Problemas NP-Completos

Fábio Botler

Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro

Aula passada

- A classe NP
- A questão P=NP
- Complementos de Problemas
- Transformações Polinomiais
- Alguns problemas NP-Completos

Aula de hoje

- Reduções de Karp $\times$ Reduções de Cook
- Restrições e Extensões de Problemas
- Algoritmos superpolinomiais
- Isomorfismo de Grafos
- Algoritmos pseudopolinomiais
- Problemas numéricos

Definição

Uma **redução de Karp** é um processo f que transforma instâncias de um problema Π_1 em instâncias um problema Π_2 tal que:

1. $\Pi_1(I) = SIM$ se e somente se $\Pi_2(f(I)) = SIM$
2. f pode ser computada em tempo polinomial

Definição

Uma **redução de Karp** é um processo f que transforma instâncias de um problema Π_1 em instâncias um problema Π_2 tal que:

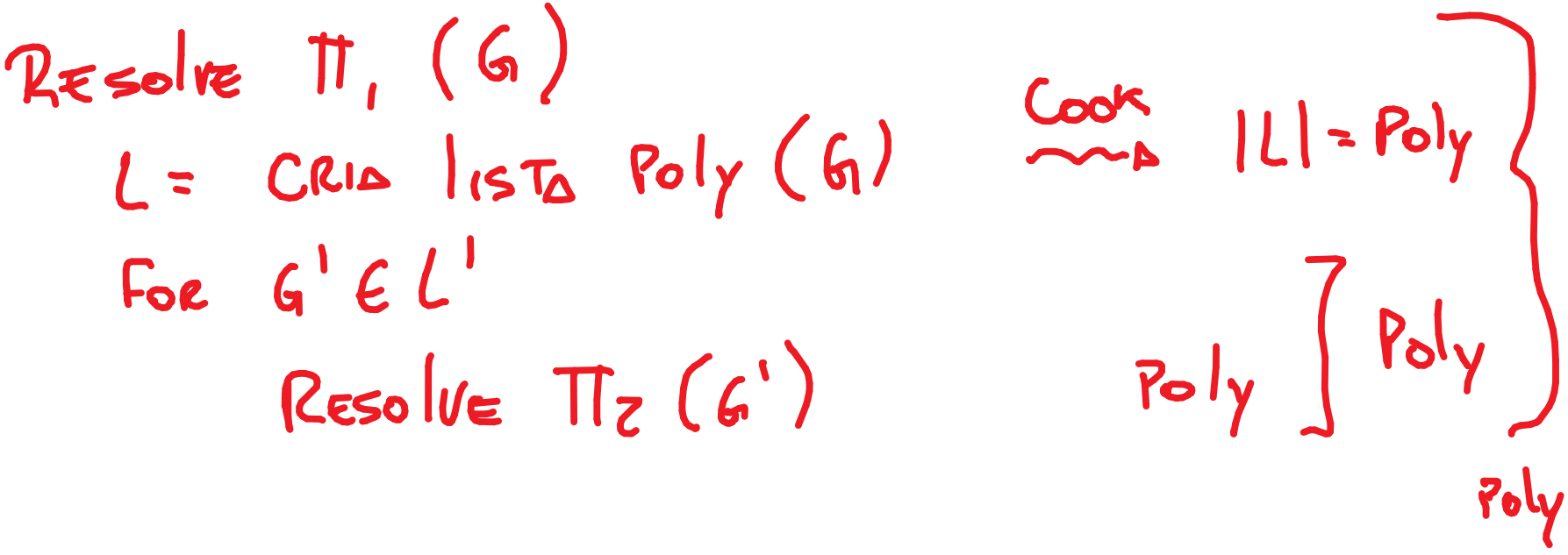
1. $\Pi_1(I) = SIM$ se e somente se $\Pi_2(f(I)) = SIM$
2. f pode ser computada em tempo polinomial

Definição alternativa

Uma **redução de Karp** de um problema Π_1 para um problema Π_2 é um algoritmo que resolve Π_1 fazendo uma chamada a uma subrotina que resolve o problema Π_2, e que toma tempo polinomial fora dessa chamada.

Definição

Uma **redução de Cook** de um problema Π_1 para um problema Π_2 é um algoritmo que resolve Π_1 fazendo um número polinomial de chamadas a uma subrotina que resolve o problema Π_2, e que toma tempo polinomial fora dessas chamadas.



Definição

Uma **redução de Cook** de um problema Π_1 para um problema Π_2 é um algoritmo que resolve Π_1 fazendo um número polinomial de chamadas a uma subrotina que resolve o problema Π_2, e que toma tempo polinomial fora dessas chamadas.

Definição alternativa

Uma **redução de Cook** é um processo f que transforma instâncias de um problema Π_1 em conjuntos de instâncias de um problema Π_2 tal que:

1. $\Pi_1(I) = SIM$ se e somente se $SIM \in \{\Pi_2(I') : I' \in f(I)\}$
2. f pode ser computada em tempo polinomial

Restrições e Extensões de problemas

$\Pi'(D', Q')$ é uma **restrição** de $\Pi(D, Q)$ se

(i) $D' \subseteq D$

(ii) $Q' = Q$

Também dizemos que Π é uma **extensão** de Π'.

~~Usamos restrições de problemas~~

Exemplo: SAT × 3SAT

SAT

DADOS: Uma expressão booleana E
na Forma Normal Conjuntiva (FNC)

OBJETIVO: E é satisfatível?

3SAT

DADOS: Uma expressão booleana E
na Forma Normal Conjuntiva (FNC)
contendo apenas cláusulas com *três literais*

OBJETIVO: E é satisfatível?

▷ 3SAT é restrição de SAT.

Clique

DADOS: Um grafo G e um inteiro k
OBJETIVO: G possui uma clique de tamanho pelo menos k?

Clique

DADOS: Um grafo G e um inteiro k
OBJETIVO: G possui uma clique de tamanho pelo menos k?

EQUIV.

Clique'

DADOS: Um grafo G e uma clique H
OBJETIVO: G possui um subgrafo isomorfo a H?

$f(G,k) = (G, K_k)$

$g(G, K_k) = (G, k)$

Exemplo: Clique $\times$ Isomorfismo de Subgrafo

Clique'

DADOS: Um grafo G e uma clique H
OBJETIVO: G possui um subgrafo isomorfo a H?

Exemplo: Clique × Isomorfismo de Subgrafo

Clique'

DADOS: Um grafo G e uma clique H
OBJETIVO: G possui um subgrafo isomorfo a H?

Isomorfismo de Subgrafo

DADOS: Um grafo G e um grafo H
OBJETIVO: G possui um subgrafo isomorfo a H?

Exemplo: Clique × Isomorfismo de Subgrafo

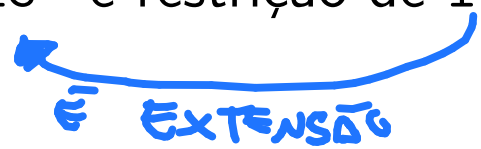
Clique'

DADOS: Um grafo G e uma clique H
OBJETIVO: G possui um subgrafo isomorfo a H?

Isomorfismo de Subgrafo

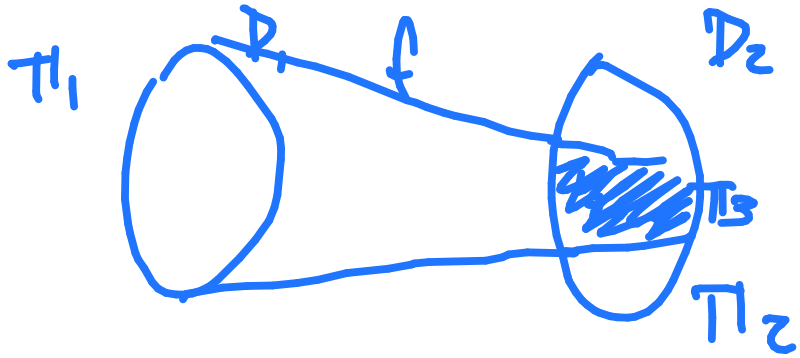
DADOS: Um grafo G e um grafo H
OBJETIVO: G possui um subgrafo isomorfo a H?

