Complexidade Computacional

Disclaimer

- Slides baseados nos slides dos professores
    - Carla N. Lintzmayer
    - Lehilton Pedrosa

Algoritmo de Tempo Polinomial

Um algoritmo é polinomial se o tempo de execução for limitado por $O(n^k)$ para alguma constante k

- Neste caso, dizemos que ele é um algoritmo eficiente
  - não necessariamente é rápido na prática
    - mas exclui muitos dos algoritmos considerados lentos

Organizando os Problemas

Por que se preocupar com isso?

- desconhecemos algoritmos rápidos para vários problemas
- acreditamos que não há algoritmos eficientes para eles
- queremos saber quais deles têm <u>algoritmos polinomiais</u>

Classes

Dado um problema de decisão L, dizemos que um algoritmo A decide L se

$$A(I) = \text{sim} \iff I \text{ é uma instância sim}$$

Classe P

A classe P é o conjunto dos problemas de decisão que podem ser decididos em tempo polinomial.

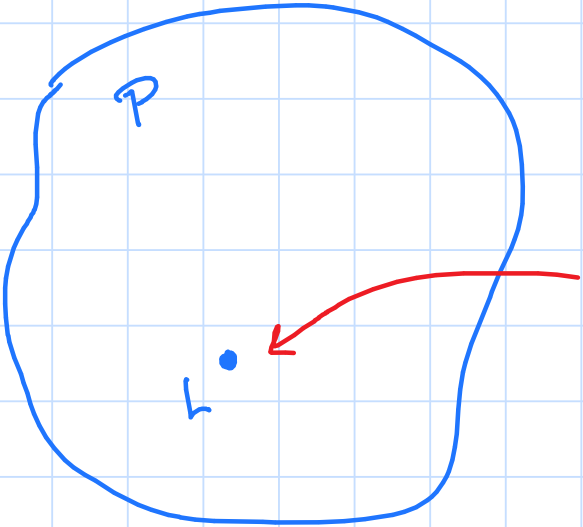
Em outras palavras, se $L \in P$, então

1. Existe algoritmo $A(x)$ que decide L
2. Esse algoritmo executa em tempo polinomial no tamanho da entrada

Como Mostramos que uma Linguagem Pertence à P?

Resolve o problema em $O(n^k)$

- Exibimos um algoritmo que decide ela em tempo polinomial.

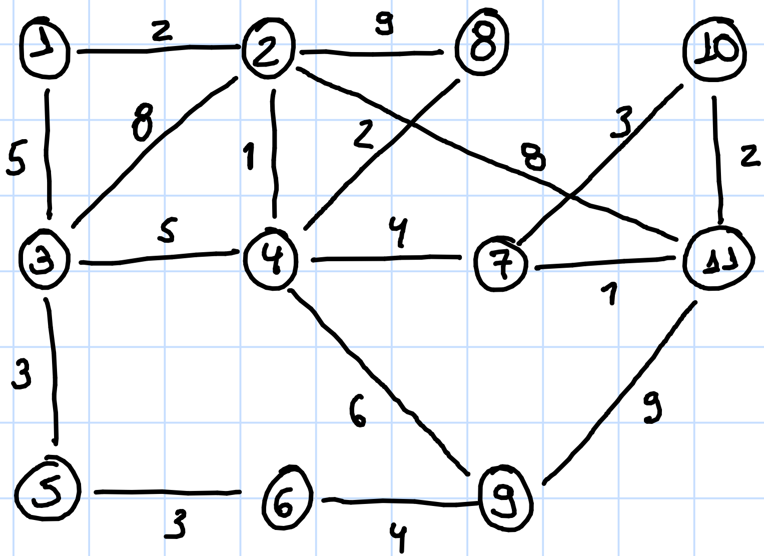


algoritmo Polinomial!

Problema Path

Entrada: um grafo G, dois vértices distintos $s, t \in V(G)$ e um inteiro k

Saída: <u>Sim</u> se $dist(s,t) \leq k$;
<u>não</u> caso contrário



$s = 3$
$t = 10$
$k = 4$

saída: sim

Teo. PATH $\in$ P

Demonstração

Vamos mostrar que o seguinte algoritmo decide PATH em tempo polinomial

```
ALG-PATH( G, u, v, k ) {
    d, pred ← BFS-dist(G, u)
    se d[v] ≤ k
          devolva sim
    senão
          devolva não
}
```

Teo. PATH $\in$ P

Demonstração (continuação)

```
ALG-PATH ( G, u, v, k ) {
 1  d, pred ← BFS-dist(G, u)
 2  se d[v] ≤ k
 3          devolva sim
 4   senão
 5          devolva não
}
```

Primeiro, note que esse algoritmo roda em tempo polinomial. Sabemos que a linha 1 roda em $O(E+V)$ e as outras linhas executam em tempo polinomial. Portanto é um algoritmo polinomial.

Teo. PATH $\in$ P

Demonstração (continuação)

- Agora vamos mostrar que esse algoritmo decide PATH.
- Primeiro suponha que $\langle G, u, v, k\rangle$ é uma instância sim
- Então existe um uv-caminho P tal que $|E(P)| \leq k$.
- Sabemos que $k \geq |E(P)| \geq dist(u,v) = d[v]$. Portanto, o Alg-PATH(G,u,v,k) = sim, o que é o comportamento esperado.

Pela correção de BFS-dist

- Agora suponha que Alg-PATH(G,u,v,k) = sim.
- Como o algoritmo retorna sim, sabemos que o teste da linha 3 deu verdadeiro e $d[v] \leq k$
- Pela correção de BFS-dist, G contém um uv-caminho P tal que $|E(P)| \leq k$.

$\square$

Certificado

Considere um problema L.

- Tome uma instância $\langle x \rangle$ do problema correspondente.
- Queremos encontrar uma estrutura/objeto y tal que y permita verificar facilmente que x é uma instância <u>sim</u> (ou <u>não</u>) do problema L

- Chamamos y de <u>certificado</u> para $\langle x \rangle$.
  - y será certificado <u>positivo</u> se atesta que $\langle x \rangle$ é uma instância sim
  - y será certificado <u>negativo</u> se atesta que $\langle x \rangle$ é uma instância não

- Normalmente um certificado positivo é uma solução da instância.

Certificado para Path

- tome uma instância $I = \langle G, u, v, k \rangle$ de Path
  - Se I é uma instância sim de Path, então existe um uv-caminho P tal que $e(P) \leq k$
  - Se I é uma instância não de Path, então $\overline{\exists}$ existe um uv-caminho P tal que $e(P) \leq k$

- No primeiro caso, podemos usar $y = P$ como um certificado de que I é uma instancia sim.

Verificador

Um (algoritmo) verificador de um problema L é um algoritmo ve recebe uma instância I, um "certificado" C para I e, então,

- devolve sim para algum certificado C se I for uma instância sim
- devolve não se I é uma instância não, independentemente do "certificado" C fornecido.

Verificador

Definição 29.4: ***Algoritmo verificador***

Seja T um problema qualquer. Um algoritmo $\mathcal{A}$ é dito **verificador** se:

de tamanho polinomial em I_T

1. para toda instância I_T que é `sim`, existe um conjunto de dados D tal que $\mathcal{A}(I_T, D)$ devolve `sim`;
2. para toda instância I_T que é `não`, qualquer conjunto de dados D faz $\mathcal{A}(I_T, D)$ devolver `não`;
3. $\mathcal{A}$ executa em tempo polinomial no tamanho de I_T.

O conjunto de dados D que satisfaz o primeiro item é um **certificado positivo**.

```
Verifica-PATH(<G,u,v,k>, <P>)
Para (i=0 ; i< |P| ; i++)
        Se P[i] ∉ V(G) ou
            Retorna não
Para (i=0; i< |P|-1 ; i++)
        Se P[i]P[i+1] ∉ E(G)
            Retorna não
se |P| - 1 ≤ k
        Retorna sim
senão
        Retorna nao
```

Tempo de Verificação

Queremos executar o verificador em tempo polinomial:

Entrada

1. O <u>tempo do verificador</u> deve ser polinomial em $|I|$ e $|C|$ ← certificado
2. O <u>tamanho do certificado</u> $|C|$ deve ser polinomial em $|I|$

- Queremos diferenciar as tarefas de decidir e verificar
- Para certos problemas, decidir uma instância é difícil, mas pode ser que <u>verificar</u> uma solução seja fácil.

Um ciclo Hamiltoniano P de um ~~di~~grafo D é um ciclo gerador de D

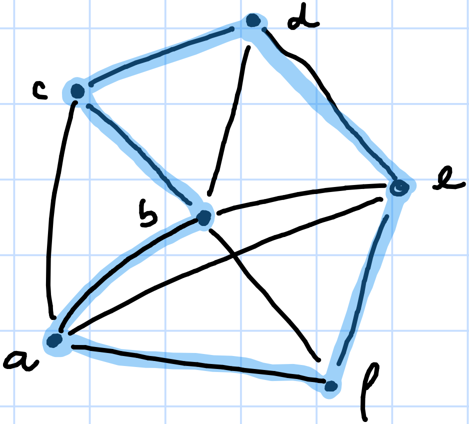
ciclo que contém todos os vértices.

Problema Ciclo Hamiltoniano (CH)

Entrada: Um ~~di~~grafo D

Saída: sim se de contém um ciclo Hamiltoniano

não caso contrário



sim

$C = a, b, c, d, e, f, a$

podemos usar o ciclo hamiltoniano como certificado.

Podemos decidir o problema do ciclo Hamiltoniano:

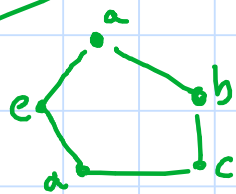
- Algoritmo trivial gasta $O(V!)$
- Os melhores algoritmos têm tempo $O(n^2 2^n)$

Verificar se um ciclo C é hamiltoniano:

- Podemos fazer em tempo linear

Verificando uma Solução

```
Verifica-HAM-ciclo (<G>, <C>)
    Se C não contém todos os vértices de G,
    devolva não
    Para i ← 1 até C.comprimento - 1
        u ← C[i]
        v ← C[i+1]
        Se uv ∉ E(G)
            devolva não
    Devolva sim
```



$C = a, b, c, d, e, a$

1 2 3 4 5 6

Classe NP

A classe NP é o conjunto dos problemas de decisão que podem ser verificados em tempo polinomial, i.e., problemas para os quais existe um algoritmo verificador para instâncias sim que roda em tempo polinomial.

A classe NP contém os problemas de decisão que admitem um certificado para instâncias sim que podem ser verificados de de forma rápida.

Determinando se o Problema é NP

- Dado um problema L, devemos seguir esses passos

1. identifique um certificado de tamanho polinomial para L
2. Construa um algoritmo verificador $A(I, C)$ polinomial

instância de L

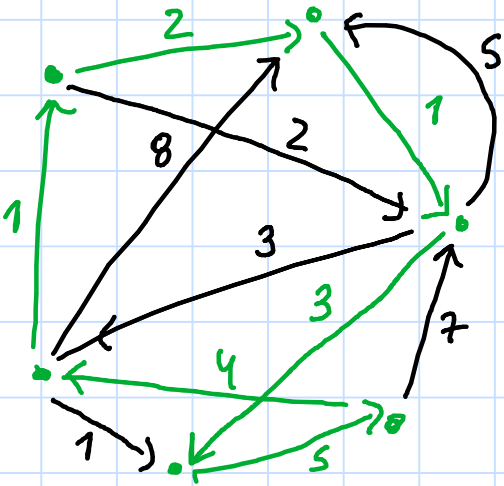
certificado

3. Demonstre que A é um algoritmo verificador

Problema TSP (Popularmente conhecido como o problema do caixeiro viajante)

Entrada: $\langle D, w, k \rangle$, onde D é um digrafo, $w: E(D) \to \mathbb{R}$ e $k \in \mathbb{R}$

Saída:
- sim, se existe um ciclo Hamiltoniano C em D tal que $w(D) \leq k$.
- não, caso contrário.



