

Arka 23

Polynomial-Time versus NP-Complete Problems

I) Caminhos mais curtos versus Caminhos mais longos

II) Euler Tour versus Hamiltonian Cycle

• Euler Tour

Uma tour de Euler de um grafo $G = (V, E)$ é um ciclo de G em que cada arco de E ocorre exatamente uma vez; embora um vértice possa ocorrer mais do que uma vez.



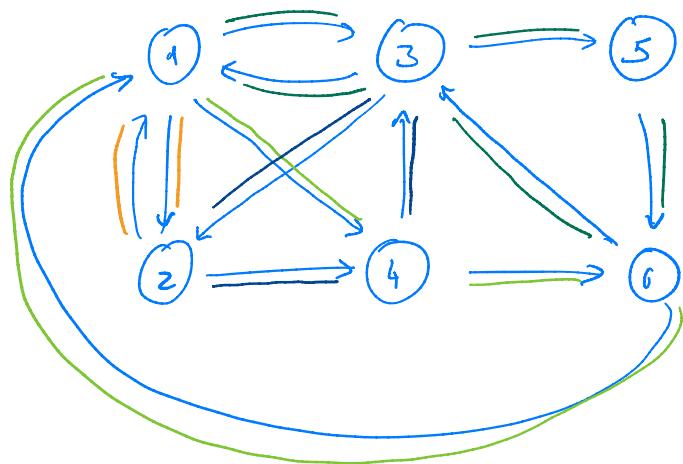
• Hamiltonian Cycle

Dado um grafo não dirigido $G = (V, E)$, um ciclo hamiltoniano de G é um ciclo simples de G que contém todos os vértices de V .

Resumo: Um grafo dirigido tem uma tour de Euler se o grafo é ligado e para todos os vértices $v \in V$:
 $\text{in-deg}(v) = \text{out-deg}(v)$

Euler Tours

Exempluz



- 1 - 3/3
- 2 - 2/2
- 3 - 3/3
- 4 - 2/2
- 5 - 1/1
- 6 - 2/2

1 4 1
↓
1 3 5 6 1 4 1
↓
1 3 5 6 1 4 3 2 4 1
↓
1 3 5 6 1 2 1 4 3 2 4 1

Polynomial-Time versus NP-Complete Problems

I) Caminhos mais curtos versus Caminhos mais longos

II) Euler Tour versus Hamiltonian Cycle

III) 2 CNF-SAT versus 3 CNF-SAT

2 CNF-SAT

- Ψ é UNSAT se e só se existe uma variável x tal que:

- $G_\Psi \vdash x \rightsquigarrow \neg x$
e.e.

- $G_\Psi \vdash \neg x \rightsquigarrow x$

Polyomial-Time versus NP-Complete Problems

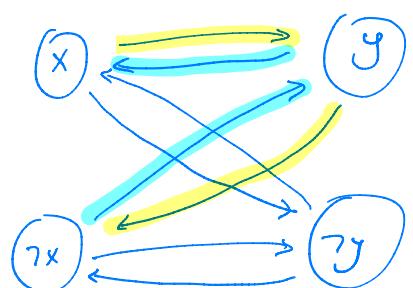
2 CNF-SAT

- Ψ é UNSAT se existe uma variável x tal que:

- $G_\Psi \vdash x \rightsquigarrow \neg x$
e.e.

- $G_\Psi \vdash \neg x \rightsquigarrow x$

$$\Psi = (x \vee y) \wedge (\neg x \vee \neg y) \wedge (x \vee \neg y) \wedge (\neg x \vee \neg y)$$
$$\begin{array}{llll} \neg x \Rightarrow y & x \Rightarrow \neg y & y \Rightarrow x & \neg y \Rightarrow \neg x \\ \neg y \Rightarrow x & \neg y \Rightarrow \neg x & \neg x \Rightarrow \neg y & x \Rightarrow \neg \neg y \end{array}$$



Polyomial-Time versus NP-Complete Problems

2 CNF-SAT

- Ψ é UNSAT se existe uma variável x tal que:

- $G_\Psi \vdash x \rightsquigarrow \neg x$
 $\&&$

- $G_\Psi \vdash \neg x \rightsquigarrow x$

Proof.

- Suponhamos que: $G_\Psi \vdash x \rightsquigarrow \neg x$ e $G_\Psi \vdash \neg x \rightsquigarrow x$.
- Suponhamos, por contradição, que Ψ é satisfazível. Conduzimos que existe uma interpretação ρ tal que $\rho(\Psi) = 1$.
- Dáis duas a considerar: $\rho(x) = 1 \vee \rho(x) = 0$.

① $\rho(x) = 1$ $\rightarrow x \Rightarrow x_1, x_1 \Rightarrow x_2, \dots, x_n \Rightarrow \neg x$
 $G_\Psi \vdash x \rightsquigarrow \neg x$ \Downarrow
 $\rho(\neg x) = 1 \Leftarrow \rho(x) = 0 \quad \text{X}$

② Análoga a ①

Polynomial-Time versus NP-Complete Problems

I Caminhos mais curtos versus Caminhos Mais longos

II Euler Tour versus Hamiltonian Cycle

III 2 CNF-SAT versus 3 CNF SAT

IV MST vs TSP

(Minimum Spanning Tree versus Travelling Salesman)

Problemas de Decisão vs Problemas de Optimização

• Problemas de Decisão

- Problemas cuja solução é sim/não
- Exemplos: SAT, Primes, Euler Tour, Hamiltonian etc
- Formalmente, um problema de decisão X corresponde ao conjunto das instâncias que satisfazem a condição do problema.

$$\text{Exemplo: } \text{Primes} = \{ n \in \mathbb{N} \mid \exists m. m+1 \leq n \wedge m \mid n \}$$

$$\text{SAT} = \{ \Psi \mid \text{Não existe valoração } \rho \text{ tal que: } \rho(\Psi) = 1 \}$$

• Problemas de Optimização

- Exemplos: caminhos mais curtos, caminhos mais longos, fluxo máximo etc
- Podem ser reformulados como problemas de decisão

Exemplo: Caminhos mais curtos

$$\text{SPath} = \{ \langle G, s, t, k \rangle \mid s, t \in G.V \text{ e } G \models s \xrightarrow{k} t \}$$

• $(G, s, t, k) \in \text{SPath}$ se existir um caminho entre s e t em G com no máximo k arcos

Algoritmos de Decisão versus Algoritmos de Verificação

• Algoritmos de Decisão

O algoritmo A decide o problema X se:

$$\forall x \in \Sigma. \quad x \in X \Leftrightarrow A(x) = 1$$

Exemplo:

$A_{SAT}(\psi)$: decide se ψ é satisfazível

• Algoritmos de Verificação

O algoritmo A verifica o problema X se:

$$\forall x \in \Sigma. \quad x \in X \Leftrightarrow \exists y. \quad A(x, y) = 1$$

↳ certificado

$A_{SAT}(\psi, \rho)$: verifica se ψ é satisfazível

Certificado: valores em ρ que satisfazem ψ ($\rho(\psi) = 1$)

$A_{Composite}(n, (m_1, \dots, m_k))$: verifica se n é um nº composto

Certificado: lista de nºs cujo produto é n

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)

Proposição 1: $P \subseteq NP$

- Suponhamos $\bar{g} X \in P$, temos de provar que $X \in NP$.
- Como $X \in P$, concluimos que existe um algoritmo A que decide X em tempo polinomial:
 $x \in X \Leftrightarrow A(x) = 1$

- Temos de mostrar que existe um algoritmo A' que verifica X em tempo polinomial. Definimos A' da seguinte:

$$A'(x, y) = A(x)$$

\hookrightarrow o certificado pode ser a string vazia

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- Exp - conjunto dos problemas decidíveis em tempo exponencial

Proposição 2: $NP \subseteq Exp$

- Suponhamos $\exists X \in NP$, temos de provar que $X \in Exp$.
- Como $X \in NP$, concluímos que existe um algoritmo A que verifica X em tempo polinomial:
 $x \in X \Leftrightarrow \exists y. |y| \leq p(|x|) \wedge A(x, y) = 1$
- Temos de mostrar que existe um algoritmo A' que decide X em tempo exponencial.
Definimos A' da seguinte forma:
$$A'(x) = \begin{cases} \text{for each } y \text{ of size } \leq |p(x)| \\ \quad \text{if } A(x, y) \\ \quad \quad \quad \text{then return 1} \\ \quad \quad \quad \text{otherwise return 0} \end{cases}$$

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- EXP - conjunto dos problemas decidíveis em tempo exponencial
- co-NP - conjunto dos problemas cujo complemento está em NP

Exemplo:

$$\text{UNSAT} = \{ \Phi \mid \Phi \text{ não é satisfazível} \}$$

Proposição 3: $P \subseteq NP \cap \text{co-NP}$

- $P \subseteq NP$ (proposição 1)
- $P \subseteq \text{co-NP}$ (fazemos)
 $\vdash x \in P \Rightarrow \bar{x} \in \text{co-NP} \Rightarrow \bar{x} \in NP$
mas $\bar{x} \in P \Rightarrow \bar{x} \in P$

• Temos \bar{x} provado que existe um algoritmo A' que decide \bar{x} .

Seja A o algoritmo que decide x .

$A'(x) :$
; if ($A(x)$)
; ; then return 0
; ; else return 1

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- EXP - conjunto dos problemas decidíveis em tempo exponencial
- co-NP - conjunto dos problemas cujo complemento está em NP

Exemplo:

$$\text{UNSAT} = \{ \Phi \mid \Phi \text{ não é satisfazível} \}$$

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

- Suponhamos que $X \in NP$, temos de provar que $\bar{X} \in NP$.
- Da $X \in NP$, concluímos que existe um algoritmo A tal que:
 $x \in X \iff \exists y. A(x, y) = 1$

↓

$$x \notin X \iff \nexists y. A(x, y) = 1$$

Temos de experimentar todas as certificações!

→ Não conseguimos usar o verificador de X para construir o verificador de \bar{X} .

Fail!

