Roteiro da Aula 23

1. Satisfatibilidade
   - $2SAT \in P$
   - $SAT \xrightarrow{O(n^4)} 3SAT$
2. NP-completude

Lógica Proposicional

- Variável Booleana, $x, y, z, \ldots$: assume valores 1 ou 0 (verdadeiro ou falso);

- Operações Booleanas: OU: $x \vee y$, E: $x \wedge y$, NÃO: $\overline{x}$;

- Fórmula Booleana: expressão envolvendo variáveis e operações booleanas. Exemplo:

$$\phi = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$$

Lógica Proposicional

Semântica: em que resulta uma operação?

Tabelas-verdade

- NÃO: $\overline{0} = 1$, $\overline{1} = 0$;
- OU: $0 \vee 0 = 0$, $0 \vee 1 = 1$, $1 \vee 0 = 1$, $1 \vee 1 = 1$;
- E: $0 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 0 = 0$, $1 \wedge 1 = 1$;

Lógica Proposicional

Satisfatibilidade

Uma fórmula ϕ é satisfatível se existe uma valoração às variáveis de ϕ que a faz resultar em 1 (verdadeiro).

- Exemplo: $\phi_1 = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$:

- Exemplo: $\phi_2 = \overline{(x \vee y)} \wedge z \wedge \overline{(z \wedge \overline{y})}$:

Lógica Proposicional

Satisfatibilidade

Uma fórmula ϕ é satisfatível se existe uma valoração às variáveis de ϕ que a faz resultar em 1 (verdadeiro).

- Exemplo: $\phi_1 = (\overline{x} \wedge y) \vee (x \wedge \overline{z})$:
  - Satisfatível, para $x = 0$, $y = 1$ e $z = 0$;
- Exemplo: $\phi_2 = \overline{(x \vee y)} \wedge z \wedge \overline{(z \wedge \overline{y})}$:
  - Insatisfatível...

Lógica Proposicional

- Literal: é uma variável ou sua negação.

Forma Normal Conjuntiva (CNF)

Uma fórmula ϕ é uma CNF-fórmula se ϕ é uma conjunção de disjunções de literais.

- Ou seja, é da forma: $\phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_k$;
- onde, $\phi_i = \beta_1 \vee \beta_2 \vee \cdots \vee \beta_\ell$;
- onde β_i são literais.

Ex.: $(\underbrace{x_1}_{\text{literal}} \vee \underbrace{\overline{x_2}}_{\text{literal}} \vee x_4 \vee x_5) \wedge (x_3 \vee \overline{x_4}) \wedge \underbrace{(x_3 \vee \overline{x_6} \vee \overline{x_1})}_{\text{cláusula}}$

Satisfatibilidade

k-Forma Normal Conjuntiva (kCNF)

Uma fórmula ϕ é uma kCNF-fórmula se é uma CNF-fórmula onde todas as cláusulas têm k literais.

Temos 3 problemas muito importantes:

- SAT $= \{\phi \mid \phi$ é uma fórmula satisfatível$\}$;
- 3SAT $= \{\phi \mid \phi$ é uma 3CNF-fórmula satisfatível$\}$;
- 2SAT $= \{\phi \mid \phi$ é uma 2CNF-fórmula satisfatível$\}$.

SAT, 3SAT e 2SAT

- SAT pertence à **NP**? SAT pertence à **P**?
- 3SAT pertence à **NP**? 3SAT pertence à **P**?
- 2SAT pertence à **NP**? 2SAT pertence à **P**?

Satisfatibilidade

- O algoritmo força bruta para SAT, 3SAT e 2SAT é claro:
  - Dada a fórmula ϕ, para cada valoração das variáveis de ϕ:
  - Substitua os valores em ϕ e verifique se resulta em 1 (verdadeiro).
- Quanto tempo custa esse algoritmo no pior caso?

