um algoritmo polinomial, então **ALG** é um algoritmo polinomial.

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

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

Esquema comum de redução

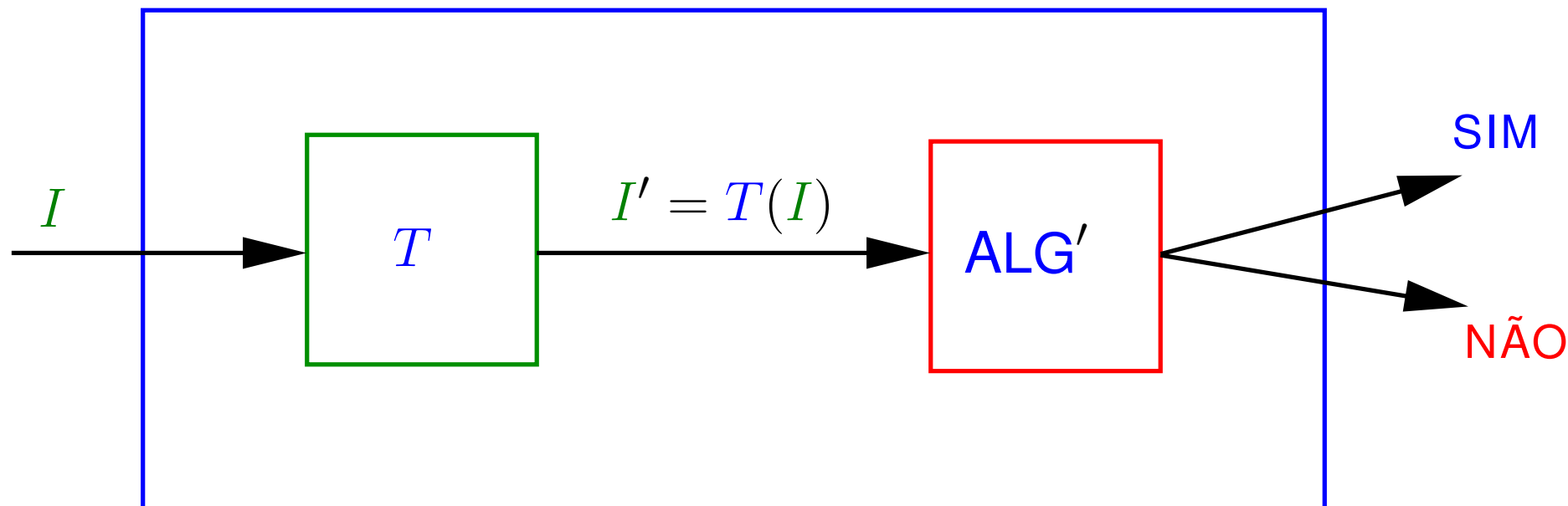
ALG



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

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

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

que torna ϕ verdadeira?

Exemplo:

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

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

3-Satisfatibilidade

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

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

que torna ϕ verdadeira?

3-Satisfatibilidade

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

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

que torna ϕ verdadeira?

Exemplo:

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

3-Satisfatibilidade

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

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

que torna ϕ verdadeira?

Exemplo:

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

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

Exemplo 4

Satisfatibilidade $\leq_P$ 3-Satisfatibilidade

Exemplo 4

Satisfatibilidade $\leq_P$ 3-Satisfatibilidade

Descreveremos um **algoritmo polinomial** T que recebe uma fórmula booleana ϕ e devolve uma fórmula booleana ϕ' com **exatamente 3 literais** por cláusula tais que

ϕ é satisfatível $\Leftrightarrow \phi'$ é satisfatível.

Exemplo 4

Satisfatibilidade $\leq_P$ 3-Satisfatibilidade

Descreveremos um **algoritmo polinomial** T que recebe uma fórmula booleana ϕ e devolve uma fórmula booleana ϕ' com **exatamente 3 literais** por cláusula tais que

ϕ é satisfatível $\Leftrightarrow \phi'$ é satisfatível.

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

Exemplo 4 (cont.)

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

Exemplo 4 (cont.)

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

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 ϕ .

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 ϕ .

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 ϕ é satisfável $\Leftrightarrow$ nova fórmula é satisfável.

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 ϕ é satisfável $\Leftrightarrow$ nova fórmula é satisfável.

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

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 $P = NP$.

Demonstração de NP-completude

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

Demonstração de NP-completude

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

Para isto devemos:

- Demonstrar que Π' está em NP.
- Escolher um problema Π sabidamente NP-completo.
- Demonstrar que $\Pi \leq_P \Pi'$.

