
Au^h 21



Redutibilidade NP

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

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

Escrevemos: $X \leq_p Y$

Proposição 4: $Y \in P \wedge X \leq_p Y \Rightarrow X \in P$

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, concluímos 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:
 $A'(y) :$
retorna $A(h(y))$

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

Prova: Hé que provam 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 $Z \leq_p X$ (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: $X \in NP \wedge \exists Y \leq_p X \wedge Y \in NPC \Rightarrow X \in NPC$

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

① Provarmos que X está em **NP**

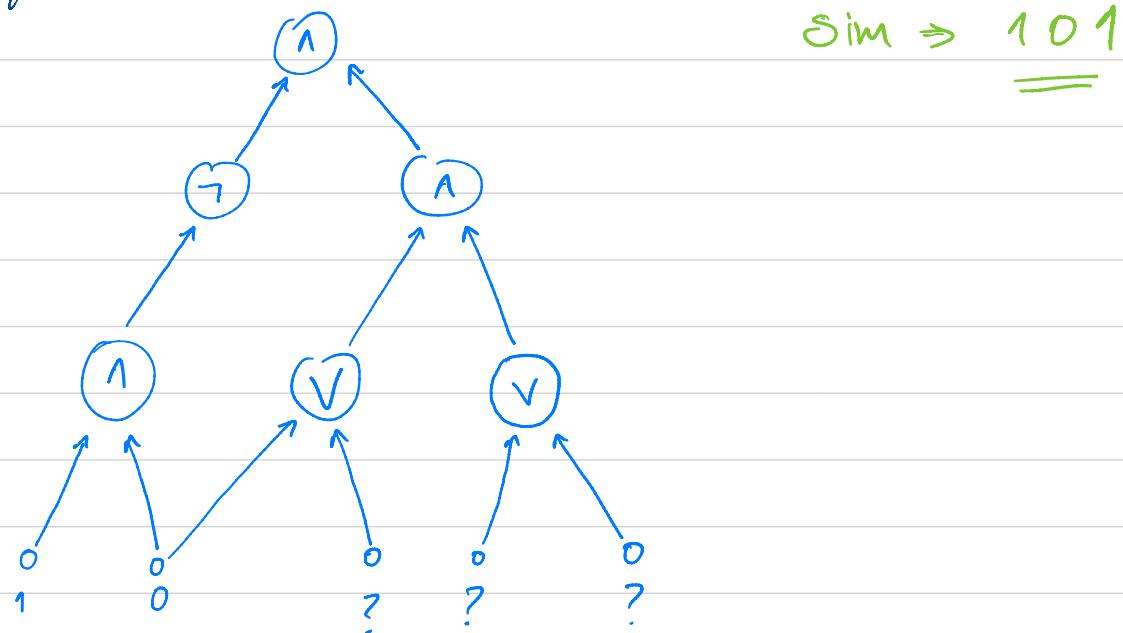
↳ Descobrir o certificado g nos permite
verificar X em tempo polinomial

② Selecionar um problema **NP-completo** Y e
construir uma redução $Y \leq_p X$.

O 1º Problema NP-Completo

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

Exemplo:



O 1º Problema NP-Completo

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

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

Esboço de Prova:

- Tome-se $X \in NP$. Da 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 combinatorio de tamanho polinomial. Seja K esse circuito
- Fixamos as l_x 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 $x \in X$).

Road Map

CNF-SAT



3 CNF-SAT



clique



Vertex-Cover

|
Set Cover

) Shared Stickers

CNF-SAT \leq_p 3-CNF-SAT

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

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

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

Exemplo: $\exists x, \forall xz$

- Uma nova variável y , 2 cláusulas de output

$(\exists x, \forall xz \vee y)$

$(\exists x, \forall xz \vee \neg y)$

• Qualquer valoração \vec{v} não satisfaz $\exists x, \forall xz$
fazendo uma das cláusulas

CNF-SAT \leq_p 3-CNF-SAT

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

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

• = 1 $\rightarrow ?$

• = 2 \rightarrow 1 nova variável + 2 novas cláusulas

• = 3 $\rightarrow \checkmark$

• $> 3 \rightarrow ?$

Exemplo: $\exists x$

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

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

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

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

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

Qualquer valoração \tilde{g} não satisfaz
 $\neg x$ falso uma das 4 cláusulas.