Satisfatibilidade

- O algoritmo força bruta para SAT, 3SAT e 2SAT é claro:
  - Dada a fórmula ϕ, para cada valoração das variáveis de ϕ:
  - Substitua os valores em ϕ e verifique se resulta em 1 (verdadeiro).
- Quanto tempo custa esse algoritmo no pior caso?
- Custo n para verificar cada uma das 2^n valorações:
- $\Theta(n2^n)$.

Reduções entre SATs

$$\begin{tikzcd}
I_{\mathsf{SAT}} \arrow[r] \arrow[d, "\Theta(n2^n)"] & I_{\mathsf{3SAT}} \arrow[r] \arrow[d, "\Theta(n2^n)"] & I_{\mathsf{2SAT}} \arrow[d, "\Theta(n2^n)"] \\
S_{\mathsf{SAT}} & S_{\mathsf{3SAT}} \arrow[l] & S_{\mathsf{2SAT}} \arrow[l]
\end{tikzcd}$$

Por enquanto, ninguém está em P!

2SAT $\in$ P

- A idéia do algoritmo é montar, dada a fórmula 2CNF ϕ, um grafo direcionado que representa as implicações entre literais de ϕ;

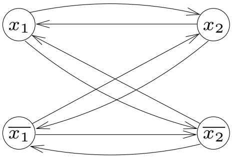
- Observe que, se $\phi = \cdots \wedge (x_1 \vee x_2) \wedge \ldots$, então para que ϕ seja satisfatível, se $x_1 = 0$, x_2 tem que ser 1, e vice-versa;

- Os vértices do grafo são os literais de ϕ. As arestas são as implicações.

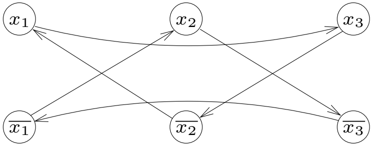
2SAT $\in$ P

Exemplos

- $\phi_1 = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$



- $\phi_2 = (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})$



2SAT $\in$ P

- Algoritmo para 2SAT:
  - Dada a fórmula ϕ, monte o grafo de implicações;
  - Para cada variável x verifique se existem os caminhos (PATH) $x \rightarrow \overline{x}$ e $\overline{x} \rightarrow x$;
  - ϕ é satisfatível sse para nenhuma variável existem os dois caminhos.
- Quanto tempo custa esse algoritmo no pior caso?

2SAT $\in$ P

- Algoritmo para 2SAT:
  - Dada a fórmula ϕ, monte o grafo de implicações;
  - Para cada variável x verifique se existem os caminhos (PATH) $x \rightarrow \overline{x}$ e $\overline{x} \rightarrow x$;
  - ϕ é satisfatível sse para nenhuma variável existem os dois caminhos.
- Quanto tempo custa esse algoritmo no pior caso?
- Custo $2n$ para montar o grafo;
- Para cada um dos $2n$ literais, $O(n^3)$ para PATH;
- $\Theta(n^4)$.

Reduções entre SATs

$$\begin{tikzcd}
I_{\text{SAT}} \arrow[r] \arrow[d, "\Theta(n2^n)"] & I_{\text{3SAT}} \arrow[r] \arrow[d, "\Theta(n2^n)"] & I_{\text{2SAT}} \arrow[d, "O(n^4)"] \\
S_{\text{SAT}} & S_{\text{3SAT}} \arrow[l] & S_{\text{2SAT}} \arrow[l]
\end{tikzcd}$$

2SAT $\in$ P

SAT $\xrightarrow{O(n^4)}$ 3SAT

Primeiro passo

fórmula $\phi \longrightarrow$ CNF-fórmula ϕ'

- Dada uma fórmula ϕ qualquer, podemos obter em tempo polinomial uma CNF-fórmula ϕ' tal que:

ϕ é satisfatível sse ϕ' também o é

- Detalhes, consultar Cormen, Ed.1, p.942

SAT $\xrightarrow{O(n^4)}$ 3SAT

Segundo passo

CNF-fórmula ϕ' fórmula $\longrightarrow$ 3CNF-fórmula ