

Melhores momentos

AULA 24

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 $O(\langle I \rangle^c)$ para resolver o problema, onde c é uma constante e I é instância do problema.

Exemplos

- Máximo divisor comum

Tamanho da instância: $\lg a + \lg b$

Consumo de tempo **Café-Com-Leite** é $O(b)$
(não-polinomial)

Consumo de tempo **EUCLIDES** é $O(\lg b)$ (polinomial)

- Subseqüência comum máxima

Tamanho da instância: $n + m$

Consumo de tempo **REC-LEC-LENGTH** é $\Omega(2^{\min\{m,n\}})$
(exponencial)

Consumo de tempo **LEC-LENGTH** é $\Theta(mn)$
(polinomial).

Mais exemplos

- Problema booleano da mochila

Tamanho da instância: $\lg n + n \lg W + n \lg V + \lg W$

Consumo de tempo MOCHILA-BOOLEANA é $\Theta(nW)$
(não-polinomial)

- Problema fracionário da mochila

Tamanho da instância: $\lg n + n \lg W + n \lg V + \lg W$

Consumo de tempo MOCHILA-FRACIONÁRIA é $\Theta(n \lg n)$ (polinomial).

- Ordenação de inteiros $A[1..n]$

Tamanho da instância: $n \lg M$,

$M := \max\{|A[1]|, |A[2]|, \dots, |A[n]|\} + 1$

Consumo de tempo MERGE-SORT é $\Theta(n \lg n)$
(polinomial).

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:

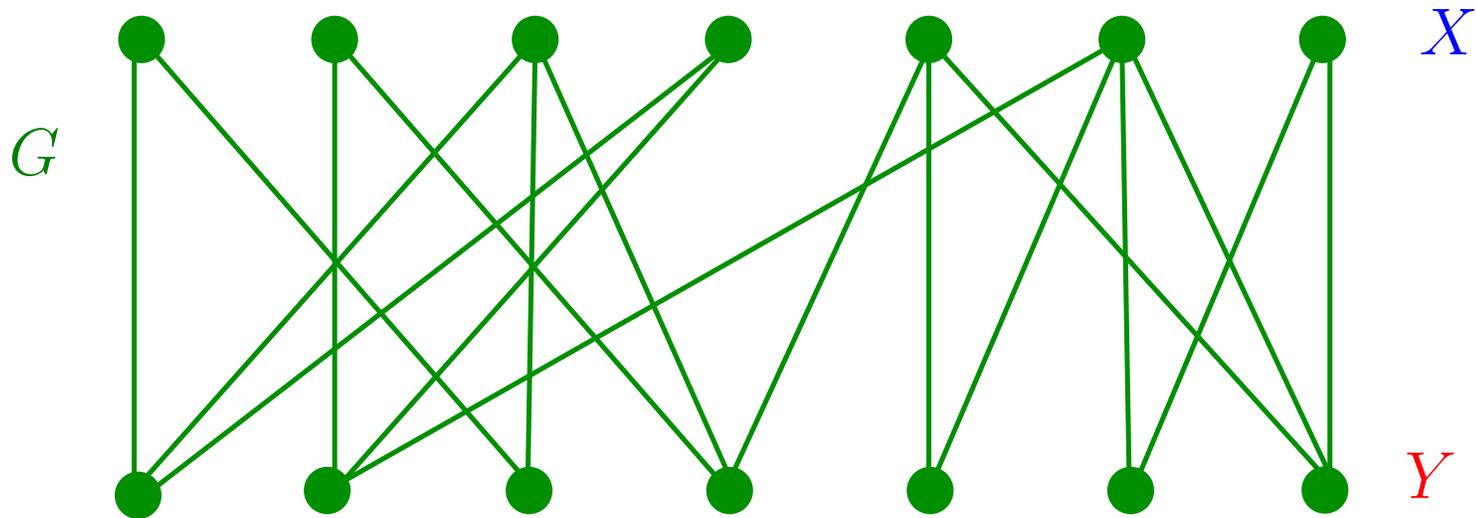
máximo divisor comum, subsequência comum
máxima e mochila fracionária

estão em **P**.

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

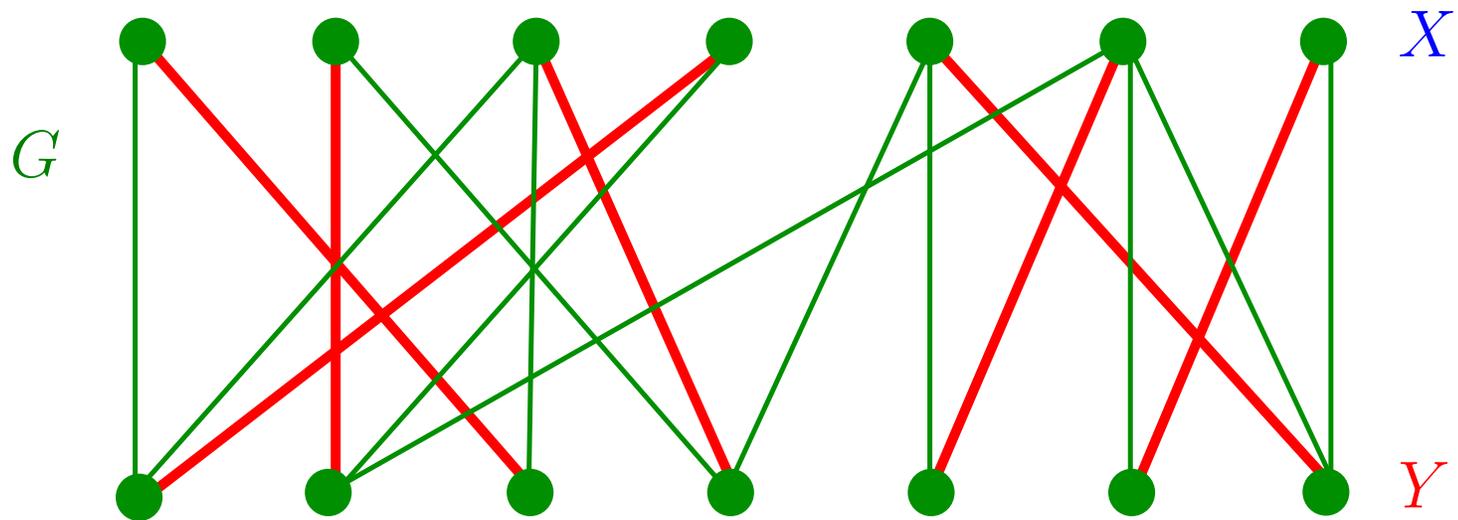
Emparelhamentos

Problema: Dado um grafo bipartido encontrar um emparelhamento perfeito.



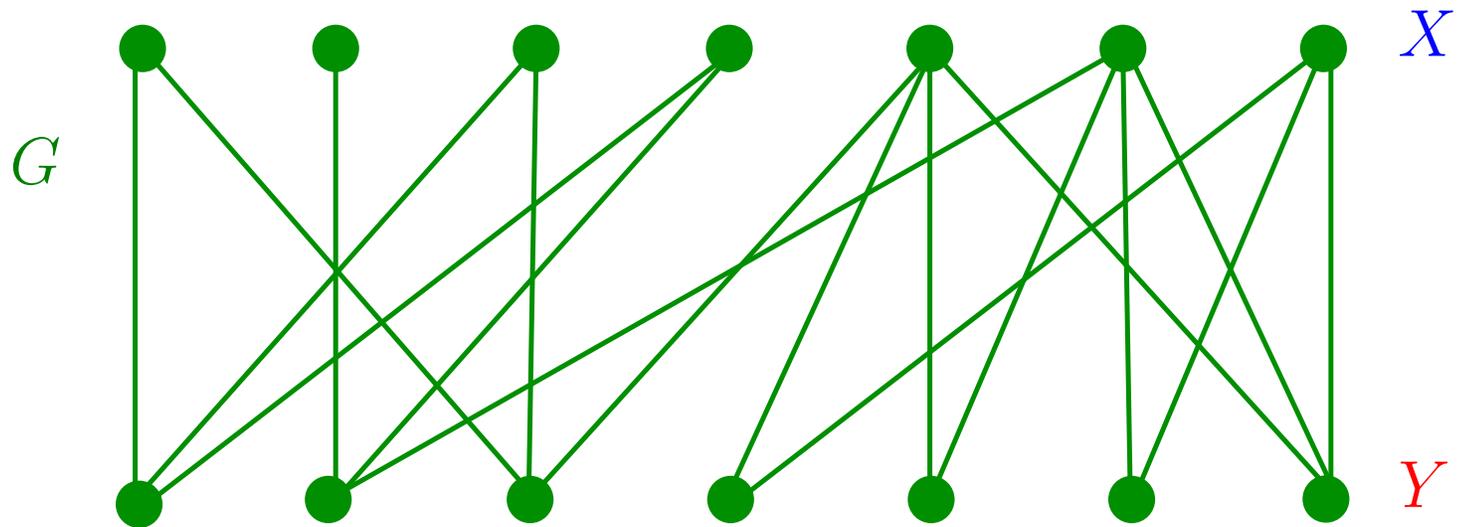
Emparelhamentos

Problema: Dado um grafo bipartido encontrar um emparelhamento perfeito.