CNF-SAT \leq_p 3-CNF-SAT

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

- 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: $(x_1 \vee \neg x_2 \vee x_3 \vee \neg x_4)$

$$\underbrace{x_1 \vee \neg x_2}_{\downarrow} \quad \underbrace{x_3 \vee \neg x_4}_{\downarrow}$$

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{y} 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 novos cláusulas
- = 2 \rightarrow 1 nova variável + 2 novas cláusulas
- = 3 \rightarrow ✓
- > 3 \rightarrow ?

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

$$\begin{array}{c} x_1 \vee \underline{\neg x_2} \vee \underline{x_3} \vee \underline{\neg x_4} \vee x_5 \\ \diagup \quad | \quad \diagdown \\ \neg y_1 \vee \neg x_2 \vee y_1 \quad \neg y_1 \vee x_3 \vee y_2 \quad \neg y_2 \vee \neg x_4 \vee x_5 \end{array}$$

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

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

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

:

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

:

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

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

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

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

e P' extende P

CNF-SAT \leq_p 3-CNF-SAT

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

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

$$\begin{aligned}C'_1 &= l_1 \vee l_2 \vee y_1 \\C'_2 &= \neg y_1 \vee l_3 \vee y_2\end{aligned}$$

:

$$C'_j = \neg y_{j-1} \vee l_{j+1} \vee y_j$$

:

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

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

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

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.

• $\rho'(y_l) = \begin{cases} 1 & \text{se } l \leq i-2 \\ 0 & \text{se } l \geq i-1 \end{cases}$

$\rho'(y_l) = f(x_l) \quad \forall x_l \in \text{dom}(f)$

f é uma extensão de f

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

Hipótese: O problema Clique é NP-completo.

3-CNF-SAT \leq_p Clique

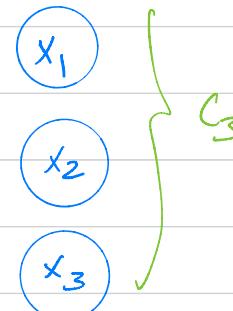
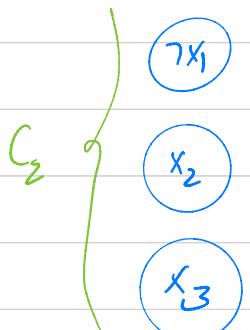
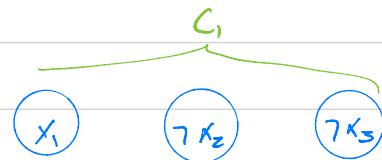
Proposição: O problema clique é NP-completo.

- 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 .

Exemplo:

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

C_1 C_2 C_3



3-CNF-SAT \leq_p Clique

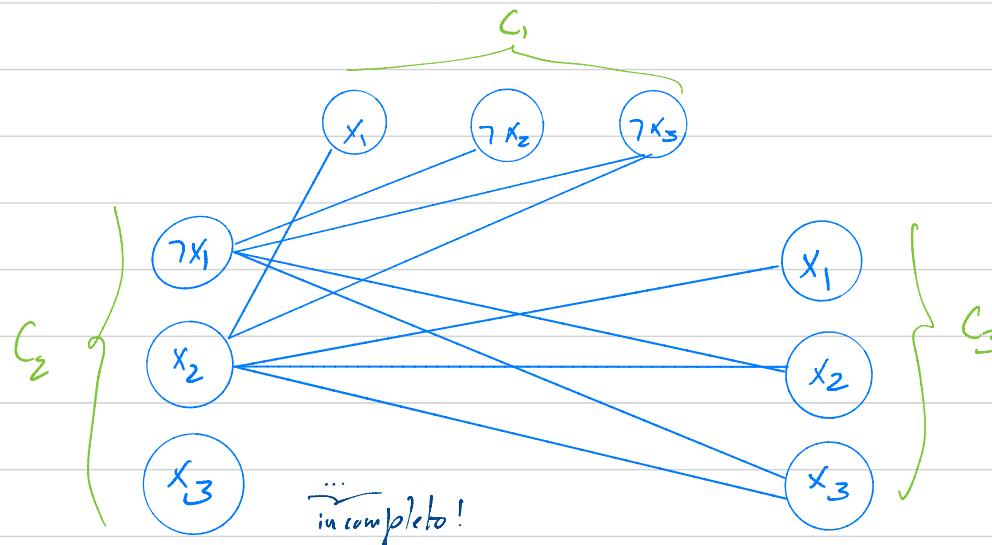
Proposição: O problema clique é NP-completo.