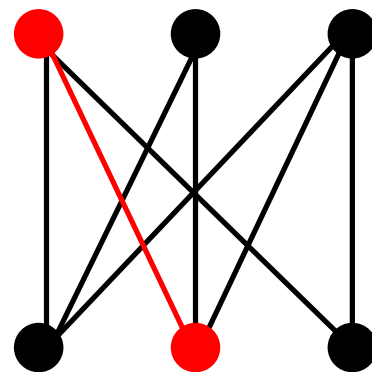
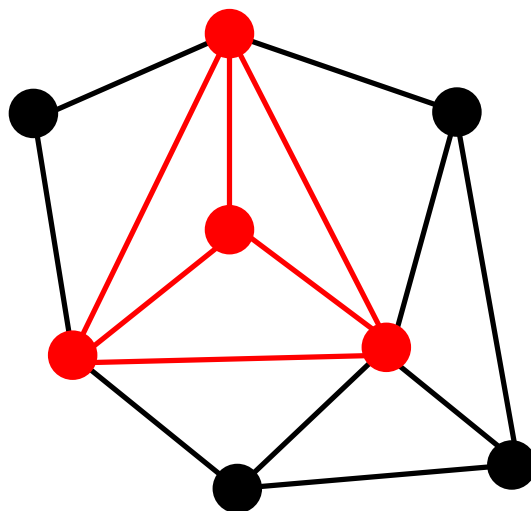
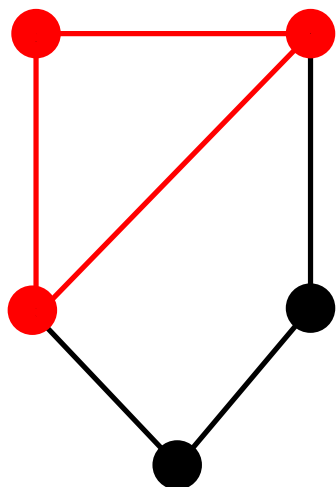
Clique

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?

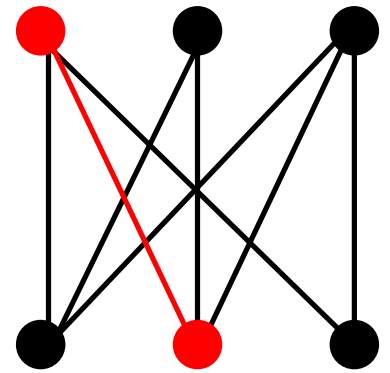
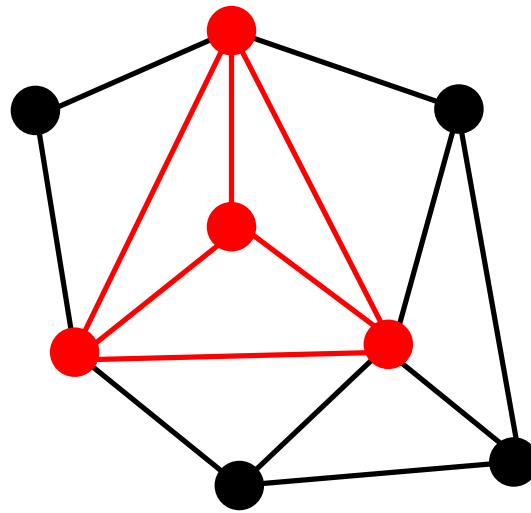
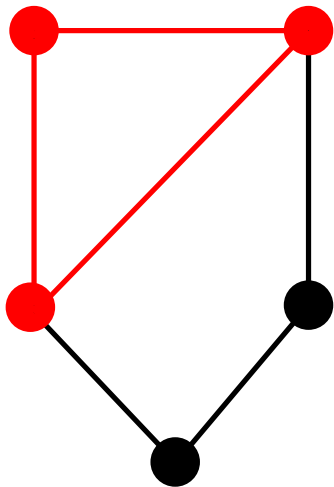
Exemplos:



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 ϕ com k cláusulas e exatamente 3 literais por cláusula e devolve um grafo G tais que

ϕ 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 ϕ com k cláusulas e exatamente 3 literais por cláusula e devolve um grafo G tais que

ϕ é satisfatível $\Leftrightarrow G$ 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

