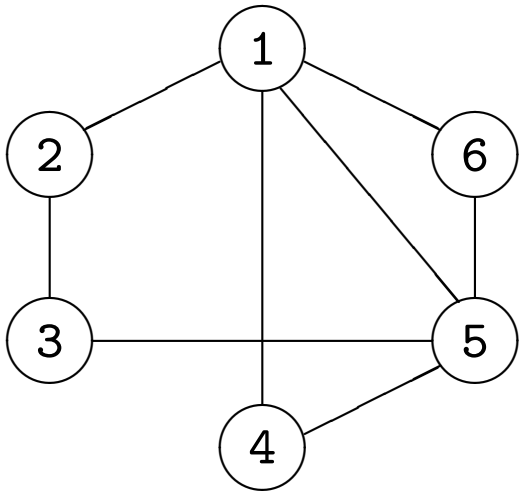
Capítulo XVI

Introdução à Teoria da Complexidade

Problema

Dado um grafo **não orientado e conexo**,

existe um circuito que passa uma, e uma só, vez por cada **arco**?



1, 6, 5, 4, 1, 2, 3, 5, 1

Circuito de EULER

- um circuito que passa uma, e uma só, vez por cada **arco**.

Dado um grafo G não orientado e conexo, G tem um circuito de Euler?

Em caso afirmativo, como encontrá-lo?

Proposições

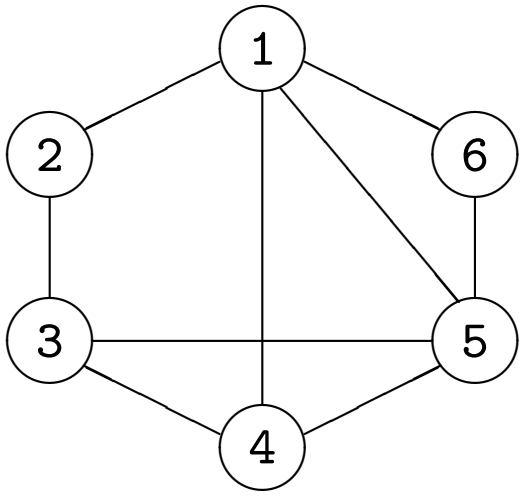
- Um grafo não orientado e conexo tem um circuito de Euler se, e só se, todos os vértices têm grau par.

  Consequência: **Existe** um **algoritmo polinomial** para descobrir se um grafo não orientado e conexo tem um circuito de Euler.

- **Existe** um **algoritmo polinomial** para encontrar um circuito de Euler num grafo não orientado e conexo cujos vértices têm grau par.

Problema

Dado um grafo **não orientado e conexo**,
existe um circuito que passa uma, e uma só, vez por cada **vértice**?



1, 2, 3, 4, 5, 6, 1

Circuito de HAMILTON

- um circuito simples que passa por todos os **vértices**.

Dado um grafo G não orientado e conexo, G tem um circuito de Hamilton? Em caso afirmativo, como encontrá-lo?

Facto

Embora os problemas do Circuito de Euler e do Circuito de Hamilton tenham enunciados muito parecidos, as respetivas soluções computacionais têm ordens de complexidade muito diferentes:

- **existem algoritmos polinomiais** para resolver os problemas do Circuito de Euler; mas
- **todos os algoritmos que se conhecem** para resolver os problemas do Circuito de Hamilton são **exponenciais**.

Problema, Instância & Solução

SAT: **Problema da Satisfazibilidade**

Dada uma fórmula proposicional f, f é satisfazível?

Instância: $p \wedge (q \vee \neg p)$ $p \wedge (q \vee \neg p)$ é satisfazível?

Solução: Sim

#-SAT: **Problema da #-Satisfazibilidade**

Dada uma fórmula proposicional f, quantas atribuições satisfazem f?

Instância: $p \wedge (q \vee \neg p)$ Quantas atribuições satisfazem $p \wedge (q \vee \neg p)$?

Solução: Uma

Um **problema de decisão** é um problema onde qualquer solução é "Sim" ou "Não".

Exemplo: **SAT** é um problema de decisão.

Algoritmo

Um **algoritmo para resolver um problema** é um procedimento que calcula a solução de **qualquer** instância do problema num número **finito** de passos.

Decidibilidade

Um **problema** de decisão diz-se:

- **decidível**, se existir algum algoritmo para o resolver;
- **indecidível**, se (foi provado que) não existe um algoritmo para o resolver.

Halting Problem

HALT: Dados um programa P e uma entrada E,

P termina com E?

Indecidível **[Turing 1936]**

O *Halting Problem* é indecidível, i.e., é impossível especificar um algoritmo que, dados um programa arbitrário e uma entrada arbitrária, decida se o programa termina com aquela entrada.

Cálculo de Predicados de Primeira Ordem

PRED: Dada uma fórmula f do Cálculo de Predicados de Primeira Ordem,

f é válida?

Instâncias

- $(\forall x\ (Px \Rightarrow Qx)) \Rightarrow ((\exists x Px) \Rightarrow (\exists x Qx))$
- $\forall y\ (\forall x Px \Rightarrow Py)$
- $\exists x(Px \vee Qx) \Rightarrow \exists x(Px \wedge Qx)$

Indecidível [Church 1936, Turing 1936/7]

Décimo Problema de Hilbert

(enunciado em 1900)

HILBERT: Dada uma equação polinomial de coeficientes inteiros,

$P(x_1, \ldots, x_n) = 0$,

$$P(x_1, \ldots, x_n) = 0 \text{ tem solução inteira?}$$

Instâncias

- $x^2 - 3x + 2 = 0$
- $7x^5y^2 - 4x^2y^3 + 14y^2 + 8y - 3 = 0$

Indecidível [Matijacevič 1970]

Ambiguidade de Gramáticas Independentes do Contexto

GIC-AMB: Dada uma gramática independente do contexto G,

G é ambígua?

Instâncias

- $(\{0,1\},\ \{S\},\ S,\ \{S \longrightarrow 0S \mid 1S \mid 0 \mid 1\})$
- $(\{0,1\},\ \{S\},\ S,\ \{S \longrightarrow SS \mid 0 \mid 1\})$

Indecidível [Bar-Hillel, Perles, Shamir 1961]

Algumas Classes de Complexidade

P, **PTIME**: a classe dos problemas de decisão resolúveis em tempo polinomial (i.e., para os quais existe algum algoritmo **determinista** cujo tempo é polinomial).

