Introdução à Teoria da Computação

Complexidade Computacional: A classe NP

Professor Murilo V. G. da Silva

Departamento de Informática

Universidade Federal do Paraná

2025 / 2

Decidir *vs* Verificar

	2		5		1		9	
8			2		3			6
	3			6			7	
		1				6		
5	4						1	9
		2				7		
	9			3			8	
2			8		4			7
	1		9		7		6	

Problema

4	2	*6*	5	*7*	1	*3*	9	*8*
8	*5*	*7*	2	*9*	3	*1*	*4*	6
1	3	*9*	*4*	6	*8*	*2*	7	*5*
9	*7*	1	*3*	*8*	*5*	6	*2*	*4*
5	4	*3*	*7*	*2*	*6*	*8*	1	9
6	*8*	2	*1*	*4*	*9*	7	*5*	*3*
7	9	*4*	*6*	3	*2*	*5*	8	*1*
2	*6*	*5*	8	*1*	4	*9*	*3*	7
3	1	*8*	9	*5*	7	*4*	6	*2*

Solução

Figura: O Problema Sudoku.

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, \ldots, x_n$ escrita em CNF

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$.

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$. neste caso $n = 3$

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$. neste caso $n = 3$

Seja ϕ uma instância verdadeira do problema SAT,

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$. neste caso $n = 3$

Seja ϕ uma instância verdadeira do problema SAT, com $n = 1000$.

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$. neste caso $n = 3$

Seja ϕ uma instância verdadeira do problema SAT, com $n = 1000$. Considere os dois cenários:

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$. neste caso $n = 3$

Seja ϕ uma instância verdadeira do problema SAT, com $n = 1000$. Considere os dois cenários:

(1) Forneço a fórmula CNF ϕ e afirmo que a fórmula é satisfazível.

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$. neste caso $n = 3$

Seja ϕ uma instância verdadeira do problema SAT, com $n = 1000$. Considere os dois cenários:

(1) Forneço a fórmula CNF ϕ e afirmo que a fórmula é satisfazível.

(2) Forneço a fórmula CNF ϕ juntamente com uma valoração $v = v_1, ..., v_n$ que satisfaz ϕ

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$. neste caso $n = 3$

Seja ϕ uma instância verdadeira do problema SAT, com $n = 1000$. Considere os dois cenários:

(1) Forneço a fórmula CNF ϕ e afirmo que a fórmula é satisfazível.

(2) Forneço a fórmula CNF ϕ juntamente com uma valoração $v = v_1, ..., v_n$ que satisfaz ϕ (i.e., valores $v_i \in \{\text{V}, \text{F}\}$, tal que, fazendo $x_i = v_i$, a fórmula ϕ assume o valor verdade V)

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$. neste caso $n = 3$

Seja ϕ uma instância verdadeira do problema SAT, com $n = 1000$. Considere os dois cenários:

(1) Forneço a fórmula CNF ϕ e afirmo que a fórmula é satisfazível.

(2) Forneço a fórmula CNF ϕ juntamente com uma valoração $v = v_1, ..., v_n$ que satisfaz ϕ (i.e., valores $v_i \in \{\mathrm{V}, \mathrm{F}\}$, tal que, fazendo $x_i = v_i$, a fórmula ϕ assume o valor verdade V)

Em qual cenário você fica mais facilmente convencido que ϕ é satisfazível?

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$. neste caso $n = 3$

Seja ϕ uma instância verdadeira do problema SAT, com $n = 1000$. Considere os dois cenários:

(1) Forneço a fórmula CNF ϕ e afirmo que a fórmula é satisfazível.

(2) Forneço a fórmula CNF ϕ juntamente com uma valoração $v = v_1, ..., v_n$ que satisfaz ϕ (i.e., valores $v_i \in \{\mathrm{V}, \mathrm{F}\}$, tal que, fazendo $x_i = v_i$, a fórmula ϕ assume o valor verdade V)

Em qual cenário você fica mais facilmente convencido que ϕ é satisfazível?

Note que no cenário (2), juntamente com a instância verdadeira, foi fornecido um *certificado* que atesta ela é realmente verdadeira.

Decidir *vs* Verificar: O problema SAT

Considere uma fórmula booleana com n variáveis $x_1, ..., x_n$ escrita em CNF

- Por exemplo, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$. neste caso $n = 3$

Seja ϕ uma instância verdadeira do problema SAT, com $n = 1000$. Considere os dois cenários:

(1) Forneço a fórmula CNF ϕ e afirmo que a fórmula é satisfazível.