▷ Clique' é restrição de Isomorfismo de Subgrafo.



Lema

Sejam Π *e* Π' *problemas de decisão tais que* Π' *é restrição de* Π*. Se* Π' *é NP-Completo, então* Π *é NP-Difícil.*



Lema

Sejam Π *e* Π' *problemas de decisão tais que* Π' *é restrição de* Π*. Se* Π' *é NP-Completo, então* Π *é NP-Difícil.*

$\triangleright$ Se $\Pi \in NP$, então Π é NP-Completo

$\Pi'' \in NP \quad \Pi'' \propto \Pi' \propto \Pi$

Lema

Sejam Π *e* Π' *problemas de decisão tais que* Π' *é restrição de* Π*.*
Se Π' *é NP-Completo, então* Π *é NP-Difícil.*

- ▷ Se $\Pi \in NP$, então Π é NP-Completo
- ▷ Se Π é NP-Completo,
  não necessariamente Π' é NP-Completo

Lema

Sejam Π *e* Π' *problemas de decisão tais que* Π' *é restrição de* Π*.*
Se Π' *é NP-Completo, então* Π *é NP-Difícil.*

- ▷ Método alternativo para provar que um problema é NP-Completo

Lema

Sejam Π *e* Π' *problemas de decisão tais que* Π' *é restrição de* Π*.*
Se Π' *é NP-Completo, então* Π *é NP-Difícil.*

- ▷ Método alternativo para provar que um problema é NP-Completo

Π é NP-Completo se

Lema

Sejam Π *e* Π' *problemas de decisão tais que* Π' *é restrição de* Π*.*
Se Π' *é NP-Completo, então* Π *é NP-Difícil.*

- ▷ Método alternativo para provar que um problema é NP-Completo

Π é NP-Completo se

(i) $\Pi \in NP$

(ii) Π é extensão de um problema NP-Completo

Isomorfismo de subgrafos

Teorema

`Isomorfismo de subgrafos` é NP-Completo.

Isomorfismo de subgrafos

Teorema

Isomorfismo de subgrafos é NP-Completo.

Prova

Isomorfismo de subgrafos está em NP

Isomorfismo de subgrafos

Teorema

Isomorfismo de subgrafos é NP-Completo.

Prova

Isomorfismo de subgrafos está em NP

Isomorfismo de subgrafos é extensão de Clique'

Isomorfismo de subgrafos

Teorema

Isomorfismo de subgrafos é NP-Completo.

Prova

Isomorfismo de subgrafos está em NP

Isomorfismo de subgrafos é extensão de Clique'

Clique' é NP-Completo

Isomorfismo de Grafos

DADOS: Um grafo G e um grafo H
OBJETIVO: G é isomorfo a H?

- ▷ Isomorfismo de Grafos é **restrição** de Isomorfismo de subrafos

Exemplo

3SAT é NP-Completo?

Exemplo

3SAT é NP-Completo? Não devido ao lema anterior

Exemplo

3SAT é NP-Completo? Não devido ao lema anterior

Teorema

3SAT é NP-Completo

Exemplo

3SAT é NP-Completo? Não devido ao lema anterior

Teorema

3SAT é NP-Completo

Prova

$\triangleright$ $SAT \propto 3SAT$

Exemplo

3SAT é NP-Completo? Não devido ao lema anterior

Teorema

3SAT é NP-Completo

Prova

- $\triangleright$ $SAT \propto 3SAT$
- $\triangleright$ $(x_1 \vee x_2 \vee \cdots \vee x_k) = (x_1 \vee x_2 \vee h) \wedge (\overline{h} \vee x_3 \vee \cdots \vee x_k)$

Exemplo

3SAT é NP-Completo? Não devido ao lema anterior

Teorema

3SAT é NP-Completo

Prova

- $SAT \propto 3SAT$
- $(x_1 \vee x_2 \vee \cdots \vee x_k) = (x_1 \vee x_2 \vee h) \wedge (\overline{h} \vee x_3 \vee \cdots \vee x_k)$
- *Uma cláusula de tamanho k é transformada em $(k-2)$ cláusulas de tamanho 3.*

Exemplo

3SAT é NP-Completo? Não devido ao lema anterior

Teorema

3SAT é NP-Completo

Prova

- $\triangleright$ *SAT* $\propto$ *3SAT*
- $\triangleright$ $(x_1 \vee x_2 \vee \cdots \vee x_k) = (x_1 \vee x_2 \vee h) \wedge (\overline{h} \vee x_3 \vee \cdots \vee x_k)$
- $\triangleright$ *Uma cláusula de tamanho* k *é transformada em* $(k-2)$ *cláusulas de tamanho* 3.
- $\triangleright$ *Uma expressão* $E = C_1 \wedge \cdots \wedge C_p$ *onde* C_i *possui* k_i *literais*

Exemplo

3SAT é NP-Completo? Não devido ao lema anterior

Teorema

3SAT é NP-Completo

Prova

- $\triangleright$ $SAT \propto 3SAT$
- $\triangleright$ $(x_1 \vee x_2 \vee \cdots \vee x_k) = (x_1 \vee x_2 \vee h) \wedge (\overline{h} \vee x_3 \vee \cdots \vee x_k)$
- $\triangleright$ *Uma cláusula de tamanho k é transformada em $(k-2)$ cláusulas de tamanho 3.*
- $\triangleright$ *Uma expressão $E = C_1 \wedge \cdots \wedge C_p$ onde C_i possui k_i literais é transformada em uma expressão E' com no máximo $(\sum k_i) - 2p$ cláusulas de tamanho 3.*

$f(\epsilon) = \epsilon'$ 3SAT

Exemplo

3SAT é NP-Completo? Não devido ao lema anterior

Teorema

3SAT é NP-Completo

Prova

- $\triangleright$ *SAT* $\propto$ *3SAT*
- $\triangleright$ $(x_1 \vee x_2 \vee \cdots \vee x_k) = (x_1 \vee x_2 \vee h) \wedge (\overline{h} \vee x_3 \vee \cdots \vee x_k)$
- $\triangleright$ *Uma cláusula de tamanho* k *é transformada em* $(k-2)$ *cláusulas de tamanho* 3.
- $\triangleright$ *Uma expressão* $E = C_1 \wedge \cdots \wedge C_p$ *onde* C_i *possui* k_i *literais é transformada em uma expressão* E' *com no máximo* $(\sum k_i) - 2p$ *cláusulas de tamanho* 3.
- $\triangleright$ *Se* E *possui* $\ell = \sum k_i$ *literais*

Exemplo

3SAT é NP-Completo? Não devido ao lema anterior

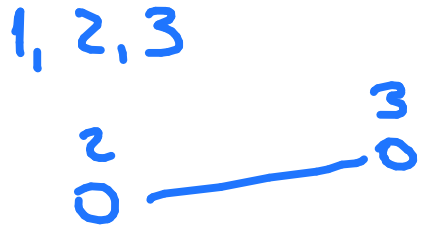
Teorema

3SAT é NP-Completo

Prova

- $\triangleright$ $SAT \propto 3SAT$
- $\triangleright$ $(x_1 \vee x_2 \vee \cdots \vee x_k) \rightarrow (x_1 \vee x_2 \vee h) \wedge (\overline{h} \vee x_3 \vee \cdots \vee x_k)$
- $\triangleright$ *Uma cláusula de tamanho k é transformada em $(k-2)$ cláusulas de tamanho 3.*
- $\triangleright$ *Uma expressão $E = C_1 \wedge \cdots \wedge C_p$ onde C_i possui k_i literais é transformada em uma expressão E' com no máximo $(\sum k_i) - 2p$ cláusulas de tamanho 3.*
- $\triangleright$ *Se E possui $\ell = \sum k_i$ literais E' possui $3(\ell - 2p)$ literais*

3-coloração



DADOS: Um grafo G

OBJETIVO: G admite uma 3-coloração própria?