EXPTIME: a classe dos problemas de decisão resolúveis em tempo exponencial.

PSPACE: a classe dos problemas de decisão resolúveis em espaço polinomial.

Um **problema** diz-se:

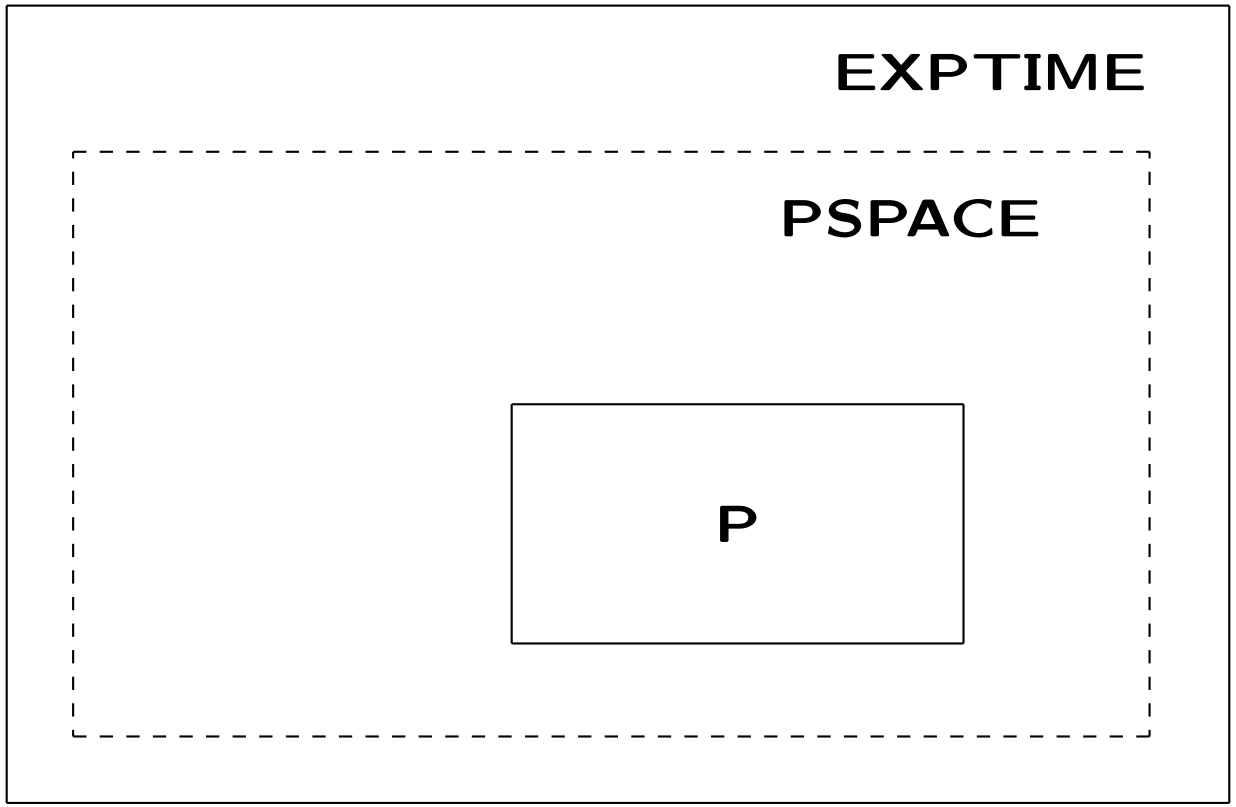
- **tratável**, se existir algum algoritmo (cujo tempo é) polinomial para o resolver;
- **intratável**, se (foi provado que) não existe um algoritmo (cujo tempo é) polinomial para o resolver.

Relações entre **P**, **PSPACE** e **EXPTIME**

$$\mathbf{P} \subseteq \mathbf{PSPACE} \subseteq \mathbf{EXP}$$

$\subset$ ou $=$? $\qquad\qquad$ $\subset$ ou $=$?

$$\mathbf{P} \subset \mathbf{EXP}$$



Igualdade de Expressões Regulares

ER-=: Dadas duas expressões regulares, E_1 e E_2,

$$\mathcal{L}(E_1) = \mathcal{L}(E_2)?$$

Instâncias

- $(a+b+c)^*$ e $(a+b+c)^*a^*$
- $(a+b)^*a^*$ e $((a+b)^*a)^*$
- $b^*a(a+b)^*$ e $(a+b)^*a(a+b)^*$

Intratável [Stockmeyer, Meyer 1973]

Satisfazibilidade / *Satisfiability*

SAT: Dada uma fórmula proposicional na forma normal conjuntiva,

$$f = \bigwedge_{1 \leq i \leq k} C_i,$$

f é satisfazível ?

Instâncias

- $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y) \wedge z$
- $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$
- $(x \vee y) \wedge (\neg x) \wedge (\neg y)$

NP

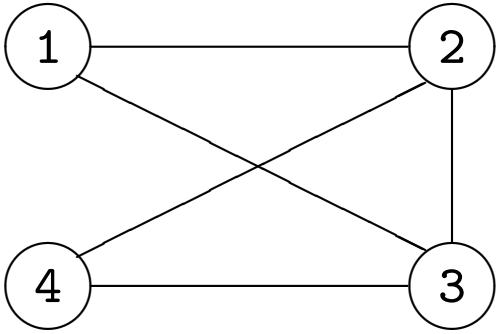
Circuito de Hamilton / *Hamiltonian Cycle*

Seja G um grafo não orientado e conexo. Um **circuito de Hamilton** em G é um circuito simples que passa por todos os vértices de G.

HAMILTON: Dado um grafo G, não orientado e conexo,

G tem um circuito de Hamilton?