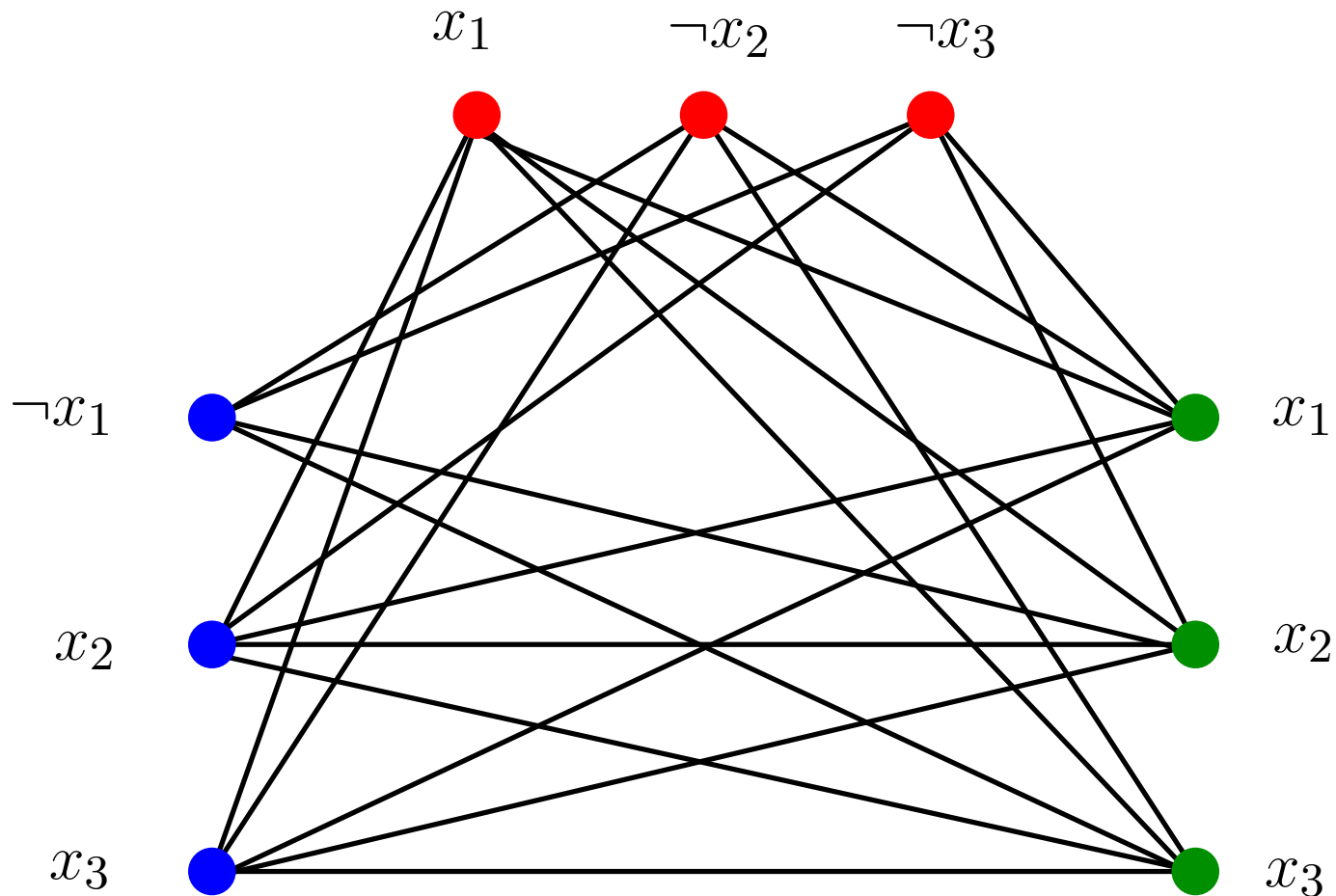
- u e v são vértices que correspondem a literais em diferentes cláusulas; e
- se u corresponde a um literal x então v não corresponde ao literal $\neg x$.

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.)

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



Problemas NP-difíceis

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

Problemas NP-difíceis

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

Todo problema NP-completo é NP-difícil.

Problemas NP-difíceis

Um problema Π , não necessariamente em NP, é NP-difícil se a existência de um algoritmo polinomial para Π 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 ...

Exercícios

Exercício 25.A

Suponha que os algoritmos ALG e ALG' transformam cadeias de caracteres em outras cadeias de caracteres. O algoritmo ALG consome $O(n^2)$ unidades de tempo e o algoritmo ALG' consome $O(n^4)$ unidades de tempo, onde n é o número de caracteres da cadeia de entrada. Considere agora o algoritmo $ALGALG'$ que consiste na composição de ALG e ALG' , com ALG' recebendo como entrada a saída de ALG . Qual o consumo de tempo de $ALGALG'$?