$K = 22$

$w(C) = 1 + 2 + 1 + 3 + 5 + 4$
$= 16$

Teo. $TSP \in NP$ $\Longrightarrow$ próximo slide

Lema A $\quad TSP \in NP$

Demonstração

Podemos usar como certificado para o TSP a sequência dos vértices de um circuito Hamiltomiano $C = u_1, u_2, u_3, \dots, u_n$ tal que $w(C) \leq K$. Assim, a verificação pode ser feita em tempo polinomial pelo seguinte algoritmo.

```
Verifica-TSP(<D, w, k>, <C>) {
    Se |C| ≠ |V(D)|+1 retorna não
    Se C[1] ≠ C[|C|] ou C[1..|C|] contém um
     vértice que aparece mais de uma vez, retorna
     não.
    z = 0
    para (i = 1 ; i < |C| - 1 ; i++)
        se C[i]C[i+1] ∉ E(D)
            retorna não
        z = z + w(C[i]C[i+1])
    Se z ≤ k , retorna sim
    senão retorna não
}
```

- Agora, vamos mostrar que Verifica-TSP realmente é um algoritmo verificador do problema.
- Primeiro, note que Verifica-TSP roda em tempo polinomial, $O(|C|) = O(V)$ para ser preciso.
- Na linha 1, o algoritmo verifica se o vetor C tem o tamanho adequado para armazenar um ciclo Hamiltoniano, caso não tenha, devolve não.
- Na linha 2, o algoritmo verifica se a sequência de vertices C inicia e termina com o mesmo vértice e se ñ há nenhum outro vertice repetido. Se esse teste falha, então retorna não
- no laço da linha 3, testamos se as arestas $w(C[i]C[i+1])$ existem. Caso alguma ñ exista, retornamos não.
- Se C sobreviveu a todos esses testes, temos que C

é um ciclo hamiltoniano de D

- O valor de $w(C)$ foi acumulado na variável z. Na sequência, testamos se $z \leq k$. Se for retornamos sim; caso contrário, não.

- Portanto Verifica-TSP($\langle D, w, k \rangle, \langle C \rangle$) devolve sim apenas qndo $C \subseteq D$ é um ciclo Hamiltoniano de D $\square$

$P \subseteq NP$

Teo $P \subseteq NP$

Demonstração

- Seja $L \in P$. Então existe um algoritmo A polinomial que decide L
- Vamos construir um verificador para L

  VerificaL(x, y)
  devolva $A(x)$

- Se x é instância sim, seja $y = \varepsilon$, e note que VerificaL(x,y) = sim, já que A decide L
- Se VerificaL(x,y) = sim, então $A(x)$ = sim e, portanto, x é uma instância sim.
- Note que Verifical(x,y) é polinomial pq $A(x)$ é!

P vs NP

Teo $P \subseteq NP$

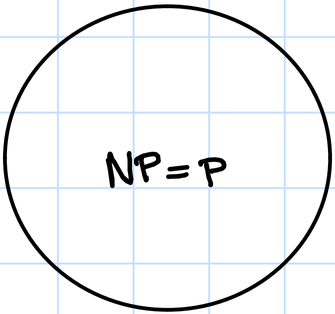
Não sabemos se $NP \subseteq P$, i.e., se $P = NP$

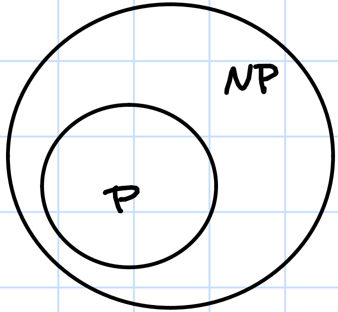
- i.e. não sabemos se os problemas que são fáceis de decidir são os problemas que são fáceis de verificar.

- Existe um prêmio de 10^6 dólares para quem resolver esse problema.

Classes de Complexidade

Possíveis configurações



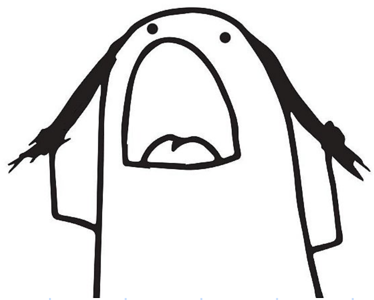


Classe P : polynomial time

Classe NP : non-polynomial time

Classe P : polynomial time

Classe NP : ~~non-polynomial time~~

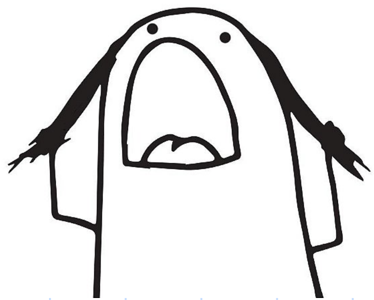


Classe P : polynomial time

Classe NP : ~~non-polynomial time~~

non-deterministic polynomial time

↳ Roda em uma máquina de Turing ñ determinística em tempo polinomial.



Teo Considere os Problemas de decisão L_1 e L_2 tais que $L_1 \leq_p L_2$ poli
Se $L_2 \in P$, então $L_1 \in P$.

Demonstração

- Suponha que $L_2 \in P$ e seja ALG-L_2 um algoritmo que decide L_2 em tempo polinomial
- Seja f uma redução de L_1 para L_2 em tempo polinomial
- Seja ALG-$L_1(x)$ = ALG-$L_2(f(x))$
- Como f é uma redução temos que x é uma instância sim de L_1 se e somente se $f(x)$ é uma inst. sim de L_2

ALG-L_1(x)
  $z \leftarrow f(x)$] Poli
  Devolva ALG-L_2(z)] Poli

- Portanto, ALG-L_1 decide L_1 em tempo polinomial. $\square$

Classe NP-difícil

n̄ precisam ser de decisão

A classe NP-difícil é o conjunto de problemas L tais que $L' \leq_p L$ para todo $L' \in NP$.

A classe NP-completo é o conjunto dos problemas L tais que $L \in NP$ e $L \in$ NP-difícil.

então L é de decisão

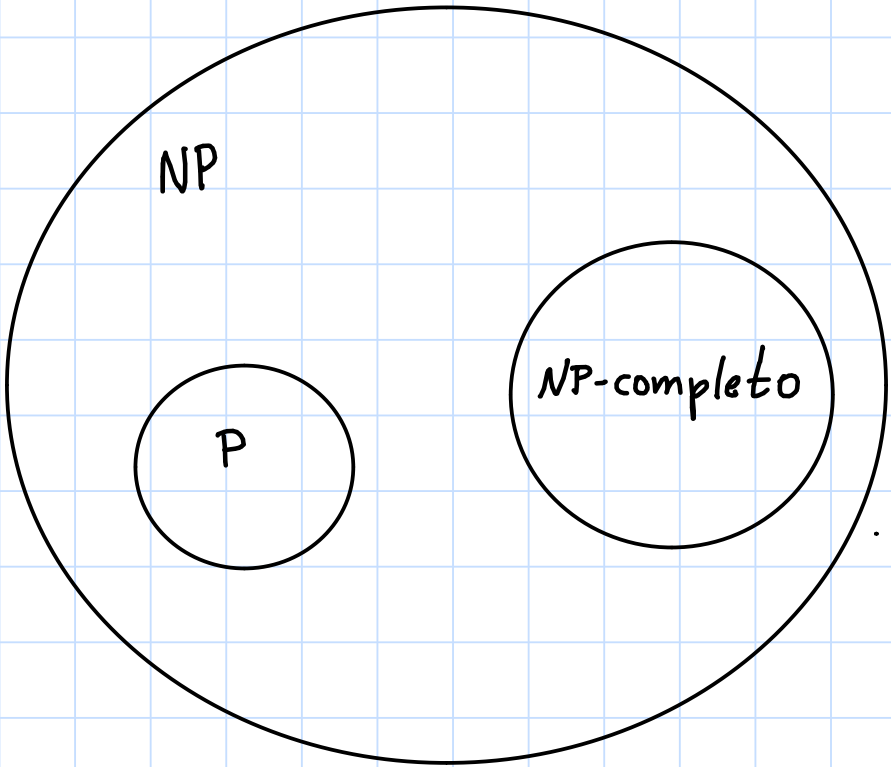
Teo. Se existe um algoritmo que decide $L \in NP$-completo em tempo polinomial, então $P = NP$.

Demonstração

- Suponha que existe $L \in P \cap NP$-completo
- Como $L \in NP$-completo, para toda linguagem $L' \in NP$, temos que $L' \leq_P L$
- Como $L \in P$, temos que $L' \in P$ pelo teorema anterior.
- Logo $NP \subseteq P$ e, portanto, $NP = P$. $\square$

Cor. Se existe um problema $L \in NP$ tal que $L \notin P$, então $NP\text{-completo} \cap P = \emptyset$.

Como acreditamos que é



Por que acreditamos que não há algoritmo rápido para problemas NP-difíceis?

- Demonstrou-se que um problema é NP-completo pela primeira vez na década de 1970 (Cook-Levin)
- bem antes, vários desses problemas já eram estudados
- Desde então, mostrou-se que inúmeros problemas importantes também são NP-completos ou NP-difíceis
- Vários pesquisadores estudaram seus problemas NP-completos preferidos, mas ninguém descobriu qualquer algoritmo polinomial.
- basta que um algoritmo NP-difícil tenha algoritmo de tempo polinomial para que <u>todos</u> os problemas em NP tenham algoritmos de tempo polinomial.

O que fazer se o problema for NP-difícil?

Se soubermos que o problema é NP-difícil, podemos concentrar nossos esforços em busca de

- Algoritmo para instâncias pequenas
  - backtracking
  - enumeração
  - Programação Semidefinida
  - Programação Linear Inteira
  - Algoritmos Parametrizados
  - Programação por Restrição
- Soluções aproximadas
  - Heurísticas
  - MetaHeurísticas
  - Algoritmos de Aproximação

    $OPT(I) \leq A(I) \leq 2\,OPT(I)$
- Algoritmos eficientes e exatos, para casos particulares do problema.

A classe NP-Completo não é vazia

- Será que realmente existe um problema NP-completo?
  - Cook e Levin responderam que sim (independentemente)
    - Eles mostraram que SAT $\in$ NP-Completo.

Teo de Cook-Levin

SAT é NP-completo

Satisfabilidade

Considere uma fórmula booleana

- Contém um conjunto de variáveis booleanas
- é escrita usando os seguintes operadores:
  1. negação ($\neg$)
  2. Conjunção ($\wedge$)
  3. Disjunção ($\vee$)
  4. Implicação ($\rightarrow$)
  5. Equivalencia ($\leftrightarrow$)

Problema da Satisfabilidade (SAT)

Entrada: uma fórmula booleana

Saída: sim, se existe uma atribuição de valores lógicos às variáveis da fórmula tal que a fórmula dê verdadeiro.
não, caso contrário.

Fórmula Satisfazível

Uma fórmula é satisfazível se houver atribuição das variáveis para a qual a avaliação é verdadeira.

Exemplo

$$f = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$$

NÃO   sim

atribuição: $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$

$$f = ((0 \rightarrow 0) \vee \neg((\neg 0 \leftrightarrow 1) \vee 1)) \wedge \neg 0$$

$$= (1 \vee \neg((1 \leftrightarrow 1) \vee 1)) \wedge 1$$

$$= 1 \wedge 1$$

$$= 1$$

Como Mostrar que um Problema é NP-difícil?

temos duas possibilidades para mostrar que um problema Q é NP-difícil:

1. Mostrar que $L \leq_p Q$ para todo $L \in NP$

2. Mostrar que $L \leq_p Q$ para algum $L \in$ NP-Difícil.

Normalmente usamos a segunda opção

Como Mostrar que um Problema é NP-Completo?

Para provar que um problema Q é NP-Completo, fazemos:

1. Provamos que $Q \in NP$.
2. Provamos que $Q \in$ NP-difícil.

Teo Considere o problema Q e seja $L \in$ NP-difícil. Se $L \leqslant_p Q$, então $Q \in$ NP-difícil.

Demonstração

Como L é NP-difícil, para todo $L' \in NP$, vale que $L' \leqslant_p L$

Assim $L' \leqslant_p L$ e $L \leqslant_p Q$, o que implica $L' \leqslant_p Q$

Portanto $Q \in$ NP-difícil $\square$

Além disso, se $Q \in NP$, então Q é NP-Completo.