(2) Forneço a fórmula CNF ϕ juntamente com uma valoração $v = v_1, ..., v_n$ que satisfaz ϕ (i.e., valores $v_i \in \{\mathrm{V}, \mathrm{F}\}$, tal que, fazendo $x_i = v_i$, a fórmula ϕ assume o valor verdade V)

Em qual cenário você fica mais facilmente convencido que ϕ é satisfazível?

Note que no cenário (2), juntamente com a instância verdadeira, foi fornecido um *certificado* que atesta ela é realmente verdadeira. (a valoração v é o certificado de ϕ)

SAT é verificável em tempo polinomial

Verificador_SAT: (ϕ, v)

1: **for each** $C_1, C_2, \ldots C_m$ **do**
2: $\quad C_i = \text{FALSE}$
3: $\quad$ **for each** $\mathrm{l_{ij}}$ de C_i **do**
4: $\quad\quad$ **if** valoração v satisfaz $\mathrm{l_{ij}}$ **then**
5: $\quad\quad\quad C_i = \text{TRUE}$
6: $\quad\quad\quad$ Break
7: $\quad$ **if** $C_i = \text{FALSE}$ **then**
8: $\quad\quad$ Return FALSE
9: Return TRUE

SAT é verificável em tempo polinomial

Verificador_SAT: (ϕ, v)

1: **for each** $C_1, C_2, \ldots C_m$ **do**
2: $\quad C_i = \text{FALSE}$
3: $\quad$ **for each** $\mathrm{l_{ij}}$ de C_i **do**
4: $\quad\quad$ **if** valoração v satisfaz $\mathrm{l_{ij}}$ **then**
5: $\quad\quad\quad C_i = \text{TRUE}$
6: $\quad\quad\quad$ Break
7: $\quad$ **if** $C_i = \text{FALSE}$ **then**
8: $\quad\quad$ Return FALSE
9: Return TRUE

Exemplo:

SAT é verificável em tempo polinomial

Verificador_SAT: (ϕ, v)

1: **for each** $C_1, C_2, \ldots C_m$ **do**
2: $\quad C_i = \text{FALSE}$
3: $\quad$ **for each** $\mathrm{l_{ij}}$ de C_i **do**
4: $\quad\quad$ **if** valoração v satisfaz $\mathrm{l_{ij}}$ **then**
5: $\quad\quad\quad C_i = \text{TRUE}$
6: $\quad\quad\quad$ Break
7: $\quad$ **if** $C_i = \text{FALSE}$ **then**
8: $\quad\quad$ Return FALSE
9: Return TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 110$

SAT é verificável em tempo polinomial

Verificador_SAT: (ϕ, v)

1: **for each** $C_1, C_2, \ldots C_m$ **do**
2: $\quad C_i = \text{FALSE}$
3: $\quad$ **for each** $\mathrm{l_{ij}}$ de C_i **do**
4: $\quad\quad$ **if** valoração v satisfaz $\mathrm{l_{ij}}$ **then**
5: $\quad\quad\quad C_i = \text{TRUE}$
6: $\quad\quad\quad$ Break
7: $\quad$ **if** $C_i = \text{FALSE}$ **then**
8: $\quad\quad$ Return FALSE
9: Return TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 110$ Algoritmo devolve TRUE

SAT é verificável em tempo polinomial

Verificador_SAT: (ϕ, v)

1: **for each** $C_1, C_2, \ldots C_m$ **do**
2: $\quad C_i = \text{FALSE}$
3: $\quad$ **for each** $\mathrm{l_{ij}}$ de C_i **do**
4: $\quad\quad$ **if** valoração v satisfaz $\mathrm{l_{ij}}$ **then**
5: $\quad\quad\quad C_i = \text{TRUE}$
6: $\quad\quad\quad$ Break
7: $\quad$ **if** $C_i = \text{FALSE}$ **then**
8: $\quad\quad$ Return FALSE
9: Return TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 110$ Algoritmo devolve TRUE

Exemplo:

SAT é verificável em tempo polinomial

Verificador_SAT: (ϕ, v)

1: **for each** $C_1, C_2, \ldots C_m$ **do**
2: $\quad C_i = \text{FALSE}$
3: $\quad$ **for each** $\mathrm{l_{ij}}$ de C_i **do**
4: $\quad\quad$ **if** valoração v satisfaz $\mathrm{l_{ij}}$ **then**
5: $\quad\quad\quad C_i = \text{TRUE}$
6: $\quad\quad\quad$ Break
7: $\quad$ **if** $C_i = \text{FALSE}$ **then**
8: $\quad\quad$ Return FALSE
9: Return TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 110$ Algoritmo devolve TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 111$