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- EXP - conjunto dos problemas decidíveis em tempo exponencial
- co-NP - conjunto dos problemas cujo complemento está em NP

Exemplo:

$$\text{UNSAT} = \{ \Phi \mid \Phi \text{ não é satisfazível} \}$$

(characterização alternativa: $\chi \in \text{co-NP}$ se e só se existe um algoritmo de verificação polinomial)

A tal que:

$$x \in \chi \Leftrightarrow \forall y \in \text{Cert}(\chi). A(x, y) = 0$$

• Em geral, não conseguimos provar em tempo polinomial $\bar{\Psi}$ uma dada fórmula Ψ pertencente a UNSAT, mas conseguimos provar $\bar{\Psi}$ não pertence.

$$x \notin \chi \Leftrightarrow \exists y \in \text{Cert}(\chi). A(x, y) = 1$$

Classes de Complexidade

- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- EXP - conjunto dos problemas decidíveis em tempo exponencial
- co-NP - conjunto dos problemas cujo complemento está em NP
Exemplo:
 $\text{UNSAT} = \{ \Phi \mid \Phi \text{ não é satisfazível} \}$
- $X \in NP$ - conseguimos confirmar a "pertença" a X em tempo polinomial
Exemplo: SAT
- $X \in \text{co-}NP$ - conseguimos confirmar a "não-pertença" a X em tempo polinomial
Exemplo: UNSAT

Classes de Complexidade

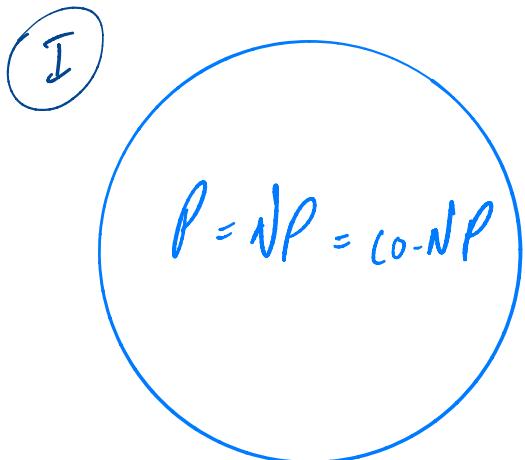
- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- EXP - conjunto dos problemas decidíveis em tempo exponencial
- co-NP - conjunto dos problemas cujo complemento está em NP
Exemplo:
 $\text{UNSAT} = \{ \Phi \mid \Phi \text{ não é satisfazível} \}$
- $X \in NP$ - conseguimos confirmar a "pertença" a X em tempo polinomial
Exemplo: SAT
- $X \in \text{co-NP}$ - conseguimos confirmar a "não-pertença" a X em tempo polinomial
Exemplo: UNSAT

Classes de Complexidade

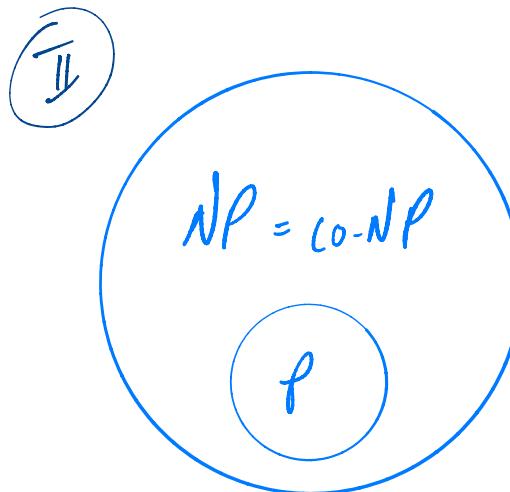
- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- EXP - conjunto dos problemas decidíveis em tempo exponencial
- co-NP - conjunto dos problemas cujo complemento está em NP
Exemplo:
 $\text{UNSAT} = \{ \Phi \mid \Phi \text{ não é satisfazível} \}$
- $NP \cap \text{co-}NP$
 - classe dos problemas para os quais conseguimos verificar a pertença e não-pertença em tempo polinomial

Classes de Complexidade - Conjecturas

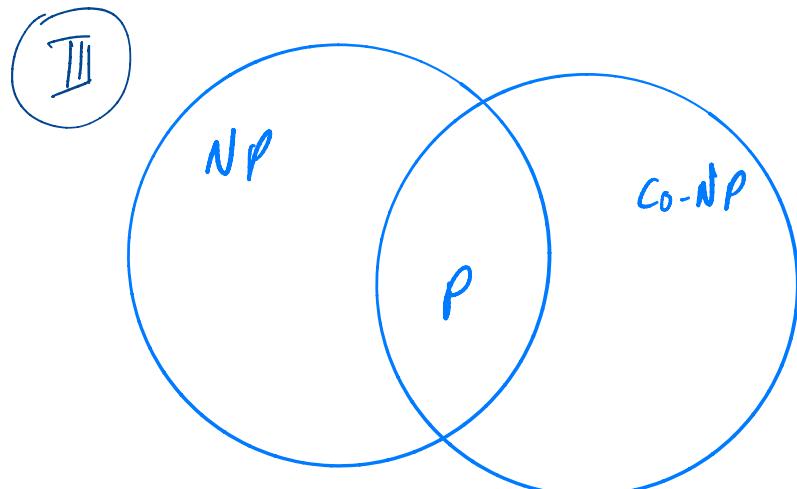
* 4 possibilidades



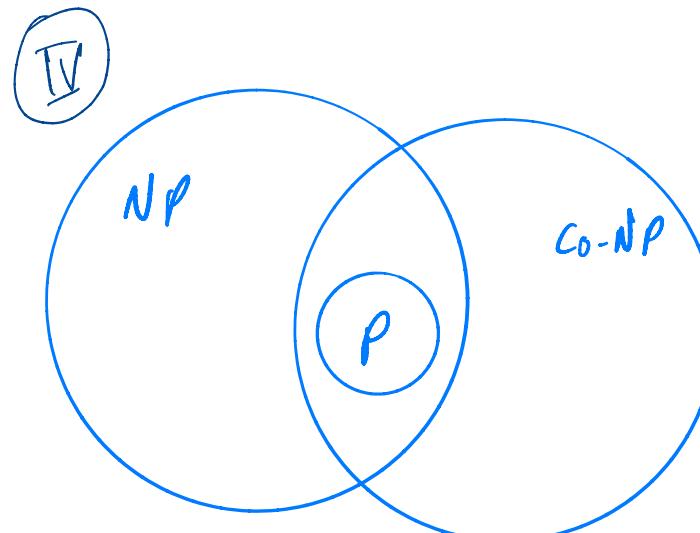
$P = NP?$



$NP = \text{Co-NP}?$



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



Classes de Complexidade - TPC

- $NP \neq co-NP \Rightarrow P \neq NP$
- P é fechada para a intersecção, união, complemento, concatenação e fecho de Kleene.
 - $X_1, X_2 \in P \Rightarrow X_1 \cap X_2 \in P$
 - $X_1, X_2 \in P \Rightarrow X_1 \cup X_2 \in P$
 - $X_1 \in P \Rightarrow \bar{X}_1 \in P$
 - $X_1, X_2 \in P \Rightarrow X_1 \cdot X_2 \in P$
 - $X_1 \in P \Rightarrow X_1^* \in P$
- P é fechada para a intersecção, união, concatenação e fecho de Kleene.

Redutibilidade NP

- O problema X é reduzível em tempo polinomial (polynomial-time reducible) ao problema Y se existe uma função f calculável em tempo polinomial tal que:

$$x \in X \Leftrightarrow f(x) \in Y$$

Escrivemos: $X \leq_p Y$

Proposição: $Y \in P \wedge X \leq_p Y \Rightarrow X \in P$

Prova:

- Suponhamos que $Y \in P$ e $X \leq_p Y$, há que provar que $X \in P$.
- Seja A o algoritmo que decide Y e f a função que reduz X a Y .
- Temos que construir um algoritmo A' que decide X em tempo polinomial

$A'(x)$:

retorna $A(f(x))$

Completo de NP

- Um problema X diz-se *NP-difícil* se:

$$\forall Y \in NP. Y \leq_p X$$

- Um problema X diz-se *NP-completo* se:
 - $X \in NP$
 - X é *NP-difícil*

Proposição 5: Se um problema *NP-completo* for resolvível em tempo polinomial então $P = NP$.

Prova:

- Assumindo que existe $X \in P$ tal que X é *NP-difícil*, temos de provar que $\forall Y \in NP. Y \in P$.
- Tomemos um qualque $Y \in NP$; como X é *NP-difícil*, concluimos que existe h , calculável em tempo polinomial, tal que:
 $y \in Y \Leftrightarrow h(y) \in X$

- Seja A o algoritmo polinomial que decide X , definimos o algoritmo A' que decide Y em tempo polinomial como se segue:
 $A'(y) :$
return $A(h(y))$

Completo de NP

- Um problema X diz-se NP -difícil sse:
 $\forall Y \in NP. Y \leq_p X$

- Um problema X diz-se NP -completo sse:
 - $X \in NP$
 - X é NP -difícil

- Seja C uma classe de problemas, dizemos que X é completo para C sse:
 - $X \in C$
 - $\forall Y \in C. Y \leq_p X$
- } Generalização do conceito de completo

Proposição 6: X é completo para NP sse \bar{X} é completo para co-NP.

⊬ Suponhamos \bar{X} é completo para NP , queremos provar que \bar{X} é completo para co-NP.
Para todo $Y \in \text{co-NP}$, há \bar{Y} mostrando $\bar{Y} \leq_p \bar{X}$.

$$\begin{array}{c} \downarrow \\ Y \in \text{co-NP} \Rightarrow \bar{Y} \in NP \Rightarrow \bar{Y} \leq_p \bar{X} \end{array} \quad \begin{array}{c} \nearrow \\ \forall y \in \text{dom}(Y). y \in Y \Leftrightarrow f(y) \notin \bar{X} \\ \forall y \in \text{dom}(Y). y \notin Y \Leftrightarrow f(y) \in \bar{X} \end{array} \quad \begin{array}{c} \nearrow \\ \forall y \in \text{dom}(Y). y \in Y \Leftrightarrow f(y) \notin \bar{X} \\ \forall y \in \text{dom}(Y). y \notin Y \Leftrightarrow f(y) \in \bar{X} \end{array} \quad \begin{array}{c} \nearrow \\ \bar{Y} \leq_p \bar{X} \end{array}$$

Completo de NP

- Um problema X diz-se NP-difícil se:
 $\forall Y \in NP. Y \leq_p X$

- Um problema X diz-se NP-completo se:
 - $X \in NP$
 - X é NP-difícil

Proposição 7: $X \in NP \wedge Y \leq_p X \wedge Y \in NPC \Rightarrow X \in NPC$

Prova: Há que provar que $\forall Z \in NP. Z \leq_p X$

- Tomemos $Z \in NP$.
- Como $Y \in NPC$, temos que $Z \leq_p Y$.
- De $Y \leq_p X$ e $Z \leq_p Y$ segue que $Z \leq_p Y$ (pela transitividade de \leq_p).

Completo de NP

- Um problema X diz-se **NP-completo** se:
 - $X \in NP$
 - X é **NP-difícil**
- Um problema X diz-se **NP-difícil** se:
 $\forall Y \in NP. Y \leq_p X$

Proposição 7: $X \in NP \wedge Y \leq_p X \wedge Y \in NPC \Rightarrow X \in NPC$

* Como provar que um problema X é **NP-completo**?

① Provar que X está em **NP**

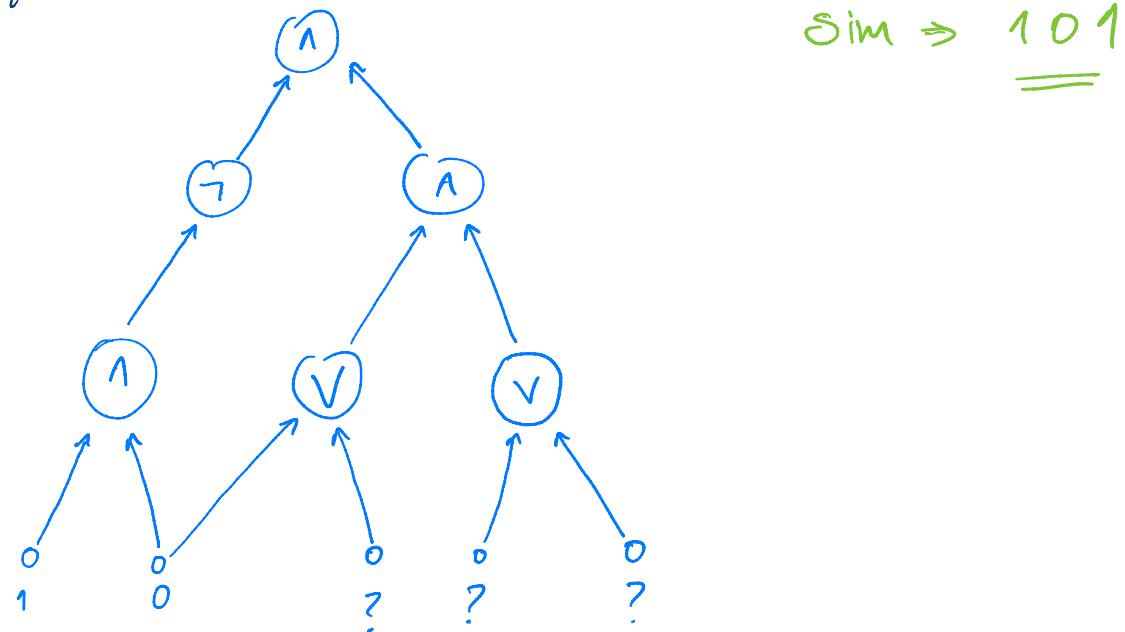
↳ Descobrir o certificado g nos permite
verificar X em tempo polinomial (fácil)

② Selecionar um problema **NP-completo** $Y \in (difícil)$
construir uma redução $Y \leq_p X$.