Fórmula 3-CNF

Dizemos que uma fórmula booleana ϕ sobre um conj. de variáveis $V = \{x_1, x_2, \dots, x_n\}$ é 3-CNF se

$$\phi = C_1 \wedge C_2 \wedge C_3 \wedge \cdots \wedge C_m,$$

onde $C_i = (l_1^i \vee l_2^i \vee l_3^i)$, para todo $i = 1, 2, \dots, m$, e l_j^i é um literal tal que $l_j^i = x$ ou $l_j^i = \neg x$, onde $x \in V$.

→ chamamos isso de cláusula

Exemplo

$$\phi = (x_1 \vee \overline{x_2} \vee x_3) \wedge \underbrace{(\overline{x_1} \vee x_2 \vee x_4)}_{\text{cláusula}} \wedge (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_3})$$

literais

Problema 3-SAT

Entrada: ϕ, onde ϕ é uma fórmula 3-CNF contendo literais de variáveis booleanas $x_1, x_2, \dots, x_n$.

Saída: Sim existe uma atribuição de valores $x_1, x_2, \dots x_n$ tal que ϕ seja satisfatível. não caso contrário.

Teo 3-SAT é NP-completo.

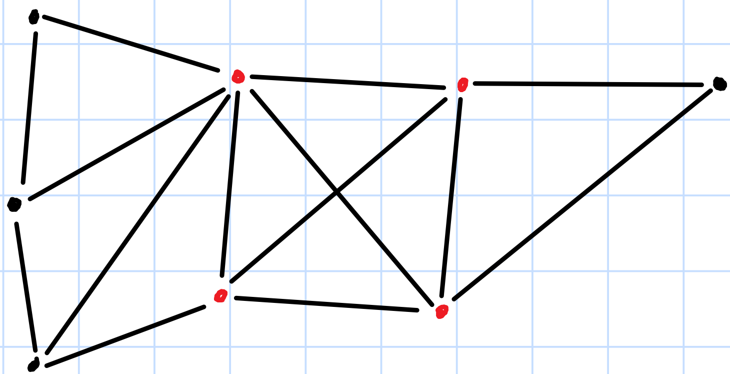
Problema Clique

Entrada: ⟨G, k⟩, onde G é um grafo e k é um inteiro positivo

Saída: sim se existe uma clique $S \subseteq V(G)$ tal que $|S| \geq k$ e não, caso contrário.

conj. de vertices tal que para todo $u, v \in S$, temos $uv \in E(G)$

$k = 3$



Teo Clique é NP-completo

Demonstração

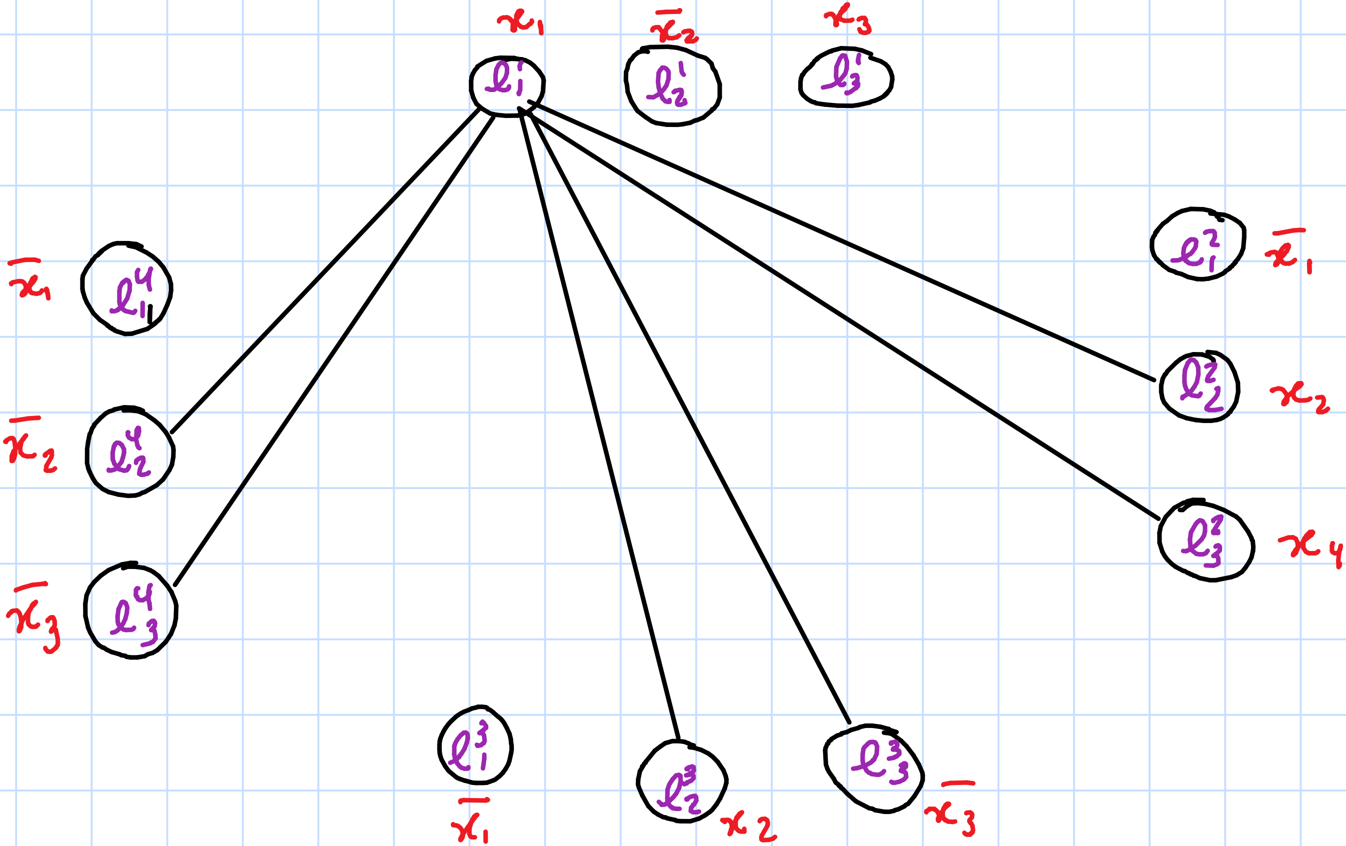
- Primeiro mostraremos que CLIQUE está em NP e depois faremos uma redução polinomial de 3-SAT para CLIQUE

- Clique está em NP pois, dados $\langle G, k \rangle$ e um certificado $\langle S \rangle$, onde $S \subseteq V(G)$ é uma clique tal que $|S| \geq k$, é fácil escrever um algoritmo para verificar se S é realmente tal clique: basta verificar se há uma aresta entre todos os pares de vértices em S, que pode ser feito em $O(V^2 \cdot E)$, e contar o número de elementos de S, que pode ser feito em $O(V)$.

Quando for provar que um problema está em NP no meio da prova de NP-completo, você pode ser menos detalhista, para a prova não ficar muito longa

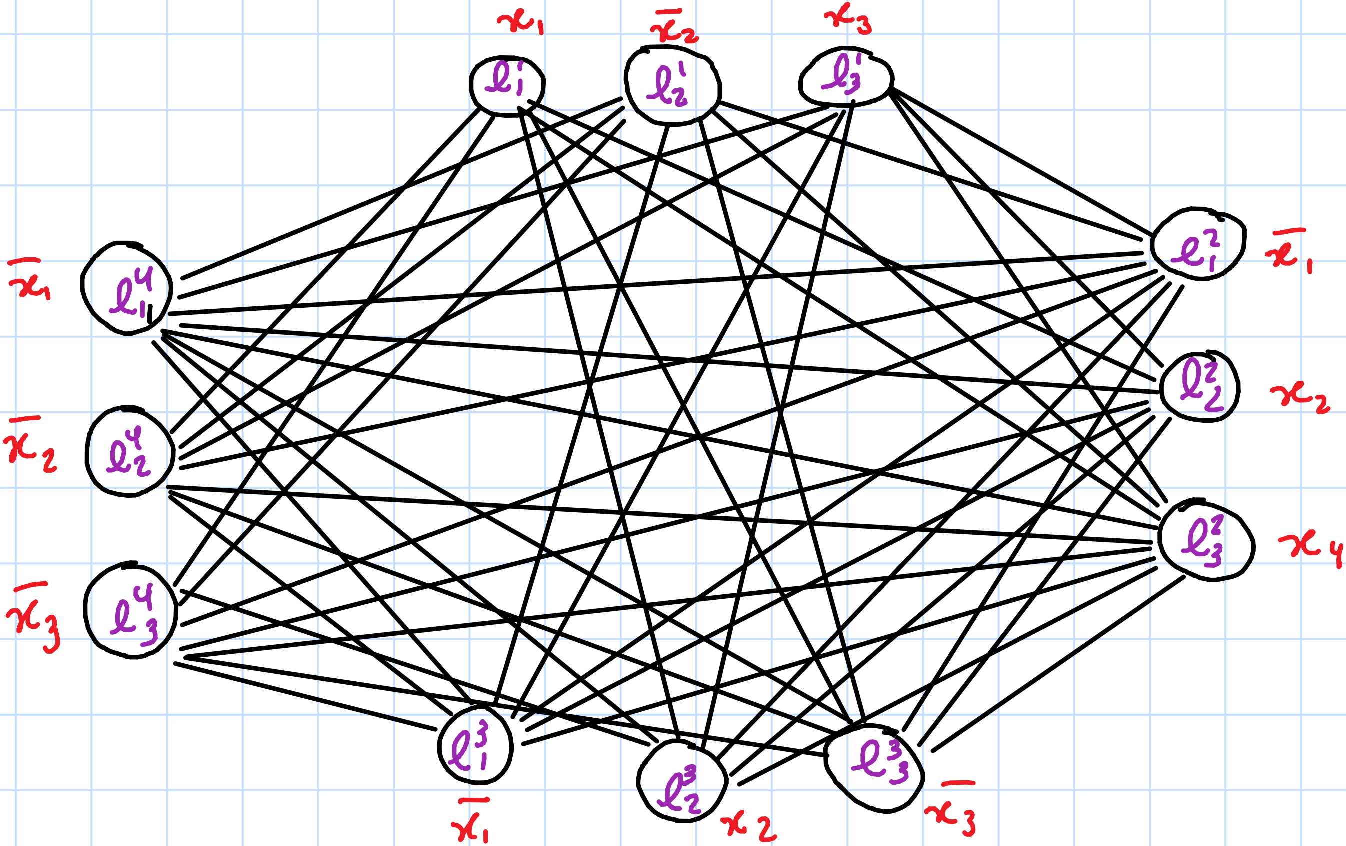
Dado $\phi = (\underbrace{\overset{\ell_1^1}{x_1} \vee \overset{\ell_2^1}{\overline{x_2}} \vee \overset{\ell_3^1}{x_3}}_{c_1}) \wedge (\underbrace{\overset{\ell_1^2}{\overline{x_1}} \vee \overset{\ell_2^2}{x_2} \vee \overset{\ell_3^2}{x_4}}_{c_2}) \wedge (\underbrace{\overset{\ell_1^3}{\overline{x_1}} \vee \overset{\ell_2^3}{x_2} \vee \overset{\ell_3^3}{\overline{x_3}}}_{c_3}) \wedge (\underbrace{\overset{\ell_1^4}{\overline{x_1}} \vee \overset{\ell_2^4}{\overline{x_2}} \vee \overset{\ell_3^4}{\overline{x_3}}}_{c_4})$

construa



Dado $$\phi = (\underbrace{\overset{\ell_1^1}{x_1} \vee \overset{\ell_2^1}{\overline{x_2}} \vee \overset{\ell_3^1}{x_3}}_{c_1}) \wedge (\underbrace{\overset{\ell_1^2}{\overline{x_1}} \vee \overset{\ell_2^2}{x_2} \vee \overset{\ell_3^2}{x_4}}_{c_2}) \wedge (\underbrace{\overset{\ell_1^3}{\overline{x_1}} \vee \overset{\ell_2^3}{x_2} \vee \overset{\ell_3^3}{\overline{x_3}}}_{c_3}) \wedge (\underbrace{\overset{\ell_1^4}{\overline{x_1}} \vee \overset{\ell_2^4}{\overline{x_2}} \vee \overset{\ell_3^4}{\overline{x_3}}}_{c_4})$$

construa



Teo Clique é NP-completo

Demonstração (continuação)

- Agora considere uma fórmula 3-CNF ϕ com m cláusulas e conjunto de variáveis $V = \{x_1, x_2, \ldots, x_n\}$, ou seja, $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, onde $C_i = (l_1^i \vee l_2^i \vee l_3^i)$ para $i = 1, \ldots, m$ e l_j^i é uma variável de V ou a negação de uma variável de V para todo $i = 1, \ldots, m$ e $j = 1, 2, 3$.