Exercício 25.B [CLRS 34.1-4]

O algoritmo $MOCHILA-BOOLEANA$ é polinomial? Justifique a sua resposta.

Exercício 25.C [CLRS 34.1-5]

Seja ALG um algoritmo que faz um número **constante** de chamadas a um algoritmo ALG' . Suponha que se o consumo de tempo de ALG' é constante então o consumo de tempo de ALG é polinomial.

1. Mostre que se o consumo de tempo de ALG' é polinomial então o consumo de tempo de ALG é polinomial.
2. Mostre que se o consumo de tempo de ALG' é polinomial e ALG faz um número polinomial de chamadas a ALG' , então é possível que o consumo de tempo de ALG seja exponencial.

Mais exercícios

Exercício 25.D [CLRS 34.2-1]

Mostre que o problema de decidir se dois grafos dados são isomorfos está em **NP**.

Exercício 25.E [CLRS 34.2-2]

Mostre que um grafo bipartido com um número ímpar de vértices não é hamiltoniano (= possui um ciclo hamiltoniano).

Exercício 25.F [CLRS 34.2-3]

Mostre que se o problema do **Ciclo hamiltoniano** está em **P**, então o problema de listar os vértices de um ciclo hamiltoniano, na ordem em que eles ocorrem no ciclo, pode ser resolvido em tempo polinomial.

Exercício 25.G [CLRS 34.2-5]

Mostre que qualquer problema em **NP** pode ser resolvido por um algoritmo de consumo de tempo $2^{O(n^c)}$, onde n é o tamanho da entrada e c é uma constante.

Exercício 25.H [CLRS 34.2-6]

Mostre que o problema do **Caminho hamiltoniano** está em **NP**.

Exercício 25.I [CLRS 34.2-7]

Mostre que o problema do caminho hamiltoniano pode ser resolvido em tempo polinomial em grafos orientado acíclicos.

Mais exercícios

Exercício 25.J [CLRS 34.2-8]

Uma fórmula booleana ϕ é uma **tautologia** se $t(\phi) = \text{VERDADE}$ para toda atribuição de $t : \{\text{variáveis}\} \rightarrow \{\text{VERDADE}, \text{FALSO}\}$. Mostre que o problema de decidir se uma dada fórmula booleana é uma tautologia está em **co-NP**.

Exercício 25.K [CLRS 34.2-9]

Prove que **P** $\subseteq$ **co-NP**.

Exercício 25.L [CLRS 34.2-10]

Prove que se **NP** $\neq$ **co-NP**, então **P** $\neq$ **NP**.

Exercício 25.M [CLRS 34.2-11]

Se G é um grafo conexo com pelo menos 3 vértices, então G^3 é o grafo que se obtém a partir de G ligando-se por uma aresta todos os pares de vértices que estão conectados em G por um caminho com no máximo três arestas. Mostre que G^3 é hamiltoniano.

Exercício 25.N [CLRS 34.3-2]

Mostre que se $\Pi_1 \leq_P \Pi_2$ e $\Pi_2 \leq_P \Pi_3$, então $\Pi_1 \leq_P \Pi_3$.

Mais exercícios

Exercício 25.O [CLRS 34.3-7]

Suponha que Π e Π' são problemas de decisão sobre o mesmo conjunto de instâncias e que $\Pi(I) = \text{SIM}$ se e somente se $\Pi'(I) = \text{NÃO}$. Mostre que Π é NP-completo se e somente se Π' é co-NP-completo. (Um problema Π' é co-NP-completo se Π' está em co-NP e $\Pi \leq_P \Pi'$ para todo problema Π em co-NP.)

Exercício 25.P [CLRS 34.4-4]

Mostre que o problema de decidir se uma fórmula booleana é uma tautologia é co-NP-completo. (Dica: veja o exercício 25.O.)

Exercício 25.Q [CLRS 34.4-6]

Suponha que ALG' é um algoritmo polinomial para Satisfatibilidade.

Descreva um algoritmo polinomial ALG que recebe um fórmula booleana ϕ e devolve uma atribuição $t : \{\text{variáveis}\} \rightarrow \{\text{VERDADE}, \text{FALSO}\}$ tal que $t(\phi) = \text{VERDADE}$.

Exercício 25.Q [CLRS 34.5-3]

Prove que o problema Sistemas lineares 0-1 é NP-completo.

Exercício 25.R [CLRS 34.5-6]

Mostre que o problema Caminho hamiltoniano é NP-completo.

Exercício 25.S [CLRS 34.5-7]

Mostre que o problema de encontrar um ciclo de comprimento máximo é NP-completo.