SAT é verificável em tempo polinomial

Verificador_SAT: (ϕ, v)

1: **for each** $C_1, C_2, \ldots C_m$ **do**
2: $\quad C_i = \text{FALSE}$
3: $\quad$ **for each** $\mathrm{l_{ij}}$ de C_i **do**
4: $\quad\quad$ **if** valoração v satisfaz $\mathrm{l_{ij}}$ **then**
5: $\quad\quad\quad C_i = \text{TRUE}$
6: $\quad\quad\quad$ Break
7: $\quad$ **if** $C_i = \text{FALSE}$ **then**
8: $\quad\quad$ Return FALSE
9: Return TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 110$ Algoritmo devolve TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 111$ Algoritmo devolve FALSE

SAT é verificável em tempo polinomial

Verificador_SAT: (ϕ, v)

1: **for each** $C_1, C_2, \ldots C_m$ **do**
2: $\quad C_i = \text{FALSE}$
3: $\quad$ **for each** $\mathrm{l_{ij}}$ de C_i **do**
4: $\quad\quad$ **if** valoração v satisfaz $\mathrm{l_{ij}}$ **then**
5: $\quad\quad\quad C_i = \text{TRUE}$
6: $\quad\quad\quad$ Break
7: $\quad$ **if** $C_i = \text{FALSE}$ **then**
8: $\quad\quad$ Return FALSE
9: Return TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 110$ Algoritmo devolve TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 111$ Algoritmo devolve FALSE

Exemplo:

SAT é verificável em tempo polinomial

Verificador_SAT: (ϕ, v)

1: **for each** $C_1, C_2, \ldots C_m$ **do**
2: $\quad C_i = \text{FALSE}$
3: $\quad$ **for each** $\mathrm{l_{ij}}$ de C_i **do**
4: $\quad\quad$ **if** valoração v satisfaz $\mathrm{l_{ij}}$ **then**
5: $\quad\quad\quad C_i = \text{TRUE}$
6: $\quad\quad\quad$ Break
7: $\quad$ **if** $C_i = \text{FALSE}$ **then**
8: $\quad\quad$ Return FALSE
9: Return TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 110$ Algoritmo devolve TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 111$ Algoritmo devolve FALSE

Exemplo: Se $\phi_2 = (\overline{x_1} \vee \overline{x_2}) \wedge (x_1) \wedge (x_2)$ e $v = 00$

SAT é verificável em tempo polinomial

Verificador_SAT: (ϕ, v)

1: **for each** $C_1, C_2, \ldots C_m$ **do**
2: $\quad C_i = \text{FALSE}$
3: $\quad$ **for each** $\mathrm{l_{ij}}$ de C_i **do**
4: $\quad\quad$ **if** valoração v satisfaz $\mathrm{l_{ij}}$ **then**
5: $\quad\quad\quad C_i = \text{TRUE}$
6: $\quad\quad\quad$ Break
7: $\quad$ **if** $C_i = \text{FALSE}$ **then**
8: $\quad\quad$ Return FALSE
9: Return TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 110$ Algoritmo devolve TRUE

Exemplo: Se $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_3})$ e $v = 111$ Algoritmo devolve FALSE

Exemplo: Se $\phi_2 = (\overline{x_1} \vee \overline{x_2}) \wedge (x_1) \wedge (x_2)$ e $v = 00$ Algoritmo devolve FALSE

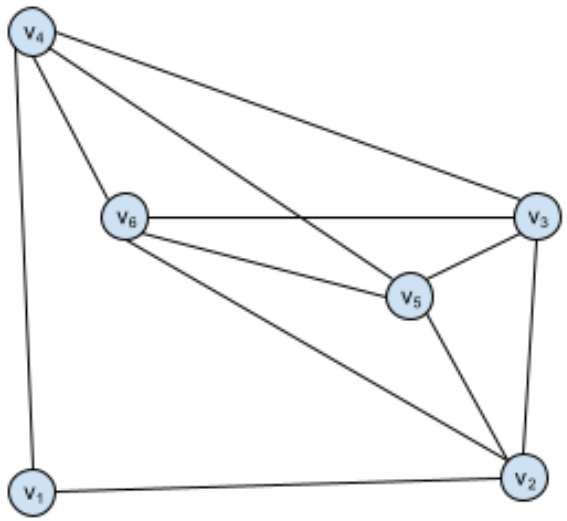
Decidir *vs* Verificar: Grafos Hamiltonianos

Considere o problema de testar se um grafo G com n vértices $v_1, \ldots, v_n$ é hamiltoniano