Emparelhamentos

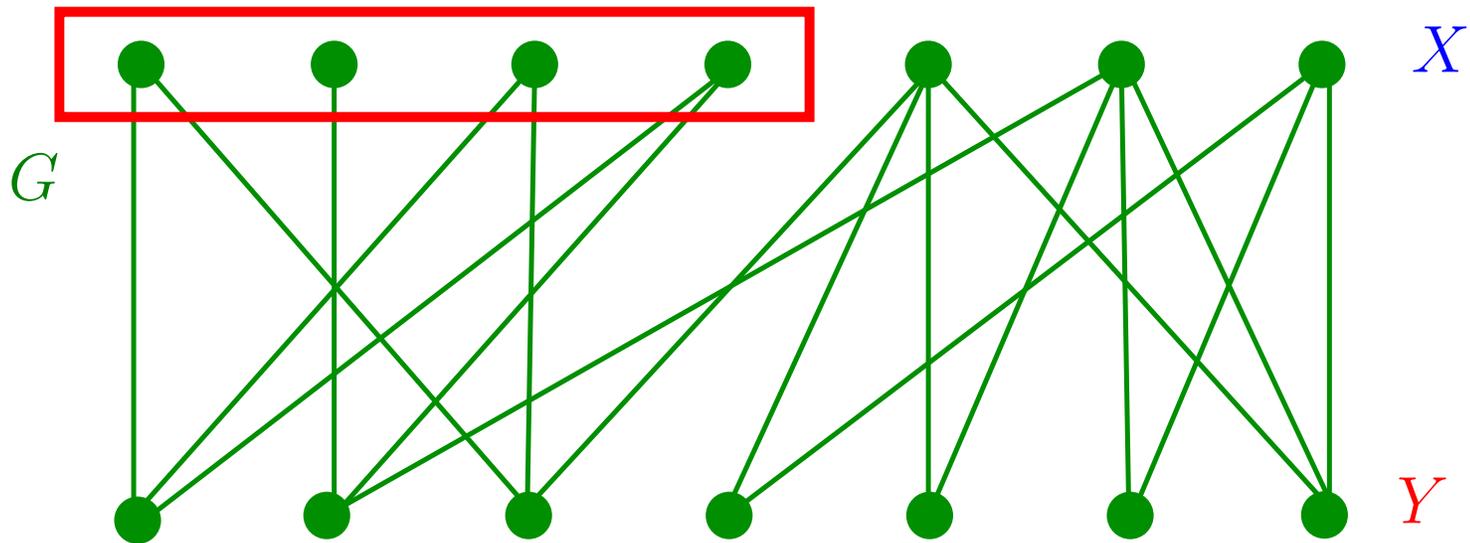
Problema: Dado um grafo bipartido encontrar um emparelhamento perfeito.



NÃO existe! Certificado?

Emparelhamentos

Problema: Dado um grafo bipartido encontrar um emparelhamento perfeito.



NÃO existe! Certificado: $S \subseteq X$ tal que $|S| > |\text{vizinhos}(S)|$.