- Vamos construir um grafo $G(\phi)$ com $3m$ vértices: para cada cláusula adicionamos 3 vértices, um para cada literal. Assim

$$V(G(\phi)) = \bigcup_{i=1}^{m} \{l_1^i, l_2^i, l_3^i\}$$

Teo Clique é NP-completo

Demonstração (continuação)

- Sejam ℓ^i_j e ℓ^k_p dois vértices de $G(\phi)$. Vamos adicionar uma aresta entre esses dois vértices em $G(\phi)$ se $i \neq k$ e ℓ^i_j não for a negação do literal ℓ^k_p (ou seja, se (a) $\ell^i_j = \ell^i_p$ ou (b) $\ell^i_j = x$ e $\ell^k_p = \bar{x}$, então não colocamos a aresta).
- Nossa redução será

$f(\langle\phi\rangle)$ {

Seja m o número de literais em ϕ

Devolve $\langle G(\phi), m\rangle$

}

Teo Clique é NP-completo

Demonstração (continuação)

- Não é difícil perceber que f executa em tempo polinomial.
- Resta verificar se $\langle \phi \rangle$ é uma instância sim do 3-SAT sse $f(\langle \phi \rangle)$ for uma instância sim de Clique
- Suponha que $\langle \phi \rangle$ é uma instância sim de 3-SAT

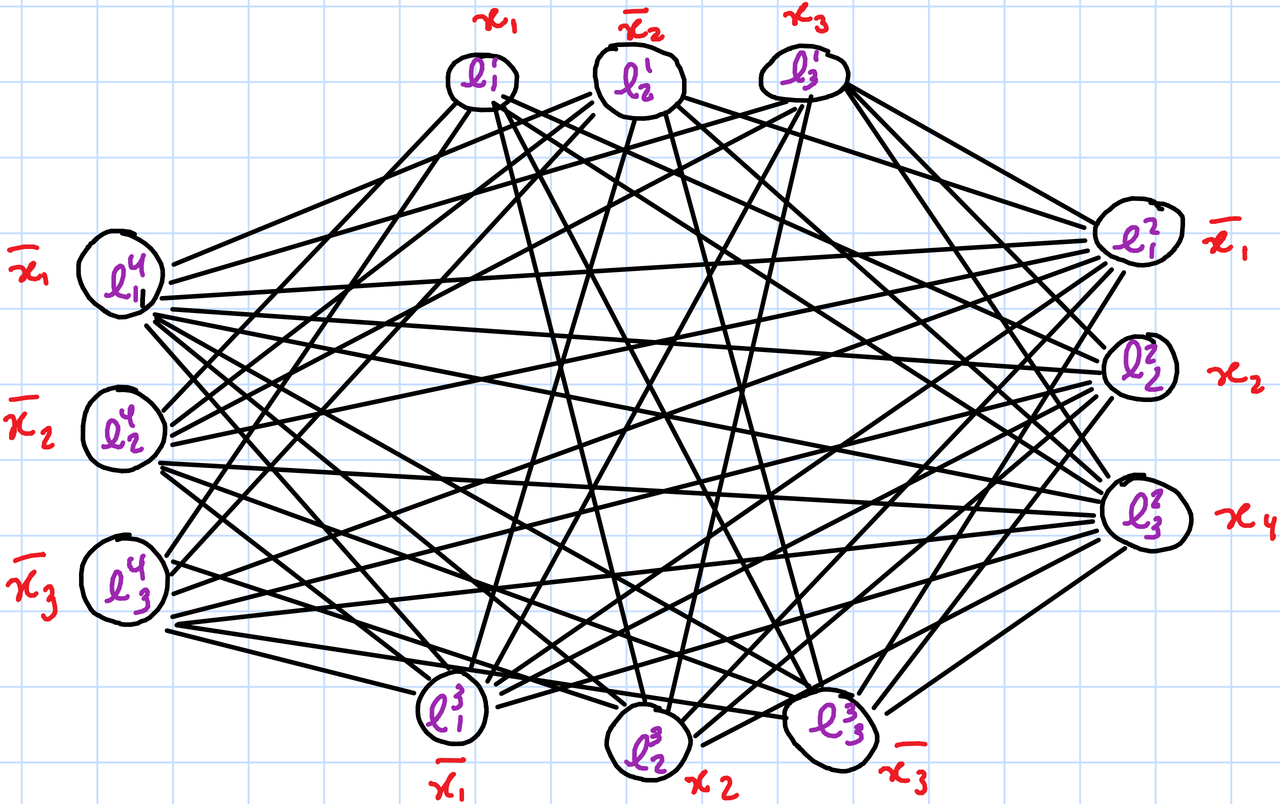
$$\phi = \overbrace{(\underset{1}{\overset{l_1^1}{x_1}} \vee \underset{0}{\overset{l_2^1}{\overline{x_2}}} \vee \underset{0}{\overset{l_3^1}{x_3}})}^{C_1} \wedge \overbrace{(\underset{0}{\overset{l_1^2}{\overline{x_1}}} \vee \underset{1}{\overset{l_2^2}{x_2}} \vee \underset{1}{\overset{l_3^2}{x_4}})}^{C_2} \wedge \overbrace{(\underset{0}{\overset{l_1^3}{\overline{x_1}}} \vee \underset{0}{\overset{l_2^3}{x_2}} \vee \underset{1}{\overset{l_3^3}{\overline{x_3}}})}^{C_3} \wedge \overbrace{(\underset{0}{\overset{l_1^4}{\overline{x_1}}} \vee \underset{0}{\overset{l_2^4}{\overline{x_2}}} \vee \underset{1}{\overset{l_3^4}{\overline{x_3}}})}^{C_4} = 1$$

$x_1 = 1$

$x_2 = 1$

$x_3 = 0$

$x_4 = 1$



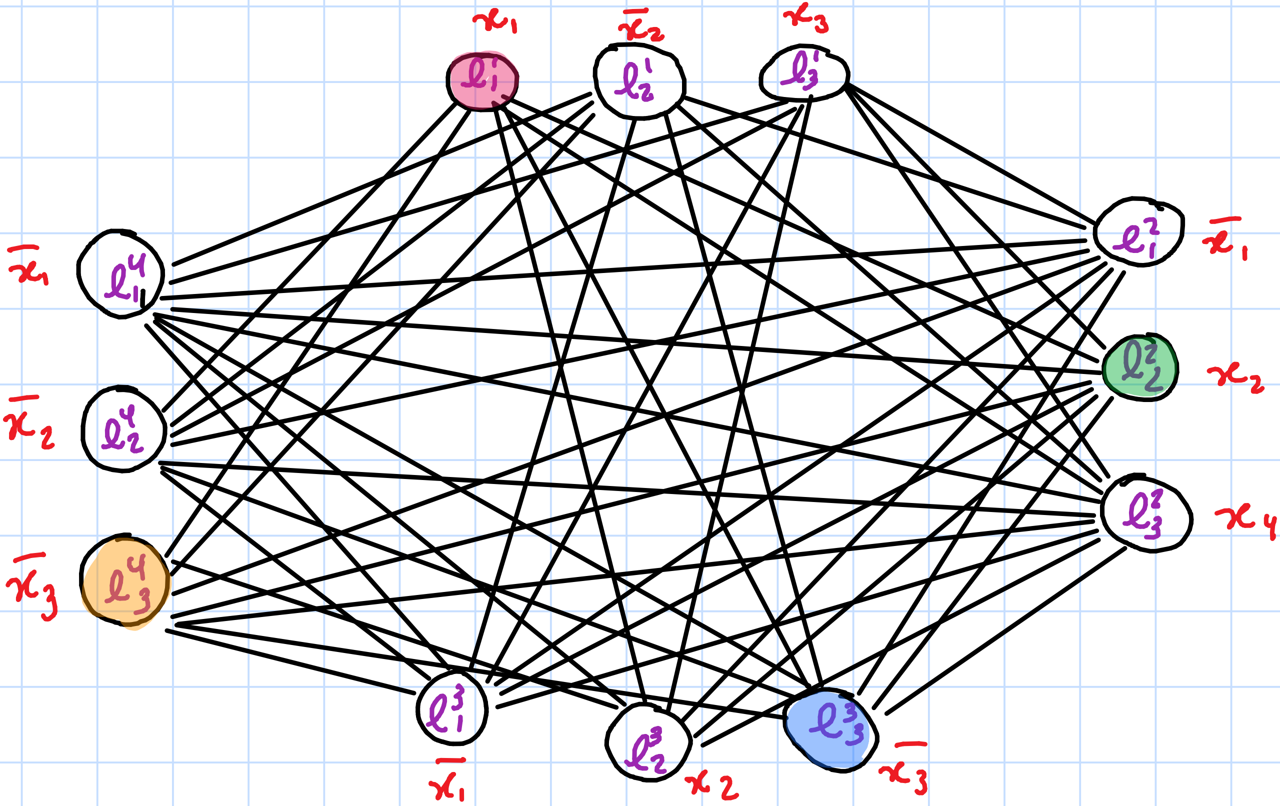
$$\phi = \overbrace{(\underset{1}{\overset{\ell_1^1}{x_1}} \vee \underset{0}{\overset{\ell_2^1}{\overline{x_2}}} \vee \underset{0}{\overset{\ell_3^1}{x_3}})}^{C_1} \wedge \overbrace{(\underset{0}{\overset{\ell_1^2}{\overline{x_1}}} \vee \underset{1}{\overset{\ell_2^2}{x_2}} \vee \underset{1}{\overset{\ell_3^2}{x_4}})}^{C_2} \wedge \overbrace{(\underset{0}{\overset{\ell_1^3}{\overline{x_1}}} \vee \underset{0}{\overset{\ell_2^3}{x_2}} \vee \underset{1}{\overset{\ell_3^3}{\overline{x_3}}})}^{C_3} \wedge \overbrace{(\underset{0}{\overset{\ell_1^4}{\overline{x_1}}} \vee \underset{0}{\overset{\ell_2^4}{\overline{x_2}}} \vee \underset{1}{\overset{\ell_3^4}{\overline{x_3}}})}^{C_4} = 1$$

