Melhores momentos

AULA PASSADA

NP e não-determinísmo

Teorema. Uma linguagem L está em NP se e somente se alguma máquina de Turing não-determinística que roda em tempo polinomial a decide.

NP e não-determinísmo

NTIME$(t(n)) = \{L : L$ é decidida por uma MT não-determinística em tempo $\mathrm{O}(t(n))\}$

Corolário. NP $= \cup_k$NTIME(n^k).

NP = Nondeterministic Polynomial time

Complemento linguagens

O **complemento** de uma linguagem L sobre o alfabeto Σ, denotada por $\overline{L}$ é o conjunto das cadeias em Σ^* que não estão em L:

$$\overline{L} = \Sigma^* \setminus L.$$

É conveniente considerarmos o complemento de uma linguagem L que codifica um problema como sendo o conjunto das cadeias que codificam entradas válidas para o problema e que não pertencem a L.

Exemplos

$$\overline{\text{CASAMENTO}} = \{\langle G \rangle : G \text{ é um grafo bipartido que } \textbf{não} \text{ possui um emparelhamento perfeito}\}$$

$$\overline{\text{CAMHAM}} = \{\langle G, s, t \rangle : G \text{ é um grafo orientado que } \textbf{não} \text{ possui um caminho hamiltoniano de } s \text{ a } t\}$$

$$\begin{aligned}\overline{\text{COMPOSTO}} &= \{\langle k \rangle : k \neq i \times j,\ i, j > 1\} \\ &= \{\langle p \rangle : p \text{ é um número primo}\} \\ &= \text{PRIME}\end{aligned}$$

AULA 9

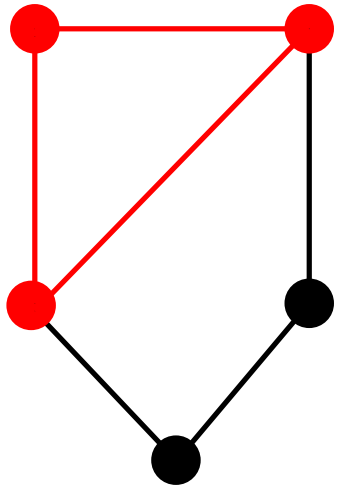
A classe NP

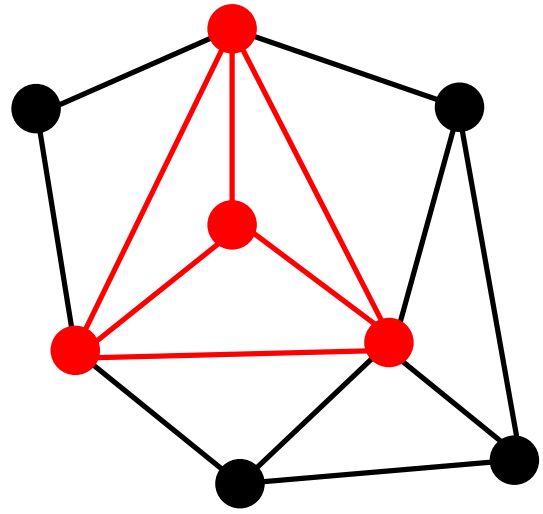
MS 7.3

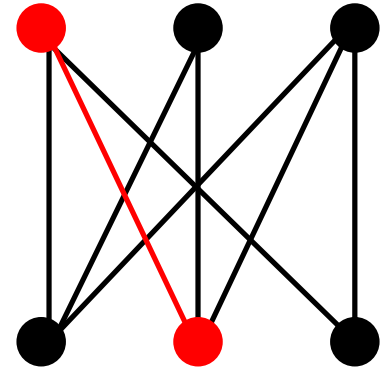
Mais exemplos de problemas em NP

$$\text{CLIQUE} = \{\langle G, k\rangle : G \text{ tem um clique de tamanho } k\}$$

Exemplos:







clique com k vértices $=$ subgrafo completo com k vértices

Mais exemplos de problemas em NP

Teorema. CLIQUE está em NP.