Teorema de Hall: G tem um emparelhamento perfeito se e somente se

$$|S| \leq |\text{vizinhos}(S)|, \quad \text{para todo } S \subseteq X.$$

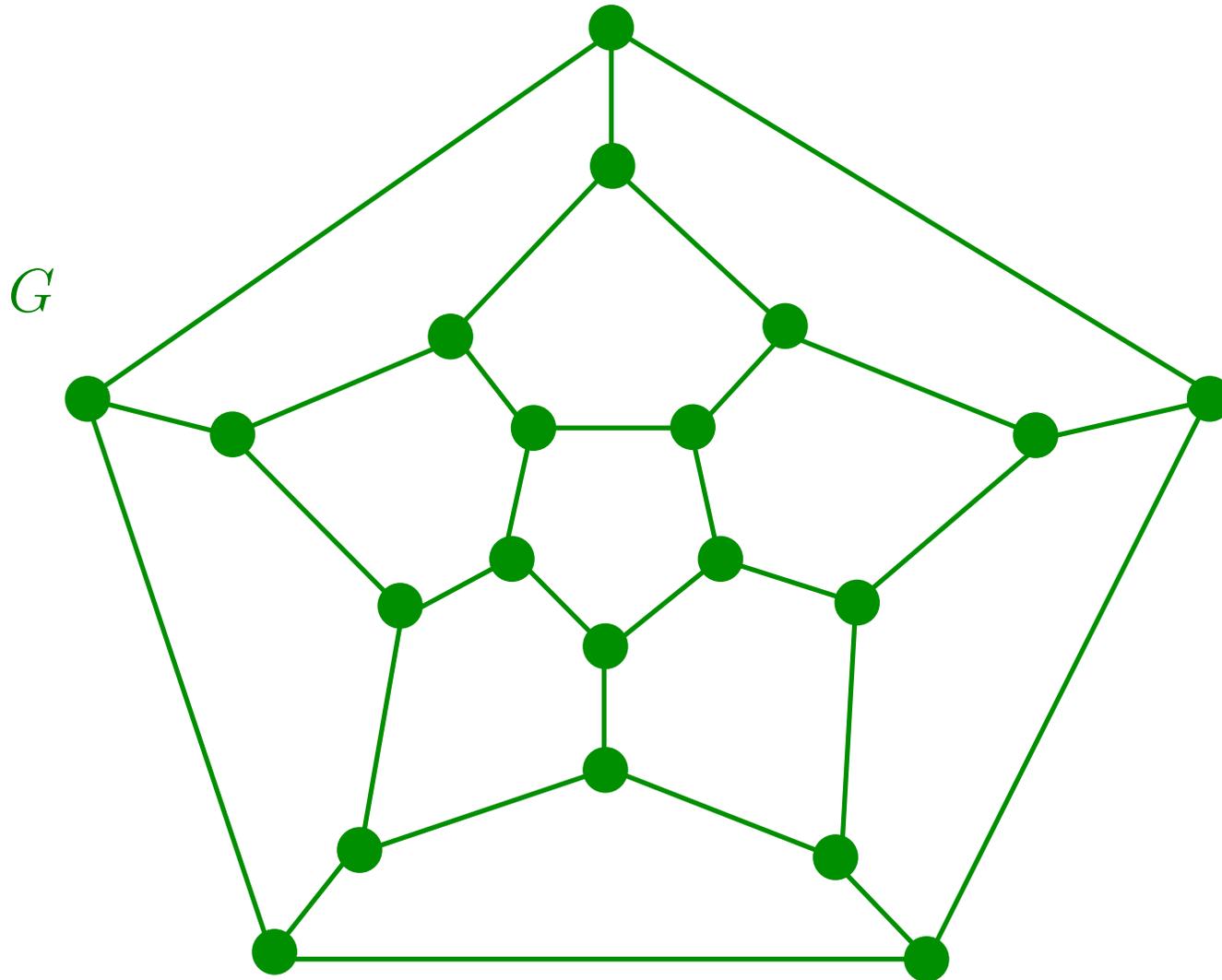
AULA 25

Mais complexidade computacional

CLR 36 ou CLRS 34

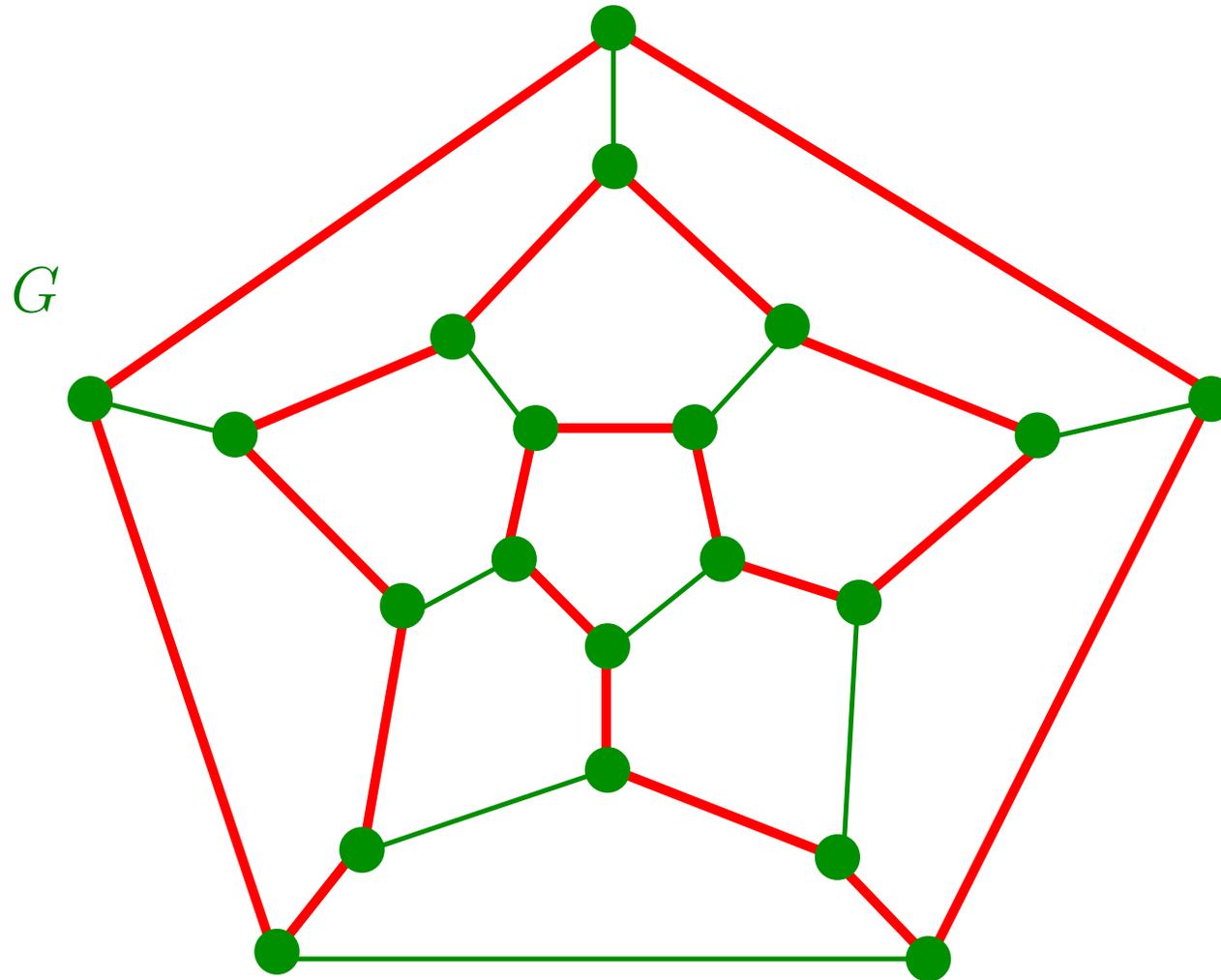
Grafos hamiltonianos

Problema: Dado um grafo encontrar um ciclo hamiltoniano.



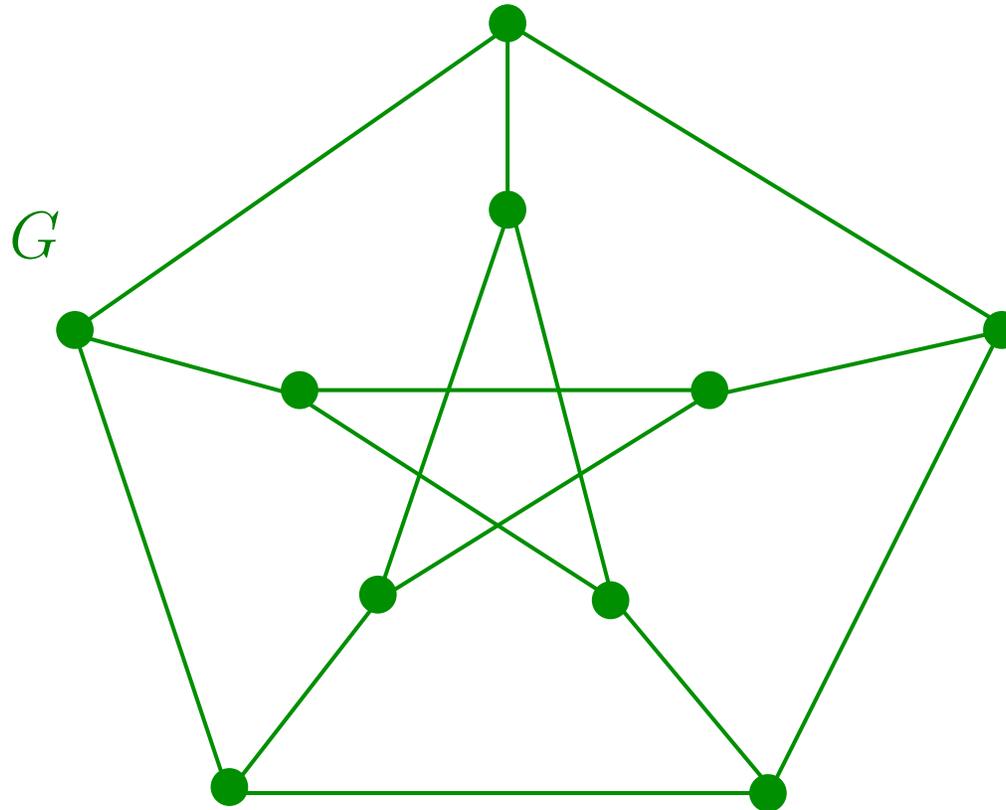
Grafos hamiltonianos

Problema: Dado um grafo encontrar um ciclo hamiltoniano.



Grafos hamiltonianos

Problema: Dado um grafo encontrar um ciclo hamiltoniano.



NÃO existe! Certificado? Hmmm ...

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 $O(\langle I \rangle^c)$ para alguma constante c

e

$\text{ALG}(I, C) = \text{SIM}$ se e somente se a resposta a $\Pi(I)$ é **SIM**.

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

Exemplos

- Se G é hamiltoniano, então um ciclo hamiltoniano de G é um certificado polinomial:
 - dados um grafo G e C pode-se verificar em tempo $O(\langle G \rangle)$ se C é um ciclo hamiltoniano.
- se $X[1..m]$ e $Y[1..n]$ possuem uma sscos $\geq k$, então uma subsequência comum $Z[1..k]$ é um certificado polinomial resposta:
 - dados $X[1..m]$, $Y[1..n]$ e $Z[1..k]$ pode-se verificar em tempo $O(m+n)$ se Z é sscos de X e Y .
- se n é um número composto, então um divisor $d > 1$ de n é um certificado polinomial.

Verificado polinomial para NÃO

Um **verificador polinomial para a resposta NÃO** de 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 $O(\langle I \rangle^c)$ para alguma constante c

e **devolve**

SIM se e somente se a resposta a $\Pi(I)$ é **NÃO**, em caso contrário **ALG** devolve **NÃO**.

O objeto C é dito um **certificado polinomial** ou **certificado curto** da resposta **NÃO** a $\Pi(I)$.

Classe NP

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

Em outras palavras, a classe NP é formada pelos problemas de decisão Π para os quais existe um problema Π' em P e uma função polinomial $p(n)$ tais que, para cada instância I do problema Π , existe um objeto C com $\langle C \rangle \leq p(\langle I \rangle)$ tal que a

resposta a $\Pi(I)$ é SIM se e somente se a resposta a $\Pi'(I, C)$ é SIM.

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

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**.

$$P \subseteq NP$$

Prova:

se Π é um problema em P , então pode-se tomar a seqüência de instruções realizadas por um algoritmo polinomial para resolver $\Pi(I)$ como certificado polinomial da resposta **SIM** a $\Pi(I)$.

Outra prova:

Pode-se construir um verificador polinomial para a resposta **SIM** a Π utilizando-se um algoritmo polinomial para Π como subrotina e ignorando-se o certificado C .

$P \neq NP?$

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 “ $P \neq NP?$ ”

Não confunda NP com “não-polinomial”.

Classe co-NP

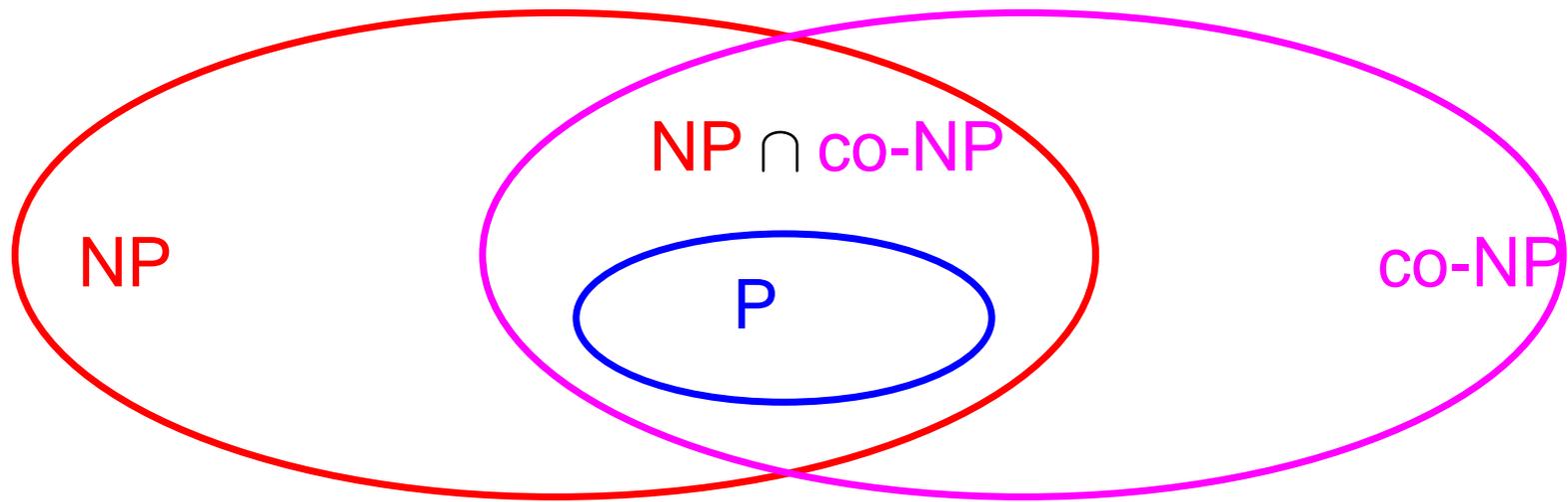
A classe **co-NP** é definida trocando-se **SIM** por **NÃO** na definição de **NP**.

Um problema de decisão 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**.

P, NP e co-NP



$P \neq NP?$

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

$NP \neq co-NP?$

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?