Decidir *vs* Verificar: Grafos Hamiltonianos

Considere o problema de testar se um grafo G com n vértices $v_1, \ldots, v_n$ é hamiltoniano

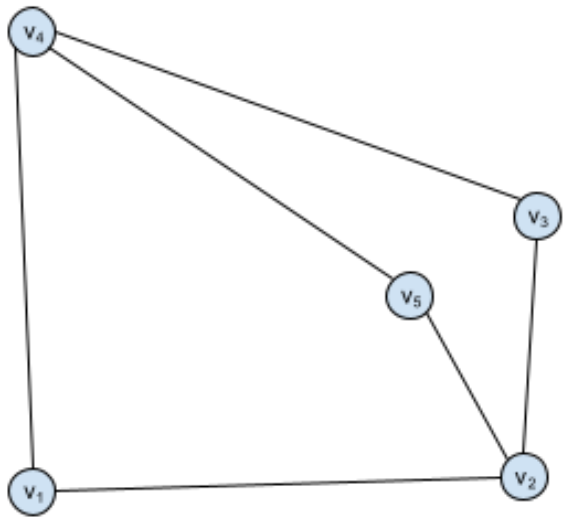
- e.g., o grafo G abaixo é hamiltoniano:



Decidir *vs* Verificar: Grafos Hamiltonianos

Considere o problema de testar se um grafo G com n vértices $v_1, \ldots, v_n$ é hamiltoniano

- e.g., o grafo G abaixo NÃO hamiltoniano:



Decidir *vs* Verificar: Grafos Hamiltonianos

Considere o problema de testar se um grafo G com n vértices $v_1, ..., v_n$ é hamiltoniano

Decidir *vs* Verificar: Grafos Hamiltonianos

Considere o problema de testar se um grafo G com n vértices $v_1, ..., v_n$ é hamiltoniano

Seja G uma instância verdadeira do problema L_{HAM}.

Decidir *vs* Verificar: Grafos Hamiltonianos

Considere o problema de testar se um grafo G com n vértices $v_1, ..., v_n$ é hamiltoniano

Seja G uma instância verdadeira do problema L_{HAM}. Considere os dois cenários:

Decidir *vs* Verificar: Grafos Hamiltonianos

Considere o problema de testar se um grafo G com n vértices $v_1, ..., v_n$ é hamiltoniano

Seja G uma instância verdadeira do problema L_{HAM}. Considere os dois cenários:

Decidir *vs* Verificar: Grafos Hamiltonianos

Considere o problema de testar se um grafo G com n vértices $v_1, ..., v_n$ é hamiltoniano

Seja G uma instância verdadeira do problema L_{HAM}. Considere os dois cenários:

(1) Forneço um grafo G e afirmo que o grafo é hamiltoniano.

(2) Forneço um grafo G juntamente com uma permutação $[x_1, ..., x_n]$ de $V(G)$ que é um circuito hamiltoniano

Decidir *vs* Verificar: Grafos Hamiltonianos

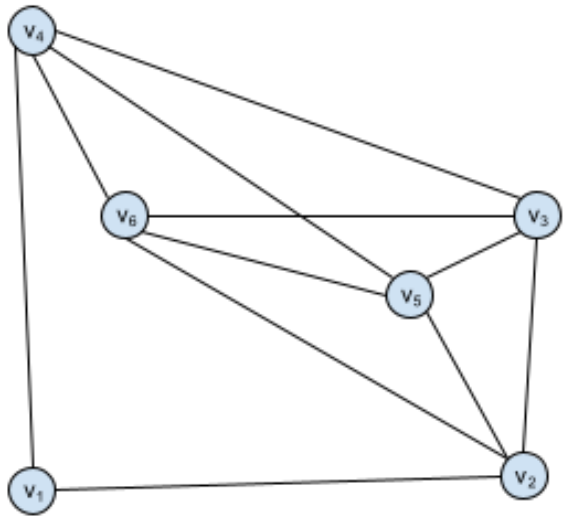
Considere o problema de testar se um grafo G com n vértices $v_1, ..., v_n$ é hamiltoniano

Seja G uma instância verdadeira do problema L_{HAM}. Considere os dois cenários:

(1) Forneço um grafo G e afirmo que o grafo é hamiltoniano.

(2) Forneço um grafo G juntamente com uma permutação $[x_1, ..., x_n]$ de $V(G)$ que é um circuito hamiltoniano

- e.g., o grafo



- junto com a permutação $[v_1, v_4, v_3, v_6, v_5, v_2]$

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado?

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado?

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado? (sim, mas exponencialmente grande!)

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado? (sim, mas exponencialmente grande!)