$x_1 = 1$

$x_2 = 1$

$x_3 = 0$

$x_4 = 1$



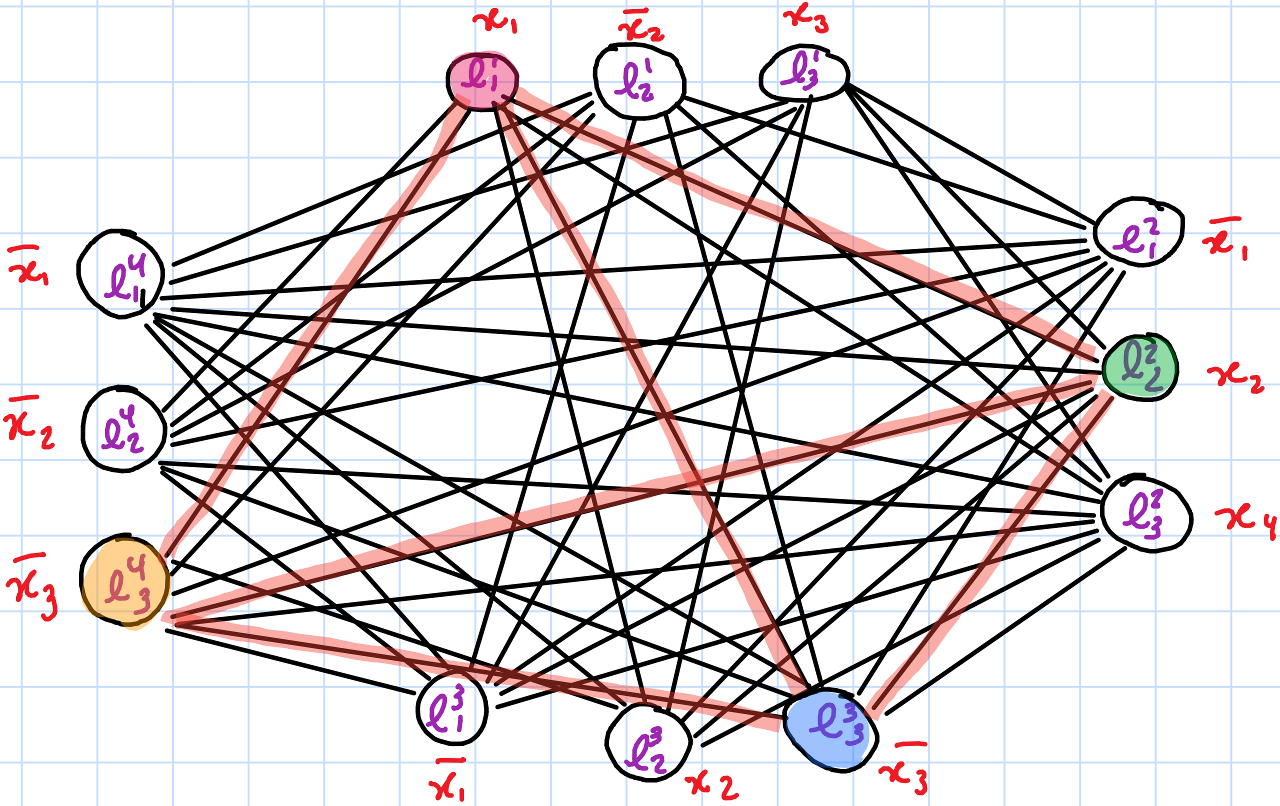
$$\phi = \overbrace{(\underset{1}{\overset{\ell_1^1}{x_1}} \vee \underset{0}{\overset{\ell_2^1}{\overline{x_2}}} \vee \underset{0}{\overset{\ell_3^1}{x_3}})}^{C_1} \wedge \overbrace{(\underset{0}{\overset{\ell_1^2}{\overline{x_1}}} \vee \underset{1}{\overset{\ell_2^2}{x_2}} \vee \underset{1}{\overset{\ell_3^2}{x_4}})}^{C_2} \wedge \overbrace{(\underset{0}{\overset{\ell_1^3}{\overline{x_1}}} \vee \underset{0}{\overset{\ell_2^3}{x_2}} \vee \underset{1}{\overset{\ell_3^3}{\overline{x_3}}})}^{C_3} \wedge \overbrace{(\underset{0}{\overset{\ell_1^4}{\overline{x_1}}} \vee \underset{0}{\overset{\ell_2^4}{\overline{x_2}}} \vee \underset{1}{\overset{\ell_3^4}{\overline{x_3}}})}^{C_4} = 1$$

$x_1 = 1$

$x_2 = 1$

$x_3 = 0$

$x_4 = 1$



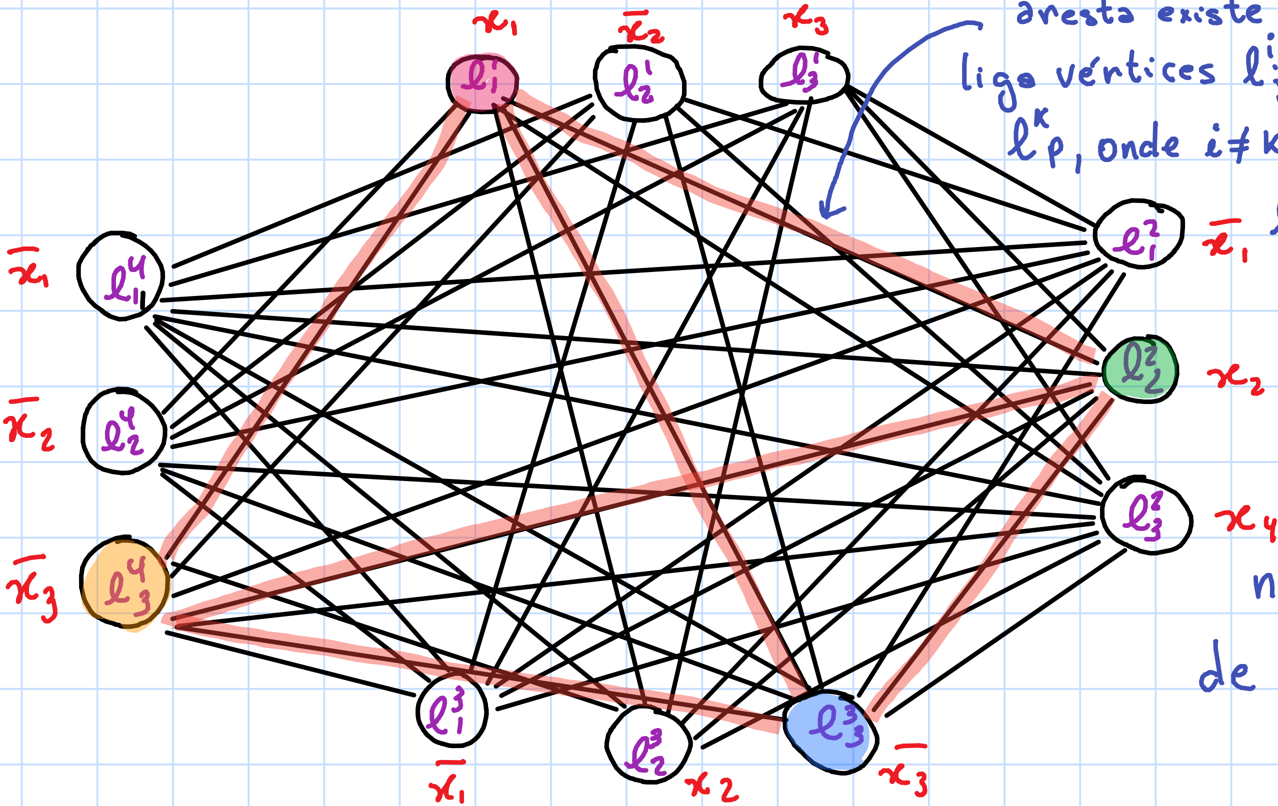
$$\phi = \overbrace{(\underset{1}{\overset{\ell^1_1}{x_1}} \vee \underset{0}{\overset{\ell^1_2}{\overline{x_2}}} \vee \underset{0}{\overset{\ell^1_3}{x_3}})}^{C_1} \wedge \overbrace{(\underset{0}{\overset{\ell^2_1}{\overline{x_1}}} \vee \underset{1}{\overset{\ell^2_2}{x_2}} \vee \underset{1}{\overset{\ell^2_3}{x_4}})}^{C_2} \wedge \overbrace{(\underset{0}{\overset{\ell^3_1}{\overline{x_1}}} \vee \underset{0}{\overset{\ell^3_2}{x_2}} \vee \underset{1}{\overset{\ell^3_3}{\overline{x_3}}})}^{C_3} \wedge \overbrace{(\underset{0}{\overset{\ell^4_1}{\overline{x_1}}} \vee \underset{0}{\overset{\ell^4_2}{\overline{x_2}}} \vee \underset{1}{\overset{\ell^4_3}{\overline{x_3}}})}^{C_4} = 1$$

$x_1 = 1$

$x_2 = 1$

$x_3 = 0$

$x_4 = 1$



aresta existe pois liga vértices ℓ^i_j e ℓ^k_p, onde $i \neq k$ e ℓ^i_j não é a negação de ℓ^k_p.

Teo Clique é NP-completo

Demonstração (continuação)

- Seja $\psi : V \to \{0,1\}$ uma atribuição de valores lógicos à ϕ de forma que ϕ seja satisfazível
- Então, ao menos um literal em cada cláusula C_i deve ter avaliado para 1. Para cada cláusula C_i, seja t_i um literal dessa cláusula que tenha avaliado para 1.
- Seja $S = \{t_1, t_2, \dots, t_m\}$
- Note que $|S| = m$. Agora, vamos argumentar que S é uma clique em $G(\phi)$
- Como todos os elementos de S são de cláusulas distintas,

Teo Clique é NP-completo

Demonstração (continuação)

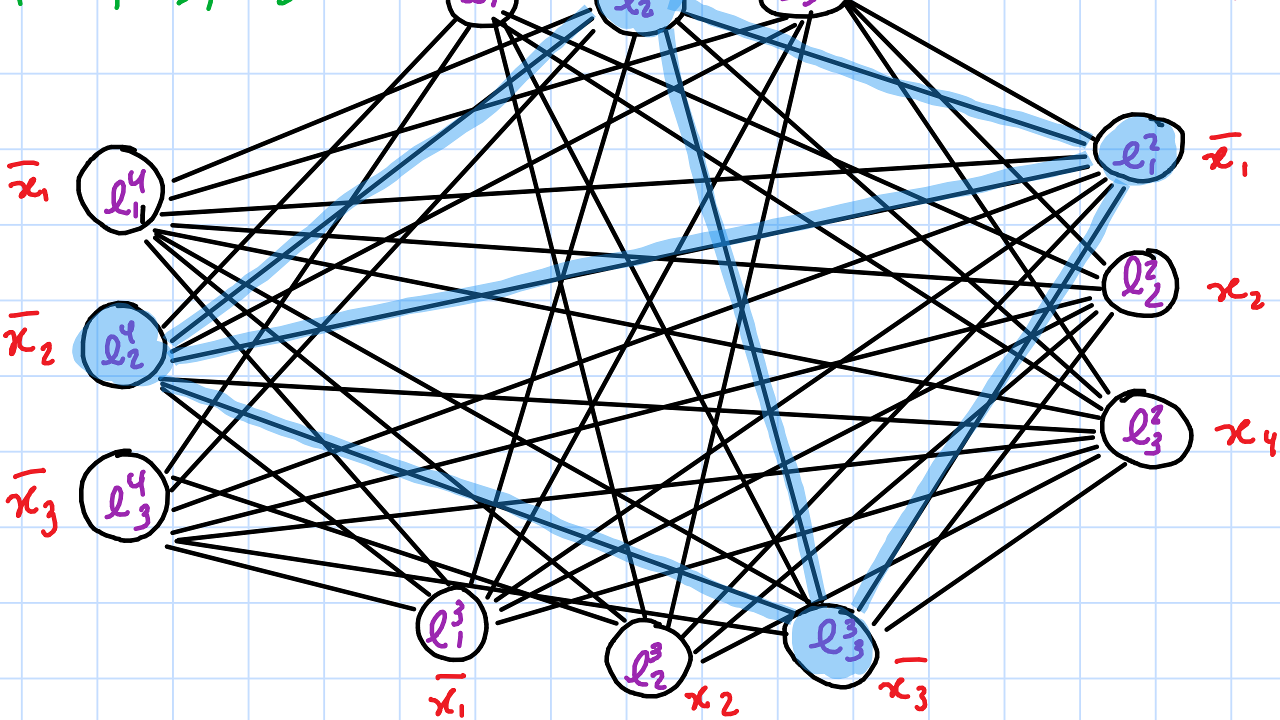
pela construção de $G(\phi)$, a única possibilidade para não haver uma aresta entre dois vértices $t_i, t_j \in S$ seria se $t_i = x$ e $t_j = \bar{x}$, mas neste caso, não seria possível que esses dois literais tivessem avaliado para 1. Assim, S é uma clique em $G(\phi)$ e $f(\langle\phi\rangle) = \langle G(\phi), m\rangle$ é uma instância sim de clique.