3-coloração

DADOS: Um grafo G
OBJETIVO: G admite uma 3-coloração própria?

Teorema

3-coloração é NP-Completo

3-coloração

DADOS: Um grafo G
OBJETIVO: G admite uma 3-coloração própria?

Teorema

3-coloração é NP-Completo

Prova

$\triangleright 3SAT \propto$ *3-coloração*

3-coloração

DADOS: Um grafo G
OBJETIVO: G admite uma 3-coloração própria?

Teorema

3-coloração é NP-Completo

Prova

- $\triangleright$ $SAT \propto$ *3-coloração*
- $\triangleright$ *Seja* $E = C_1 \wedge \cdots \wedge C_k$ *uma expressão Booleana*

3-coloração

DADOS: Um grafo G
OBJETIVO: G admite uma 3-coloração própria?

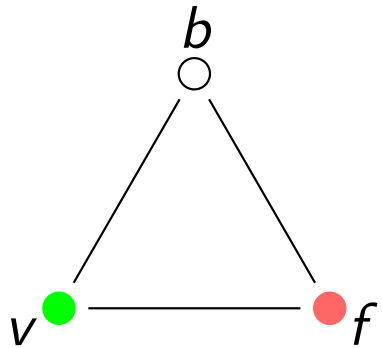
Teorema

3-coloração é NP-Completo

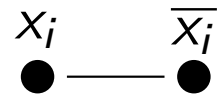
Prova

- ▷ $SAT \propto$ *3-coloração*
- ▷ *Seja* $E = C_1 \wedge \cdots \wedge C_k$ *uma expressão Booleana*
- ▷ *Definimos três artefatos: base, variável, cláusula*

Prova



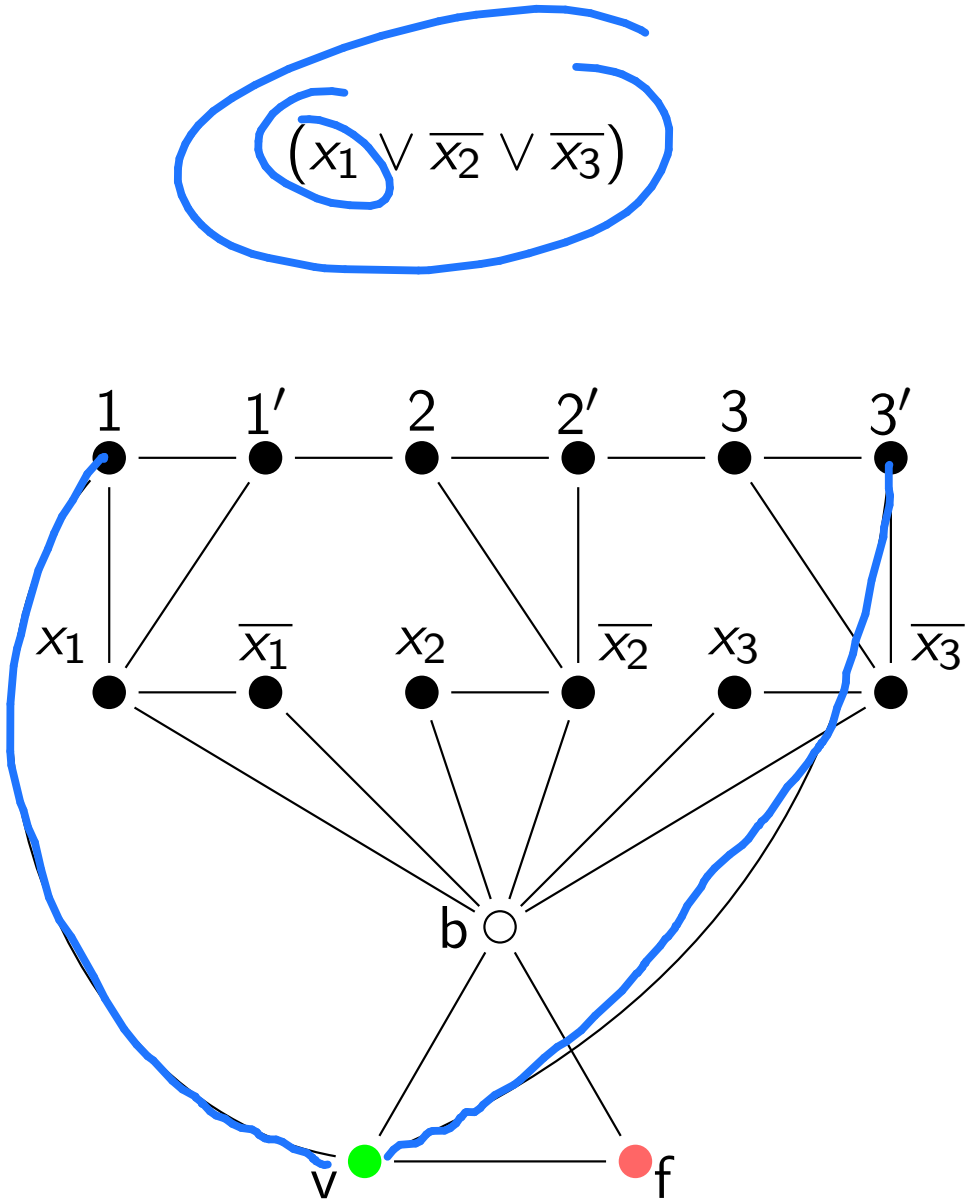
Triângulo base



Artefato Variável

Artefato Cláusula

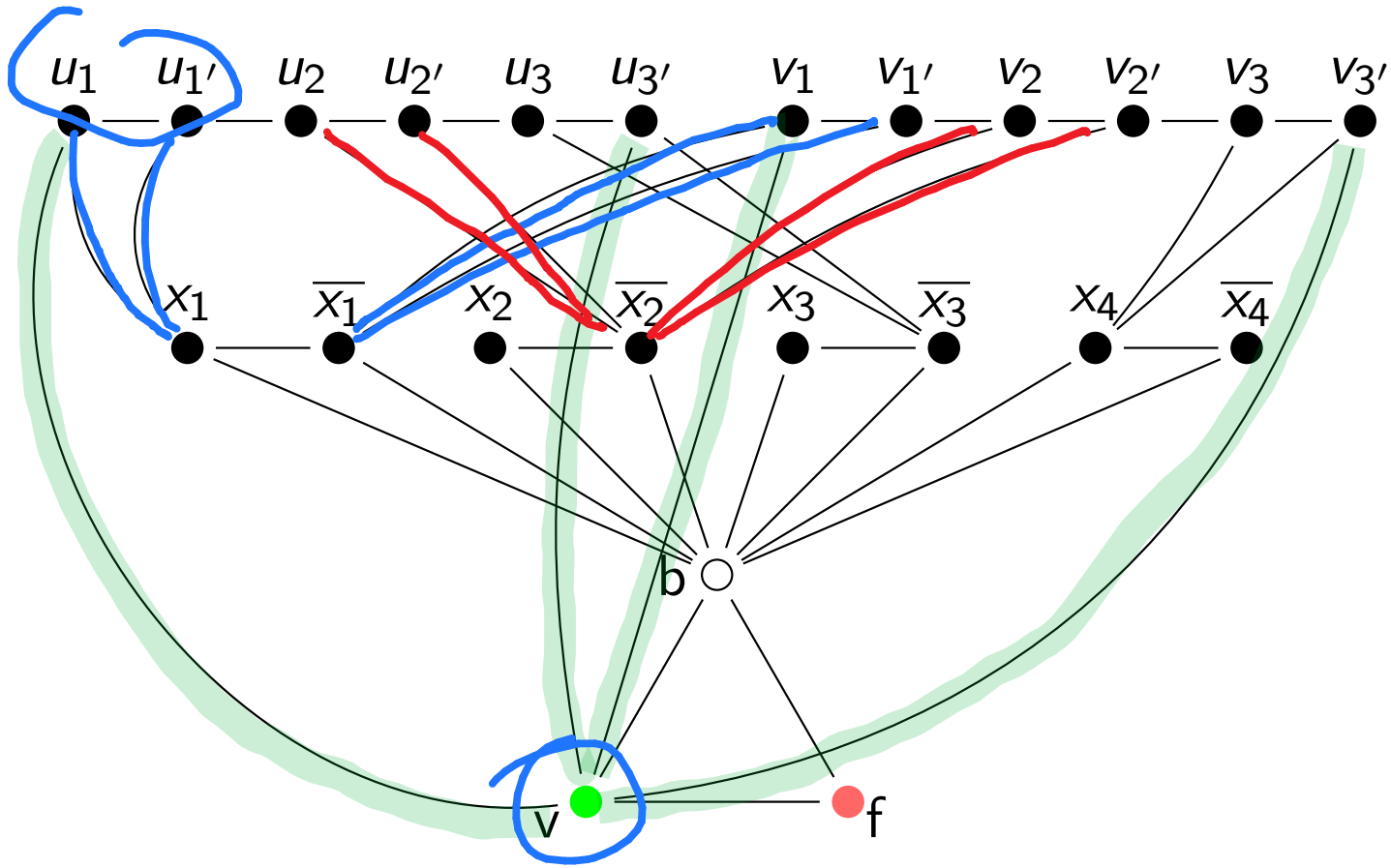
- um triângulo base
- um artefato variável para cada variável
- os dois vértices de cada artefato variável são ligados ao vértice b do triângulo base
- os vértices finais de cada artefato cláusula são ligados ao vértice v do triângulo base
- para $i \in \{1, 2, 3\}$, os vértices i e i' de cada artefato cláusula são ligados ao vértice do artefato variável correspondente ao literal presente em sua cláusula