O 1º Problema NP-Completo

Circuit-SAT: Dado um circuito combinatorio construído com portas And, Or e Not, existem inputs que fornecem o output do circuito 1.

Exemplo:



O 1º Problema NP-Completo

Circuit-SAT: Dado um circuito combinatório construído com portas And, Or e Not, existem inputs \bar{y} fornecendo o output do circuito $\underline{1}$.

Teorema [Cook-Lovlin] Circuit-SAT é NP-completo.

Esboço de prova:

- Tome-se $X \in NP$. De definição de NP segue que existe um algoritmo de verificação A tal que:
 $x \in X \Leftrightarrow \exists y. A(x, y) = 1$
- Um algoritmo polinomial pode ser implementado por um circuito combinatório de tamanho polinomial. Seja K esse circuito
- Fixamos as l_x^s primeiras entradas de K com os bits de x . As restantes $|y|$ entradas ficam com ?.
- O circuito K é satisfazível se $\exists y. A(x, y) = 1$ (sse $x \in X$).

Ayla ZG

Reduc Map

Circuit-SAT



CNF-SAT



3-CNF-SAT



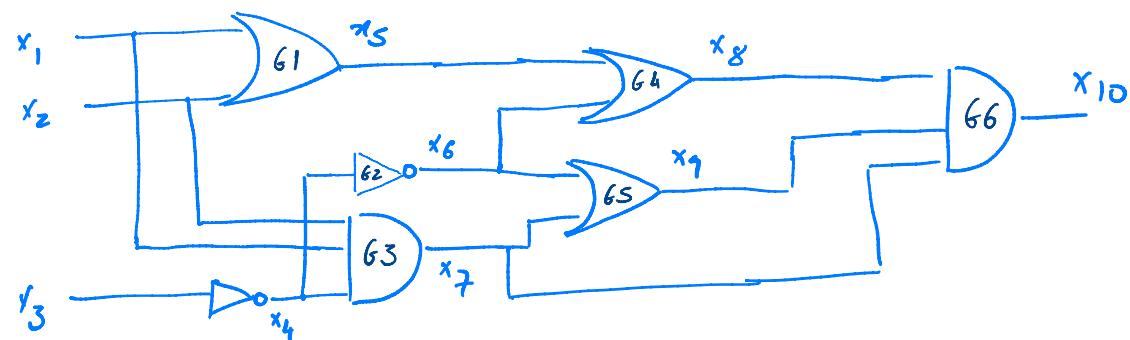
Clique



Vertex-Cover

Circuit SAT \leq_p SAT

Exemplo de Circuito:

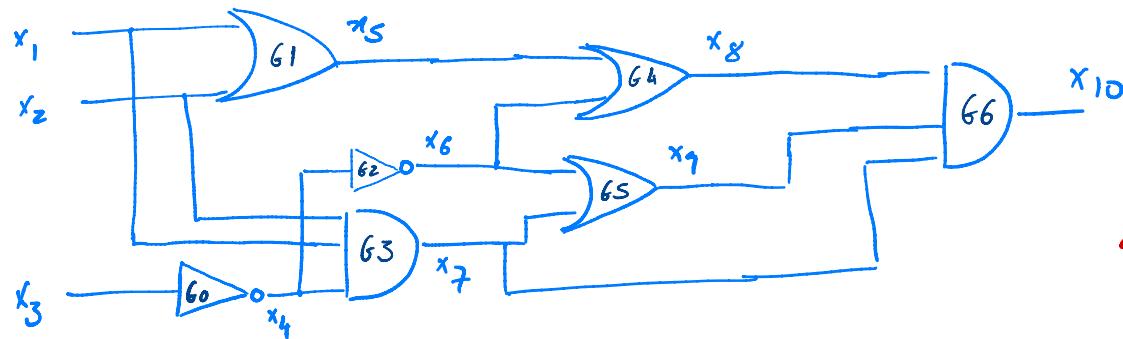


• Reduzão

$$\underbrace{\text{VAR_Out}}_{\substack{\text{variáveis} \\ \text{de output}}} \wedge \bigwedge_{i=1}^n \underbrace{\text{Formula}_i(G_i)}_{\substack{\text{formula SAT} \\ \text{descritiva} \\ \text{a pntz } G_i}}$$

Circuit SAT \leq_p CNF-SAT

Exemplo de circuito:



- Problema: Pontas lógicas cujas fórmulas não são clausulas!

$$\underbrace{\text{Var_Out}}_{\substack{\text{variável} \\ \text{de output}}} \wedge \bigwedge_{i=1}^n \underbrace{\text{Fórmula } (G_i)}_{\substack{\text{fórmula SAT} \\ \text{que descreve} \\ \text{a ponta } G_i}}$$

$$G_1: x_5 \Leftrightarrow x_1 \vee x_2 \rightarrow \begin{cases} x_5 \Rightarrow x_1 \vee x_2 \rightarrow \neg x_5 \vee x_1 \vee x_2 \\ x_1 \vee x_2 \Rightarrow x_5 \rightarrow \neg(x_1 \vee x_2) \vee x_5 \end{cases}$$

$$G_0: \neg x_3 \Leftrightarrow x_4 \rightarrow \begin{cases} \neg x_3 \Rightarrow x_4 \rightarrow x_3 \vee x_4 \\ x_4 \Rightarrow \neg x_3 \rightarrow \neg x_4 \vee \neg x_3 \end{cases}$$

$$G_3: x_4 \wedge x_1 \wedge x_2 \Leftrightarrow x_7 \rightarrow \begin{cases} x_4 \wedge x_1 \wedge x_2 \Rightarrow x_7 \rightarrow \neg(x_4 \wedge x_1 \wedge x_2) \vee x_7 \\ x_7 \Rightarrow x_4 \wedge x_1 \wedge x_2 \rightarrow \neg x_7 \vee (x_4 \wedge x_1 \wedge x_2) \end{cases}$$

Circuito SAT \leq_p CNF-SAT

$$G1: \quad x_5 \Leftrightarrow x_1 \vee x_2 \rightarrow \begin{cases} x_5 \Rightarrow x_1 \vee x_2 & \rightarrow \neg x_5 \vee x_1 \vee x_2 \\ x_1 \vee x_2 \Rightarrow x_5 & \rightarrow \neg(x_1 \vee x_2) \vee x_5 \end{cases}$$

- Como construir para cada porta lógica uma fórmula CNF?

① Tabela de verdade de $\neg\phi$

② A partir de ①, construir uma representação DNF de $\neg\phi$

DNF - Disjunctive Normal Form
 \hookrightarrow "Ors de Ands"

③ A partir de ②, usar as Leis de DeMorgan para obter uma representação CNF de $\phi^T (= \top \neg \phi)$

CNF - Conjunctive Normal Form
 \hookrightarrow "AND de Ors"

Circuit SAT \leq_p CNF-SAT

Exemplo: $\phi = x_5 \Leftrightarrow x_1 \vee x_2$

①

x_1	x_2	x_5	$x_1 \vee x_2$	$x_5 \Leftrightarrow x_1 \vee x_2$	$\rightarrow (x_5 \Leftrightarrow x_1 \vee x_2)$
0	0	0	0	1	0
0	0	1	0	0	1
0	1	0	1	0	1
0	1	1	1	1	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	0

②

$$\begin{aligned} \neg\phi \equiv & (\neg x_1 \wedge \neg x_2 \wedge \neg x_5) \vee (\neg x_1 \wedge x_2 \wedge \neg x_5) \\ & \vee (x_1 \wedge \neg x_2 \wedge \neg x_5) \vee (x_1 \wedge x_2 \wedge \neg x_5) \end{aligned}$$

Circuito SAT \leq_p CNF-SAT

Exemplo: $\phi = x_5 \Leftrightarrow x_1 \vee x_2$

$$\textcircled{2} \quad \neg\phi \equiv (\neg x_1 \wedge \neg x_2 \wedge x_5) \vee (\neg x_1 \wedge x_2 \wedge \neg x_5) \\ \vee (x_1 \wedge \neg x_2 \wedge \neg x_5) \vee (x_1 \wedge x_2 \wedge \neg x_5)$$

$$\textcircled{3} \quad \neg\neg\phi = \phi :$$

$$\phi = (x_1 \vee x_2 \vee \neg x_5) \wedge (x_1 \vee \neg x_2 \vee x_5) \\ \wedge (\neg x_1 \vee x_2 \vee x_5) \wedge (\neg x_1 \vee \neg x_2 \vee x_5)$$

Importante: Ité que mover a gá a construção da fórmula lógica se faz em tempo polinomial no tamanho do circuito.

Circuit SAT \leq_p CNF-SAT

Importante: Há que provar q a construção da fórmula lógica se faz em tempo polinomial no tamanho do circuito.

- Construção da fórmula de cada porta lógica: $O(1)$

↳ Admitindo que têm um n° de entradas inferiores a uma constante

- Construção da fórmula do circuito: $O(n)$

↳ n - n° de portas lógicas

Food for Thought: O q fizemos com circuitos cujas portas lógicas têm um n° arbitrário de entradas?

CNF-SAT \leq_p 3-CNF-SAT

- Input: Fórmula CNF, $\phi = C_1 \wedge C_2 \wedge \dots \wedge C_n$ $\xrightarrow{\text{Cada cláusula de } \phi \text{ pode ter um nº arbitrário de literais}}$
- Output: Fórmula 3-CNF, $\phi' = C'_1 \wedge \dots \wedge C'_m$
 \hookrightarrow Cada cláusula de ϕ' tem exactamente 3 literais

Ideia Chave: Vamos transformar cada cláusula de ϕ num conjunto de cláusulas com 3 literais cada

$$C_i \rightarrow C_{i1}^!, \dots, C_{ik}^!$$

- 4 casos a considerar; o número de literais de C_i é:

- = 1 $\rightarrow ?$
- = 2 $\rightarrow ?$
- = 3 $\rightarrow \checkmark$
- $> 3 \rightarrow ?$

CNF-SAT \leq_p 3-CNF-SAT

Ideia Chave: $C_i \rightarrow C_{i_1}^1, \dots, C_{i_k}^1$

- 4 casos a considerar; o número de literais de C_i é:

- = 1 $\rightarrow ?$
- = 2 $\rightarrow ?$
- = 3 $\rightarrow \checkmark$
- > 3 $\rightarrow ?$

Exemplo: $\neg x_1 \vee x_2$

- Uma nova variável y , 2 cláusulas de output

$$(\neg x_1 \vee x_2 \vee y)$$

$$(\neg x_1 \vee x_2 \vee \neg y)$$

• Qualquer valoração \bar{x} não satisfaz $\neg x_1 \vee x_2$
fazendo uma das cláusulas

CNF-SAT \leq_p 3-CNF-SAT

Ideia Chave: $C_i \rightarrow C_{i_1}^1, \dots, C_{i_k}^1$

- 4 casos a considerar; o número de literais de C_i é:

- = 1 $\rightarrow ?$
- = 2 \rightarrow 1 nova variável + 2 novas cláusulas
- = 3 $\rightarrow \checkmark$
- $> 3 \rightarrow ?$

• Seja $C_i = l_1 \vee l_2$

- Uma nova variável y , 2 cláusulas de output

$$(l_1 \vee l_2 \vee y)$$

$$(l_1 \vee l_2 \vee \neg y)$$

CNF-SAT \leq_p 3-CNF-SAT

Ideia Chave: $C_i \rightarrow C_{i_1}^1, \dots, C_{i_k}^1$

- 4 casos a considerar; o número de literais de C_i é:

- = 1 \rightarrow ?
- = 2 \rightarrow 1 nova variável + 2 novas cláusulas
- = 3 \rightarrow ✓
- > 3 \rightarrow ?

Exemplo: $\exists x$

- duas novas variáveis y_1 e y_2 , 4 cláusulas de output

$$(y_1 \vee y_2 \vee \neg x)$$

$$(\neg y_1 \vee y_2 \vee \neg x)$$

$$(\neg y_1 \vee \neg y_2 \vee \neg x)$$

$$(\neg y_1 \vee \neg y_2 \vee y_1)$$

Qualquer valoração \bar{y} não satisfaz
 $\neg x$ em uma das 4 cláusulas.

CNF-SAT \leq_p 3-CNF-SAT

Ideia Chave: $C_i \rightarrow C_{i_1}^1, \dots, C_{i_k}^1$

- 4 casos a considerar; o número de literais de C_i é:

- = 1 \rightarrow 2 novas variáveis, 4 cláusulas de output
- = 2 \rightarrow 1 nova variável + 2 novas cláusulas
- = 3 \rightarrow ✓
- $> 3 \rightarrow ?$

• Seja $C_i = l$

- duas novas variáveis y_1 e y_2 , 4 cláusulas de output

$$(y_1 \vee y_2 \vee l)$$

$$(y_1 \vee \neg y_2 \vee l)$$

$$(\neg y_1 \vee y_2 \vee l)$$

$$(\neg y_1 \vee \neg y_2 \vee l)$$

CNF-SAT \leq_p 3-CNF-SAT

Ideia Chave: $C_i \rightarrow C_{i_1}^1, \dots, C_{i_k}^1$

- 4 casos a considerar; o número de literais de C_i é:

- = 1 \rightarrow 2 novas variáveis + 4 novas cláusulas
- = 2 \rightarrow 1 nova variável + 2 novas cláusulas
- = 3 \rightarrow ✓
- > 3 \rightarrow ?

• Exemplo: $(x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4)$

$x_1 \vee \neg x_2$ $x_3 \vee \neg x_4$

só precisamos de satisfazer uma das cláusulas

$$x_1 \vee \neg x_2 \vee y \quad \wedge \quad \neg y \vee x_3 \vee \neg x_4$$

\hookrightarrow O \underline{j} escolhe a cláusula original \bar{y} para satisfazer

CNF-SAT \leq_p 3-CNF-SAT

Ideia Chave: $C_i \rightarrow C_{i_1}^1, \dots, C_{i_k}^1$

- 4 casos a considerar; o número de literais de C_i é:

- = 1 \rightarrow 2 novas variáveis + 4 novas cláusulas
- = 2 \rightarrow 1 nova variável + 2 novas cláusulas
- = 3 \rightarrow ✓
- > 3 \rightarrow ?

• Exemplo: $(x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4 \vee x_5)$

$$\underline{(x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4 \vee x_5)}$$
$$(x_1 \vee \neg x_2 \vee y_1) \vee (\neg y_1 \vee x_3 \vee y_2) \vee (\neg y_2 \vee \neg x_4 \vee x_5)$$

CNF-SAT \leq_p 3-CNF-SAT

Ideia Chave: $C_i \rightarrow C'_{i_1}, \dots, C'_{i_k}$

- 4 casos a considerar; o número de literais de C'_i é:

- = 1 \rightarrow 2 novas variáveis + 4 novas cláusulas
- = 2 \rightarrow 1 nova variável + 2 novas cláusulas
- = 3 \rightarrow ✓
- $> 3 \rightarrow ?$

• Seja $C = l_1 \vee \dots \vee l_k$, $k \geq 4$

$$C'_1 = l_1 \vee l_2 \vee y_1$$

$$C'_2 = \neg y_1 \vee l_3 \vee j_2$$

:

$$C'_j = \neg j_{j-1} \vee l_{j+1} \vee j_j$$

:

$$C'_{k-3} = \neg j_{k-4} \vee l_{k-2} \vee j_{k-3}$$

$$C'_{k-2} = \neg j_{k-3} \vee l_{k-1} \vee l_{k-2}$$

Proposição: (C é satisfazível) $\Leftrightarrow \bigwedge_{i=1}^{k-2} C'_i$ é satisfazível

Proposição: P satisfaz C $\Leftrightarrow P$ satisfaz $\bigwedge_{i=1}^{k-2} C'_i$

CNF-SAT \leq_p 3-CNF-SAT

Proposição: ρ satisfaz C se e só se existe uma extensão de ρ , ρ' , que satisfaz $\bigwedge_{i=1}^{k-2} C'_i$

- $C = l_1 \vee \dots \vee l_k$, $k \geq 4$

$$C'_1 = l_1 \vee l_2 \vee j_1$$

$$C'_2 = \neg j_1 \vee l_3 \vee j_2$$

:

$$C'_j = \neg j_{j-1} \vee l_{j+1} \vee j_j$$

:

$$C'_{k-3} = \neg j_{k-4} \vee l_{k-2} \vee j_{k-3}$$

$$C'_{k-2} = \neg j_{k-3} \vee l_{k-1} \vee l_{k-2}$$

\Rightarrow Existe $1 \leq i \leq k$ tal que $\rho(l_i) = 1$

$$\text{Seja } j = \begin{cases} 1 & \text{se } i = 1, 2 \\ k-2 & \text{se } i = k-1, k \\ i-1 & \text{c.c.} \end{cases}$$

Concluímos que $\rho'(C'_j) = 1$ para qualquer ρ' que estende ρ

- Temos de "encontrar" a extensão de ρ , ρ' que satisfaz todas as outras cláusulas.

$$\rho'(y_l) = \begin{cases} 1 & \text{se } l \leq i-2 \\ 0 & \text{se } l \geq i-1 \end{cases} \quad \rho'(x_l) = f(x_l) \quad \forall x_l \in \text{dom}(C)$$

CNF-SAT \leq_p 3-CNF-SAT

Proposição: ρ satisfaz C se e só se ρ satisfaz $\bigwedge_{i=1}^{k-2} C'_i$

- $C = l_1 \vee \dots \vee l_k$, $k \geq 4$

$$C'_1 = l_1 \vee l_2 \vee j_1$$

$$C'_2 = \neg j_1 \vee l_3 \vee j_2$$

:

$$C'_j = \neg j_{j-1} \vee l_{j+1} \vee j_j$$

:

$$C'_{k-3} = \neg j_{k-4} \vee l_{k-2} \vee j_{k-3}$$

$$C'_{k-2} = \neg j_{k-3} \vee l_{k-1} \vee l_{k-2}$$

\Leftarrow Suponhamos por contradição que ρ' satisfaz $\bigwedge_{i=1}^{k-2} C'_i$ e ρ' não satisfaz C .

- $\forall i \leq k$. $\rho'(l_i) = 0$

- Isto significa que $\underbrace{\rho(j_1) = 1, \dots, \rho(j_{k-3}) = 1}$

↓

para satisfazermos as clausulas C'_1, \dots, C'_{k-3}

- A clausula C'_{k-2} não é satisfeita por ρ' .

3-CNF-SAT \leq_p Clique

Clique: Um clique num grafo não dirigido $G = (V, E)$ é subconjunto $V' \subseteq V$ tal que todos os pares de vértices em V' estão ligados por um arco em E .

V' é um clique de G sse $\forall u, v \in V'. (u, v) \in E$

Problema Clique:

Clique = $\{ \langle G, k \rangle \mid G \text{ é um grafo não dirigido que contém um clique de tamanho } k \}$

Proposição: O problema Clique é NP-completo.

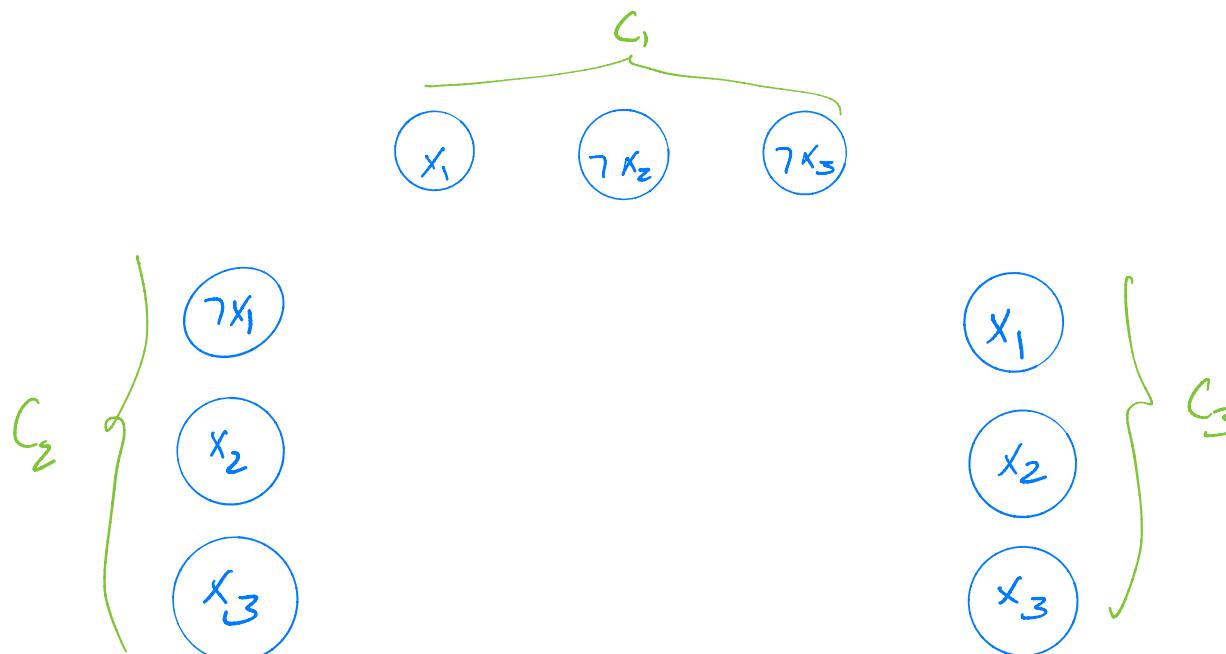
3-CNF-SAT \leq_p Clique

Hipótese: O problema Clique é NP-completo.

- Seja ϕ uma fórmula proposicional no formato 3-CNF, temos de calcular um grafo $G = (V, E)$ e um inteiro k tal que ϕ é satisfazível se e só se G tem um clique de tamanho k .