- Agora, suponha que $f(\langle\phi\rangle) = \langle G(\phi), m\rangle$ é uma instância sim de clique

$$\phi = \overbrace{(\overset{l_1^1}{x_1} \vee \overset{l_2^1}{\overline{x_2}} \vee \overset{l_3^1}{x_3})}^{C_1} \wedge \overbrace{(\overset{l_1^2}{\overline{x_1}} \vee \overset{l_2^2}{x_2} \vee \overset{l_3^2}{x_4})}^{C_2} \wedge \overbrace{(\overset{l_1^3}{\overline{x_1}} \vee \overset{l_2^3}{x_2} \vee \overset{l_3^3}{\overline{x_3}})}^{C_3} \wedge \overbrace{(\overset{l_1^4}{\overline{x_1}} \vee \overset{l_2^4}{\overline{x_2}} \vee \overset{l_3^4}{\overline{x_3}})}^{C_4}$$

$S = \{ l_2^1, l_1^2, l_3^3, l_2^4 \}$

$\langle G(\phi), \overset{m}{4} \rangle$

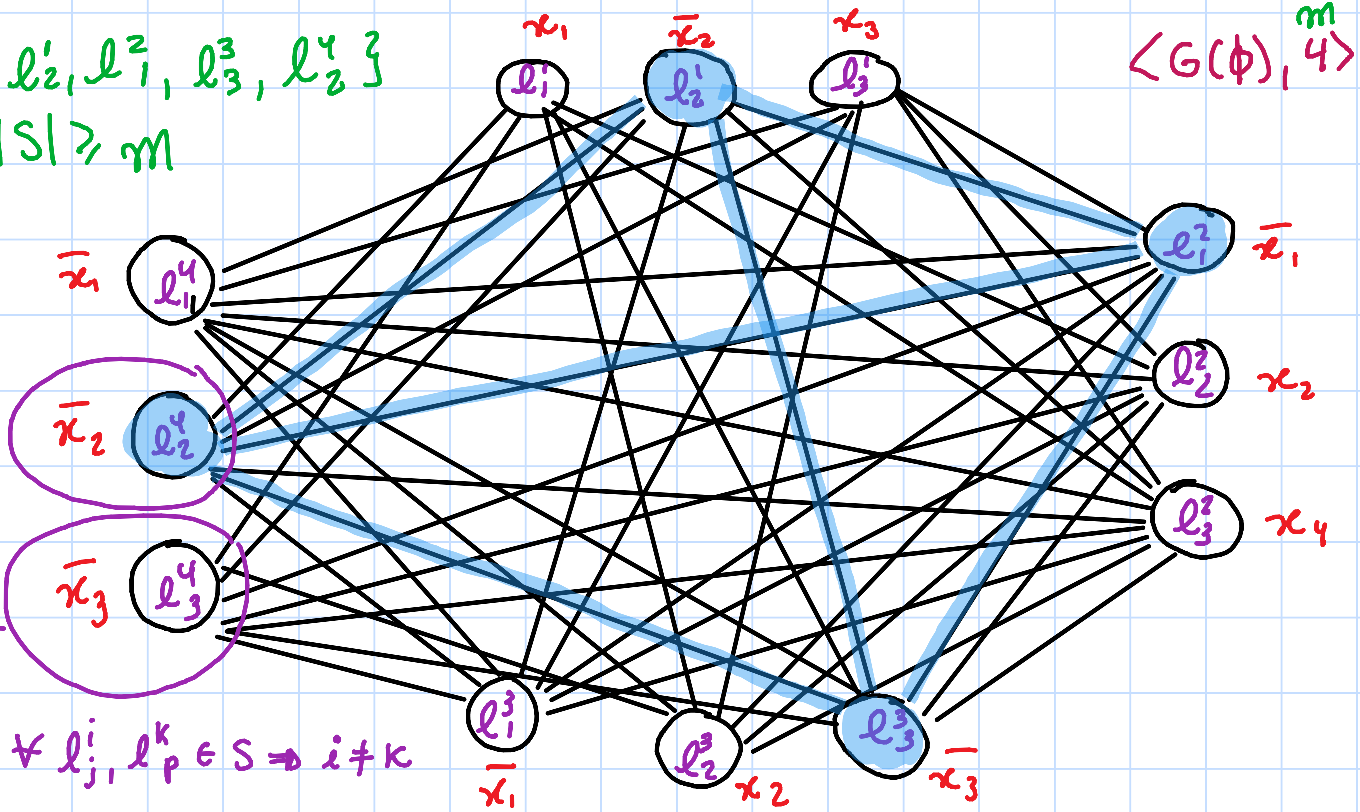


$$\phi = \overbrace{(\overset{l^1_1}{x_1} \lor \overset{l^1_2}{\overline{x_2}} \lor \overset{l^1_3}{x_3})}^{C_1} \land \overbrace{(\overset{l^2_1}{\overline{x_1}} \lor \overset{l^2_2}{x_2} \lor \overset{l^2_3}{x_4})}^{C_2} \land \overbrace{(\overset{l^3_1}{\overline{x_1}} \lor \overset{l^3_2}{x_2} \lor \overset{l^3_3}{\overline{x_3}})}^{C_3} \land \overbrace{(\overset{l^4_1}{\overline{x_1}} \lor \overset{l^4_2}{\overline{x_2}} \lor \overset{l^4_3}{\overline{x_3}})}^{C_4}$$

$S = \{ l^1_2, l^2_1, l^3_3, l^4_2 \}$

$|S| \geq m$

$\langle G(\phi), \overset{m}{4} \rangle$



$\forall\, l^i_j, l^k_p \in S \Rightarrow i \neq k$

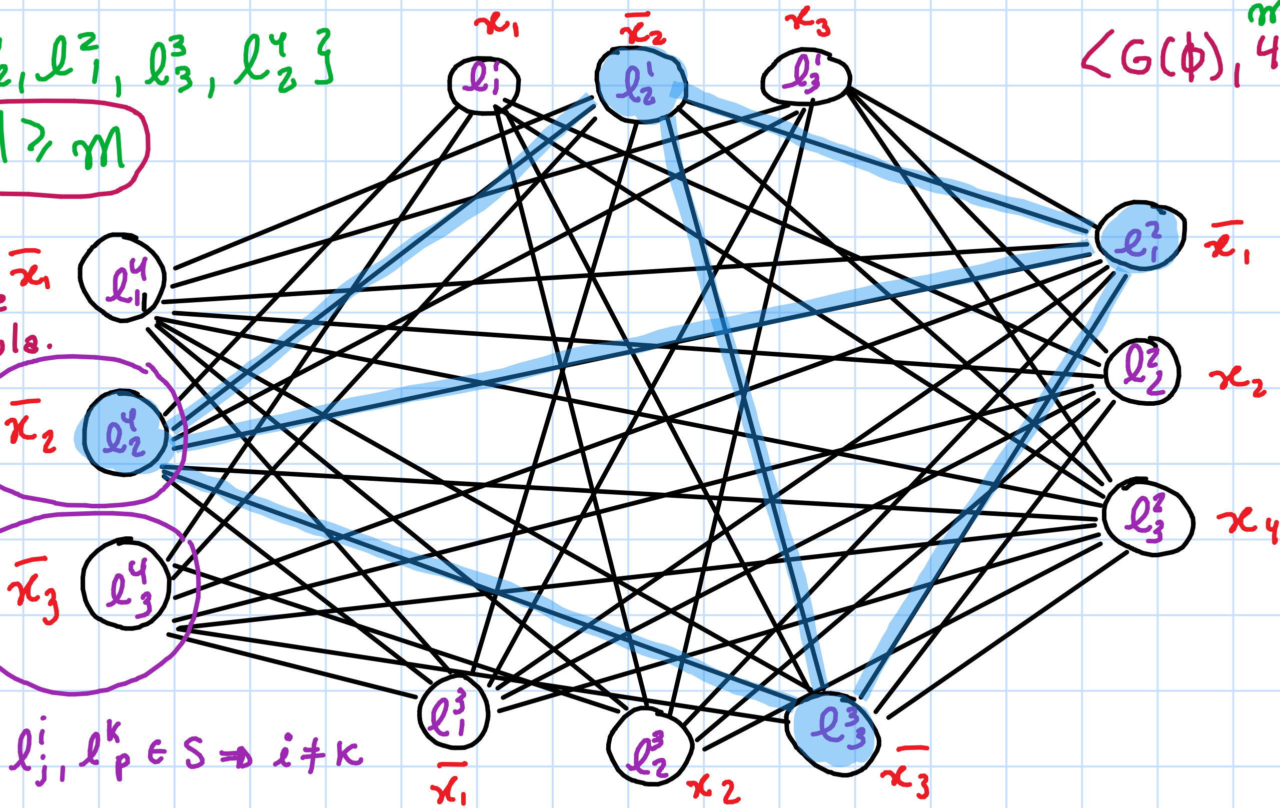
$$\phi = \overbrace{(\overset{\ell_1^1}{x_1} \vee \overset{\ell_2^1}{\overline{x_2}} \vee \overset{\ell_3^1}{x_3})}^{C_1} \wedge \overbrace{(\overset{\ell_1^2}{\overline{x_1}} \vee \overset{\ell_2^2}{x_2} \vee \overset{\ell_3^2}{x_4})}^{C_2} \wedge \overbrace{(\overset{\ell_1^3}{\overline{x_1}} \vee \overset{\ell_2^3}{x_2} \vee \overset{\ell_3^3}{\overline{x_3}})}^{C_3} \wedge \overbrace{(\overset{\ell_1^4}{\overline{x_1}} \vee \overset{\ell_2^4}{\overline{x_2}} \vee \overset{\ell_3^4}{\overline{x_3}})}^{C_4}$$

$S = \{\ell_2^1, \ell_1^2, \ell_3^3, \ell_2^4\}$

$|S| \geq m$

Pegamos um vértice por cláusula.

$\langle G(\phi), \overset{m}{4} \rangle$



$\forall\ \ell_j^i, \ell_p^k \in S \Rightarrow i \neq k$

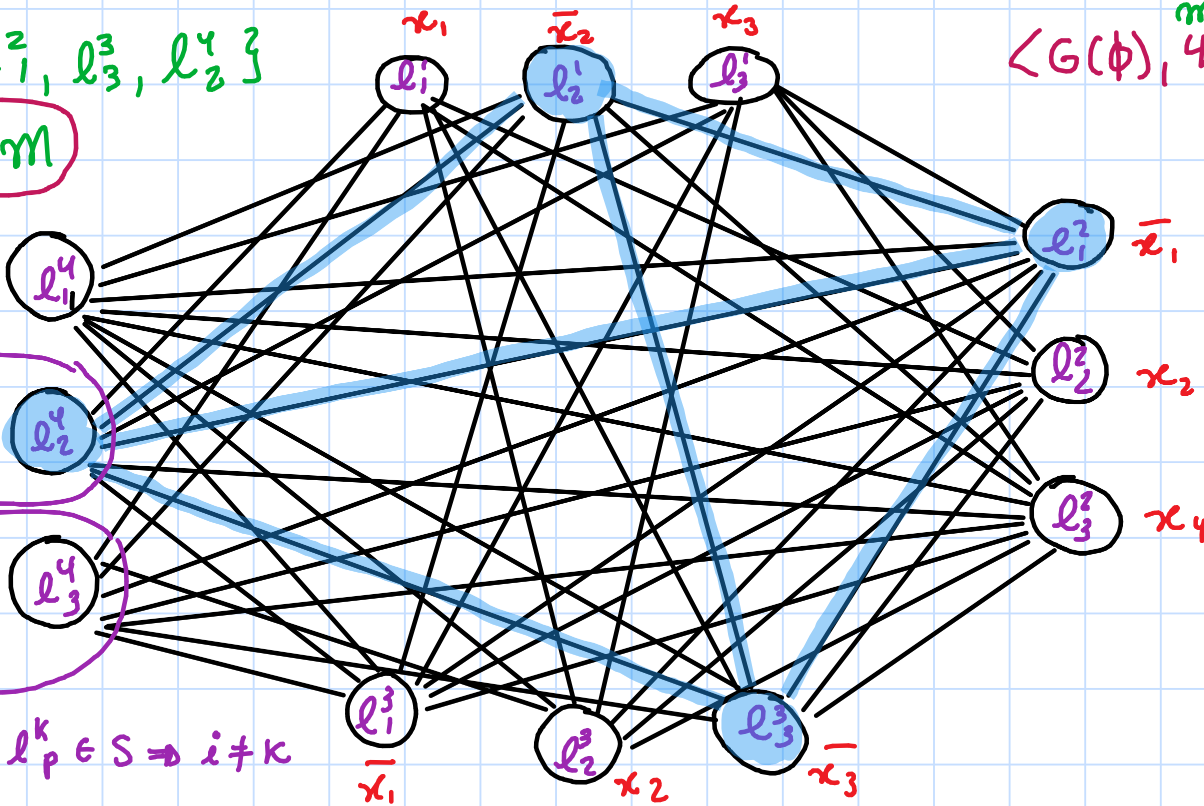
$$\phi = \overbrace{(\overset{l_1^1}{x_1} \vee \underset{1}{\overset{l_2^1}{\overline{x_2}}} \vee \overset{l_3^1}{x_3})}^{C_1} \wedge \overbrace{(\underset{1}{\overset{l_1^2}{\overline{x_1}}} \vee \overset{l_2^2}{x_2} \vee \overset{l_3^2}{x_4})}^{C_2} \wedge \overbrace{(\overset{l_1^3}{\overline{x_1}} \vee \overset{l_2^3}{x_2} \vee \underset{1}{\overset{l_3^3}{\overline{x_3}}})}^{C_3} \wedge \overbrace{(\overset{l_1^4}{\overline{x_1}} \vee \underset{1}{\overset{l_2^4}{\overline{x_2}}} \vee \overset{l_3^4}{\overline{x_3}})}^{C_4}$$