$(x_1 \vee \overline{x_2} \vee \overline{x_3})$
1
1′
2
2′
3
3′
x_1
$\overline{x_1}$
x_2
$\overline{x_2}$
x_3
$\overline{x_3}$
b
v
f

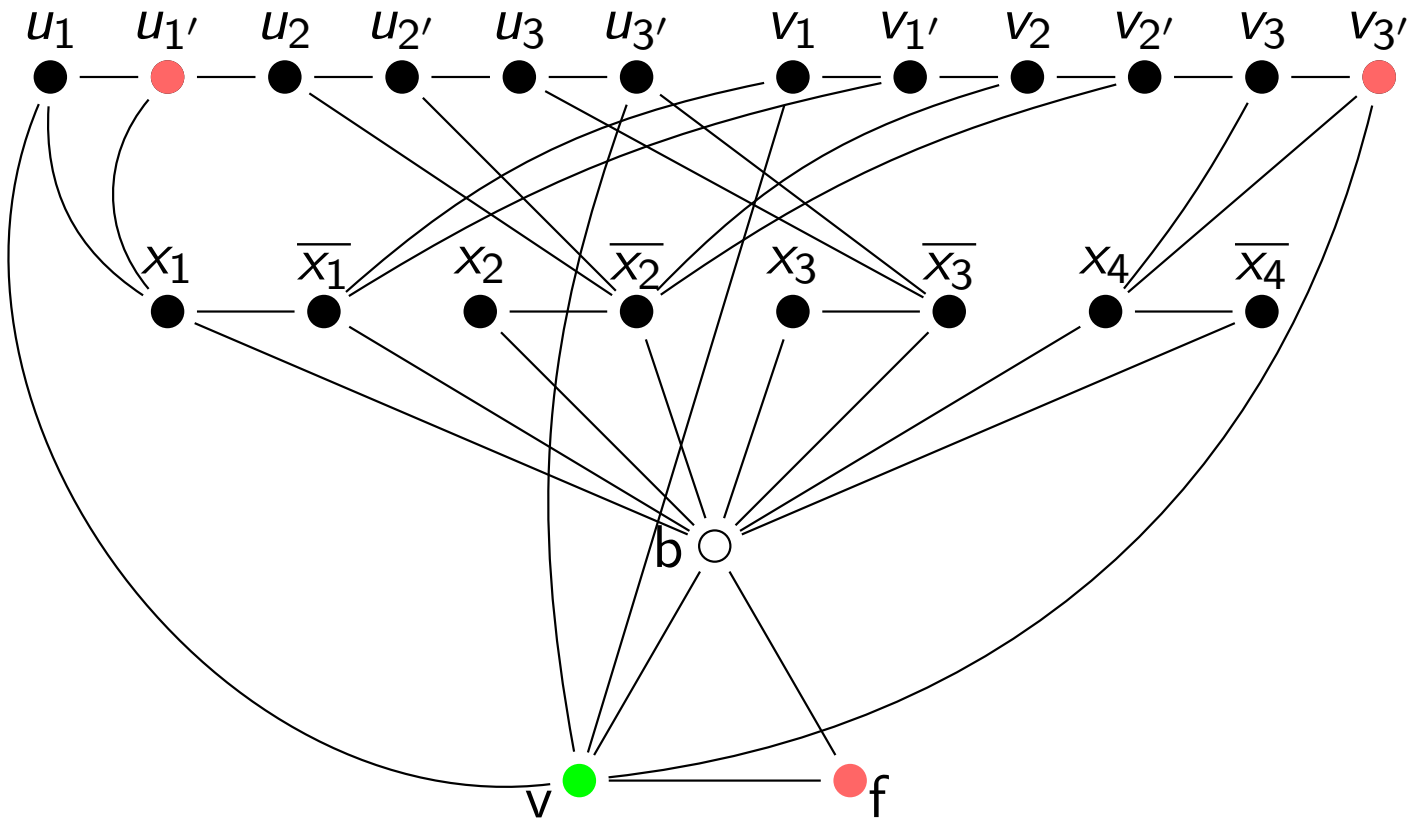
Lema

Seja G o grafo obtido pelo processo acima. Se existe uma 3-coloração de G então E é satisfatível.