- Exemplo:

$$\phi = (\underbrace{(x_1 \vee \neg x_2 \vee \neg x_3)}_{C_1} \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_3)}_{C_3})$$



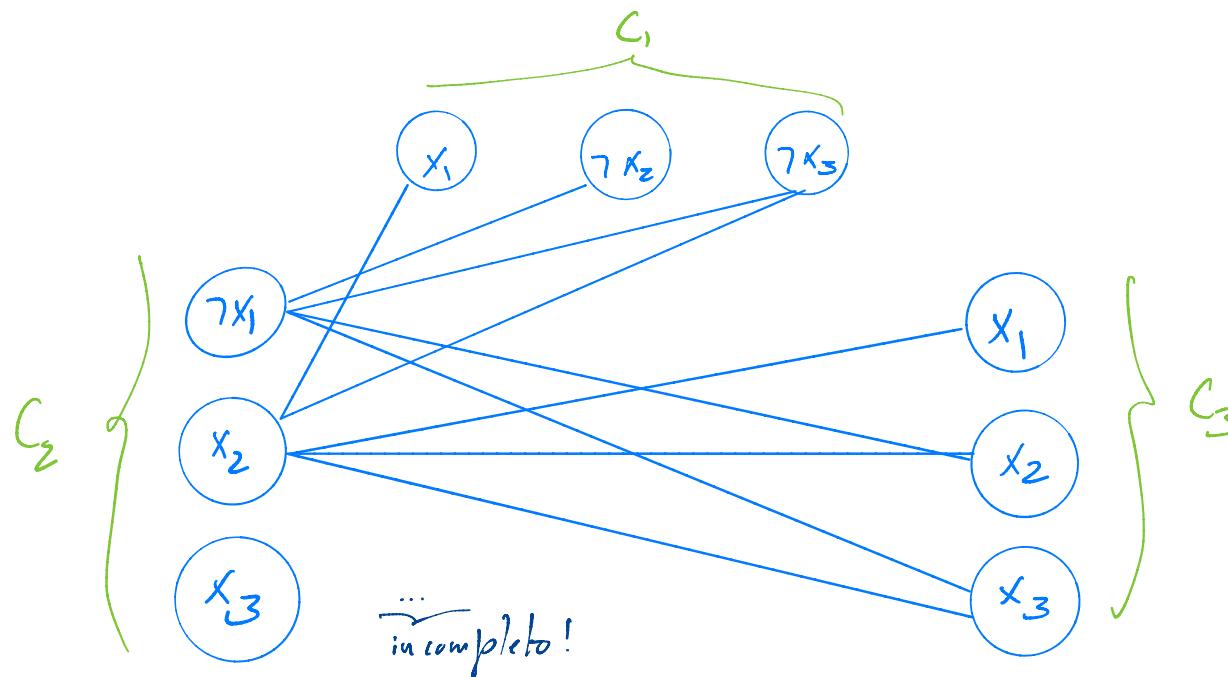
3-CNF-SAT \leq_p Clique

Hipótese: O problema Clique é NP-completo.

- Seja ϕ uma fórmula proposicional no formato 3-CNF, temos de calcular um grafo $G = (V, E)$ e um inteiro k tal que ϕ é satisfazível se e só se G tem um clique de tamanho k .

- Exemplo:

$$\phi = (\underbrace{(x_1 \vee \neg x_2 \vee \neg x_3)}_{C_1} \wedge \underbrace{(\neg x_1 \vee x_2 \vee x_3)}_{C_2} \wedge \underbrace{(x_1 \vee x_2 \vee x_3)}_{C_3})$$



3-CNF-SAT \leq_p Clique

Proposição: O problema Clique é NP-completo.

- Seja ϕ uma fórmula proposicional no formato 3-CNF, temos de calcular um grafo $G = (V, E)$ e um inteiro k tal que ϕ é satisfazível se e só se G tem um clique de tamanho k .
- Seja $\phi = C_1 \vee \dots \vee C_k$, construímos o seguinte grafo:

$$G_\phi = (V_\phi, E_\phi)$$

$$V_\phi = \{ (l, r) \mid \exists l', l''. \; C_l = (l' \vee l \vee l'') \}$$

$$E_\phi = \{ ((l, r), (l', s)) \mid r \neq s \wedge l' \neq l \wedge l + r' \}$$

- Proposição: $\phi = C_1 \vee \dots \vee C_k$ se e só se $G_\phi = (V_\phi, E_\phi)$ contém um clique de tamanho k .

3-CNF-SAT \leq_p Clique

Hipótese: O problema Clique é NP-completo.

- Seja ϕ uma fórmula proposicional no formato 3-CNF, temos de calcular um grafo $G = (V, E)$ e um inteiro k tal que ϕ é satisfazível se e só se G tem um clique de tamanho k .
- Seja $\phi = C_1 \vee \dots \vee C_k$, construímos o seguinte grafo:

$$G_\phi = (V_\phi, E_\phi)$$

$$V_\phi = \{ (l, r) \mid \exists l', l''. \; C_l = (l' \vee l \vee l'') \}$$

$$E_\phi = \{ ((l, r), (l', s)) \mid r \neq s \wedge l' \neq l \wedge l + l' \}$$

$$\begin{aligned} & |V_\phi| = 3 \times k \\ & |E_\phi| = (3 \times k)^2 = 9k^2 \end{aligned} \quad \left\{ \begin{array}{l} \text{A redegradação é calculada em tempo} \\ \text{polinomial} \end{array} \right.$$

3-CNF-SAT \leq_p Clique

- Proposição: $\phi = C_1 \vee \dots \vee C_k$ se e só se $G_\phi = (V_\phi, E_\phi)$ contém um clique de tamanho k .
- Seja $\phi = C_1 \vee \dots \vee C_k$, construímos o seguinte grafo:
 $G_\phi = (V_\phi, E_\phi)$
 $V_\phi = \{ (l, r) \mid \exists l', l''. \ C_r = (l' \vee l \vee l'') \}$
 $E_\phi = \{ ((l, r), (l', s)) \mid r \neq s \wedge l' \neq r \wedge l + l' \}$
- \Rightarrow Em cada cláusola C_i , com $1 \leq i \leq k$, tem de existir um literal l_i tal que $f(l_i) = 1$. O conjunto
 $\hat{V} = \{ (l_i, i) \mid 1 \leq i \leq k \}$
é um clique em G_ϕ .
- Observamos que $(l_i, i), (l_j, j) \in \hat{V}$ então
 $((l_i, i), (l_j, j)) \in E_\phi$
 - $i \neq j$
 - $l_i \neq l_j \Leftrightarrow f_j \neq f_i$

3-CNF-SAT \leq_p Clique

- Proposição: $\phi = C_1 \vee \dots \vee C_k$ se e só se $G_\phi = (V_\phi, E_\phi)$ contém um clique de tamanho k .

- Seja $\phi = C_1 \vee \dots \vee C_k$, construímos o seguinte grafo:

$$G_\phi = (V_\phi, E_\phi)$$

$$V_\phi = \left\{ (l, r) \mid \exists l', l'' \text{ s.t. } C_r = (l' \vee l \vee l'') \right\}$$

$$E_\phi = \left\{ ((l, r), (l', s)) \mid r \neq s \wedge l' \neq r \wedge l \neq l' \right\}$$

\Leftarrow Seja \hat{V} um clique em G_ϕ de tamanho k .

Escolhemos $\rho \models \phi$ que:

$$\forall (l_i, i) \in \hat{V}. \rho(l_i) = 1$$

↓ Por construção, ρ satisfaz ϕ .

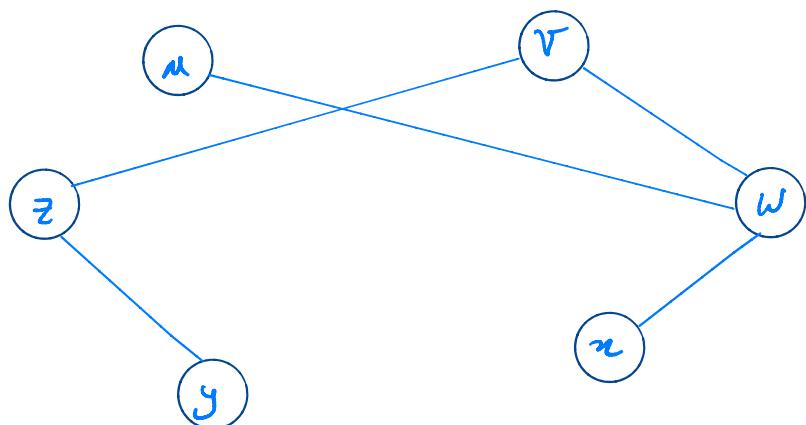
Clique \leq_p Vertex-Cover

Vertex-Cover: V' é uma cobertura de vértices do grafo não dirigido $G = (V, E)$ sse
 $\forall (u, v) \in E, u \in V' \text{ ou } v \in V'$

Vertex-Cover Problem: Encontrar a cobertura de vértices de cardinalidade mínima

↓ Problema de Decisão

VertexCover = $\{ \langle G, k \rangle \mid G \text{ é um grafo não dirigido com uma cobertura de vértices de tamanho } k \}$



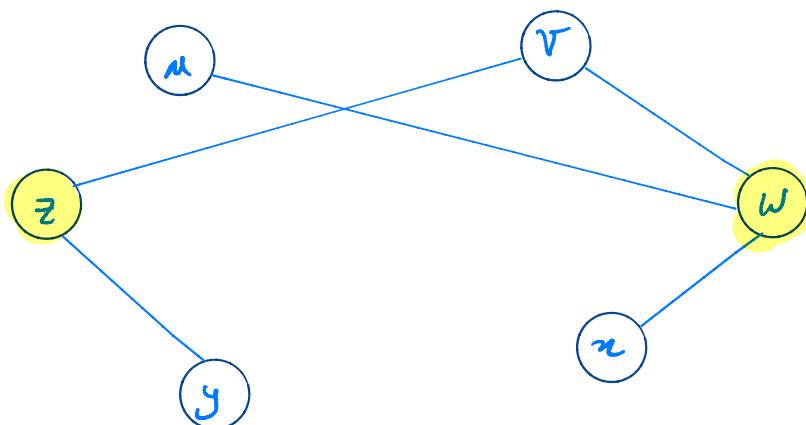
Clique \leq_p Vertex-Cover

Vertex-Cover: V' é uma cobertura de vértices do grafo não dirigido $G = (V, E)$ sse
 $\forall (u, v) \in E, u \in V' \text{ ou } v \in V'$

Vertex-Cover Problem: Encontrar a cobertura de vértices de cardinalidade mínima

\Downarrow Problema de Decisão

VertexCover = $\{ \langle G, k \rangle \mid G \text{ é um grafo não dirigido com uma cobertura de vértices de tamanho } k \}$



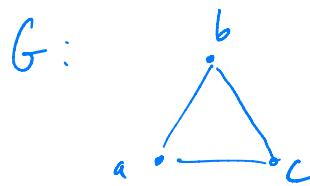
Clique \leq_p Vertex-Cover

Proposição: $G = (V, E)$ tem um clique de tamanho k sse $\bar{G} = (\bar{V}, \bar{E})$ tem uma cobertura de vértices de tamanho $|V| - k$.

Onde:

$$\bar{G} = (\bar{V}, \bar{E}) \quad \bar{E} = \{(u, v) \in V^2 \mid (u, v) \notin E \wedge u \neq v\}$$

Exemplo:



$$\bar{G}: \quad \bullet^b$$

$$\bullet^a \quad \bullet^c$$

$$\text{Clique} = \{a, b, c\}$$

$$VC = \{a, b, c\}$$

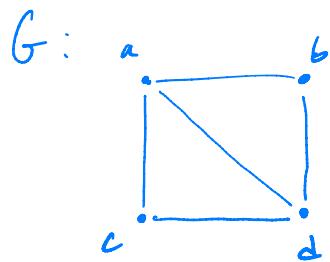
Clique \leq_p Vertex-Cover

Proposição: $G = (V, E)$ tem um clique de tamanho k sse $\bar{G} = (\bar{V}, \bar{E})$ tem uma cobertura de vértices de tamanho $|V| - k$.