$S = \{ l_2^1, l_1^2, l_3^3, l_2^4 \}$

$|S| \geq m$

Pegamos um vértice por cláusula.

$\langle G(\phi), \overset{m}{4} \rangle$



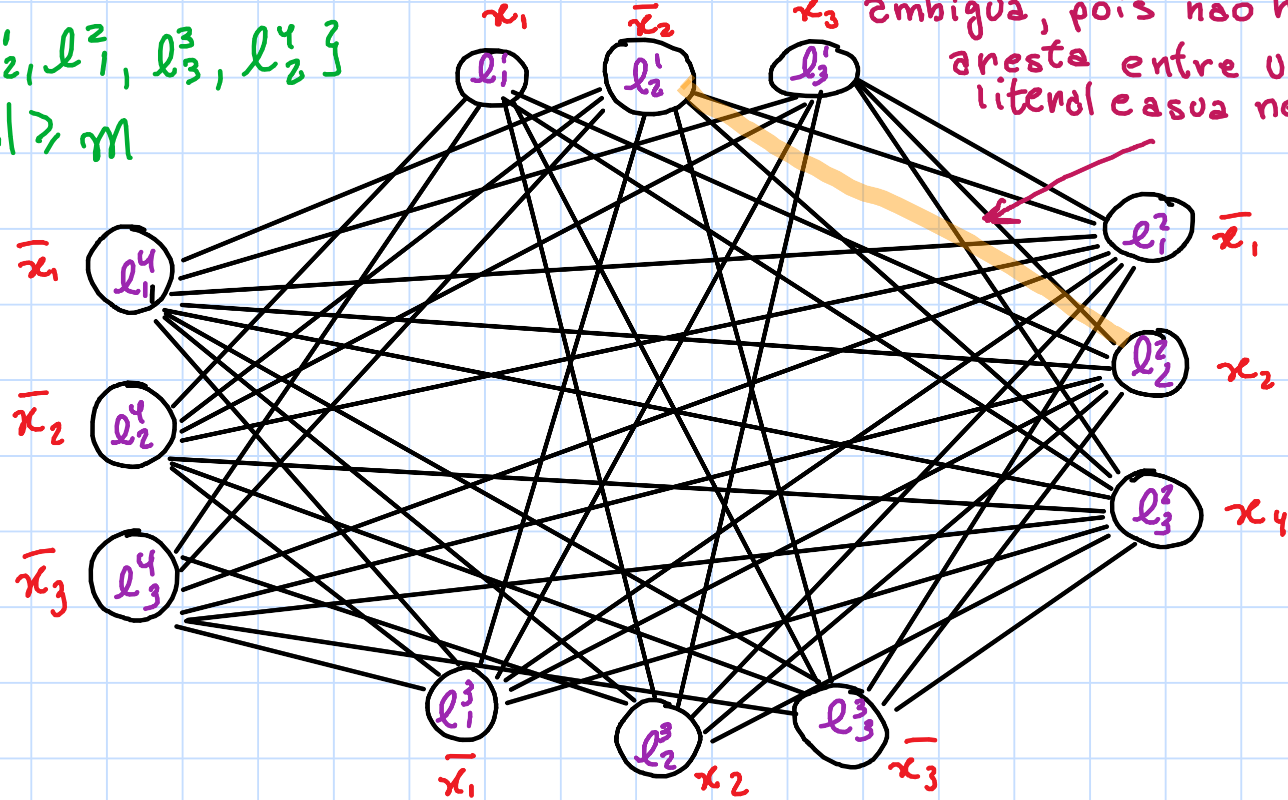
$\forall\, l_j^i, l_p^k \in S \Rightarrow i \neq k$

$$\phi = \overbrace{(\overset{\ell^1_1}{x_1} \vee \overset{\ell^1_2}{\underset{1}{\overline{x_2}}} \vee \overset{\ell^1_3}{x_3})}^{C_1} \wedge \overbrace{(\overset{\ell^2_1}{\overline{x_1}} \vee \overset{\ell^2_2}{\underset{1}{x_2}} \vee \overset{\ell^2_3}{x_4})}^{C_2} \wedge \overbrace{(\overset{\ell^3_1}{\overline{x_1}} \vee \overset{\ell^3_2}{x_2} \vee \overset{\ell^3_3}{\overline{x_3}})}^{C_3} \wedge \overbrace{(\overset{\ell^4_1}{\overline{x_1}} \vee \overset{\ell^4_2}{\overline{x_2}} \vee \overset{\ell^4_3}{\overline{x_3}})}^{C_4}$$

← não há risco de fazer uma atribuição ambígua, pois não há aresta entre um literal e a sua neg.

$S = \{\ell^1_2, \ell^2_1, \ell^3_3, \ell^4_2\}$

$|S| \geq m$

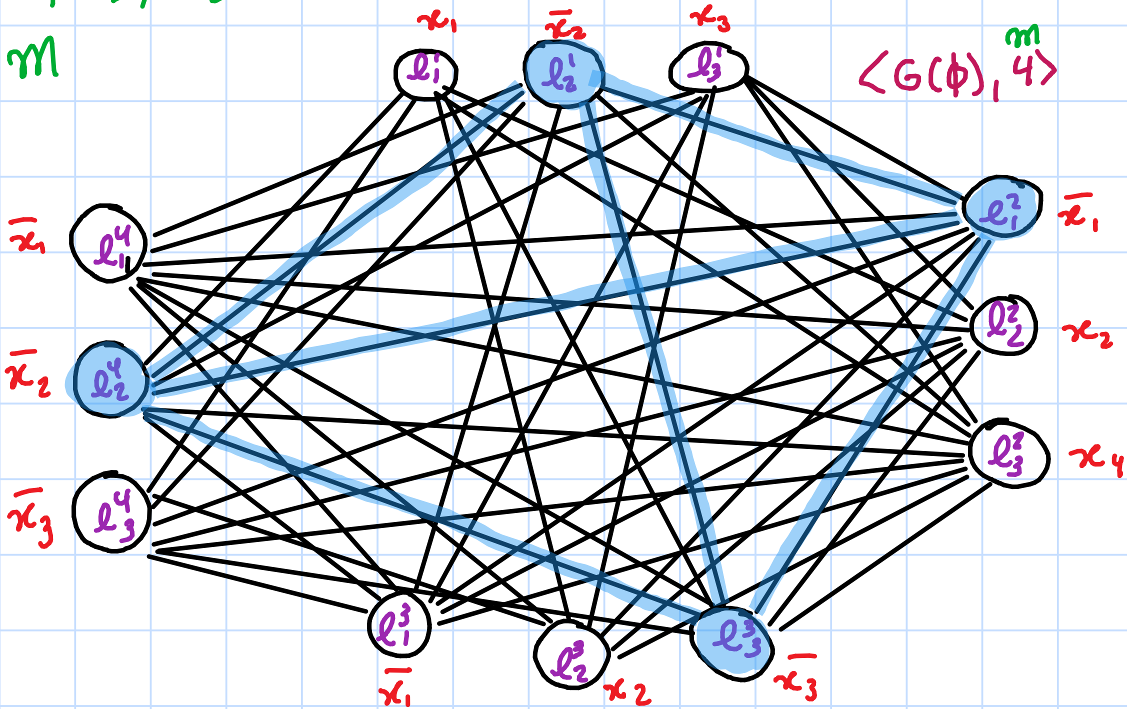


$$\phi = \overbrace{(\overset{\ell^1_1}{x_1} \vee \underset{1}{\overset{\ell^1_2}{\overline{x_2}}} \vee \overset{\ell^1_3}{x_3})}^{C_1} \wedge \overbrace{(\underset{1}{\overset{\ell^2_1}{\overline{x_1}}} \vee \overset{\ell^2_2}{x_2} \vee \overset{\ell^2_3}{x_4})}^{C_2} \wedge \overbrace{(\overset{\ell^3_1}{\overline{x_1}} \vee \overset{\ell^3_2}{x_2} \vee \underset{1}{\overset{\ell^3_3}{\overline{x_3}}})}^{C_3} \wedge \overbrace{(\overset{\ell^4_1}{\overline{x_1}} \vee \underset{1}{\overset{\ell^4_2}{\overline{x_2}}} \vee \overset{\ell^4_3}{\overline{x_3}})}^{C_4}$$

$$S = \{\ell^1_2, \ell^2_1, \ell^3_3, \ell^4_2\}$$

$$|S| \geq m$$

$\langle G(\phi), \overset{m}{4} \rangle$



x	$\psi(x)$
x_1	0
x_2	0
x_3	0
x_4	0 ou 1

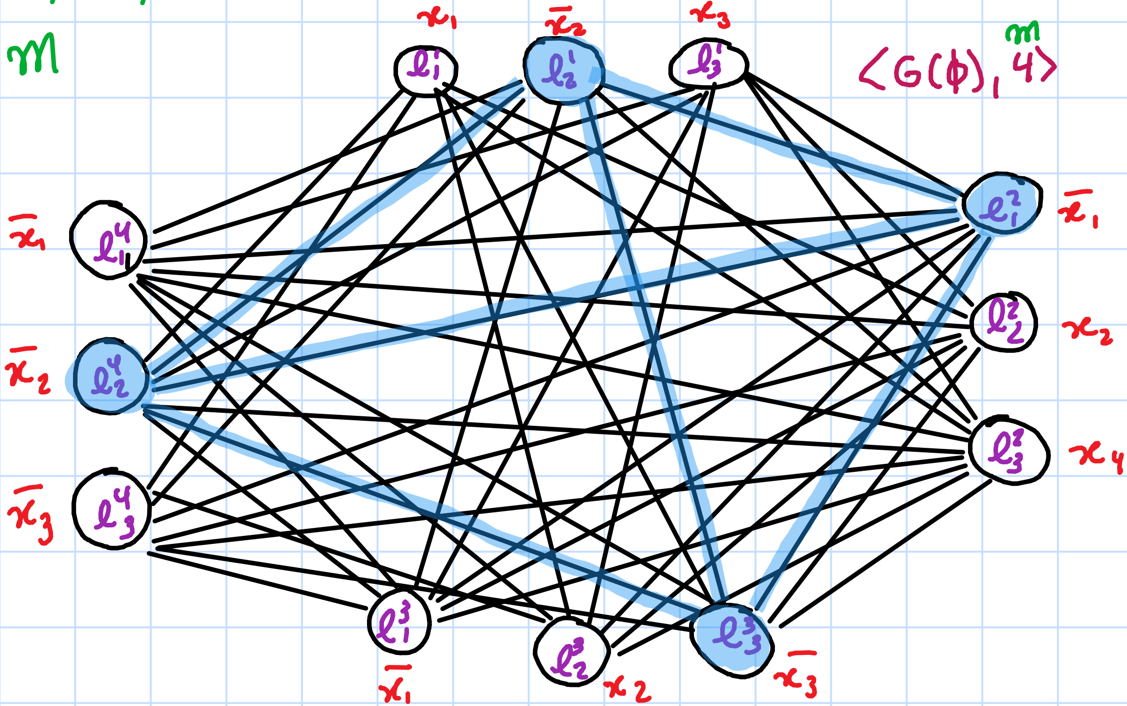
$$\phi = \overbrace{(\underset{0}{\overset{\ell_1^1}{x_1}} \vee \underset{1}{\overset{\ell_2^1}{\overline{x_2}}} \vee \underset{0}{\overset{\ell_3^1}{x_3}})}^{C_1} \wedge \overbrace{(\underset{1}{\overset{\ell_1^2}{\overline{x_1}}} \vee \underset{0}{\overset{\ell_2^2}{x_2}} \vee \underset{1}{\overset{\ell_3^2}{x_4}})}^{C_2} \wedge \overbrace{(\underset{1}{\overset{\ell_1^3}{\overline{x_1}}} \vee \underset{0}{\overset{\ell_2^3}{x_2}} \vee \underset{1}{\overset{\ell_3^3}{\overline{x_3}}})}^{C_3} \wedge \overbrace{(\underset{1}{\overset{\ell_1^4}{\overline{x_1}}} \vee \underset{1}{\overset{\ell_2^4}{\overline{x_2}}} \vee \underset{1}{\overset{\ell_3^4}{\overline{x_3}}})}^{C_4} = 1$$