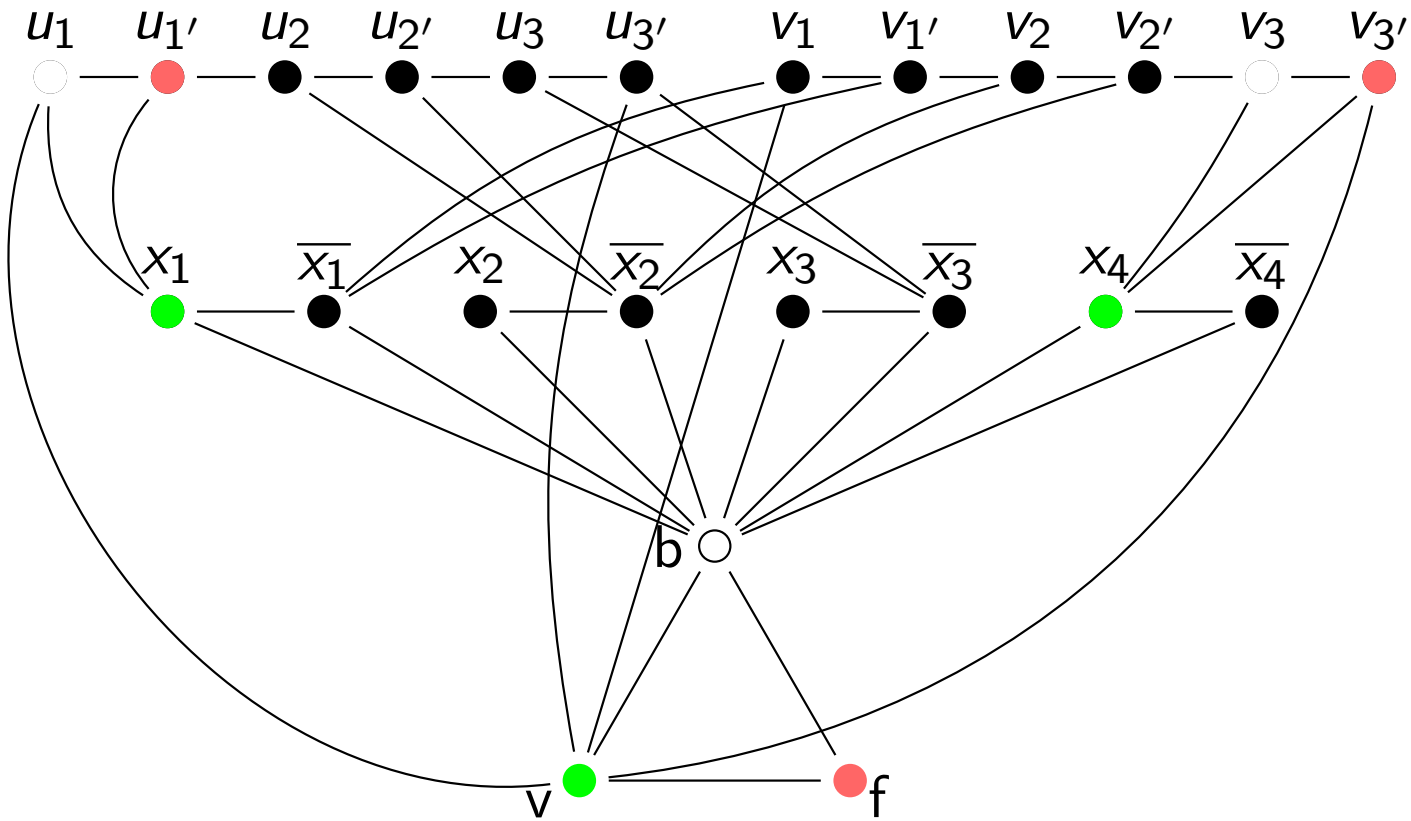
$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$$



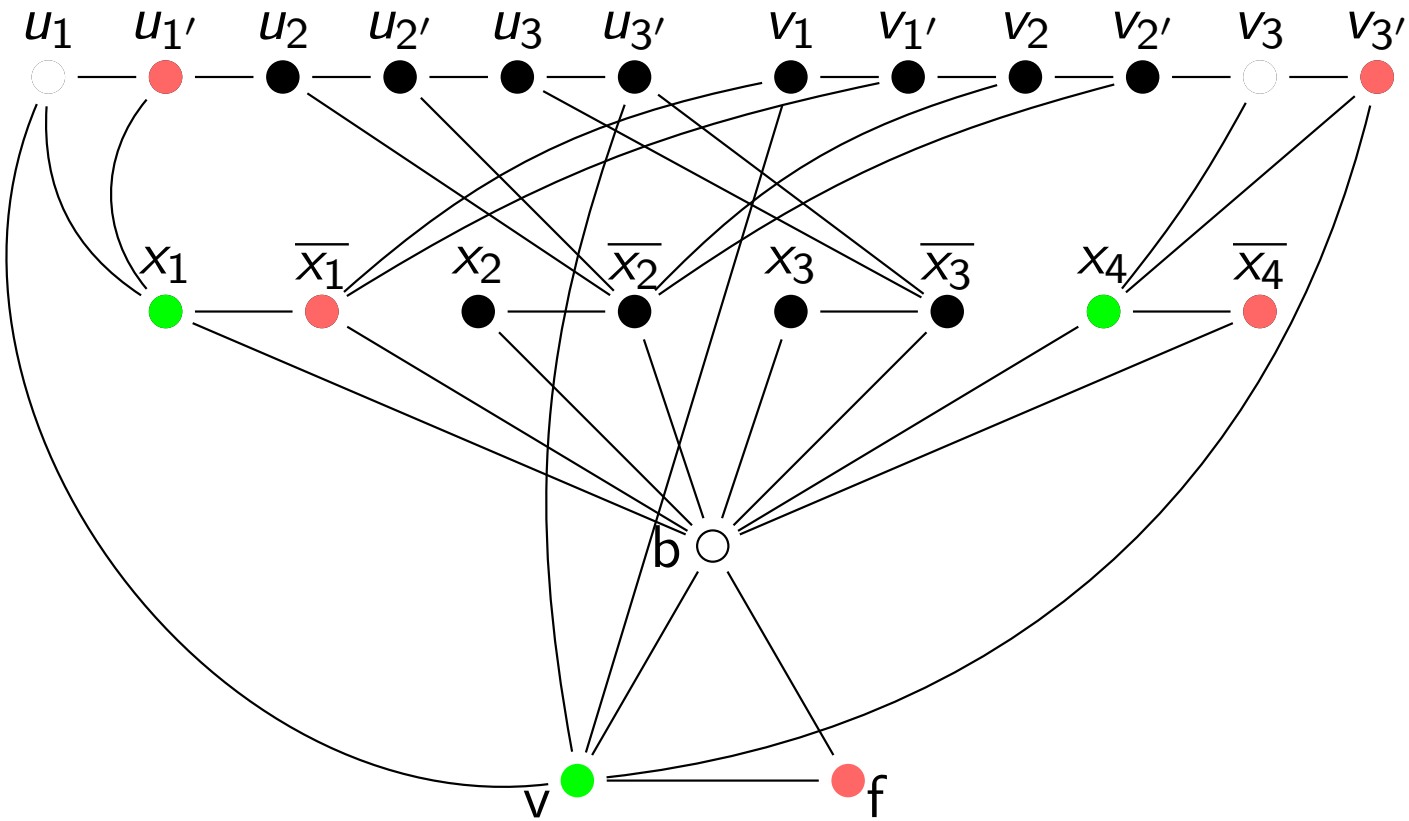
$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$$