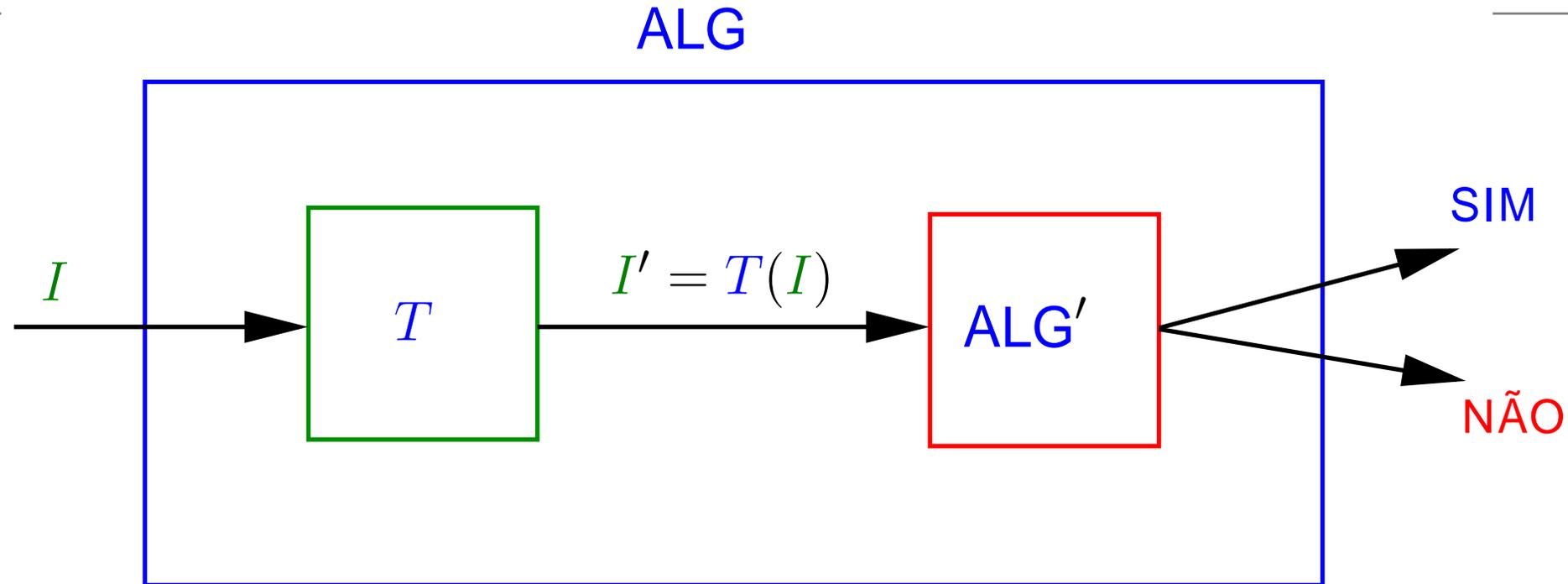
Redução de Π a Π' : ALG' é um algoritmo que resolve Π'