$S = \{ \ell_2^1, \ell_1^2, \ell_3^3, \ell_2^4 \}$

$|S| \geq m$

$\langle G(\phi), \overset{m}{4} \rangle$

x	$\psi(x)$
x_1	0
x_2	0
x_3	0
x_4	1



Teo Clique é NP-completo

Demonstração (continuação)

- Seja $S = \{w_1, w_2, \dots, w_k\}$ uma clique em $G(\phi)$ tal que $k \geq m$.
- Como não há arestas entre vértices que representam literais da mesma cláusula, sabemos que cada $w_i \in S$ pertence a uma cláusula distinta. Como temos m cláusulas e $k \geq m$, concluímos que $k = m$.
- Portanto, podemos supor, sem perda de generalidade, que $w_i \in C_i$, para todo $i = 1, \dots, m$
  - Seja $\psi : V \to \{0,1\}$ definida da seguinte forma

$$\psi(x) = \begin{cases} 1 & \text{se } \exists\, w_i \in S \text{ tal que } w_i = x \\ 0 & \text{se } \exists\, w_i \in S \text{ tal que } w_i = \overline{x} \\ \perp & \text{caso contrário} \end{cases}$$

Teo Clique é NP-completo

Demonstração (continuação)

- Como não há arestas entre literais l_j^i e l_k^p de cláusulas distintas ($i \neq p$) tais que $l_j^i = x$ e $l_k^p = \overline{x}$ e como S é uma clique, temos que ψ é uma atribuição válida e não ambígua.

- Note que a atribuição ψ faz com que o literal ω_i avalie como verdadeiro na cláusula C_i. Assim, ψ é uma fórmula satisfazível e, portanto, $\langle \phi \rangle \in$ 3-SAT.

$\square$

Problema CicloHamiltoniano

Entrada: $\langle D \rangle$, onde D é um digrafo

Saída: sim, se existe um ciclo hamiltoniano em D

não, caso contrário

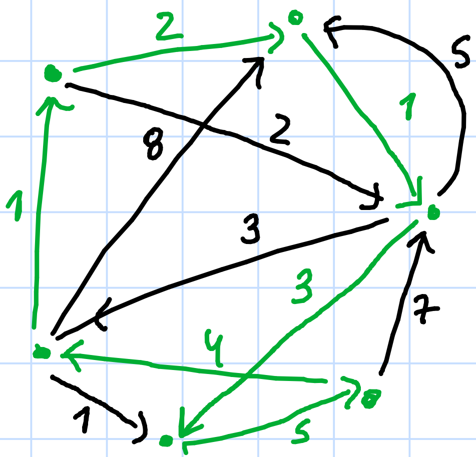
Teo CicloHamiltoniano $\in$ NP-Completo.

Problema TSP (Popularmente conhecido como o problema do caixeiro viajante)

Entrada: $\langle D, w, k \rangle$, onde D é um digrafo, $w: E(D) \to \mathbb{R}$ e $k \in \mathbb{R}$

Saída: <u>sim</u>, se existe um ciclo Hamiltoniano C em D tal que $w(D) \leq k$.

<u>não</u>, caso contrário.



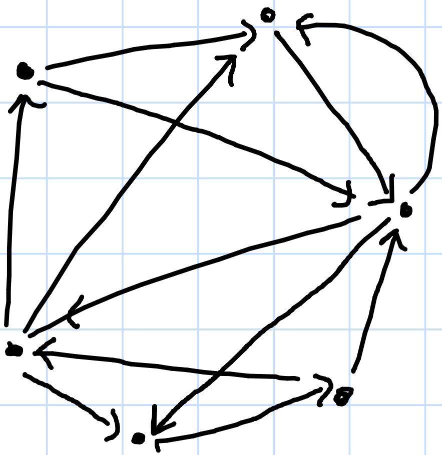
$K = 22$

$$w(C) = 1 + 2 + 1 + 3 + 5 + 4$$
$$= 16$$

Lema B $\quad$ CicloHamiltoniano $\preccurlyeq_p$ TSP

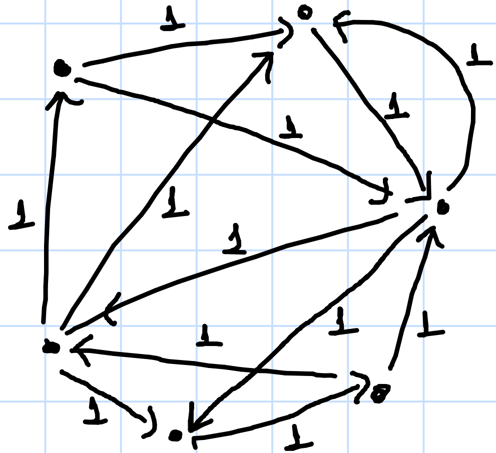
Demonstração

CicloHamiltoniano



$\langle D \rangle$

TSP



$\rho(D) = \langle D, w, |V(D)| \rangle$

Lema B $\text{CicloHamiltoniano} \preccurlyeq_P \text{TSP}$

Demonstração

Seja f o procedimento definido como:

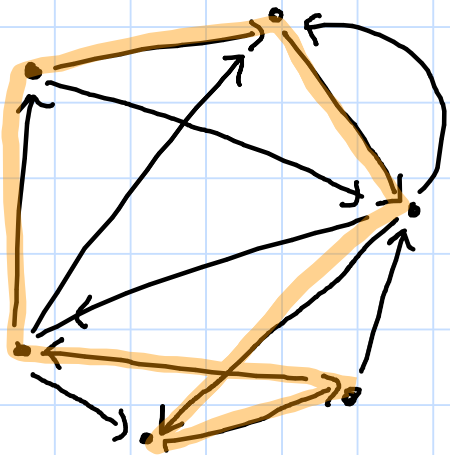
$f(\langle D \rangle)$ {
  Seja $w: E(D) \to \mathbb{R}$ definida como
    $w(e) = 1 \quad \forall\ e \in E(D)$
  Seja $K = |V(D)|$
  Retorne $\langle D, w, k \rangle$
}

É fácil perceber que f toma tempo polinomial. Agora, vamos mostra que f é uma redução.

Lema B CicloHamiltoniano $\preccurlyeq_p$ TSP

Demonstração

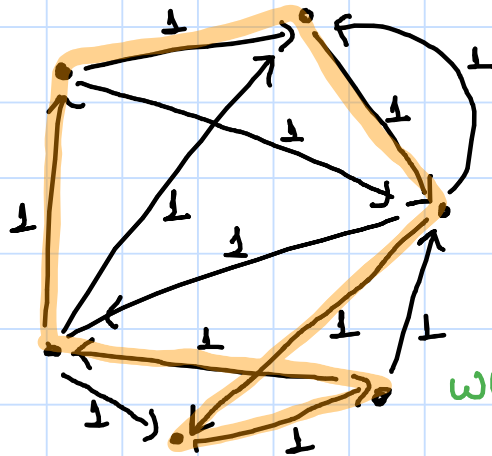
CicloHamiltoniano



C

$\Longrightarrow$

TSP



$w(C) = 6$

$\langle D \rangle$

$\rho(D) = \langle D, w, \underset{|V(D)|}{6} \rangle$

Lema B CicloHamiltoniano $\preccurlyeq_P$ TSP

Demonstração (continuação)

- Suponha que $\langle D \rangle$ é uma instância sim de cicloHamiltoniano e seja $C \subseteq D$ um ciclo hamiltoniano de D.
- Seja $\langle D', w, k \rangle = f(\langle D \rangle)$.
- Note que $C \subseteq D' = D$ é um ciclo Hamiltoniano tal que

$$w(C) = \sum_{e \in E(C)} w(e) = \sum_{e \in E(C)} 1 = |E(C)| = |V(D')| = k.$$

- Portanto, $\langle D', w, k \rangle$ é uma instância sim de TSP.

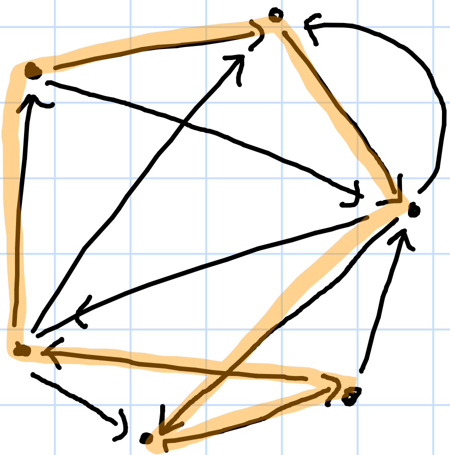
Lema B CicloHamiltoniano $\preccurlyeq_p$ TSP

Demonstração

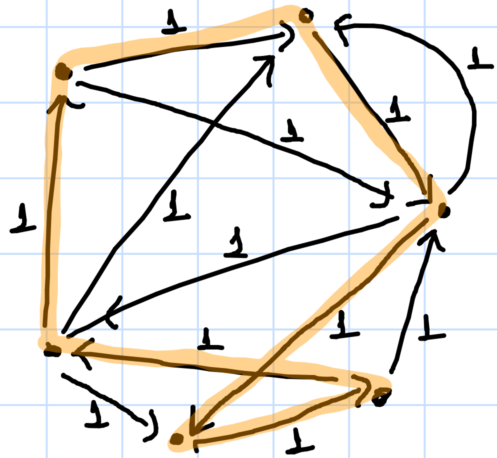
CicloHamiltoniano

TSP

C



$\Leftarrow$



$\langle D \rangle$

$\rho(D) = \langle D, w, G \rangle$

$|V(D)|$

Lema B CicloHamiltoniano $\preccurlyeq_P$ TSP

Demonstração (continuação)

- Seja $f(\langle D\rangle) = \langle D', w, K\rangle$ uma instância sim de TSP e seja $C \subseteq D'$ um ciclo hamiltoniano tal que $w(C) \leq K$.
- Note que $C \subseteq D' = D$ e $V(C) = V(D') = V(D)$.

- Portanto D é uma instância sim de cicloHamiltoniano. $\square$

Teo TSP $\in$ NP-difícil

Demonstração

Consequência direta do Lema B $\square$

Teo TSP é NP-completo

Demonstração

Consequência direta dos Lemas A e B $\square$

$$\boxed{\textbf{Lema A} \quad \text{TSP} \in \text{NP}}$$