- 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 .

Exemplo:

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

$\overbrace{\quad\quad\quad}$ C_1 $\overbrace{\quad\quad\quad}$ C_2 $\overbrace{\quad\quad\quad}$ 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 somente se G tem um clique de tamanho k .
- Seja $\phi = C_1 \vee \dots \vee C_k$, construímos o seguinte grafo:

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

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

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

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

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 somente se G tem um clique de tamanho k .
- Seja $\phi = C_1 \vee \dots \vee C_k$, construímos o seguinte grafo:

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

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

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

$$\cdot |V_\phi| = 3 \times k$$

$$\cdot |E_\phi| = (3 \times k)^2 = 9k^2$$

A redução é calculada em tempo polinomial

3-CNF-SAT \leq_p Clique

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

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

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

$$V_\phi = \{ (l, r) \mid \exists l', l'' \quad C_r = (l' \vee l \vee l'') \}$$

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

\Rightarrow Em cada cláusula 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 se $(l_i, i), (l_j, j) \in \hat{V}$ então $((l_i, i), (l_j, j)) \in E_\phi$
 - $i \neq j$
 - $l_i + r_{l_j} < l_j + r_{l_i}$

3-CNF-SAT \leq_p Clique

- Proposição: $\phi = C_1 \wedge \dots \wedge C_k$ satisfazível se e só se $G_\phi = (V_\phi, E_\phi)$ contém um clique de tamanho k .
- Seja $\phi = C_1 \wedge \dots \wedge C_k$, construímos o seguinte grafo:
 $G_\phi = (V_\phi, E_\phi)$
 $V_\phi = \{ (l, r) \mid \exists l, l'. C_i = (l' \vee l \vee l'') \}$
 $E_\phi = \{ ((l, r), (l', s)) \mid r \neq s \wedge l' \neq l \wedge l \neq l'' \}$
- \Leftarrow Seja \tilde{V} um clique em G_ϕ de tamanho k .
Escolhemos $\rho \in \tilde{V}$ que:
 $\forall (l_i, i) \in \tilde{V}. \rho(l_i) = 1$
↓ Por construção, ρ satisfaz ϕ .

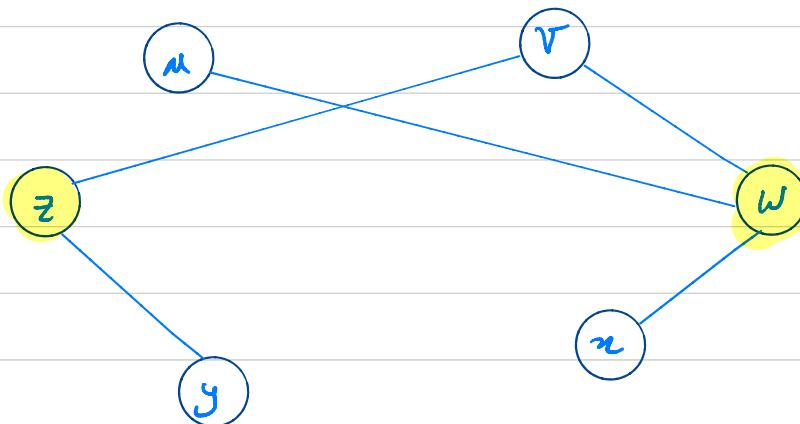
Clique \leq_p Vertex-Cover

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

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

\Downarrow Problema de Decisão

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



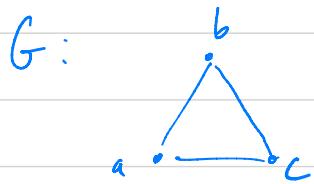
Clique \leq_p Vertex-Cover

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:

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

Exemplo:



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

$$VC = \emptyset$$

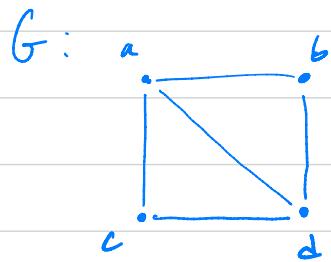
Clique \leq_p Vertex-Cover

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

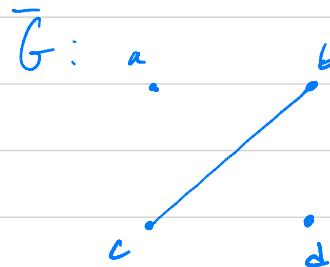
Onde:

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

Exemplos:



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



$Vc = \{c\}$

Clique \leq_p Vertex-Cover

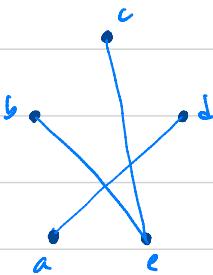
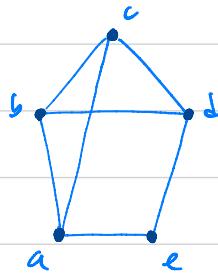
Proposição: $G = (V, E)$ tem um clique de tamanho k sse $\bar{G} = (\bar{V}, \bar{E})$ tem uma cobertura de vértices de tamanho $|\bar{V}| - k$.

Onde:

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

Exemplo 3

G :



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

$VC = \{a, e\}$

Clique \Leftrightarrow 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:

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

Basta provar que:

$$\hat{V} \text{ é um clique em } G \quad \text{sse} \quad V \setminus \hat{V} \text{ é um VC em } \bar{G}$$

\hat{V} é um clique em G

$$\Leftrightarrow \forall u, v. \quad u \in \hat{V} \wedge v \in \hat{V} \Rightarrow (u, v) \in E$$

$$\Leftrightarrow \forall u, v. \quad (u, v) \notin E \Rightarrow u \notin \hat{V} \vee v \notin \hat{V}$$

$$\Leftrightarrow \forall u, v. \quad (u, v) \in \bar{E} \Rightarrow u \in V \setminus \hat{V} \vee v \in V \setminus \hat{V}$$

$$\Leftrightarrow V \setminus \hat{V} \text{ é um VC em } \bar{G}$$

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

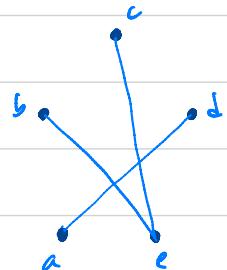
Proposição: Subset Cover é NP-difícil

Vertex-Cover \leq_p Subset-Cover

• Subset Cover:

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

Proposição: SubsetCover é NP-difícil



$$X_a = \{(a, d)\} \quad X_d = \{(a, d)\}$$

$$X_b = \{(b, e)\} \quad X_e = \{(b, e), (c, e)\}$$

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

$$\sqrt{C} = \{a, e\}$$

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

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 • $G = (V, E)$

$$\cdot \bar{F}_G = \{ E_r \mid r \in V \}$$

$$\cdot E_r = \{ (u, v) \mid (u, v) \in E \} \cup \{ (v, u) \mid (v, u) \in E \}$$

• O resultado segue diretamente, observando que:

$$\bigvee_{v \in V} \text{uma VC de } G \text{ satisfaaz } \bigcup_{u \in V} E_u = E$$