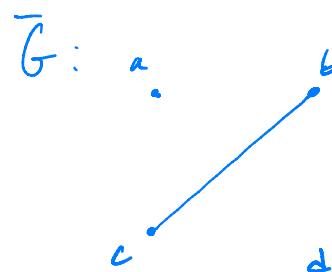
Onde:

$$\bar{G} = (\bar{V}, \bar{E}) \quad \bar{E} = \{(u, v) \in V^2 \mid (u, v) \notin E \wedge u \neq v\}$$

Exemplo 2:



Clique: $\{a, b, d\}$



$\bar{V}C = \{c\}$

Clique \leq_p Vertex-Cover

Proposição: $G = (V, E)$ tem um clique de tamanho k sse $\bar{G} = (\bar{V}, \bar{E})$ tem u-a cobertura de vértices de tamanho $|\bar{V}| - k$.

Onde:

$$\bar{G} = (\bar{V}, \bar{E}) \quad \bar{E} = \{(u, v) \in \bar{V}^2 \mid (u, v) \notin E \wedge u \neq v\}$$

\Rightarrow

- Seja V' um clique de tamanho k em $G = (V, E)$
- Temos de encontrar uma VC de tamanho $|\bar{V}| - k$ em \bar{G} .
- Consideremos o conjunto $\bar{V} = V \setminus V'$. Vamos mostrar que \bar{V} é uma VC em \bar{G} .
- Suponhamos $\bar{g} (u, v) \in \bar{E}$; segue $\bar{g} (u, v) \notin E$ e $u \neq v$

Uma vez que $(u, v) \notin E$, concluimos que u e v não podem pertencer simultaneamente a V' pelo que:
 $u \in V \setminus V'$ ou $v \in V \setminus V'$

Clique \leq_p Vertex-Cover

Proposição: $G = (V, E)$ tem um clique de tamanho k sse $\bar{G} = (\bar{V}, \bar{E})$ tem u-a cobertura de vértices de tamanho $|\bar{V}| - k$.

Onde:

$$\bar{G} = (\bar{V}, \bar{E}) \quad \bar{E} = \{(u, v) \in \bar{V}^2 \mid (u, v) \notin E \wedge u \neq v\}$$

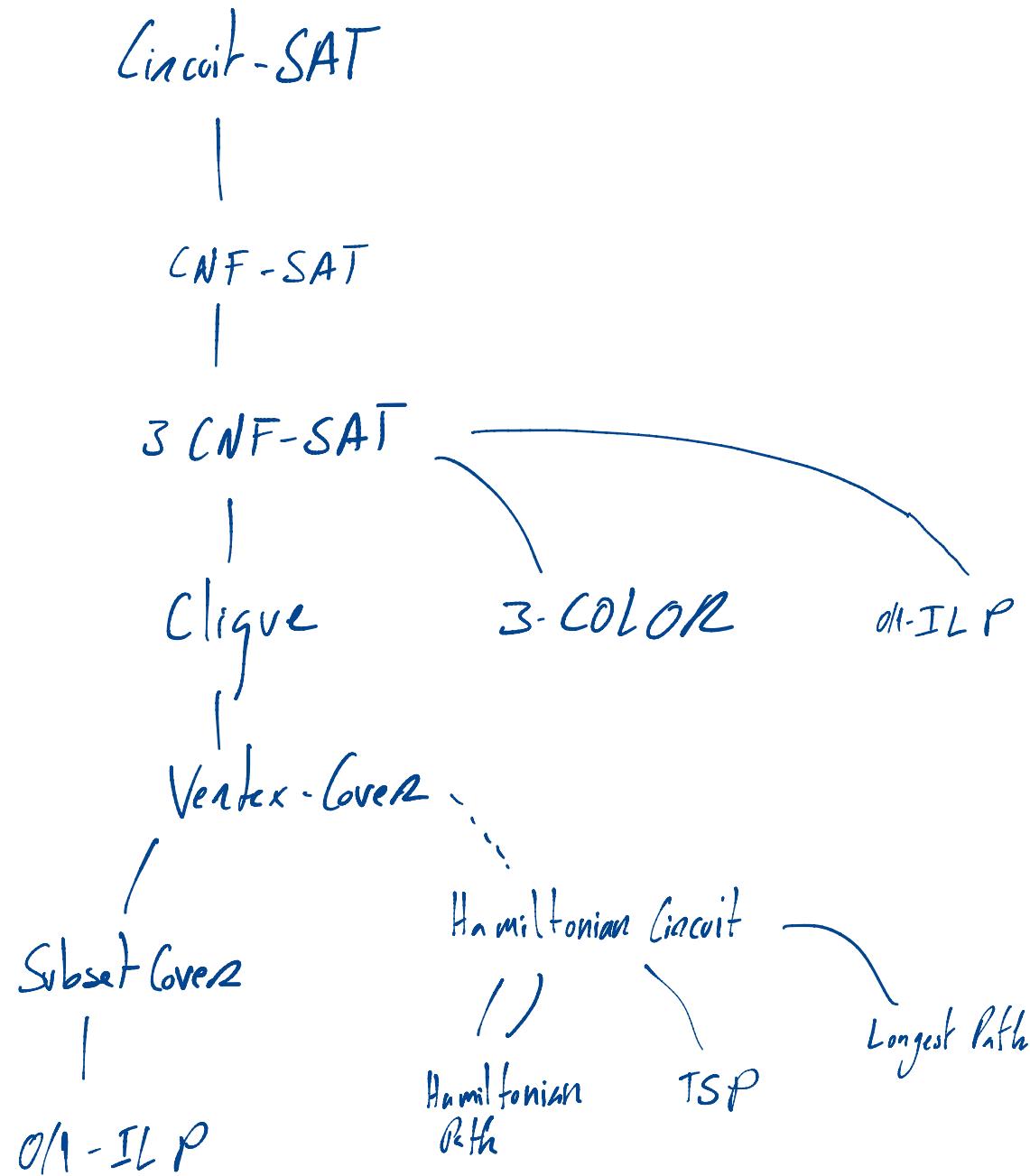


- Seja V' uma VC de tamanho k em $\bar{G} = (\bar{V}, \bar{E})$
- Temos de encontrar um clique de tamanho $|V| - k$ em $G = (V, E)$
- Seja $\hat{V} = V \setminus V'$. Vamos provar que \hat{V} é um clique em G .
 - Como V' é uma VC, sabemos que:

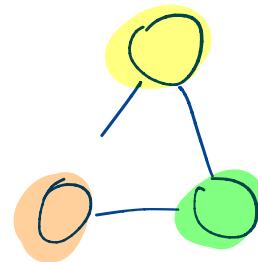
$$\begin{aligned} & \forall (u, v) \in \bar{V}^2. (u, v) \in \bar{E} \Rightarrow u \in V' \vee v \in V' \\ \Leftrightarrow & \forall (u, v) \in \bar{V}^2. (u, v) \notin E \Rightarrow u \notin V' \vee v \notin V' \\ \Leftrightarrow & \forall (u, v) \in \bar{V}^2. u \in \hat{V} \wedge v \in \hat{V} \Rightarrow (u, v) \in E \end{aligned}$$

Arsh 25

Road Map



3-CNF-SAT \leq_f 3-Color



3-Color:

- $G = (V, E)$, grafo não dirigido
- Coloração válida para G - atribuição de cores aos vértices de G tal que vértices adjacentes têm cores diferentes
- Problema 3-Color \Rightarrow Decidir se G pode ser colorido com 3 cores.

$$3\text{-Color} = \{ \langle G \rangle \mid G \text{ pode ser colorido c/ 3 cores} \}$$

Proposição: 3-Color $\in NP$

Proposição: 3-Color é NP-difícil

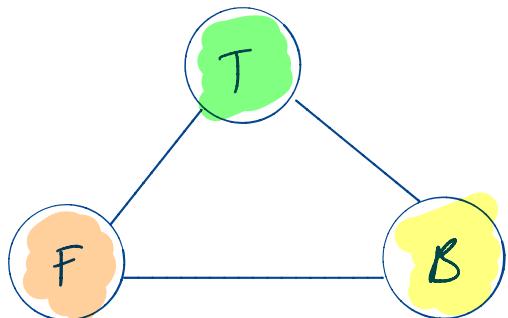
3-CNF-SAT \leq_f 3-Color

Hipótese: 3-Color é NP-difícil

Prova: 3-CNF-SAT \leq_p 3-Color

$$\phi = \underbrace{C_1 \wedge C_2 \wedge \dots \wedge C_n}_{\Downarrow}$$

Instância do problema 3-Color



$$l ::= x \mid \neg x$$

$$C = l_1 \vee l_2 \vee l_3$$

T - True

F - False

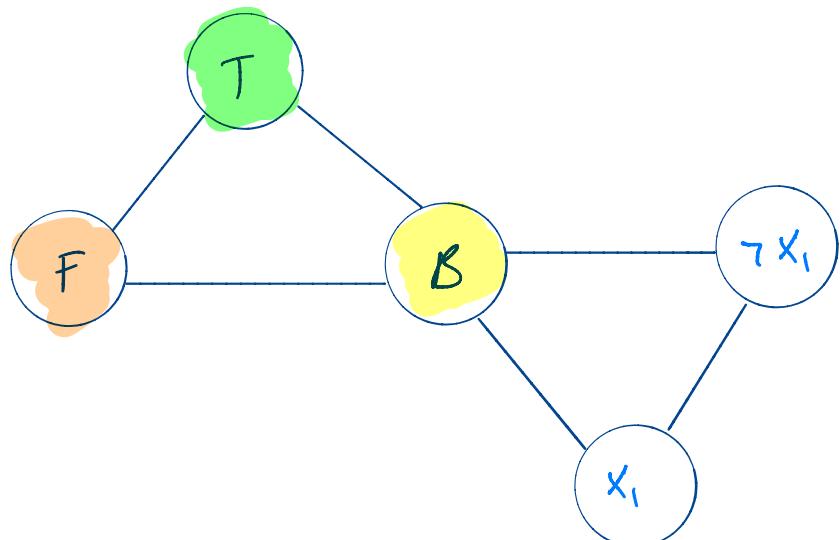
B - Base

3-CNF-SAT \leq_f 3-Color

Proposição: 3-Color é NP-difícil

Prova: 3-CNF-SAT \leq_p 3-Color

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_n$$



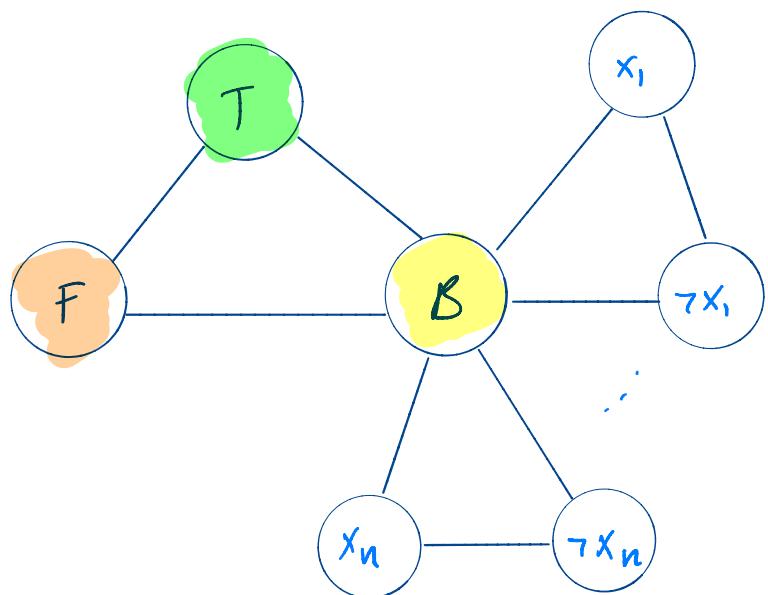
- Por cada variável x adicionamos dois vértices: x e $\neg x$
 - ligamos os dois vértices entre si
 - ligamos os dois vértices à base
- x_1 tem de ser "verde" ou "vermelho", verdadeiro ou falso e $\neg x_1$ vai ter a cor oposta à cor de x_1