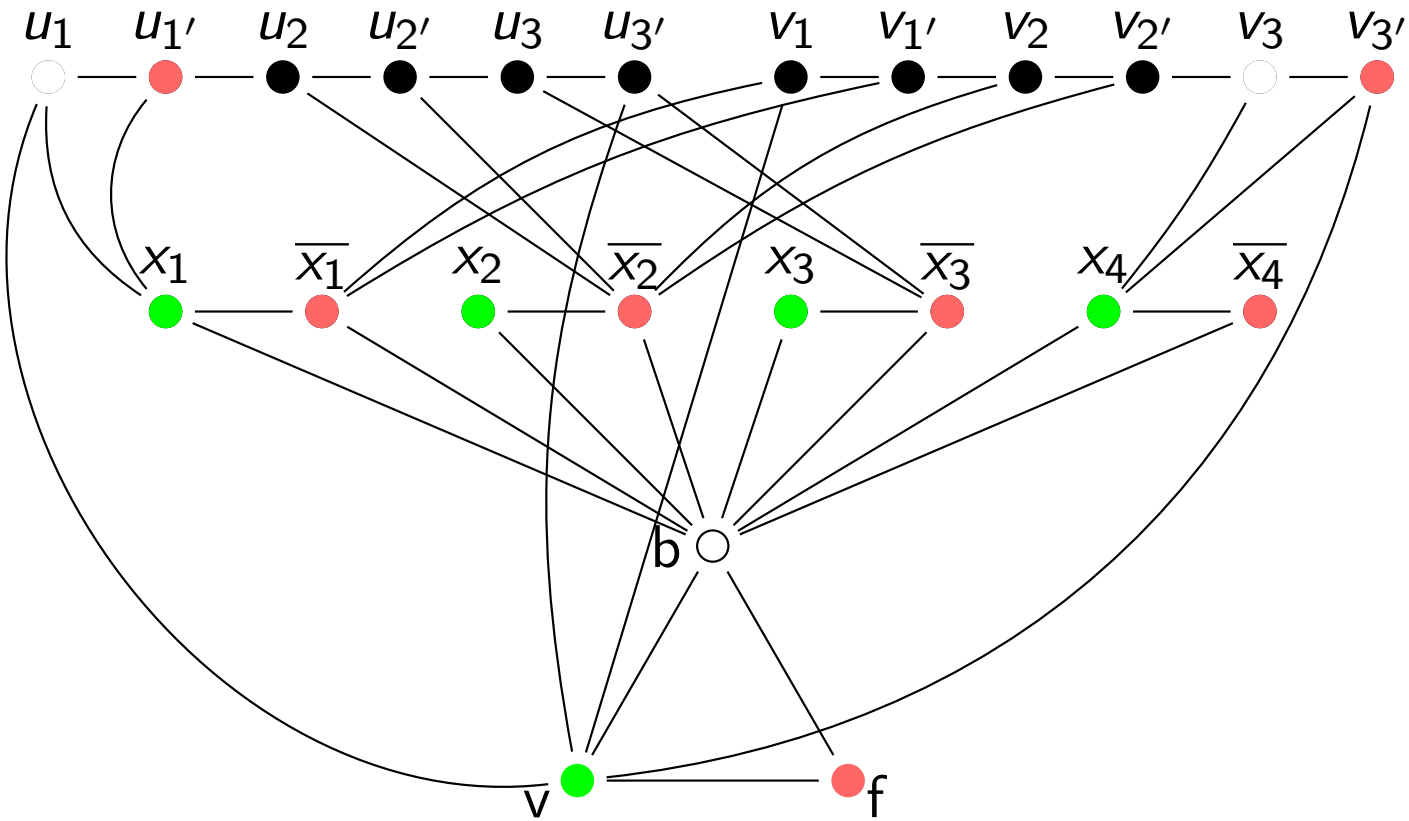
$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$$



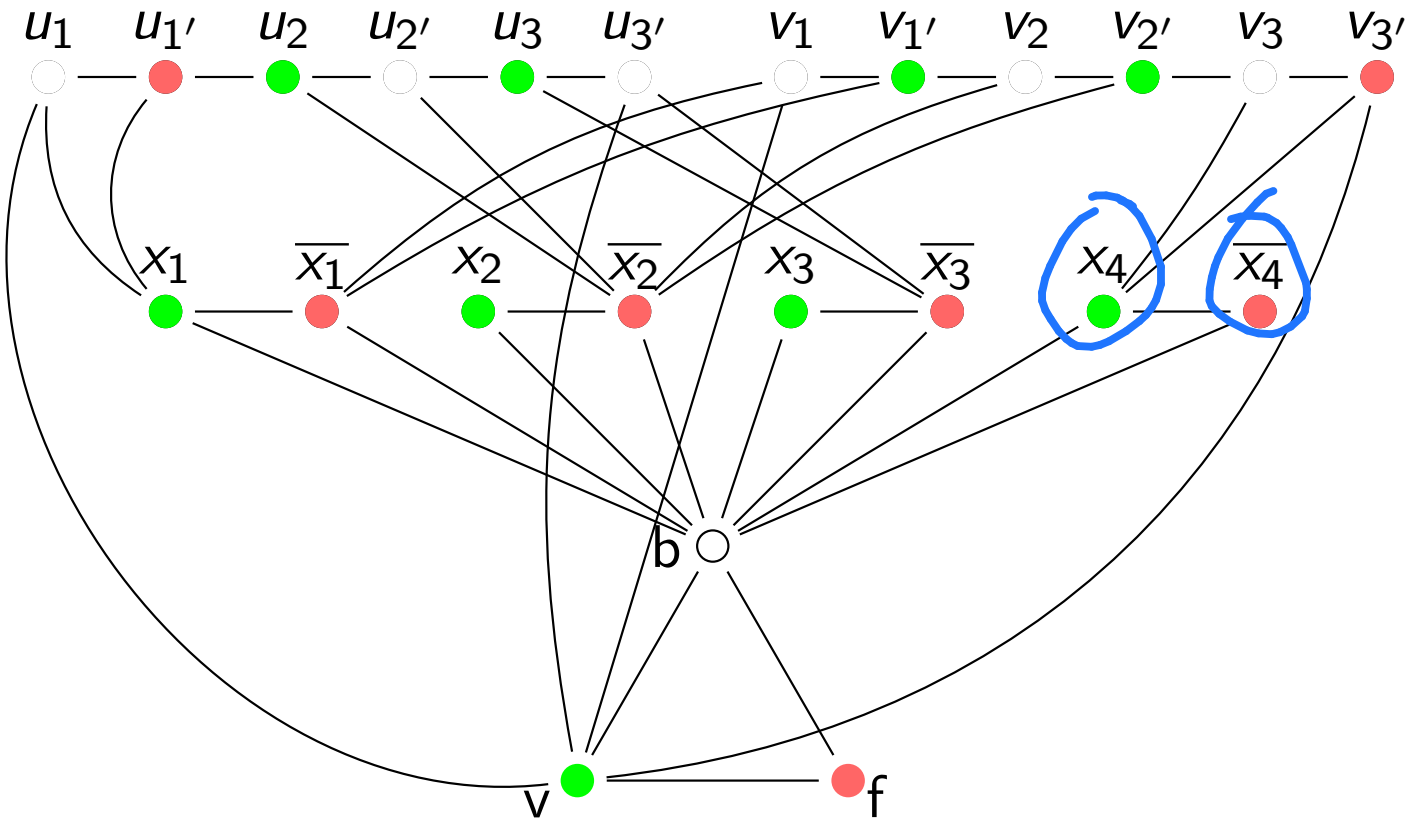
$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$$



$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$$



$$(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_2} \vee x_4)$$



Definição

Um algoritmo é **superpolinomial** *se sua complexidade é* $\omega(n^c)$ *para toda constante* c.

Definição

Um algoritmo é **exponencial** *se sua complexidade é* $2^{O(n^c)}$ *para alguma constante* c.

$2^{n^{1000}}$ $\quad$ $2^{7n^8+5n^{100}}$

Definição (sem consenso)

Um algoritmo é **subexponencial** *se sua complexidade é* $2^{o(n)}$.

Definição

Um algoritmo é **quasipolinomial** *quando sua complexidade é* $2^{O((\log n)^c)}$ *para alguma constante* c.

$2^{3(\log n)^7+5\log n}$

Polylog Polinômio em $\log n$

Isomorfismo de Grafos

DADOS: Um grafo G e um grafo H
OBJETIVO: G é isomorfo a H?

`Isomorfismo de Grafos`

DADOS: Um grafo G e um grafo H
OBJETIVO: G é isomorfo a H?

- ▷ `Isomorfismo de Grafos` é **restrição** de `Isomorfismo de subgrafos` (que é NP-Completo)

`Isomorfismo de Grafos`

DADOS: Um grafo G e um grafo H
OBJETIVO: G é isomorfo a H?

▷ `Isomorfismo de Grafos` é **restrição** de `Isomorfismo de subgrafos` (que é NP-Completo)

Teorema (Babai, 2015+)

Isomorfismo de Grafos pode ser resolvido em tempo quasipolinomial.