Exemplo: Cromos Partilhados

II.d) Seja $C = \{c_1, \dots, c_n\}$ uma colecção de cromos e $\mathcal{A} = \{a_1, \dots, a_m\}$ um grupo de amigos que coleccionam cromos. Cada membro do grupo detém um subconjunto de C ; seja C_i o conjunto de cromos detido por a_i e $\mathcal{C} = \{C_i \mid 1 \leq i \leq m\}$ o conjunto dos conjuntos de cromos de todos os membros do grupo. Os membros do grupo pretendem determinar o mais pequeno conjunto de cromos que contém pelo menos um cromo detido por cada membro do grupo. Formalmente, este problema pode ser modelado através do seguinte problema de decisão:

$$\text{SharedStickers} = \{\langle C, \mathcal{C}, k \rangle \mid \exists X \subseteq C. |X| = k \wedge \forall_{1 \leq i \leq m}. C_i \cap X \neq \emptyset\}$$

• Mostre que o problema SharedStickers é NP-difícil por redução a partir do problema Vertex

Exemplo: Cromos Parfilhados

II.d) Seja $C = \{c_1, \dots, c_n\}$ uma colecção de cromos e $\mathcal{A} = \{a_1, \dots, a_m\}$ um grupo de amigos que colecionam cromos. Cada membro do grupo detém um subconjunto de C ; seja C_i o conjunto de cromos detido por a_i e $\mathcal{C} = \{C_i \mid 1 \leq i \leq m\}$ o conjunto dos conjuntos de cromos de todos os membros do grupo. Os membros do grupo pretendem determinar o mais pequeno conjunto de cromos que contém pelo menos um cromo detido por cada membro do grupo. Formalmente, este problema pode ser modelado através do seguinte problema de decisão:

$$\text{SharedStickers} = \{\langle C, \mathcal{C}, k \rangle \mid \exists X \subseteq C. |X| = k \wedge \forall_{1 \leq i \leq m}. C_i \cap X \neq \emptyset\}$$

• Mostre que o problema SharedStickers é NP-difícil por redução a partir do problema VCover

$$\langle G, k \rangle \in \text{VCover} \Leftrightarrow \langle C, \mathcal{C}, k \rangle \in \text{SharedStickers}$$

$$C = V$$

$$\mathcal{C} = \{ \{u, v\} \mid (u, v) \in E \}$$

Complexidade
de redução: $O(\underline{V+E})$

Exemplo: Incompatibilidades

II.d) Uma matriz de incompatibilidades é uma matriz quadrada cujas células guardam valores decimais entre 0 e 1. Intuitivamente, dada uma matriz de incompatibilidades M , $n \times n$, a célula M_{ij} guarda a incompatibilidade entre os índices i e j ; $M_{ij} = 0$ se i e j são completamente compatíveis e $M_{ij} = 1$ se i e j são completamente incompatíveis. Dado um sub-conjunto de índices $I \subseteq \{1, \dots, n\}$, o nível de incompatibilidade do conjunto é dado por: $\sum_{i,j \in I} M_{ij}$. O problema das incompatibilidades define-se formalmente da seguinte maneira:

Incompat = $\{\langle M, k, v \rangle \mid M \text{ contém um sub-conjunto de índices de tamanho } k \text{ e incompatibilidade igual ou inferior a } v\}$

- Nota: é o problema Incompat é NP-difícil por redução a partir do problema ISet.

Problema ISet:

$\langle G, k \rangle \in \text{ISet}$ se e só se G contém um conjunto de vértices independentes de tamanho k

Problema ISet

Definição [Conjunto de Vértices Independentes]

X é um conjunto de vértices independentes de G se e só se

$$\forall u, v \in X. (u, v) \notin E$$

$$\boxed{\text{ISet} \leq_p \text{In}(G, k)}$$

• $\langle G, k \rangle \in \text{ISet} \Leftrightarrow \langle H_G, k, 0 \rangle \in \text{Incompat}$

$$G = (V, E) \quad V = \{v_1, \dots, v_n\}$$

$$H_{ij} = \begin{cases} 0 & \text{se } (v_i, v_j) \notin E \\ 1 & \text{c.c.} \end{cases}$$

• Complexidade da Redução: $O(V^2)$