Instância



NP

Partição de Conjunto / *Set Partition*

PARTCONJ: Dado um conjunto finito K de números inteiros positivos (cuja soma $\sum_{x \in K} x$ é par), existe um subconjunto $A \subseteq K$ tal que:

$$\sum_{x \in A} x = \sum_{x \in K \setminus A} x \, ?$$

Instâncias

$$\{1, 2, 3, 4, 5, 7\}$$

$$\{1, 2, 3, 4, 6, 10\}$$

NP

A Classe **NP** — Definição Formal

NP: a classe dos problemas de decisão X para os quais existe:

- um algoritmo polinomial $\mathcal{A}$

  (que verifica se *um candidato é a prova do sim*),

- uma função $\mathcal{P}$ (que fornece a *prova do sim*, quando ela existe) e
- uma constante k

tais que, para qualquer instância I de X:

- se a solução de I for "Sim" e o tamanho de I for n,

  então o tamanho de $\mathcal{P}(I)$ é $O(n^k)$ e $\mathcal{A}(I, \mathcal{P}(I))$ retorna "Sim";

- se a solução de I for "Não",

  então não existe nenhum valor para $\mathcal{P}(I)$

  que faça $\mathcal{A}(I, \mathcal{P}(I))$ retornar "Sim".

A Classe **NP** — Definições Intuitivas

NP: a classe dos problemas de decisão resolúveis em tempo polinomial **não-determinista**, ou seja, para os quais existe um algoritmo **não determinista** que, "*com muita sorte ou por magia*" , encontra a solução em tempo polinomial.

NP: a classe dos problemas de decisão para os quais é possível verificar em tempo polinomial "*se um qualquer candidato é uma prova do sim*" . (Esta é a definição usada nas demonstrações, como veremos a seguir.)

Satisfazibilidade / *Satisfiability*

SAT: Dada uma fórmula proposicional na forma normal conjuntiva,

$$f = \bigwedge_{1 \leq i \leq k} C_i,$$

f é satisfazível ?

Instâncias

- $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y) \wedge z$
- $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$
- $(x \vee y) \wedge (\neg x) \wedge (\neg y)$

Satisfazibilidade / *Satisfiability*

SAT: Dada uma fórmula proposicional na forma normal conjuntiva,

$$f = \bigwedge_{1 \leq i \leq k} C_i,$$

f é satisfazível ?

Instâncias

- $(x \vee y \vee \neg z) \wedge (\neg x \vee \neg y) \wedge z$
- $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$
- $(x \vee y) \wedge (\neg x) \wedge (\neg y)$

Candidato a Prova do Sim: uma atribuição de valores de verdade às variáveis da fórmula. Exemplo para 1ª instância: $((x, T), (y, F), (z, T))$.

NP-completo [Cook 1971]

SAT é um Problema **NP** (1)

O **Problema da Satisfazibilidade** é NP pelos dois seguintes motivos.

1. Existe um **algoritmo polinomial** que, dados:
   - uma fórmula proposicional $f = \bigwedge_{1 \leq i \leq k} C_i$, na forma normal conjuntiva, e
   - uma atribuição $\theta = \{(v_1, b_1), (v_2, b_2), \ldots, (v_n, b_n)\}$ de valores de verdade às variáveis de f,

   verifica se θ satisfaz f.

   O algoritmo avalia a fórmula, calculando o valor lógico de cada um dos termos C_i, para $i = 1, 2, \ldots, k$. Para avaliar $C_i = \bigvee_{1 \leq j \leq m_i} D_{ij}$, percorre os literais D_{ij}, para $j = 1, 2, \ldots, m_i$. Se o literal D_{ij} for uma variável v_u, o seu valor lógico é b_u. Se o literal tiver a forma $\neg v_u$, o seu valor lógico é o complementar de b_u. (Continua)

SAT é um Problema **NP** (2)

Os valores b_u (com $u = 1, 2, \ldots, n$) são obtidos através de pesquisas em θ. A avaliação de um termo C_i é **verdade** se o valor lógico de algum dos literais D_{ij} for **verdade**. O algoritmo retorna **true** se, e só se, a avaliação de todos os termos C_i for **verdade**.

A complexidade temporal do algoritmo é polinomial no número de símbolos de f. Se a atribuição θ estiver guardada num vetor e se f tiver n variáveis distintas, o_v ocorrências dessas variáveis e o_p ocorrências de operadores lógicos ($\neg$, $\vee$ e $\wedge$), o número de passos do algoritmo é $O(o_v\, n + o_p)$.

2. O **tamanho** da atribuição θ é **polinomial** no tamanho de f: θ tem $2n$ elementos, f tem $o_v + o_p$ elementos e $n \leq o_v$.

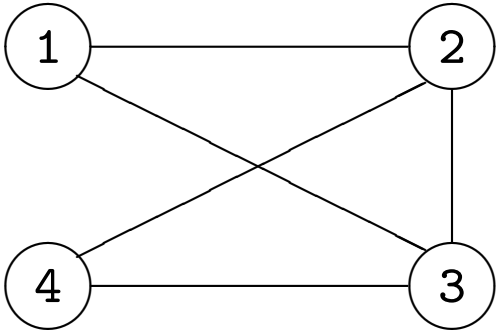
Circuito de Hamilton / *Hamiltonian Cycle*

Seja G um grafo não orientado e conexo. Um **circuito de Hamilton** em G é um circuito simples que passa por todos os vértices de G.

HAMILTON: Dado um grafo G, não orientado e conexo,

G tem um circuito de Hamilton?

Instância



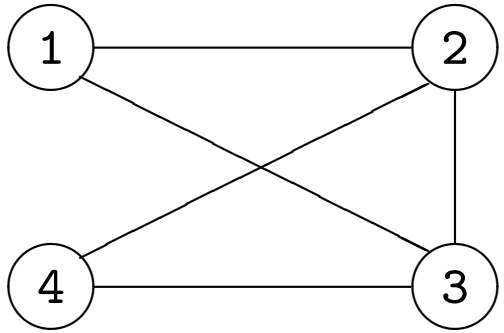
Circuito de Hamilton / *Hamiltonian Cycle*

Seja G um grafo não orientado e conexo. Um **circuito de Hamilton** em G é um circuito simples que passa por todos os vértices de G.

HAMILTON: Dado um grafo G, não orientado e conexo,

G tem um circuito de Hamilton?

Instância



Candidato a Prova do Sim: uma permutação dos vértices do grafo. Exemplo: 1 2 4 3.

NP-completo [Karp 1972]

HAMILTON é um Problema **NP** (1)

O **Problema do Circuito de Hamilton** é NP pelos dois seguintes motivos.

1. Existe um **algoritmo polinomial** que, dados:
   - um grafo $G = (V, A)$, não orientado e conexo, e
   - uma permutação $\rho = v_1\, v_2\, \cdots\, v_n$ dos vértices de G,

   verifica se $v_1\, v_2\, \cdots\, v_n\, v_1$ é um circuito de Hamilton em G.

   O algoritmo percorre a permutação ρ, testando se os arcos (v_i, v_{i+1}) pertencem a A, para todo o $i = 1, \ldots, n-1$. Depois, testa se o arco (v_n, v_1) também pertence a A. O algoritmo retorna **true** se, e só se, todos os testes tiverem sucesso. (Continua)

HAMILTON é um Problema **NP** (2)

A complexidade temporal do algoritmo é polinomial no número de vértices ($|V|$) e no número de arcos ($|A|$). Se o conjunto dos arcos estiver guardado num vetor, o número de passos do algoritmo é $O(n \times |A|) = O(|V| \times |A|)$.

2. O **tamanho** da permutação ρ é **polinomial** no tamanho de G, porque ρ tem $|V|$ vértices.

Partição de Conjunto / *Set Partition*

PARTCONJ: Dado um conjunto finito K de números inteiros positivos (cuja soma $\sum_{x \in K} x$ é par), existe um subconjunto $A \subseteq K$ tal que:

$$\sum_{x \in A} x = \sum_{x \in K \setminus A} x \, ?$$

Instâncias

$$\{1, 2, 3, 4, 5, 7\}$$

$$\{1, 2, 3, 4, 6, 10\}$$

Partição de Conjunto / *Set Partition*

PARTCONJ: Dado um conjunto finito K de números inteiros positivos (cuja soma $\sum_{x \in K} x$ é par), existe um subconjunto $A \subseteq K$ tal que:

$$\sum_{x \in A} x = \sum_{x \in K \setminus A} x \, ?$$

Instâncias

$$\{1, 2, 3, 4, 5, 7\}$$

$$\{1, 2, 3, 4, 6, 10\}$$

Candidato a Prova do Sim: um subconjunto do conjunto.
Exemplo para a 2ª instância: $\{3, 10\}$.

NP-completo

PARTCONJ é um Problema **NP** (1)

O **Problema da Partição de Conjunto** é NP pelos dois seguintes motivos.

1. Existe um **algoritmo polinomial** que, dados:
   - um conjunto finito K de números inteiros positivos (cuja soma é par) e
   - um subconjunto A de K,

   verifica se $\sum_{x \in A} x = \sum_{x \in K \setminus A} x$.

   O algoritmo percorre o conjunto K e calcula a soma T dos seus elementos. Depois, percorre o conjunto A e calcula a soma S dos seus elementos. O algoritmo retorna **true** se, e só se, o valor S for igual a $T - S$.

(Continua)

PARTCONJ é um Problema **NP** (2)

A complexidade temporal do algoritmo é polinomial no número de elementos dos conjuntos: $\Theta(|K| + |A|) = \Theta(|K|)$.

2. O **tamanho** do conjunto A é **polinomial** no tamanho de K, porque A é um subconjunto de K.

Redução de Problemas

Um problema X **reduz-se** a um problema Y se existir uma função ϕ que transforma qualquer instância I de X numa instância $\phi(I)$ de Y,

$$\begin{array}{rccc} \phi & : & X & \longrightarrow & Y \\ & & I & \longmapsto & \phi(I) \end{array}$$

de tal forma que:

$$\text{Solução}_X(I) = \text{Solução}_Y(\phi(I)).$$

Se ϕ pertencer à classe Ω, ϕ diz-se uma **redução Ω** de X para Y.

Notação:

$X \leq_P Y$ significa que há uma redução polinomial de X para Y.

PAR $\leq_P$ MULT

PAR: Dado um inteiro positivo k, k é par?

MULT: Dados dois inteiros positivos m e n, m é múltiplo de n?

PAR reduz-se a **MULT** porque a função

$$f : \textbf{PAR} \longrightarrow \textbf{MULT}$$
$$k \longmapsto$$

PAR $\leq_P$ MULT

PAR: Dado um inteiro positivo k, k é par?

MULT: Dados dois inteiros positivos m e n, m é múltiplo de n?

PAR reduz-se a **MULT** porque a função

$$\begin{array}{rcl} f : \textbf{PAR} & \longrightarrow & \textbf{MULT} \\ k & \longmapsto & (k, 2) \end{array}$$

é tal que:

$$\text{Solução}_{\textbf{PAR}}(k) = \text{Solução}_{\textbf{MULT}}((k, 2)),$$

ou seja:

$$k \text{ é par} \iff k \text{ é múltiplo de } 2.$$

A redução é polinomial porque a transformação f pode ser efetuada em tempo polinomial.

A **dificuldade** de **PAR** é **menor ou igual** à **dificuldade** de **MULT**.

Teorema **(T1)**

Redução $\phi : \begin{array}{ccc} X & \longrightarrow & Y \\ x & \longmapsto & \phi(x) \end{array}$

X indecidível $\Rightarrow$ Y **??????????**

Teorema **(T1)**

Redução $\phi : \begin{array}{ccc} X & \longrightarrow & Y \\ x & \longmapsto & \phi(x) \end{array}$

X indecidível $\Rightarrow$ Y **indecidível**

Demonstração (por contra-recíproco)

Suponhamos que Y é decidível, existindo um algoritmo para resolver Y.

Seja x uma instância qualquer de X.

Calculando $\phi(x)$ e executando o algoritmo que resolve Y com o input $\phi(x)$, obtém-se a solução de x em dois passos.

Estes dois passos definem um algoritmo para resolver X,

concluindo-se que X é decidível.

Teorema **(T2)**

Redução Polinomial $\phi : \begin{array}{ccc} X & \longrightarrow & Y \\ x & \longmapsto & \phi(x) \end{array}$

X intratável $\Rightarrow$ Y **??????????**

Teorema **(T2)**

Redução Polinomial $\phi : \begin{array}{ccc} X & \longrightarrow & Y \\ x & \longmapsto & \phi(x) \end{array}$

X intratável $\Rightarrow$ Y **intratável**

Demonstração (por contra-recíproco)

Suponhamos que Y é tratável, existindo um algoritmo polinomial para resolver Y.

Seja x uma instância qualquer de X.

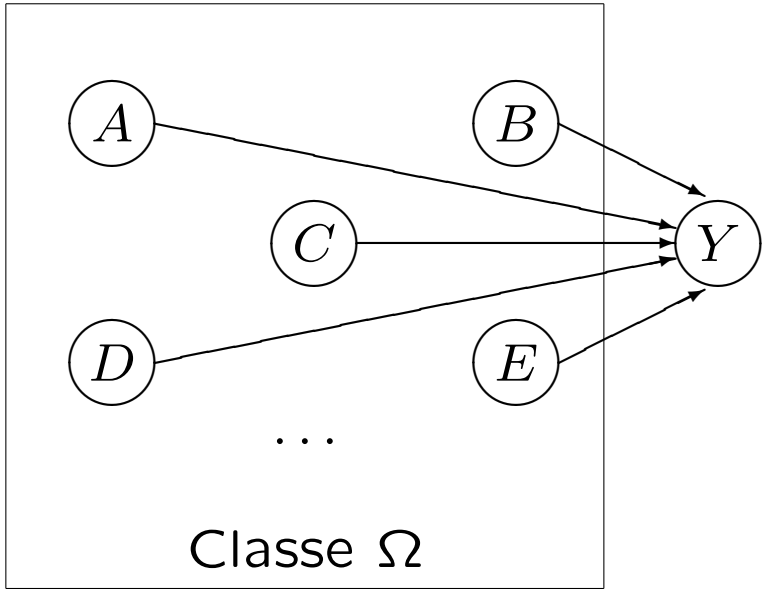
Calculando $\phi(x)$ e executando o algoritmo polinomial que resolve Y com o input $\phi(x)$, obtém-se a solução de x em dois passos polinomiais.

Estes dois passos definem um algoritmo polinomial para resolver X, concluindo-se que X é tratável.

O Sufixo Difícil (*Hard*)

Seja Ω a classe **P**, **NP**, **EXPTIME** ou **PSPACE**.

Um problema Y é Ω-**difícil** se qualquer problema em Ω se reduz a Y através de uma redução polinomial: $(\forall X \in \Omega)\ X \leq_P Y$.

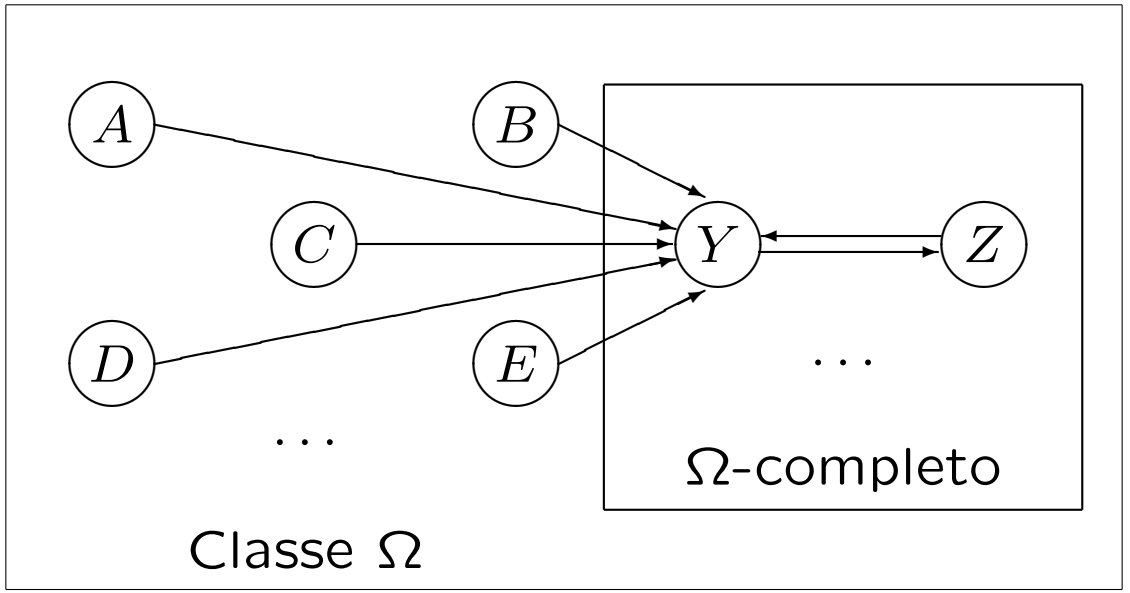


Y é **pelo menos tão difícil como** os problemas em Ω

O Sufixo Completo (*Complete*)

Seja Ω a classe **P**, **NP**, **EXPTIME** ou **PSPACE**.

Um problema Y é Ω-**completo** se Y é Ω-difícil e $Y \in \Omega$.



Y é **um dos problemas mais difíceis** de Ω

Teorema **(T3)**

Redução Polinomial $\phi : \begin{array}{ccc} X & \longrightarrow & Y \\ x & \longmapsto & \phi(x) \end{array}$

$$X\ \Omega\text{-completo} \quad \Rightarrow \quad Y\ \textbf{Ω-difícil}$$

Demonstração

Seja A um problema qualquer na classe Ω.

Como X é Ω-completo, é Ω-difícil e existe uma redução polinomial

$$\psi : A \longrightarrow X.$$

Efetuando a redução $\psi : A \longrightarrow X$, seguida da redução $\phi : X \longrightarrow Y$, obtém-se uma redução polinomial de A para Y.

Exemplos

GIC-$\emptyset$: Dada uma gramática independente do contexto G,

$$\mathcal{L}(G) = \emptyset?$$

P-completo [Jones, Laaser 1977]

HORN: Dados um conjunto H, de cláusulas de Horn fechadas, e uma fórmula atómica fechada f,

$$H \vdash f?$$

P-completo [Jones, Laaser 1977]

ER-T^*: Dada uma expressão regular R, sobre um alfabeto T,

$$\mathcal{L}(R) = T^*?$$

PSPACE-completo [Meyer, Stockmeyer 1972]

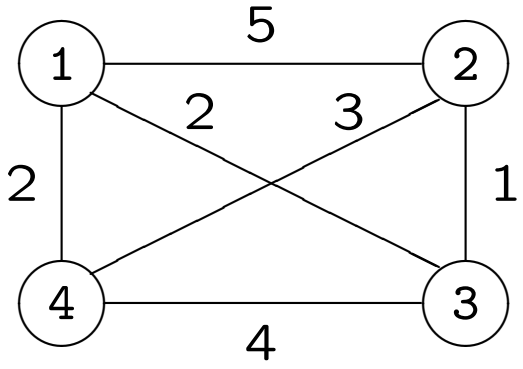
Caixeiro Viajante / *Travelling Salesman*

CAIXEIRO: Dados um grafo G, não orientado, pesado e completo, cujos arcos têm custo inteiro positivo, e um inteiro positivo k,

G tem um circuito de Hamilton

cujo comprimento pesado não excede k?

Instância



$k = 8$

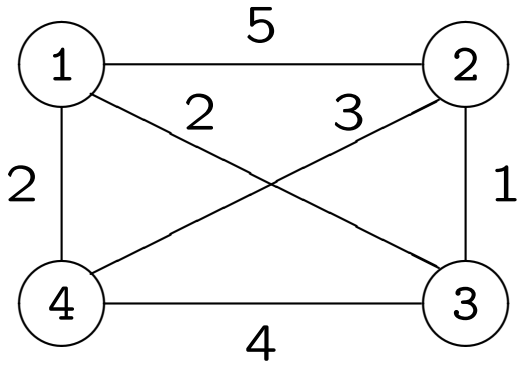
Caixeiro Viajante / *Travelling Salesman*

CAIXEIRO: Dados um grafo G, não orientado, pesado e completo, cujos arcos têm custo inteiro positivo, e um inteiro positivo k,

G tem um circuito de Hamilton

cujo comprimento pesado não excede k?

Instância



$k = 8$

Candidato a Prova do Sim: uma permutação dos vértices do grafo. Exemplo: 1 3 2 4.

NP-completo

CAIXEIRO é um Problema **NP-completo** (1)

O **Problema do Caixeiro Viajante** é NP-completo porque é NP e NP-difícil.

CAIXEIRO é um Problema **NP-completo** (2)

O **Problema do Caixeiro Viajante** é NP pelos dois seguintes motivos.

1. Existe um **algoritmo polinomial** que, dados:
   - um grafo $G = (V, A)$, não orientado, pesado e completo, cujos arcos têm custo inteiro positivo,
   - um inteiro positivo k e
   - uma permutação $\rho = v_1\, v_2\, \cdots\, v_n$ dos vértices de G,

   verifica se o comprimento pesado de $v_1\, v_2\, \cdots\, v_n\, v_1$ não excede k.

   O algoritmo percorre a permutação ρ, calculando a soma dos pesos dos arcos (v_i, v_{i+1}), para todo o $i = 1, \ldots, n-1$. Ao resultado, soma o peso do arco (v_n, v_1). O algoritmo retorna **true** se, e só se, o resultado final for inferior ou igual a k. (Continua)

CAIXEIRO é um Problema **NP-completo** (3)

A complexidade temporal do algoritmo é polinomial no número de vértices ($|V|$) e no número de arcos ($|A|$). Se o conjunto dos arcos estiver guardado num vetor, o número de passos do algoritmo é $O(n \times |A|) = O(|V| \times |A|)$.

Curiosidade: A complexidade temporal do algoritmo também é $O(|V|^3)$ porque, como o grafo é não orientado e completo,

$$|A| = \frac{|V| \times (|V| - 1)}{2} = \Theta(|V|^2).$$

2. O **tamanho** da permutação ρ é **polinomial** no tamanho de (G, k), porque ρ tem $|V|$ vértices.

CAIXEIRO é um Problema **NP-completo** (4)

O **Problema do Caixeiro Viajante** é NP-difícil porque o Problema do Circuito de Hamilton é NP-completo e a seguinte redução é **polinomial**.

$$\begin{array}{ccccc} \phi & : & \textbf{HAMILTON} & \longrightarrow & \textbf{CAIXEIRO} \\ & & ((V,A)) & \longmapsto & ??? \end{array}$$

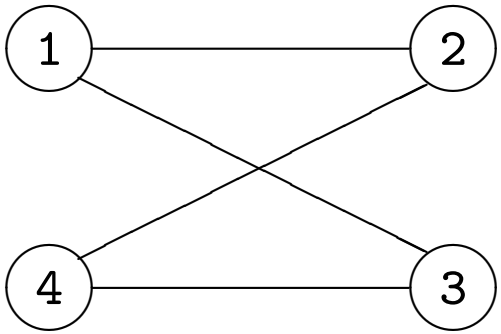
HAMILTON: Dado um grafo G, não orientado e conexo,

G tem um circuito de Hamilton?

HAMILTON $\longrightarrow$ CAIXEIRO

Grafo não orientado
e conexo.

$\exists$ circuito de Hamilton?



SIM

$\Longleftrightarrow$

Grafo não orientado, completo
e pesado (pesos positivos);
$k \geq 1$.

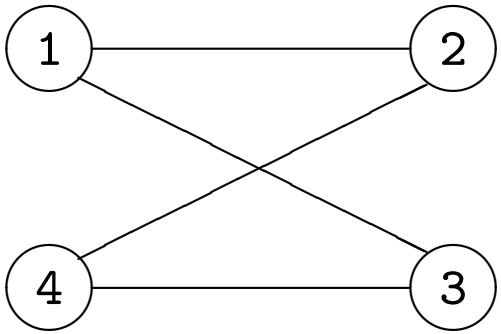
$\exists$ circuito de Hamilton com
comprimento pesado $\leq k$?

SIM

HAMILTON $\longrightarrow$ CAIXEIRO

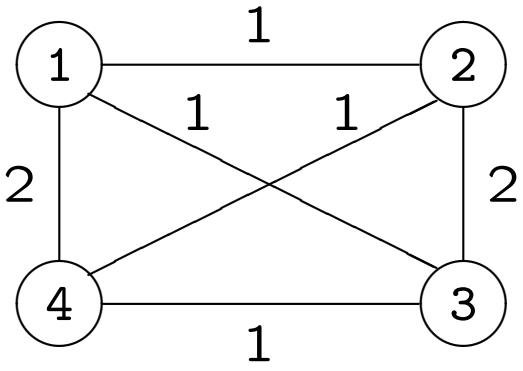
Grafo não orientado
e conexo.

$\exists$ circuito de Hamilton?



Grafo não orientado, completo
e pesado (pesos positivos);
$k \geq 1$.

$\exists$ circuito de Hamilton com
comprimento pesado $\leq k$?



$\boxed{k = 4}$

SIM 1 3 4 2 1 $\iff$ **SIM** 1 3 4 2 1

dificuldade de **HAMILTON** $\leq$ **dificuldade** de **CAIXEIRO**

CAIXEIRO é um Problema **NP-completo** (4)

O **Problema do Caixeiro Viajante** é NP-difícil porque o Problema do Circuito de Hamilton é NP-completo e a seguinte redução é **polinomial**.

$$\begin{array}{rcl} \phi \ : \ \textbf{HAMILTON} & \longrightarrow & \textbf{CAIXEIRO} \\ ((V, A)) & \longmapsto & ((V, A'),\ |V|) \end{array}$$

onde:

$$A' = \{(x, y, 1) \mid (x, y) \in A\} \cup \{(x, y, 2) \mid x \in V,\ y \in V,\ x \neq y,\ (x, y) \notin A\}$$

Mochila 0-1 / *0-1 Knapsack*

MOCH01: Seja I um conjunto finito de itens. Cada item $i \in I$ tem um peso w_i e um valor v_i, ambos inteiros não negativos. Sejam $C \geq 0$ a capacidade da mochila e $V \geq 0$. Existe um subconjunto $S \subseteq I$ tal que:

$$\sum_{i \in S} w_i \leq C \quad \text{e} \quad \sum_{i \in S} v_i \geq V \, ?$$

Forma das Instâncias

$$(\{(1, w_1, v_1),\ (2, w_2, v_2),\ \ldots,\ (n, w_n, v_n)\},\ \ C,\ \ V)$$

Instâncias

$$(\{(1,5,10),\ (2,8,70),\ (3,6,30),\ (4,3,50)\},\ \ 10,\ \ 80)$$
$$(\{(1,5,10),\ (2,8,70),\ (3,6,30),\ (4,3,50)\},\ \ 10,\ \ 100)$$

Mochila 0-1 / *0-1 Knapsack*

MOCH01: Seja I um conjunto finito de itens. Cada item $i \in I$ tem um peso w_i e um valor v_i, ambos inteiros não negativos. Sejam $C \geq 0$ a capacidade da mochila e $V \geq 0$. Existe um subconjunto $S \subseteq I$ tal que:

$$\sum_{i \in S} w_i \leq C \quad \text{e} \quad \sum_{i \in S} v_i \geq V \, ?$$

Forma das Instâncias

$$(\{(1, w_1, v_1),\ (2, w_2, v_2),\ \ldots,\ (n, w_n, v_n)\},\ \ C,\ \ V)$$

Instâncias

$$(\{(1,5,10),\ (2,8,70),\ (3,6,30),\ (4,3,50)\},\ \ 10,\ \ 80)$$
$$(\{(1,5,10),\ (2,8,70),\ (3,6,30),\ (4,3,50)\},\ \ 10,\ \ 100)$$

Candidato a Prova do Sim: um subconjunto do conjunto de itens. Exemplo: $\{(3,6,30),\ (4,3,50)\}$.

NP-completo

MOCH01 é um Problema **NP-completo** (1)

- **Problema da Mochila 0-1** é NP-completo porque é NP e NP-difícil.

MOCH01 é um Problema **NP-completo** (2)

O **Problema da Mochila 0-1** é NP pelos dois seguintes motivos.

1. Existe um **algoritmo polinomial** que, dados:
   - um conjunto finito I de itens, onde cada item $i \in I$ tem um peso w_i e um valor v_i (que são inteiros não negativos),
   - dois inteiros positivos C e V e
   - um subconjunto S de I,

   verifica se $\sum_{i \in S} w_i \leq C$ e $\sum_{i \in S} v_i \geq V$.

   O algoritmo percorre o conjunto S e calcula a soma W_S dos pesos dos seus itens e a soma V_S dos valores dos seus itens. O algoritmo retorna **true** se, e só se, o valor W_S for inferior ou igual a C e o valor V_S for superior ou igual a V. (Continua)

MOCH01 é um Problema **NP-completo** (3)

A complexidade temporal do algoritmo é polinomial no número de elementos de S: $\Theta(|S|)$.

2. O **tamanho** do conjunto S é **polinomial** no tamanho de (I, C, V), porque S é um subconjunto de I.

MOCH01 é um Problema **NP-completo** (4)

O **Problema da Mochila 0-1** é NP-difícil porque o Problema da Partição de Conjunto é NP-completo e a seguinte redução é **polinomial**.

$$\begin{array}{rcl} \phi : & \textbf{PARTCONJ} & \longrightarrow \textbf{MOCH01} \\ & K = \{x_1, x_2, \ldots, x_n\} & \longmapsto \quad ??? \end{array}$$

PARTCONJ: Dado um conjunto finito K de números inteiros positivos (cuja soma $\sum_{x \in K} x$ é par), existe um subconjunto $A \subseteq K$ tal que:

$$\sum_{x \in A} x = \sum_{x \in K \setminus A} x \, ?$$

PARTCONJ $\longrightarrow$ MOCH01

Conjunto finito de inteiros positivos.	Conjunto finito de itens, cada um com peso e valor; capacidade $C \geq 0$; valor $V \geq 0$.
$\exists$ subconjunto cuja soma é metade do total?	$\exists$ subconjunto tal que $\sum$ pesos $\leq C$ e $\sum$ valores $\geq V$?
$\{1, 2, 3, 4, 6, 10\}$	
SIM	**SIM**

$\Longleftrightarrow$

PARTCONJ $\longrightarrow$ MOCH01

Conjunto finito de inteiros positivos.	Conjunto finito de itens, cada um com peso e valor; capacidade $C \geq 0$; valor $V \geq 0$.
$\exists$ subconjunto cuja soma é metade do total?	$\exists$ subconjunto tal que $\sum$ pesos $\leq C$ e $\sum$ valores $\geq V$?
$\{1, 2, 3, 4, 6, 10\}$	$\{(1,1,1),\ (2,2,2),\ (3,3,3),$ $(4,4,4),\ (5,6,6),\ (6,10,10)\}$ $C = 13 \quad V = 13$
SIM $\{3,\ 10\}$	$\Longleftrightarrow$ **SIM** $\{(3,3,3),\ (6,10,10)\}$

dificuldade de **PARTCONJ** $\leq$ **dificuldade** de **MOCH01**

MOCH01 é um Problema **NP-completo** (4)