Prova: Um verificador polinomial recebe $\langle G, k \rangle$ e $\langle C \rangle$ onde C é um conjunto com k vértices e verifica se C é um clique. ■

Outra prova: Uma MT não-determinística chuta o conjunto C e verifica se é um clique. ■

Algoritmo não-determinístico

Recebe um grafo $G = (N, E)$ e um inteiro k e decide se G possui um clique com k vértices.

CLIQUE-ND (N, E, k)

0 se $|N| < k$ **então** rejeite $\langle G, k \rangle$

1 $C \leftarrow \emptyset \quad S \leftarrow N$
2 **para** $i = 1, 2, \ldots, k$ **faça**
3 **escolha** v em S
4 $C \leftarrow C \cup \{v\}$
5 $S \leftarrow S \setminus \{v\}$

6 **para** $u \in C$ **faça**
7 **para** $v \in C \setminus u$ **faça**
8 **se** $uv \notin E$ **então** *rejeite* $\langle G, k \rangle$

9 *aceite* $\langle G, k \rangle$

Mais exemplos de problemas em NP

$$\text{SUBSETSUM} = \{\langle X, t\rangle : X = \{x_1, \ldots, x_k\} \text{ e para algum } Y = \{y_1, \ldots, y_e\} \subseteq \{x_1, \ldots, x_k\} \text{ temos que } \sum y_i = t\ \}$$

Exemplos:

- $\langle\{4, 4, 11, 16, 21, 21, 27\}, 19\rangle$ está em SUBSETSUM, pois $4 + 4 + 11 = 19$
- $\langle\{4, 4, 11, 16, 21, 21, 27\}, 46\rangle$ está em SUBSETSUM, pois $21 + 21 + 4 = 46$
- $\langle\{4, 4, 11, 16, 21, 21, 27\}, 12\rangle$ não está em SUBSETSUM

Mais outro exemplo de problema em NP

Teorema. SUBSETSUM está NP.

Prova: Um verificador polinomial recebe $\langle X, t \rangle$ e $\langle Y \rangle$ onde Y é um multiconjunto de números. O verificador verifica se $Y \subseteq X$ e se $\sum_{y \in Y} y = t$. ■

Outra prova: Uma MT não-determinística chuta o conjunto $Y \subset X$ e verifica se $\sum_{y \in Y} y = t$. ■

Algoritmo não-determinístico

Recebe um multiconjunto X de números decide se existe $Y \subseteq X$ tal que $\sum_{y \in Y} y = t$

SUBSETSUM (X, t)

```
 0   k ← |X|   Y ← ∅
 1   escolha e ∈ {0, 1, ..., k}
 2   para i = 1, 2, ..., e faça
 3       escolha y em X
 4       Y ← Y ∪ {y}
 5       X ← X \ {y}

 6   soma ← 0
 7   para cada y ∈ Y faça soma ← soma + y

 8   se soma = t
 9       então aceite ⟨X, t⟩
10       senão rejeite ⟨X, t⟩
```

$\overline{\text{CLIQUE}}$ e $\overline{\text{SUBSETSUM}}$

$$\overline{\text{CLIQUE}} = \{\langle G, k\rangle : G \text{ **não** tem um clique de tamanho } k\}$$

$$\overline{\text{SUBSETSUM}} = \{\langle X, t\rangle : X = \{x_1, \ldots, x_k\} \text{ para **todo**}$$

$$Y = \{y_1, \ldots, y_e\} \subseteq \{x_1, \ldots, x_k\}$$

$$\text{temos que } \sum y_i \neq t \ \}$$

Não se sabe se $\overline{\text{CLIQUE}}$ ou $\overline{\text{SUBSETSUM}}$ está em NP.

A classe coNP

coNP é a classe de linguagens que são complementos das linguagens em NP.

Equivalentemente,

coNP é a classe de linguagens que tem um verificador polinomial "para resposta **não**"= "*rejeite*".

Exemplos:

- $\overline{\text{CAM}}$ está em coNP (e também em P)
- $\overline{\text{PRI-MES}}$ está em coNP (e também em P)
- $\overline{\text{COMPOSTO}}$ está em coNP (e também em P)
- $\overline{\text{CASAMENTO}}$ está em coNP (e também em P)
- $\overline{\text{CAMHAM}}$ está em coNP (não se sabe de está P)
- CAMHAM não se sabe de está coNP tão pouco em P

NP ∩ coNP

NP ∩ coNP é a classe de linguagem que tem verificador polinomial para as respostas **sim** (*aceite*) e **não** (*rejeite*).

Exemplos:

- CAM está em NP ∩ coNP
- PRI-MES está em NP ∩ coNP
- COMPOSTO está em NP ∩ coNP
- CASAMENTO está em NP ∩ coNP

NP ∩ coNP é a classe das linguagens que têm uma **boa caracterização** (Jack Edmonds)

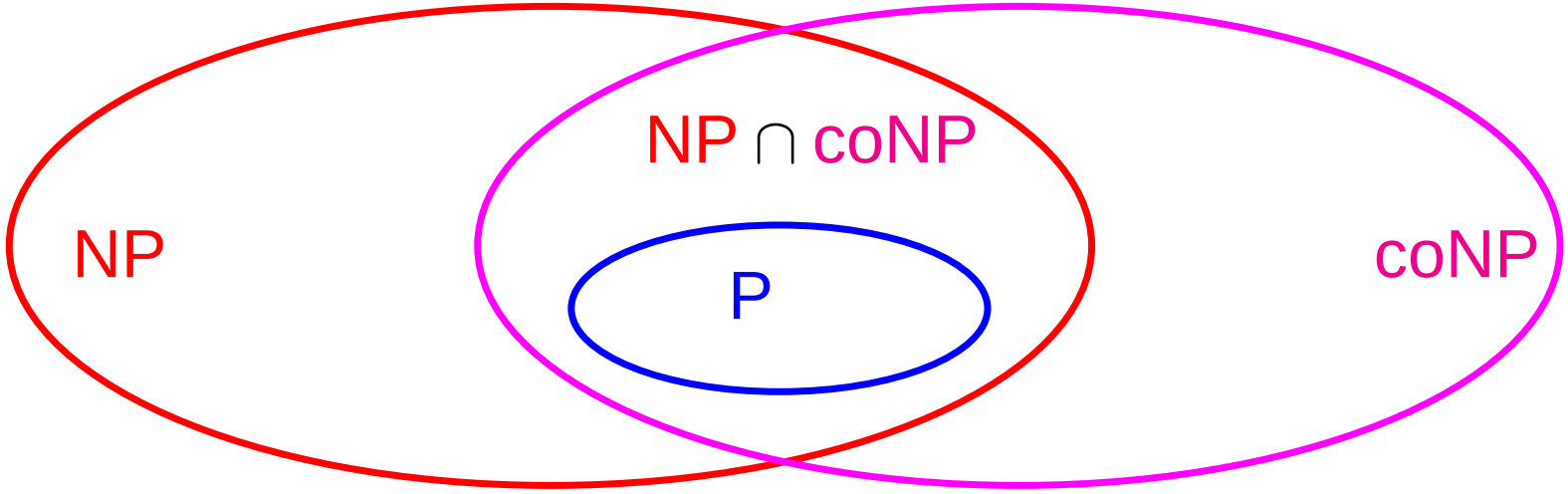
P, NP e coNP

Classe P	formada por linguagens para as quais pertinência pode ser decidida em tempo polinomial
Classe NP	formada por linguagens para as quais pertinência pode ser verificada em tempo polinomial
Classe coNP	formada por linguagens para as quais **não** pertinência pode ser verificada em tempo polinomial

P, NP e coNP

Classe P	formada por problemas de decisão que podem ser resolvidos em tempo polinomial
Classe NP	formada por problemas de decisão que possuem um verificador polinomial para a resposta SIM
Classe coNP	formada por problemas de decisão que possuem um verificador polinomial para a resposta NÃO

P, NP e coNP



$P \neq NP$?

$NP \cap coNP \neq P$?

$NP \neq coNP$?

EXPTIME

$$\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{EXPTIME} = \cup_k \mathsf{TIME}(2^{n^k})$$

Redução Polinomial

MS 7.4

Função computável em tempo polinomial

Um função f de Σ^* em Σ^* é **computável em tempo polinomial** se existe alguma MT determinística que consome tempo polinomial e que pára com exatamente $f(w)$ na sua fita, quando iniciada com qualquer cadeia w.