Outra maneira de pensar sobre problemas que podem ser verificados em tampo polinomial:

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado? (sim, mas exponencialmente grande!)

Outra maneira de pensar sobre problemas que podem ser verificados em tampo polinomial:

Existe algoritmo polinomial que

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado? (sim, mas exponencialmente grande!)

Outra maneira de pensar sobre problemas que podem ser verificados em tampo polinomial:

Existe algoritmo polinomial que

- Responde SIM para instâncias verdadeiras

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado? (sim, mas exponencialmente grande!)

Outra maneira de pensar sobre problemas que podem ser verificados em tampo polinomial:

Existe algoritmo polinomial que

- Responde SIM para instâncias verdadeiras **acompanhadas de certificado correto**

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado? (sim, mas exponencialmente grande!)

Outra maneira de pensar sobre problemas que podem ser verificados em tampo polinomial:

Existe algoritmo polinomial que

- Responde SIM para instâncias verdadeiras **acompanhadas de certificado correto**
  O certificado é polinomial no tamanho da instância
- Responde NÃO para instâncisa falsas

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado? (sim, mas exponencialmente grande!)

Outra maneira de pensar sobre problemas que podem ser verificados em tampo polinomial:

Existe algoritmo polinomial que

- Responde SIM para instâncias verdadeiras **acompanhadas de certificado correto**
  O certificado é polinomial no tamanho da instância
- Responde NÃO para instâncisa falsas (sempre, pois não existe certificado)

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado? (sim, mas exponencialmente grande!)

Outra maneira de pensar sobre problemas que podem ser verificados em tampo polinomial:

Existe algoritmo polinomial que

- Responde SIM para instâncias verdadeiras **acompanhadas de certificado correto**
  O certificado é polinomial no tamanho da instância
- Responde NÃO para instâncisa falsas (sempre, pois não existe certificado)

Atenção: Existem DOIS polinômios envolvidos:

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado? (sim, mas exponencialmente grande!)

Outra maneira de pensar sobre problemas que podem ser verificados em tampo polinomial:

Existe algoritmo polinomial que

- Responde SIM para instâncias verdadeiras **acompanhadas de certificado correto**
  O certificado é polinomial no tamanho da instância
- Responde NÃO para instâncisa falsas (sempre, pois não existe certificado)

Atenção: Existem DOIS polinômios envolvidos:
Um polinômio é a complexidade do algoritmo verificador

Decidir *vs* Verificar e certificados pequenos

Considere os problemas SUDOKO, SAT e o problema do grafo hamiltoniano:

- Instâncias verdadeiras têm *certificados*
- Instâncias falsas não têm *certificados*
- Existe algoritmo eficiente para verificar se um certificado é correto

Note a diferença em relação ao problema do xadrez $n \times n$:

- Dado um tabuleiro com cheque mate garantido para as brancas:
  - Uma jogada ótima para as brancas é um certificado? (não parece fácil de verficar!)
  - A árvore completa do jogo é um certificado? (sim, mas exponencialmente grande!)

Outra maneira de pensar sobre problemas que podem ser verificados em tampo polinomial:

Existe algoritmo polinomial que

- Responde SIM para instâncias verdadeiras **acompanhadas de certificado correto**
  O certificado é polinomial no tamanho da instância
- Responde NÃO para instâncisa falsas (sempre, pois não existe certificado)

Atenção: Existem DOIS polinômios envolvidos:
Um polinômio é a complexidade do algoritmo verificador
O outro é um polinômio que fornece o limite para o tamanho dos certificados

Cerficados e Verificadores Polinomiais

Cerficados e Verificadores Polinomiais

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma MT polinomial V_L, chamada de verificador de L,

Cerficados e Verificadores Polinomiais

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma MT polinomial V_L, chamada de verificador de L,
e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

Cerficados e Verificadores Polinomiais

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma MT polinomial V_L, chamada de verificador de L, e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V_L(x, c) = 1$.

Cerficados e Verificadores Polinomiais

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma MT polinomial V_L, chamada de verificador de L,
e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V_L(x, c) = 1$. Além disso, a string c é no máximo "polinomialmente grande" em função de $|x|$.

Cerficados e Verificadores Polinomiais

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma MT polinomial V_L, chamada de verificador de L,
e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V_L(x, c) = 1$. Além disso, a string c é no máximo "polinomialmente grande" em função de $|x|$.
Mais precisamente, $|c| = TC(|x|)$.