Algoritmos Pseudopolinomiais

Definição

Um algoritmo é **pseudopolinomial** *se sua complexidade for polinomial no tamanho da instância quando codificada em unário.*

Definição

Um algoritmo é **pseudopolinomial** *se sua complexidade for polinomial no tamanho da instância quando codificada em unário.*

$$\text{Valores} \times \text{Tamanho}$$

Definição

Um algoritmo é **pseudopolinomial** *se sua complexidade for polinomial no tamanho da instância quando codificada em unário.*

$$\text{Valores} \times \text{Tamanho}$$

Exemplo

Algoritmo natural para `Fatorização de Inteiros`*:* $O(n)$

Definição

Um algoritmo é **pseudopolinomial** *se sua complexidade for polinomial no tamanho da instância quando codificada em unário.*

Valores $\times$ Tamanho

Exemplo

Algoritmo natural para `Fatorização de Inteiros`*:* $O(n)$

UNÁRIO
$|\Sigma| = 2^n$

Exemplo

Algoritmo de programação dinâmica para o `Problema da Mochila`*:* $O(nW)$

$\Theta(n \cdot 2^n) = \Theta(|\Sigma|)$ $W = 2^n$ $\sim n$ BITS BINÁRIO
$|\Sigma| = O(n)$

Definição

Um problema NP-Completo que admite algoritmo pseudopolinomial é chamado **NP-Completo fraco**.

Definição

Um problema NP-Completo que admite algoritmo pseudopolinomial é chamado **NP-Completo fraco**.

$O(nW) \overset{W = n^3}{=} O(n^4)$

Exemplo

O `Problema da Mochila`

Exemplo

Circuito Hamiltoniano:

- ▷ *o valor do maior dado de entrada (índices de vértices/arestas) é polinomial no tamanho do grafo.*
- ▷ *Representar tais dados em forma unária não muda a natureza do tamanho dos dados, i.e., os dados continuam polinomiais.*

Exemplo

Circuito Hamiltoniano:

- ▷ *o valor do maior dado de entrada (índices de vértices/arestas) é polinomial no tamanho do grafo.*
- ▷ *Representar tais dados em forma unária não muda a natureza do tamanho dos dados, i.e., os dados continuam polinomiais.*
- ▷ *Um algoritmo pseudopolinomial, nesse caso, é um algoritmo polinomial.*

- ▷ Há problemas NP-Completos para os quais a existência de algoritmos pseudopolinomiais implica em $P = NP$.

- ▷ Há problemas NP-Completos para os quais a existência de algoritmos pseudopolinomiais implica em $P = NP$.

Definição

Um problema NP-Completo para o qual a existência de um algoritmo pseudopolinomial que o decida implica na existência de um algoritmo polinomial que o decida é chamado **NP-Completo forte**.

- ▷ Há problemas NP-Completos para os quais a existência de algoritmos pseudopolinomiais implica em $P = NP$.

Definição

Um problema NP-Completo para o qual a existência de um algoritmo pseudopolinomial que o decida implica na existência de um algoritmo polinomial que o decida é chamado **NP-Completo forte**.

- ▷ Todos os problemas com a "natureza" do Problema `Circuito Hamiltoniano` são NP-Completos fortes

Caixeiro-viajante

DADOS: Um grafo completo G com pesos $w\colon E(G) \to \mathbb{N}$, e um inteiro k

OBJETIVO: G possui um circuito Hamiltoniano com peso no máximo k?

Caixeiro-viajante

DADOS: Um grafo completo G com pesos $w\colon E(G) \to \mathbb{N}$, e um inteiro k

OBJETIVO: G possui um circuito Hamiltoniano com peso no máximo k?

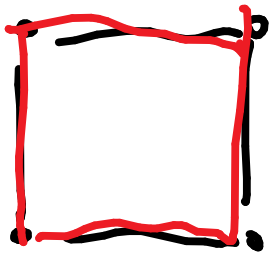
▷ É NP-Completo:

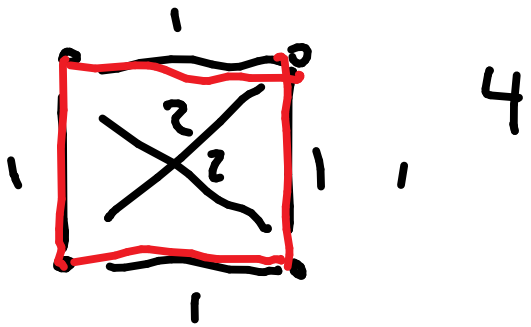
`Circuito Hamiltoniano` $\propto$ `Caixeiro-viajante`:

Dado grafo G, seja G' um grafo completo tal que $V(G') = V(G)$, e faça

$w(e) = 1$ se $e \in E(G)$

$w(e) = 2$ se $e \notin E(G)$





`Caixeiro-viajante`

DADOS: Um grafo completo G com pesos $w\colon E(G) \to \mathbb{N}$, e um inteiro k

OBJETIVO: G possui um circuito Hamiltoniano com peso no máximo k?

- ▷ É NP-Completo:
  `Circuito Hamiltoniano` $\propto$ `Caixeiro-viajante`:
  Dado grafo G, seja G' um grafo completo tal que $V(G') = V(G)$, e faça

  $$w(e) = 1 \text{ se } e \in E(G)$$
  $$w(e) = 2 \text{ se } e \notin E(G)$$

- ▷ Claramente, G possui circuito Hamiltoniano se e somente se G' possui um circuito Hamiltoniano de peso no máximo n.

Exemplo

Seja $W = \sum_{e \in E(G)} w(e)$, *e suponha que exista um algoritmo* A *de complexidade* $O\big((nWk)^c\big)$ *que decida* `Caixeiro-viajante`, *para algum* $c \in \mathbb{N}$.

Exemplo

Seja $W = \sum_{e \in E(G)} w(e)$, *e suponha que exista um algoritmo* A *de complexidade* $O\big((nWk)^c\big)$ *que decida* `Caixeiro-viajante`, *para algum* $c \in \mathbb{N}$.

$\hookrightarrow 2 \cdot n^2$

$w \leq 2n^2$

Esse algoritmo aplicado na instância obtida (G', w, n) *é polinomial no tamanho de* G, *pois* W *e* n *são polinomiais em* n.

$O\left(n \cdot 2 \cdot n^2 \cdot n\right) = O(n^4)$

Exemplo

Seja $W = \sum_{e \in E(G)} w(e)$, *e suponha que exista um algoritmo* A *de complexidade* $O\big((nWk)^c\big)$ *que decida* `Caixeiro-viajante`, *para algum* $c \in \mathbb{N}$.

Esse algoritmo aplicado na instância obtida (G', w, n) *é polinomial no tamanho de* G, *pois* W *e* n *são polinomiais em* n.

Logo, isso implica $P = NP$.

Exemplo

Seja $W = \sum_{e \in E(G)} w(e)$, *e suponha que exista um algoritmo* A *de complexidade* $O\big((nWk)^c\big)$ *que decida* `Caixeiro-viajante`, *para algum* $c \in \mathbb{N}$.

Esse algoritmo aplicado na instância obtida (G', w, n) *é polinomial no tamanho de* G, *pois* W *e* n *são polinomiais em* n.

Logo, isso implica $P = NP$.

Portanto, `Caixeiro-viajante` *é NP-Completo forte.*

Um algoritmo pseudopolinoimal é polinomial para instâncias nas quais o valor do maior dado é polionomial no tamanho da instância.

Um algoritmo pseudopolinoimal é polinomial para instâncias nas quais o valor do maior dado é polionomial no tamanho da instância.

Exemplo

O Problema da Mochila:

Um algoritmo pseudopolinoimal é polinomial para instâncias nas quais o valor do maior dado é polionomial no tamanho da instância.

Exemplo

O `Problema da Mochila`:

Se o peso total W for polinomial no tamanho da instância, i.e., $W = O(n^c)$, então o algoritmo $O(nW)$ é $O(n^{c+1})$.

Problemas Numéricos

Definição

Um problema no qual o valor do maior dado de uma instância não é necessariamente polinomial no tamanho da instância é chamado de **Problema Numérico**.

Definição

Um problema no qual o valor do maior dado de uma instância não é necessariamente polinomial no tamanho da instância é chamado de **Problema Numérico**.

Exemplo

Problema da Mochila

Definição

Um problema no qual o valor do maior dado de uma instância não é necessariamente polinomial no tamanho da instância é chamado de **Problema Numérico**.

