

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.



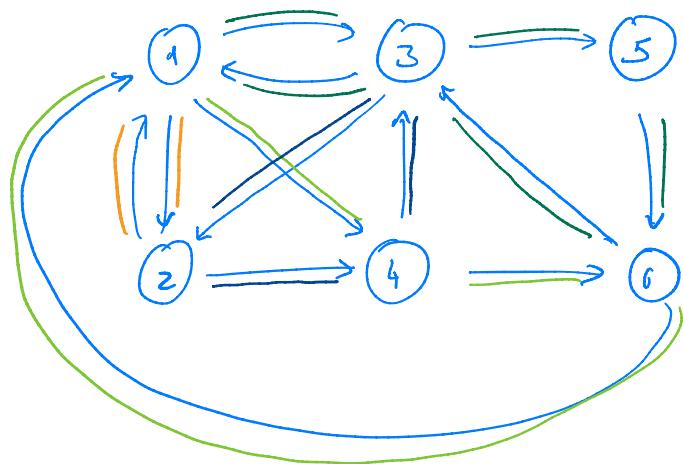
• Hamiltonian Cycle

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&

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

Polyomial-Time versus NP-Complete Problems

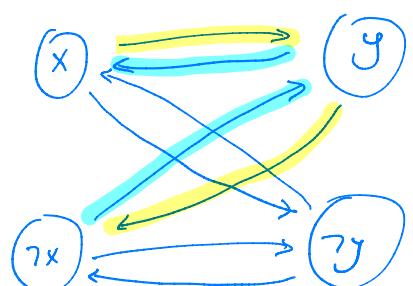
2 CNF-SAT

- 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:
$$A'(x) =$$

for each y of size $\leq |p(x)|$
if $A(x, y)$
then return 1
return 0

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:

$$x \in X \iff \exists j. A(x, j) = 1$$



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

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

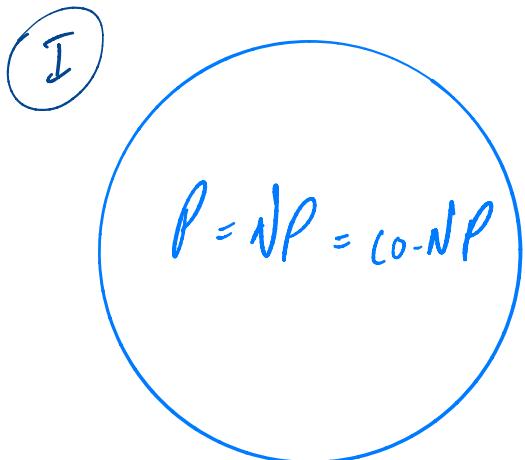
- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- EXP - conjunto dos problemas decidíveis em tempo exponencial
- co-NP - conjunto dos problemas cujo complemento está em NP
Exemplo:
 $\text{UNSAT}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

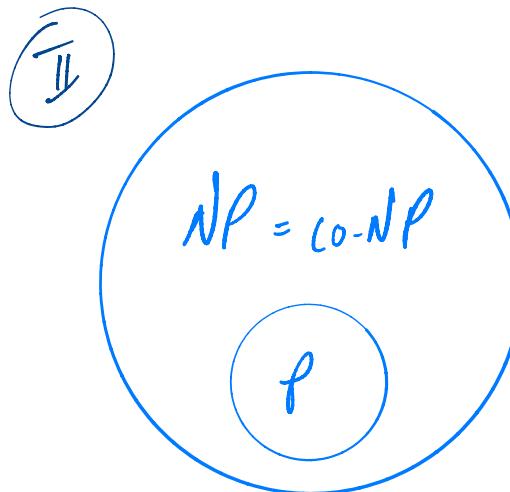
- P - conjunto dos problemas decidíveis em tempo polinomial
- NP - conjunto dos problemas verificáveis em tempo polinomial (com um certificado de tamanho polinomial)
- EXP - conjunto dos problemas decidíveis em tempo exponencial
- co-NP - conjunto dos problemas cujo complemento está em NP
Exemplo:
 $\text{UNSAT}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