O **Problema da Mochila 0-1** é NP-difícil porque o Problema da Partição de Conjunto é NP-completo e a seguinte redução é **polinomial**.

$$\begin{array}{rcl} \phi : & \textbf{PARTCONJ} & \longrightarrow \textbf{MOCH01} \\ & K = \{x_1, x_2, \ldots, x_n\} & \longmapsto \quad (I, m, m) \end{array}$$

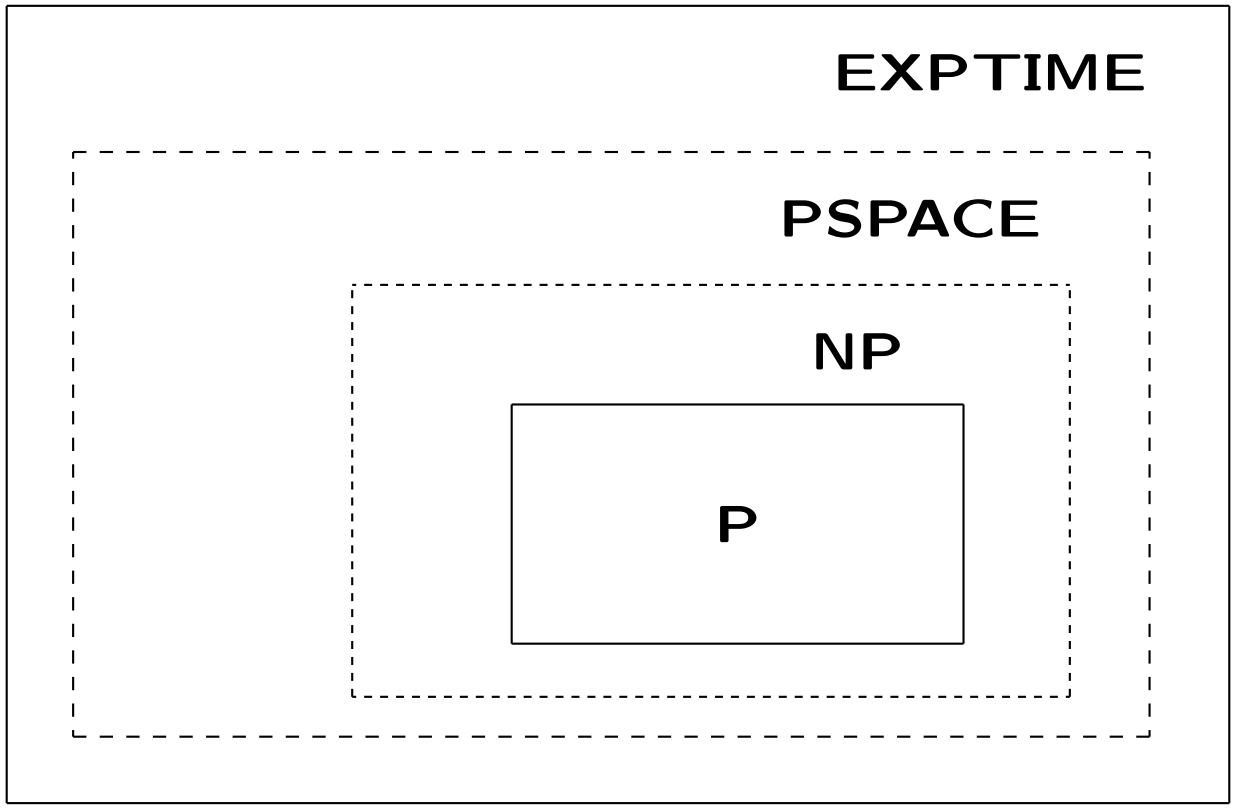
onde:

$$I = \{(1, x_1, x_1),\ (2, x_2, x_2),\ \ldots,\ (n, x_n, x_n)\} \quad \text{e}$$

$$m = \frac{1}{2} \sum_{i=1}^{n} x_i$$

Onde Colocar a Classe **NP**?

$$\mathbf{P} \underset{\subset \text{ ou } = ?}{\subseteq} \mathbf{NP} \underset{\subset \text{ ou } = ?}{\subseteq} \mathbf{PSPACE}$$



Principal Questão em Aberto: $\mathbf{P} = \mathbf{NP}$?

Se alguém criar um algoritmo polinomial que resolve algum problema **NP-completo**, então:

$$\mathbf{P} = \mathbf{NP}.$$

Se alguém provar que algum problema **NP** não pode ser resolvido por um algoritmo polinomial, então:

$$\mathbf{P} \subset \mathbf{NP} \quad \text{e} \quad \mathbf{P} \cap \mathbf{NP\text{-}completo} = \emptyset.$$

P = NP?

Se alguém criar um algoritmo polinomial que resolve algum problema X **NP-completo**, então:

$$\mathbf{P} = \mathbf{NP}.$$

Demonstração

Sabe-se que **P** $\subseteq$ **NP**. Provemos que **NP** $\subseteq$ **P**.

Seja A um problema **NP** qualquer.

Como X é **NP-completo**, existe uma redução polinomial $\phi : A \longrightarrow X$.

Como X é tratável, pelo Teorema **(T2)**, A é tratável.

Portanto, A pertence à classe **P** (é um problema de decisão tratável).

P = NP?

Se alguém provar que algum problema **NP** não pode ser resolvido por um algoritmo polinomial, então:

$$\textbf{P} \subset \textbf{NP} \quad \text{e} \quad \textbf{P} \cap \textbf{NP-completo} = \emptyset.$$

Se existir um problema X, **NP** e **intratável**, então:

$$\textbf{NP-completo} \subseteq \textbf{Intratável}.$$

Demonstração

Seja A um problema **NP-completo** qualquer.

Se X é **NP**, existe uma redução polinomial $\phi : X \longrightarrow A$.

Como X é intratável, pelo Teorema **(T2)**, A é intratável.

Ainda há muitas Questões em Aberto

e,

de vez em quando, surgem Novos Resultados

Teste à Primalidade

PRIMO: Dado um número inteiro $p \geq 2$,

p é primo?

P [Agrawal, Kayal, Saxena **2002**]