$ALG(G)$

```
1  se  $ALG'(G) = \text{NÃO}$ 
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' .

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, \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 ?

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

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:

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

$$x_1 \geq 1$$

$$1 - x_1 + 1 - x_2 + x_3 \geq 1$$

$$1 - x_3 \geq 0$$

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 $\phi(G)$ tais 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 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).

- “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 construir 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 $t(\phi(G)) = \text{VERDADE}$.

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 π é uma permutação dos vértices de G .
Então

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

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

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

3-Satisfatibilidade

Problema: Dada um 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

Descreveremos um **algoritmo polinomial** T que recebe um 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 $(l_1 \vee l_2 \vee \dots \vee l_k)$ 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 **variáveis 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**

Verifique que ϕ 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 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 .

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.

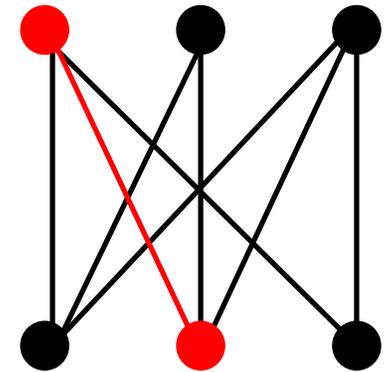
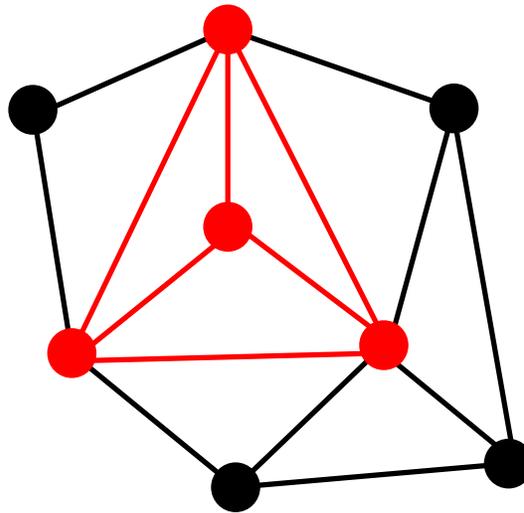
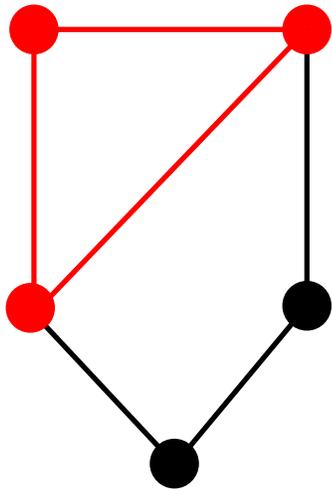
Para isto devemos:

- 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?

Exemplos:



Clique é NP-completo

Clique está em NP e 3-Satisfatibilidade \leq_P Clique.

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

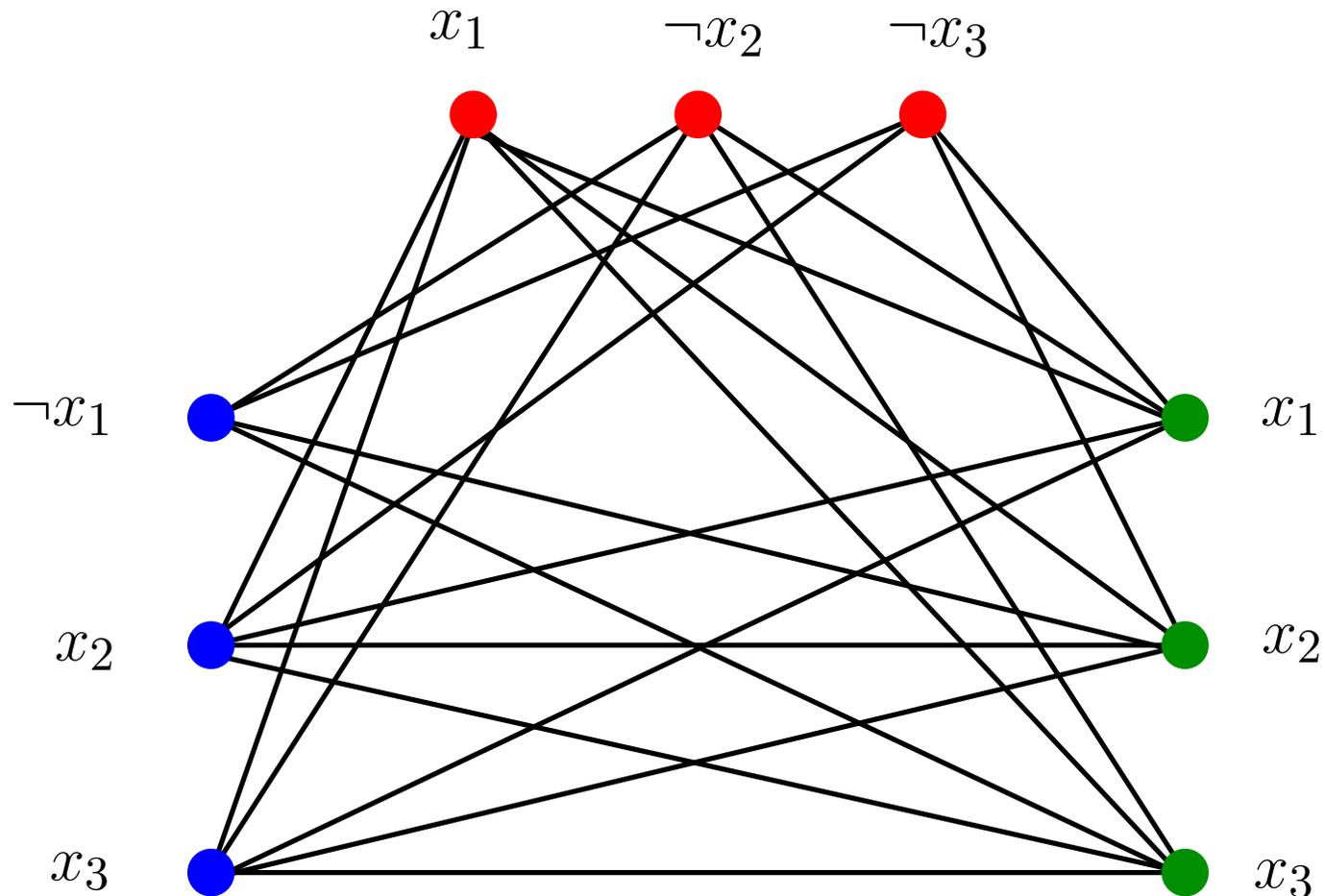
ϕ é satisfatível $\Leftrightarrow G$ possui um clique $\geq k$.

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 correspondes a literais em diferentes cláusula; e
- se u corresponde a um literal x então v não corresponde ao literal $\neg x$.

Clique é **NP**-completo (cont.)

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



Problemas NP-difíceis

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

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 \mathcal{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 $P \subseteq \text{co-NP}$.

Exercício 25.L [CLRS 34.2-10]

Prove que se $\text{NP} \neq \text{co-NP}$, então $P \neq \text{NP}$.

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.

Mais um exercício

Exercício 25.S [CLRS 34.5-7]

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