Cerficados e Verificadores Polinomiais

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma MT polinomial V_L, chamada de verificador de L,
e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V_L(x, c) = 1$. Além disso, a string c é no máximo "polinomialmente grande" em função de $|x|$.
Mais precisamente, $|c| = TC(|x|)$. (TC: tamanho do certificado)

Cerficados e Verificadores Polinomiais

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma MT polinomial V_L, chamada de verificador de L,
e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V_L(x, c) = 1$. Além disso, a string c é no máximo "polinomialmente grande" em função de $|x|$.
Mais precisamente, $|c| = TC(|x|)$. (TC: tamanho do certificado)

(b) Se $x \notin L$, então $\forall c \in \Sigma^*$, $V_L(x, c) = 0$

Cerficados e Verificadores Polinomiais

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma MT polinomial V_L, chamada de verificador de L, e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V_L(x, c) = 1$. Além disso, a string c é no máximo "polinomialmente grande" em função de $|x|$.
Mais precisamente, $|c| = TC(|x|)$. (TC: tamanho do certificado)

(b) Se $x \notin L$, então $\forall c \in \Sigma^*$, $V_L(x, c) = 0$
(i.e., se x é uma instância falsa de L, não importa qual string c passamos como candidata a certificado de x, o verificador V_L rejeita x).

SAT é verificável em tempo polinomial

Verificação em Tempo Polinomial

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma
MT polinomial V_L, chamada de verificador de L, e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V(x, c) = 1$ e $|c| = TC(|x|)$.

(b) Se $x \notin L$, então $\forall c \in \Sigma^*$, $V(x, c) = 0$.

Prova de que L_{SAT} é verificável em tempo polinomial:

SAT é verificável em tempo polinomial

Verificação em Tempo Polinomial

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma
MT polinomial V_L, chamada de verificador de L, e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V(x, c) = 1$ e $|c| = TC(|x|)$.

(b) Se $x \notin L$, então $\forall c \in \Sigma^*$, $V(x, c) = 0$.

Prova de que L_{SAT} é verificável em tempo polinomial:

- existe uma MT polinomial V_{SAT} (equivalente ao pseudo-código Verificador_SAT).

SAT é verificável em tempo polinomial

Verificação em Tempo Polinomial

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma
MT polinomial V_L, chamada de verificador de L, e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V(x, c) = 1$ e $|c| = TC(|x|)$.

(b) Se $x \notin L$, então $\forall c \in \Sigma^*$, $V(x, c) = 0$.

Prova de que L_{SAT} é verificável em tempo polinomial:

- existe uma MT polinomial V_{SAT} (equivalente ao pseudo-código Verificador_SAT).
- Se $\llcorner \phi \lrcorner \in L_{\text{SAT}}$, então $\exists c$ tal que $V_{\text{SAT}}(\llcorner \phi \lrcorner, c) = 1$.

SAT é verificável em tempo polinomial

Verificação em Tempo Polinomial

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma
MT polinomial V_L, chamada de verificador de L, e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V(x, c) = 1$ e $|c| = TC(|x|)$.

(b) Se $x \notin L$, então $\forall c \in \Sigma^*$, $V(x, c) = 0$.

Prova de que L_{SAT} é verificável em tempo polinomial:

- existe uma MT polinomial V_{SAT} (equivalente ao pseudo-código Verificador_SAT).
- Se $\llcorner\phi\lrcorner \in L_{\text{SAT}}$, então $\exists c$ tal que $V_{\text{SAT}}(\llcorner\phi\lrcorner, c) = 1$. Além disso $|c| = poli(|\llcorner\phi\lrcorner|)$.

SAT é verificável em tempo polinomial

Verificação em Tempo Polinomial

Seja $L \subseteq \Sigma^*$ um problema de decisão. Dizemos que L pode ser *verificado em tempo polinomial* se existe uma
MT polinomial V_L, chamada de verificador de L, e existe um polinômio $TC(n)$ tal que o seguinte é satisfeito:

(a) Se $x \in L$, $\exists c \in \Sigma^*$, chamada de *certificado de* x tal que $V(x, c) = 1$ e $|c| = TC(|x|)$.

(b) Se $x \notin L$, então $\forall c \in \Sigma^*$, $V(x, c) = 0$.

Prova de que L_{SAT} é verificável em tempo polinomial:

- existe uma MT polinomial V_{SAT} (equivalente ao pseudo-código Verificador_SAT).
- Se $\llcorner\phi\lrcorner \in L_{\text{SAT}}$, então $\exists c$ tal que $V_{\text{SAT}}(\llcorner\phi\lrcorner, c) = 1$. Além disso $|c| = poli(|\llcorner\phi\lrcorner|)$.
- Se $\llcorner\phi\lrcorner \notin L_{\text{SAT}}$, então $\forall c$, $V_{\text{SAT}}(\llcorner\phi\lrcorner, c) = 0$.

Verificação em tempo polinomial e a classe **NP**

Verificação em tempo polinomial e a classe **NP**

- Decisão em tempo polinomial implica verificação em tempo polinomial

Verificação em tempo polinomial e a classe **NP**

- Decisão em tempo polinomial implica verificação em tempo polinomial (provaremos a seguir)

Verificação em tempo polinomial e a classe **NP**

- Decisão em tempo polinomial implica verificação em tempo polinomial (provaremos a seguir)
- Verificação em tempo polinomial não nessariamente implica em decisão em tempo polinomial (?)

Verificação em tempo polinomial e a classe **NP**

- Decisão em tempo polinomial implica verificação em tempo polinomial (provaremos a seguir)
- Verificação em tempo polinomial não nessariamente implica em decisão em tempo polinomial (?)

Teorema

Seja $L \subseteq \Sigma^*$ se $L \in P$, então L pode ser verificada em tempo polinomial.

Verificação em tempo polinomial e a classe **NP**

- Decisão em tempo polinomial implica verificação em tempo polinomial (provaremos a seguir)
- Verificação em tempo polinomial não nessariamente implica em decisão em tempo polinomial (?)

Teorema

Seja $L \subseteq \Sigma^*$ se $L \in P$, então L pode ser verificada em tempo polinomial.

Ideia da Prova:

Verificação em tempo polinomial e a classe **NP**

- Decisão em tempo polinomial implica verificação em tempo polinomial (provaremos a seguir)
- Verificação em tempo polinomial não nessariamente implica em decisão em tempo polinomial (?)

Teorema

Seja $L \subseteq \Sigma^*$ se $L \in P$, então L pode ser verificada em tempo polinomial.

Ideia da Prova:

1. Se $L \in P$, então existe uma MT polinomial M que decide L.

Verificação em tempo polinomial e a classe **NP**

- Decisão em tempo polinomial implica verificação em tempo polinomial (provaremos a seguir)
- Verificação em tempo polinomial não nessariamente implica em decisão em tempo polinomial (?)

Teorema

Seja $L \subseteq \Sigma^*$ se $L \in P$, então L pode ser verificada em tempo polinomial.

Ideia da Prova:

1. Se $L \in P$, então existe uma MT polinomial M que decide L.
   isto é, $M(x) = 1 \Leftrightarrow x \in L$

Verificação em tempo polinomial e a classe **NP**

- Decisão em tempo polinomial implica verificação em tempo polinomial (provaremos a seguir)
- Verificação em tempo polinomial não nessariamente implica em decisão em tempo polinomial (?)

Teorema

Seja $L \subseteq \Sigma^*$ se $L \in P$, então L pode ser verificada em tempo polinomial.

Ideia da Prova:

1. Se $L \in P$, então existe uma MT polinomial M que decide L.
   isto é, $M(x) = 1 \Leftrightarrow x \in L$
2. Algoritmo M pode ser o verificador de L com pequena alteração:
   Algoritmo M' pega dois argumentos, i.e, $M'(x, c)$, ignora o segundo argumento e se comporta como M

Verificação em tempo polinomial e a classe **NP**

- Decisão em tempo polinomial implica verificação em tempo polinomial (provaremos a seguir)
- Verificação em tempo polinomial não nessariamente implica em decisão em tempo polinomial (?)

Teorema

Seja $L \subseteq \Sigma^*$ se $L \in P$, então L pode ser verificada em tempo polinomial.

Ideia da Prova:

1. Se $L \in P$, então existe uma MT polinomial M que decide L.
   isto é, $M(x) = 1 \Leftrightarrow x \in L$
2. Algoritmo M pode ser o verificador de L com pequena alteração:
   Algoritmo M' pega dois argumentos, i.e, $M'(x, c)$, ignora o segundo argumento e se comporta como M
3. Para toda string $x \in L$, existe certificado (por exemplo, ϵ) que faz verificador aceitar.
   O verificador M' se comporta exatamente como M, portanto $M'(x, \epsilon) = 1$.

Verificação em tempo polinomial e a classe **NP**

- Decisão em tempo polinomial implica verificação em tempo polinomial (provaremos a seguir)
- Verificação em tempo polinomial não nessariamente implica em decisão em tempo polinomial (?)

Teorema

Seja $L \subseteq \Sigma^*$ se $L \in P$, então L pode ser verificada em tempo polinomial.