Exemplo

Problema da Mochila

Exemplo

Fatorização de Inteiros

Como mostramos que um problema Π é NP-Completo fraco?

Como mostramos que um problema Π é NP-Completo fraco?

- ▷ Basta mostrar um algoritmo pseudopolinomial que o resolva

Como mostramos que um problema Π é NP-Completo fraco?

- ▷ Basta mostrar um algoritmo pseudopolinomial que o resolva

Como mostramos que um problema Π é NP-Completo forte?

Como mostramos que um problema Π é NP-Completo fraco?

- ▷ Basta mostrar um algoritmo pseudopolinomial que o resolva

Como mostramos que um problema Π é NP-Completo forte?

- ▷ Mostramos que se houver algoritmo pseudopolinomial para Π então existe algoritmo polinomial para um problema NP-Completo

Como mostramos que um problema Π é NP-Completo fraco?

- ▷ Basta mostrar um algoritmo pseudopolinomial que o resolva

Como mostramos que um problema Π é NP-Completo forte?

- ▷ Mostramos que se houver algoritmo pseudopolinomial para Π então existe algoritmo polinomial para um problema NP-Completo
- ▷ Equivalentemente, mostramos que há uma restrição de Π que é NP-Completa

Ex: $\{1, 3, 7\}$

$t = 6$

$t = 4$

Subconjunto Soma

DADOS: Um conjunto finito $S \subseteq \mathbb{N}$
e um inteiro t

OBJETIVO: existe $S' \subseteq S$ tal que $\sum_{s \in S'} s = t$?

Subconjunto Soma

DADOS: Um conjunto finito $S \subseteq \mathbb{N}$
e um inteiro t
OBJETIVO: existe $S' \subseteq S$ tal que $\sum_{s \in S'} s = t$?

- ▷ É um problema numérico, pois o valor de $s \in S$ não é limitado pelo tamanho de S.

Teorema

Subconjunto Soma é NP-Completo.

Teorema

Subconjunto Soma é NP-Completo.

Prova

$\triangleright$ *Cobertura por Vértices* $\propto$ *Subconjunto Soma*

Teorema

Subconjunto Soma é NP-Completo.

Prova

- ▷ *Cobertura por Vértices* $\propto$ *Subconjunto Soma*
- ▷ *Seja* (G, k) *uma instância do problema Cobertura por Vértices.*
- ▷ *Ordene as arestas e fixe os índices delas*

Teorema

Subconjunto Soma é NP-Completo.

Prova

- ▷ *Cobertura por Vértices* $\propto$ *Subconjunto Soma*
- ▷ *Seja* (G, k) *uma instância do problema Cobertura por Vértices.*
- ▷ *Ordene as arestas e fixe os índices delas*
- ▷ *Para a i-ésima aresta* e_i, *seja* $s_{e_i} = 4^i$

Teorema

Subconjunto Soma é NP-Completo.

Prova

- ▷ *Cobertura por Vértices* $\propto$ *Subconjunto Soma*
- ▷ *Seja* (G, k) *uma instância do problema Cobertura por Vértices.*
- ▷ *Ordene as arestas e fixe os índices delas*
- ▷ *Para a* i*-ésima aresta* e_i*, seja* $s_{e_i} = 4^i$
- ▷ *Para* $v \in V(G)$*, seja* $E(v)$ *o conjunto de arestas incidentes a* v *e seja*

$$s_v = 4^m + \sum_{e \in E(v)} s_e$$

Teorema

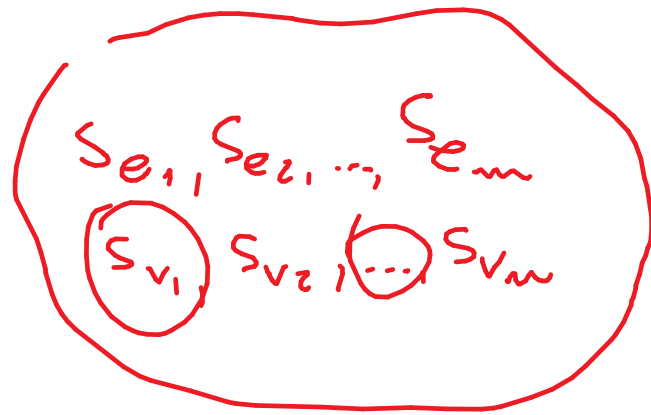
Subconjunto Soma é NP-Completo.

$m = |E(G)|$

Prova

- ▷ *Cobertura por Vértices* $\propto$ *Subconjunto Soma*
- ▷ *Seja* (G, k) *uma instância do problema Cobertura por Vértices.*
- ▷ *Ordene as arestas e fixe os índices delas*
- ▷ *Para a* i*-ésima aresta* e_i*, seja* $s_{e_i} = 4^i$
- ▷ *Para* $v \in V(G)$*, seja* $E(v)$ *o conjunto de arestas incidentes a* v *e seja*

$$s_v = 4^m + \sum_{e \in E(v)} s_e$$



- ▷ *Seja* $t = k4^m + \sum_{e \in E(G)} 2s_e$

Prova

$\triangleright$ $S = \{s_x : x \in V(G) \cup E(G)\}$.

Prova

- $S = \{s_x : x \in V(G) \cup E(G)\}$.

- *G possui uma cobertura por vértices de tamanho k se e somente se existe $S' \subseteq S$ tal que $\sum_{s \in S'} s = t$*

Teorema

Subconjunto Soma é NP-Completo fraco.

Teorema

- *Existe algoritmo de programação dinâmica de complexidade* $O\big(n(t+1)\big)$ *que resolve Subconjunto Soma.*
- *Para cada* $k \in \{1, \ldots, n\}$, *seja* $S_k = \{s_1, \ldots, s_k\}$.
- *Temos que Subconjunto Soma*$(S_k, t) =$ *SIM se e somente se*
  Subconjunto Soma$(S_{k-1}, t) =$ *SIM ou*
  Subconjunto Soma$(S_{k-1}, t - s_k) =$ *SIM.*

Denotaremos por $x_{i,j}$ o resultado de $\texttt{Subconjunto Soma}(S_i, j)$

$\texttt{Subconjunto Soma}(S, t)$

Para cada $i = 1, \ldots, n$

    $x_{i,0} = \text{SIM}$

Para cada $j = 1, \ldots, t$

    Para cada $i = 1, \ldots, n$

        se $x_{i-1,j} = SIM$ ou $x_{i-1,j-s_j} = SIM$,

            então $x_{i,j} = SIM$

            senão $x_{i,j} = \text{NÃO}$

Retorna $x_{n,t}$

Problemas NP-Completos

Fábio Botler

Programa de Engenharia de Sistemas e Computação
Universidade Federal do Rio de Janeiro

Π_1 EU ACHO QUE É Poly

Π_2 EU SEI QUE É Poly

OBJETIVO MOSTRAR QUE Π_1 É Poly

↳ PRECISO DE UMA REDUÇÃO DE Π_1 PARA Π_2

↳ UMA FUNÇÃO f QUE TRANSFORMA CADA INSTÂNCIA DE Π_1 EM UMA INSTÂNCIA DE Π_2 T.Q.

$$\underset{\text{SIM/NÃO}}{\Pi_1(I)} = \underset{\text{SIM/NÃO}}{\Pi_2(f(I))}$$

Π_1 EU SEI QUE É NP-COMPLETO

Π_2 EU ACHO QUE É NP-COMPLETO

OBJETIVO MOSTRAR QUE Π_2 É NP-COMPLETO

↳ PRECISO DE UMA REDUÇÃO DE Π_1 PARA Π_2

↳ UMA FUNÇÃO f QUE TRANSFORMA CADA INSTÂNCIA DE Π_1 EM UMA INSTÂNCIA DE Π_2 T.Q.

$$\underset{\text{SIM/NÃO}}{\Pi_1(I)} = \underset{\text{SIM/NÃO}}{\Pi_2(f(I))}$$