3-CNF-SAT \leq_f 3-Color

Proposição: 3-Color é NP-difícil

Prova: 3-CNF-SAT \leq_p 3-Color

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_n$$



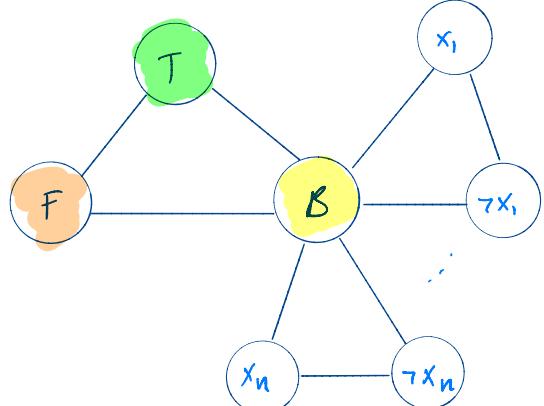
• O que é que nos falta?

3-CNF-SAT \leq_f 3-Color

Proposição: 3-Color é NP-difícil

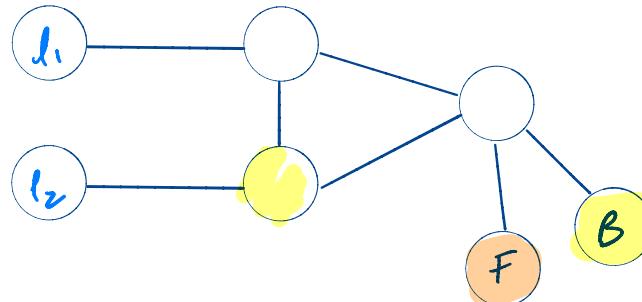
Prova: 3-CNF-SAT \leq_p 3-Color

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_n$$



* Clause satisfiability gadget

$$l_1 \vee l_2$$

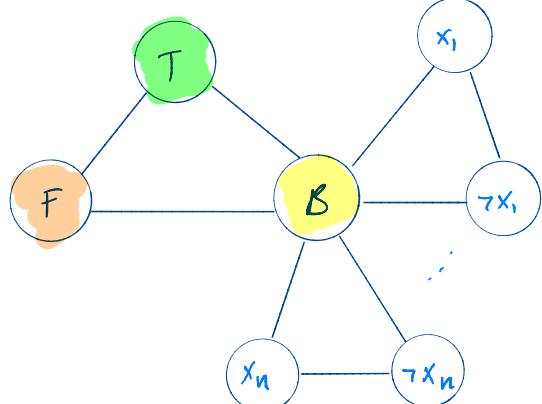


3-CNF-SAT \leq_f 3-Color

Proposição: 3-Color é NP-difícil

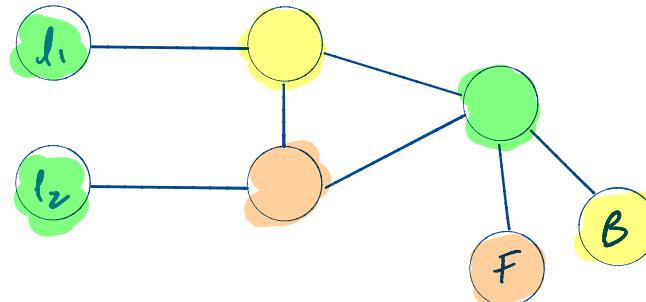
Prova: 3-CNF-SAT \leq_p 3-Color

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_n$$



* Clause satisfiability gadget

$$l_1 \vee l_2$$

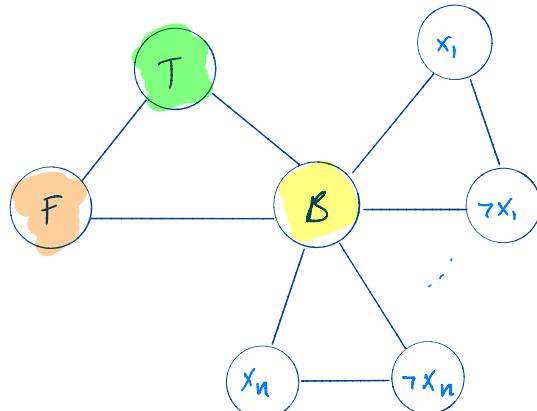


3-CNF-SAT \leq_f 3-Color

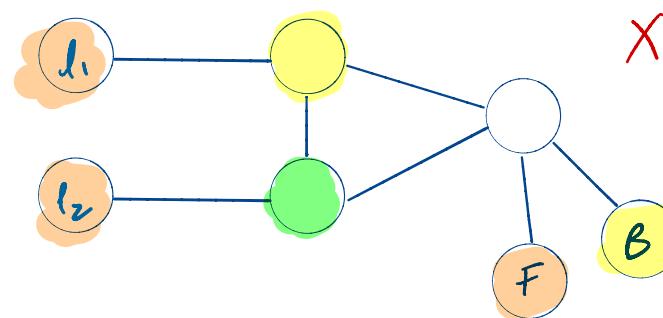
Proposição: 3-Color é NP-difícil

Prova: 3-CNF-SAT \leq_p 3-Color * Clause Satisfiability Gadget

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_n$$



$$l_1 \vee l_2$$



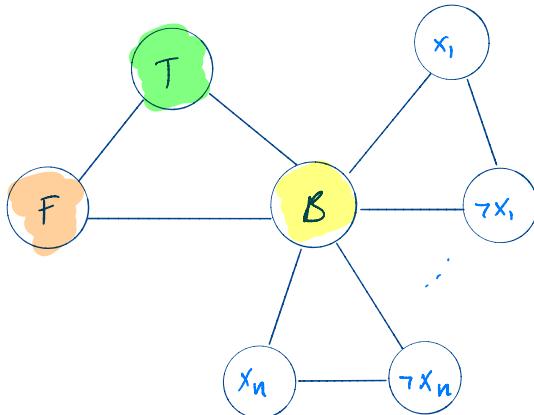
Proposição: P satisfaz a disjunção $l_1 \vee l_2$ se existe uma extensão de P, P' , que é uma coloração válida do 2Or-gadget.

3-CNF-SAT \leq_f 3-Color

Proposição: 3-Color é NP-difícil

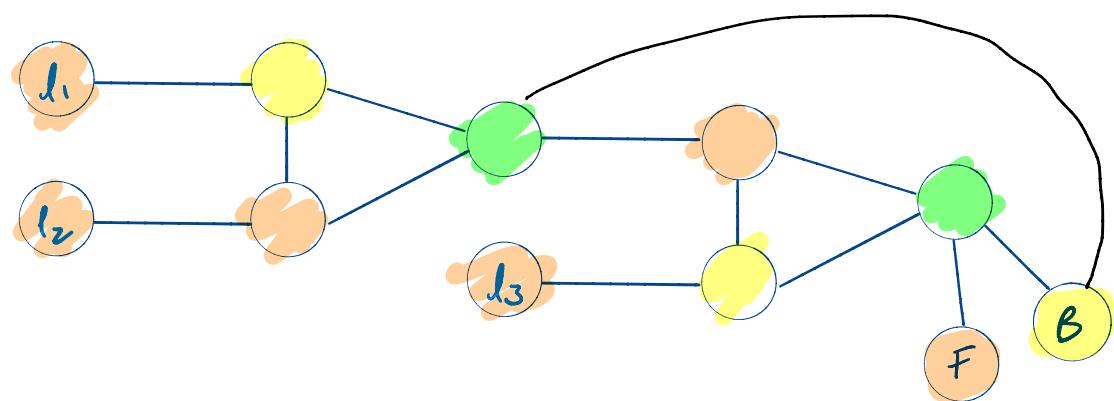
Prova: 3-CNF-SAT \leq_p 3-Color

$$\phi = C_1 \wedge C_2 \wedge \dots \wedge C_n$$



* Clause Satisfaction Gadget

$$l_1 \vee l_2 \vee l_3$$



Proposição: ρ satisfaz a disjunção $l_1 \vee l_2 \vee l_3$ se existe uma extensão de ρ, ρ' , que é uma coloração válida do 3Color-gadget

3-CNF-SAT \leq_p 0/1-ILP

• 0/1-ILP: Determinar o vetor $x \in \mathbb{Z}^n$ que: $Ax = b$

onde:

- A : matriz $n \times m$ com valores inteiros
- b : vetor de inteiros de dimensão m
- $x \in \{0, 1\}^n$

Proposição: 0/1-ILP $\in NP$

Proposição: 0/1-ILP é NP-difícil

3-CNF-SAT \leq_p 0/1-ILP

- 0/1-ILP: Determinar o vetor $x \in \mathbb{Z}^n$ que: $Ax = b$
 - A : matriz $n \times m$ com valores inteiros
 - b : vetor de inteiros de dimensão m
 - $x \in \{0, 1\}^n$

Proposição: 0/1-ILP é NP-difícil

Prova: Seja $\phi = C_1 \wedge \dots \wedge C_n$ uma fórmula no formato 3-CNF

- Ideia:
- Das variáveis do problema 0/1-ILP por cada variável da fórmula: $x_j \mapsto x_j^1, x_j^0$
 \downarrow \downarrow
 T F
 - Transformar cada cláusola C_i numa desigualdade
 $C_i \mapsto \hat{a}_i^T \cdot x \leq \hat{b}_i$
 - Duas desigualdades por cada variável da fórmula original

$$x_j \rightsquigarrow \begin{cases} \hat{a}_j^T \cdot x \leq \hat{b}_j \\ \hat{\bar{a}}_j^T \cdot x \leq \hat{\bar{b}}_j \end{cases}$$

3-CNF-SAT \leq_p 0/1-ILP

- Ideia:
- Das variáveis do programa 0/1-ILP por cada variável da fórmula: $x_j \rightarrow x_j^1, x_j^0$
 - Transformar cada cláusula C_i numa desigualdade
 $C_i \rightarrow a_i^T \cdot x \leq b_i$
- $$A \cdot x \leq b$$

Exemplo: $C_1: x_0 \vee \neg x_2 \vee x_4$

$$x_0^1 + x_2^0 + x_4^1 \geq 1$$

$$\Leftrightarrow -x_0^1 - x_2^0 - x_4^1 \leq -1$$

$$a_1^T = [x_0^0 \ x_0^1 \ x_1^0 \ x_1^1 \ x_2^0 \ x_2^1 \ x_3^0 \ x_3^1 \ x_4^0 \ x_4^1]$$

$$b_1 = -1$$

3-CNF-SAT \leq_p 0/1-ILP

- Ideia:
- Das variáveis do programa 0/1-ILP por cada variável x_j formula: $x_j \rightarrow x_j^1, x_j^0$
 - Transformar cada cláusula C_i numa desigualdade
 $C_i \rightarrow a_i^T \cdot x \leq b_i$