Ideia da Prova:

1. Se $L \in P$, então existe uma MT polinomial M que decide L.
   isto é, $M(x) = 1 \Leftrightarrow x \in L$
2. Algoritmo M pode ser o verificador de L com pequena alteração:
   Algoritmo M' pega dois argumentos, i.e, $M'(x, c)$, ignora o segundo argumento e se comporta como M
3. Para toda string $x \in L$, existe certificado (por exemplo, ϵ) que faz verificador aceitar.
   O verificador M' se comporta exatamente como M, portanto $M'(x, \epsilon) = 1$.
3. Para toda string $x \notin L$
   O verificador se comporta exatamente como M e rejeita x.

Verificação em tempo polinomial e a classe **NP**

Verificação em tempo polinomial e a classe **NP**

Teorema

Seja $\mathcal{V}$ a classe de problemas que podem ser verificados em tempo polinomial. Então $\mathcal{V} = \text{NP}$.

Verificação em tempo polinomial e a classe **NP**

Teorema

Seja $\mathcal{V}$ a classe de problemas que podem ser verificados em tempo polinomial. Então $\mathcal{V} = \text{NP}$.

Classe **NP** (definição equivalente)

Uma linguagem L está em NP se existe um polinômio $p : \mathbb{N} \to \mathbb{N}$ e uma MT polinomial M (verificador de L) tal que para toda string x:

$$x \in L \Leftrightarrow \exists u \in \Sigma^{p(|x|)}; M(x, u) = 1$$

Verificação em tempo polinomial e a classe **NP**

Teorema

Seja $\mathcal{V}$ a classe de problemas que podem ser verificados em tempo polinomial. Então $\mathcal{V} = \text{NP}$.

Classe **NP** (definição equivalente)

Uma linguagem L está em NP se existe um polinômio $p : \mathbb{N} \to \mathbb{N}$ e uma MT polinomial M (verificador de L) tal que para toda string x:

$$x \in L \Leftrightarrow \exists u \in \Sigma^{p(|x|)}; M(x, u) = 1$$

Para $x \in L$ e $u \in \Sigma^{p(|x|)}$ satisfazendo $M(x, u) = 1$

- u é chamada de certificado de x (em relação à L e à M).

Verificação em tempo polinomial e a classe **NP**

Teorema

Seja $\mathcal{V}$ a classe de problemas que podem ser verificados em tempo polinomial. Então $\mathcal{V} = \text{NP}$.

Verificação em tempo polinomial e a classe **NP**

Teorema

Seja $\mathcal{V}$ a classe de problemas que podem ser verificados em tempo polinomial. Então $\mathcal{V} = \text{NP}$.

Para provar isso:

Verificação em tempo polinomial e a classe **NP**

Teorema

Seja $\mathcal{V}$ a classe de problemas que podem ser verificados em tempo polinomial. Então $\mathcal{V} = \text{NP}$.

Para provar isso: Mostrar que $\mathcal{V} \subseteq \text{NP}$ e $\text{NP} \subseteq \mathcal{V}$

- $\mathcal{V} \subseteq \text{NP}$: Dada $L \in \mathcal{V}$, mostrar como construir MTN N que decide L

  para isso, usa-se a existência do verificador V_L e do fato que instâncias verdadeiras tem certificado

Verificação em tempo polinomial e a classe **NP**

Teorema

Seja $\mathcal{V}$ a classe de problemas que podem ser verificados em tempo polinomial. Então $\mathcal{V} = \text{NP}$.

Para provar isso: Mostrar que $\mathcal{V} \subseteq \text{NP}$ e $\text{NP} \subseteq \mathcal{V}$

- $\mathcal{V} \subseteq \text{NP}$: Dada $L \in \mathcal{V}$, mostrar como construir MTN N que decide L

  para isso, usa-se a existência do verificador V_L e do fato que instâncias verdadeiras tem certificado

- $\text{NP} \subseteq \mathcal{V}$:

Verificação em tempo polinomial e a classe **NP**

Teorema

Seja $\mathcal{V}$ a classe de problemas que podem ser verificados em tempo polinomial. Então $\mathcal{V} = \text{NP}$.

Para provar isso: Mostrar que $\mathcal{V} \subseteq \text{NP}$ e $\text{NP} \subseteq \mathcal{V}$

- $\mathcal{V} \subseteq \text{NP}$: Dada $L \in \mathcal{V}$, mostrar como construir MTN N que decide L

  para isso, usa-se a existência do verificador V_L e do fato que instâncias verdadeiras tem certificado

- $\text{NP} \subseteq \mathcal{V}$: Dada $L \in NP$, mostrar que $\exists V_L$ e que instâncias verdadeiras tem certificado

  para isso usa-se a existência de MTN N que decide L