Redução polinomial

Permite comparar o "grau de complexidade" de problemas.

Um linguagem A é **redutível em tempo polinomial** à linguagem B, denotado por $A \leq_P B$, se existe uma função computável em tempo polinomial f de Σ^* em Σ^* tal que

$$w \in A \Leftrightarrow f(w) \in B \ .$$

A função f é chamada de **redução de tempo polinomial** de A para B.

Redução polinomial

Teorema. Se $A \leq_P B$ e B está em P, então A está em P.

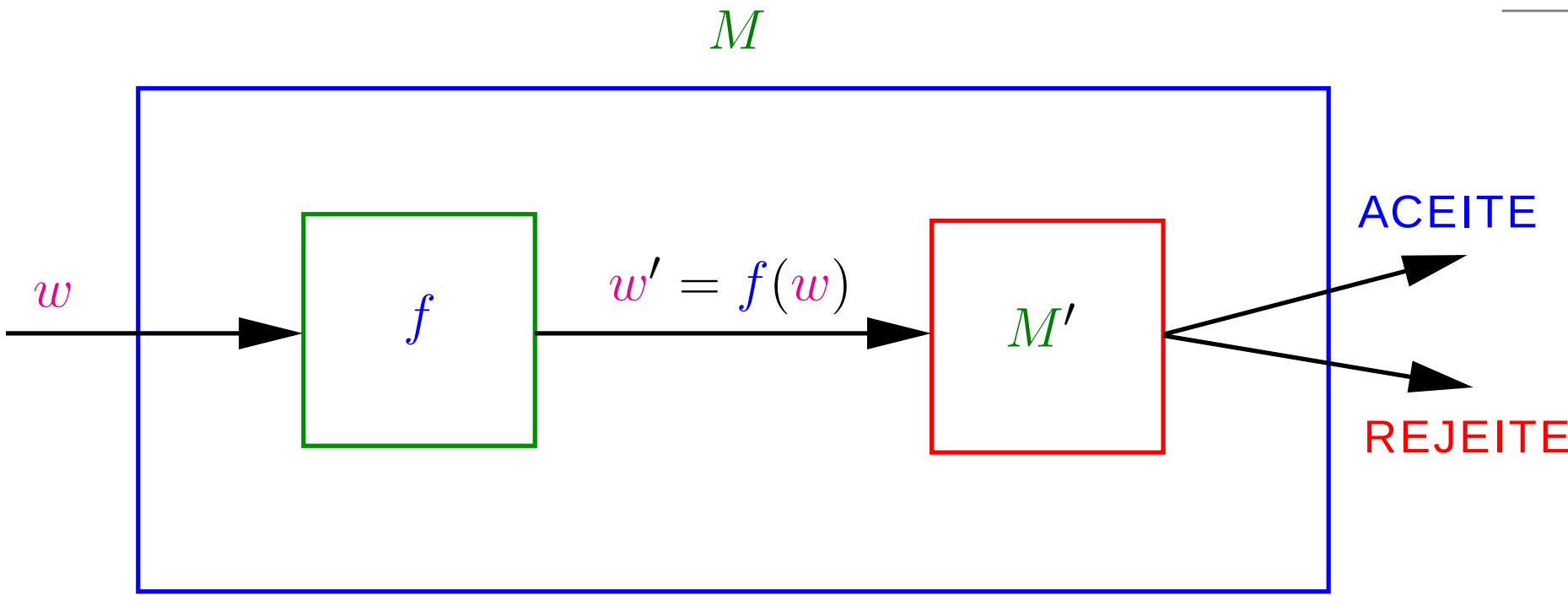
Prova: Seja M' MT que decide B em tempo polinomial e f uma redução polinomial de A em B. A MT M a seguir decide A em tempo polinomial:

$M =$ "com entrada w:

1. Calcule $w' = f(w)$
2. Simule M' com entrada w'.
   Se M' aceita w', então ACEITE w.
   Se M' rejeita w', então REJEITE w."

M consome tempo polinomial pois a composição de polinômios é um polinômio. ■

Esquema comum de reduções



Faz apenas uma chamada à MT M'.

f transforma w em $w' = f(w)$ tal que

$M(w) =$ ACEITE se e somente se $M'(w') =$ ACEITE

f é uma espécie de “filtro” ou “compilador”.

Satisfatibilidade

Problema: Dada um fórmula booleana ϕ nas variáveis $x_1, \ldots, x_n$, existe uma atribuição

$$t : \{x_1, \ldots, x_n\} \rightarrow \{\text{VERDADE}, \text{FALSO}\}$$

que torna ϕ verdadeira?

Exemplo:

$$\phi = (x_1) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3) \wedge (\overline{x}_3)$$

Se $t(x_1) = \text{VERDADE}, t(x_2) = \text{FALSO}, t(x_3) = \text{FALSO}$,
então $t(\phi) = \text{VERDADE}$

Se $t(x_1) = \text{VERDADE}, t(x_2) = \text{VERDADE}, t(x_3) = \text{FALSO}$,
então $t(\phi) = \text{FALSO}$

SAT

$$\text{SAT} = \{\langle\phi\rangle : \phi \text{ é uma fórmula booleana satisfatível}\}$$

Exemplo:

$$\phi = (x_1 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x}_3)$$

Variáveis: x_1, x_2, x_3, x_4

Literais: $x_1, \overline{x}_1, \overline{x}_2, x_3, \overline{x}_3, x_4$

Cláusula= **ou** de literais: $(\overline{x}_1 \vee \overline{x}_2 \vee x_3 \vee x_4)$

CSAT e 3SAT

Uma fórmula está na **fórmula normal conjuntiva** se ela á uma conjunção de cláusulas (**e** de cláusulas): Exemplo:

$$\phi = (x_1 \vee x_4) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x}_3)$$

$\text{CSAT} = \{\langle\phi\rangle : \phi$ é uma fórmula normal conjuntiva satisfatível$\}$

$\text{3SAT} = \{\langle\phi\rangle : \phi$ é uma fórmula normal conjuntiva satisfatível com **3 literais por cláusula**$\}$

Exemplo:

$$\phi = (x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

CSAT$\leq_P$3SAT

Descreveremos uma redução polinomial f que recebe um fórmula booleana ϕ e devolve uma fórmula booleana ϕ' com exatamente 3 literais por cláusula tais que

$$\phi \text{ é satisfatível} \Leftrightarrow \phi' \text{ é satisfatível.}$$

A transformação consiste em substituir cada cláusula de ϕ por uma coleção de cláusulas com exatamente 3 literais cada e equivalente a ϕ.

CSAT$\leq_P$3SAT

Seja $(l_1 \vee l_2 \vee \cdots \vee l_k)$ uma cláusula de ϕ.

Caso 1. $k = 1$

Troque (l_1) por

$$(l_1 \vee y_1 \vee y_2)\,(l_1 \vee \overline{y}_1 \vee y_2)\,(l_1 \vee y_1 \vee \overline{y}_2)\,(l_1 \vee \overline{y}_1 \vee \overline{y}_2)$$

onde y_1 e y_2 são variáveis novas.

Caso 2. $k = 2$

Troque $(l_1 \vee l_2)$ por $(l_1 \vee l_2 \vee y)\,(l_1 \vee l_2 \vee \overline{y})$. onde y é uma variáveis nova.

Caso 3. $k = 3$

Mantenha $(l_1 \vee l_2 \vee l_3)$.

CSAT$\leq_P$3SAT

Caso 4. $k > 3$

Troque $(l_1 \vee l_2 \vee \cdots \vee l_k)$ por

$$(l_1 \vee l_2 \vee y_1)$$

$$(\overline{y}_1 \vee l_3 \vee y_2)\,(\overline{y}_2 \vee l_4 \vee y_3)\,(\overline{y}_3 \vee l_5 \vee y_4)\,\ldots$$

$$(\overline{y}_{k-3} \vee l_{k-1} \vee l_k)$$

onde $y_1, y_2, \ldots, y_{k-3}$ são variáveis novas

Verifique que ϕ é satisfátivel $\Leftrightarrow$ nova fórmula é satisfatível.

O tamanho da nova fórmula é $\mathrm{O}(m)$, onde m é o número de literais que ocorrem em ϕ (contando-se as repetições).