Classes de Complexidade - Conjecturas

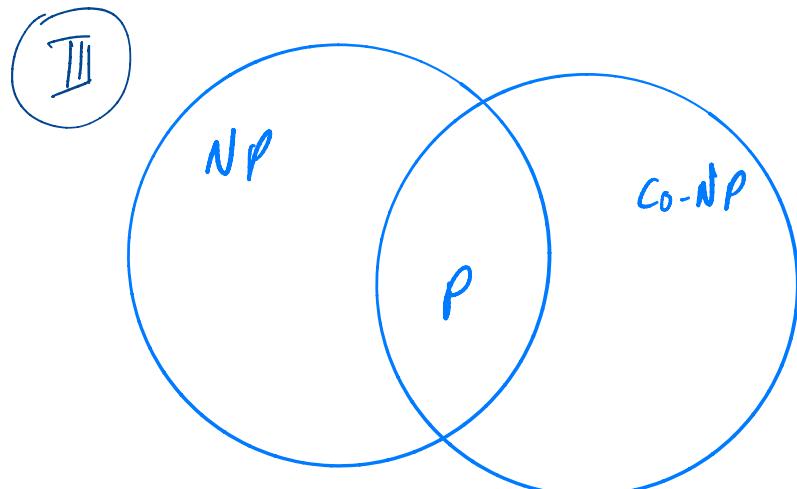
* 4 possibilidades



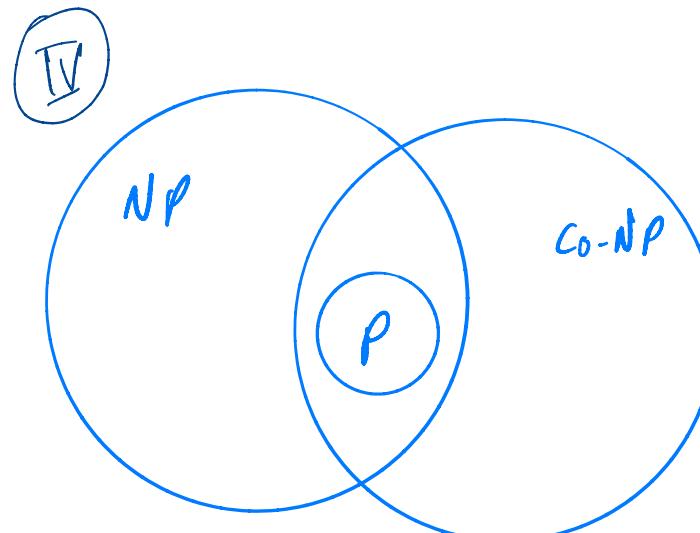
$P = NP?$



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



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



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 $Y \in co-NP$, há \bar{Y} mostrando $\bar{Y} \leq_p \bar{X}$.

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

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**

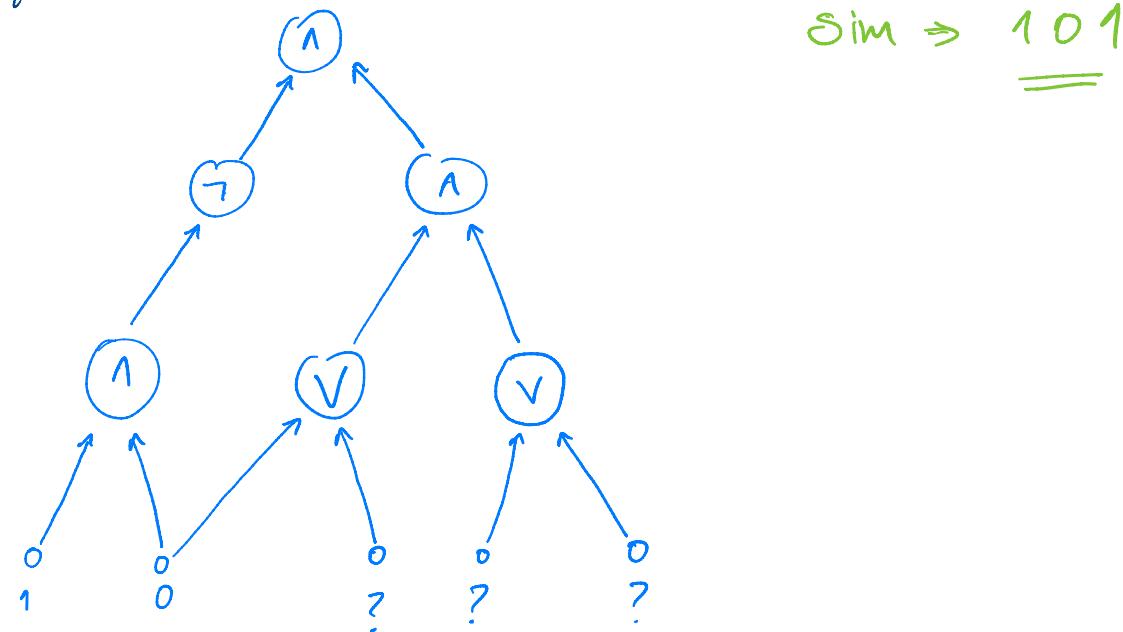
↳ Descobrir o certificado g nos permite
verificar X em tempo polinomial (fácil)

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 $x \in X$).