Exemplo: $C_1: x_0 \vee \neg x_2 \vee x_4$

$$\Downarrow$$

$$x_0^1 + x_2^0 + x_4^1 \geq 1$$

$$\Downarrow$$

$$-x_0^1 - x_2^0 - x_4^1 \leq -1$$

$$A_1^T = \begin{bmatrix} x_0^0 & x_0^1 & x_1^0 & x_1^1 & x_2^0 & x_2^1 & x_3^0 & x_3^1 & x_4^0 & x_4^1 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \end{bmatrix}$$

$$b_1 = -1$$

3-CNF-SAT \leq_p 0/1-ILP

- Ideia:
- Das variáveis do programa 0/1-ILP (ou cada variável) à formula: $x_j \rightarrow x_j^1, x_j^0$
 - Duas desigualdades para cada variável da formula original

$$x_j \sim \begin{cases} \hat{a}_j^T \cdot x \leq \hat{b}_j \\ \hat{\bar{a}}_j^T \cdot x \leq \hat{\bar{b}}_j \end{cases}$$

Exemplo: $\circledcirc x_1$ $x_1^1 \quad x_1^0 \quad x_1^1 + x_1^0 = 1 \Leftrightarrow \begin{cases} x_1^1 + x_1^0 \leq 1 \\ x_1^1 + x_1^0 \geq 1 \end{cases} \Leftrightarrow \begin{cases} x_1^1 + x_1^0 \leq 1 \\ -x_1^1 - x_1^0 \leq -1 \end{cases}$

$$\begin{aligned} \hat{a}_1^T &= [x_0^0 \ x_0^1 \ x_1^0 \ x_1^1 \ x_2^0 \ x_2^1 \ x_3^0 \ x_3^1 \ x_4^0 \ x_4^1] & \hat{b}_1 &= 1 \\ \hat{\bar{a}}_1^T &= [0 \ 0 \ -1 \ -1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0] & \hat{\bar{b}}_1 &= -1 \end{aligned}$$

3-CNF-SAT \leq_p 0/1-ILP

- Ideia:
- Das variáveis do programa 0/1-ILP (ou cada variável) à formula: $x_j \rightarrow x_j^1, x_j^0$
 - Duas desigualdades para cada variável da formula original

$$x_j \sim \begin{cases} \hat{a}_j^T \cdot x \leq \hat{b}_j \\ \hat{\bar{a}}_j^T \cdot x \leq \hat{\bar{b}}_j \end{cases}$$

Exemplo: x_1

$$x_1^1 + x_1^0 = 1 \Leftrightarrow \begin{cases} x_1^1 + x_1^0 \leq 1 \\ x_1^1 + x_1^0 \geq 1 \end{cases} \Leftrightarrow \begin{cases} x_1^1 + x_1^0 \leq 1 \\ -x_1^1 - x_1^0 \leq -1 \end{cases}$$

$$\hat{a}_1^T = \begin{bmatrix} x_0^0 & x_0^1 & x_1^0 & x_1^1 & x_2^0 & x_2^1 & x_3^0 & x_3^1 & x_4^0 & x_4^1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \hat{b}_1 = 1$$

$$\hat{\bar{a}}_1^T = \begin{bmatrix} 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \hat{\bar{b}}_1 = -1$$

Vertex-Cover \leq_p Subset-Cover

• Subset Cover:

- F é uma família de conjuntos de elementos de um dado conjunto base Ω

$$F = \{x_1, \dots, x_n\} \quad \forall 1 \leq i \leq n. \quad x_i \subseteq \Omega$$

- $\gamma \subseteq \Omega$

- $\text{SubsetCover} = \left\{ \langle F, \gamma, k \rangle \mid \exists x_1, \dots, x_k \in F. \quad x_1 \cup x_2 \cup \dots \cup x_k = \gamma \right\}$

Proposição: Subset Cover está em NP

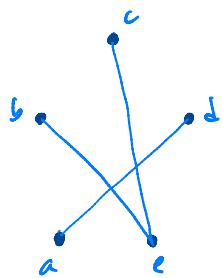
Proposição: Subset Cover é NP-difícil

Vertex-Cover \leq_p Subset-Cover

• Subset Cover:

- $\text{SubsetCover} = \left\{ \langle F, Y, k \rangle \mid \exists x_1, \dots, x_k \in F. \quad x_1 \cup x_2 \cup \dots \cup x_k = Y \right\}$

Proposição: SubsetCover é NP-difícil



$$VC = \{a, e\}$$

$$x_a = \{(a, d)\}$$

$$x_d = \{(a, d)\}$$

$$x_b = \{(b, e)\}$$

$$x_e = \{(b, e), (c, e)\}$$

$$x_c = \{(c, e)\}$$

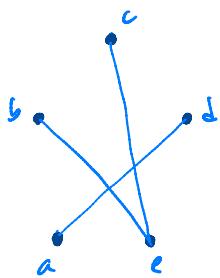
$$Y = \{(b, e), (c, e), (a, d)\}$$

Vertex-Cover \leq_p Subset-Cover

• Subset Cover:

- $\text{SubsetCover} = \left\{ \langle F, Y, k \rangle \mid \exists x_1, \dots, x_k \in F. \quad x_1 \cup x_2 \cup \dots \cup x_k = Y \right\}$

Proposição: SubsetCover é NP-difícil



$$VC = \{a, e\}$$

$$x_a = \{(a, d)\}$$

$$x_d = \{(a, d)\}$$

$$x_b = \{(b, e)\}$$

$$x_e = \{(b, e), (c, e)\}$$

$$x_c = \{(c, e)\}$$

$$Y = \{(b, e), (c, e), (a, d)\}$$

Vertex-Cover \leq_p Subset-Cover

• Subset Cover:

- $\text{SubsetCover} = \left\{ \langle F, Y, k \rangle \mid \exists x_1, \dots, x_k \in F. \quad x_1 \cup x_2 \cup \dots \cup x_k = Y \right\}$

Proposição: SubsetCover é NP-difícil

$\langle G, k \rangle \in VC \Leftrightarrow \langle \bar{F}_G, E, k \rangle \in \text{SubsetCover}$

onde • $G = (V, E)$

• $\bar{F}_G = \{E_v \mid v \in V\}$

• $E_v = \{ (u, v) \mid (u, v) \in E \} \cup \{ (v, u) \mid (v, u) \in E \}$

• O resultado segue diretamente, observando que:

\hat{V} é uma VC de G se e só se $\bigcup_{u \in \hat{V}} E_u = E$

Ciclo Hamiltoniano

Ciclo Hamiltoniano: Dado um grafo não dirigido $G = (V, E)$, um ciclo Hamiltoniano de G é um ciclo simples que contém todos os vértices de V .

$$\text{Ham-Cycle} = \left\{ \langle G \rangle \mid G \text{ é um grafo não dirigido com um ciclo hamiltoniano} \right\}$$

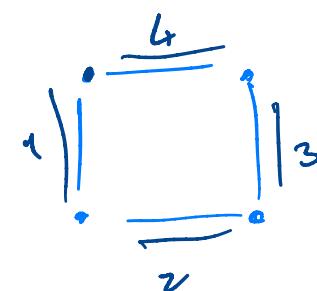
Proposição: Ham-Cycle ∈ NP

Proposição: Ham-Cycle é NP-difícil.

↳ Prova:

Redução: Vertex-Cover \leq_p Ham-Cycle

34.S.3 (CLRS)



Caminho Hamiltoniano

Ciclo Hamiltoniano: Dado um grafo não dirigido $G = (V, E)$, um caminho Hamiltoniano de G é um caminho que contém todos os vértices de V exatamente uma vez.

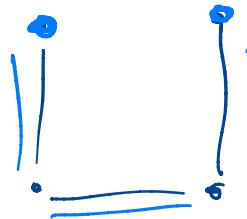
$$\text{Ham-Rth} = \left\{ \langle G \rangle \mid G \text{ é um grafo não dirigido com um caminho hamiltoniano} \right\}$$

Proposição: Ham-Rth $\in NP$

Proposição: Ham-Rth é NP-difícil.

↳ Prova:

Redução: Ham-Cycle \leq_p Ham-Rth

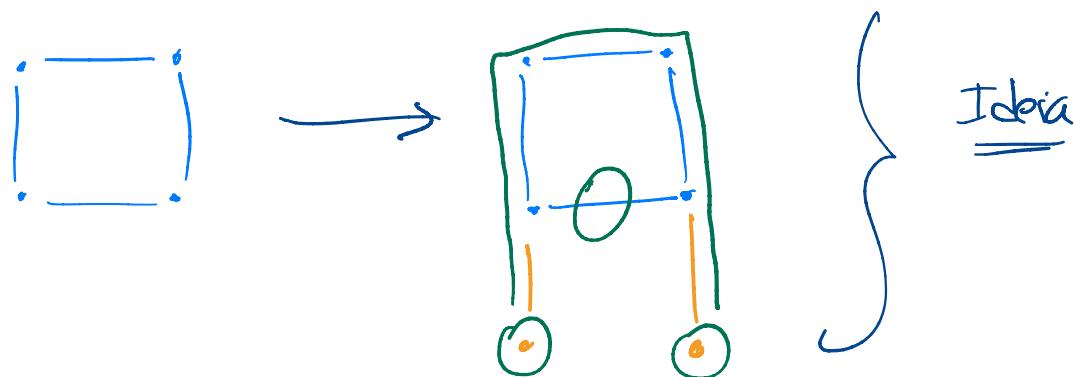


Caminho Hamiltoniano

Proposição: Ham-Rath é NP-difícil.

↳ Prova:

Redução: Ham-Cycle \leq_g Ham-Rath

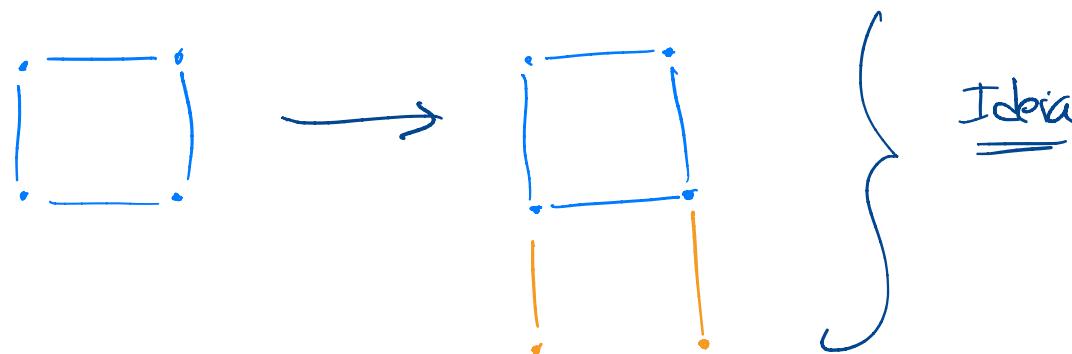


Caminho Hamiltoniano

Proposição: Ham-Path é NP-difícil.

↳ Prova:

Redução: Ham-Cycle \leq_p Ham-Path



Proposição:

$G \in \text{Ham-Cycle} \iff G' \in \text{Ham Path}$

onde:

$$G = (V, E) \wedge (u, v) \in E$$

$$G' = (V \cup \{u', v'\}, E \cup \{(u', u), (v', v)\})$$

Observação: É a redução:

Ham-Path \leq_p Ham-Cycle ?

= (TPC)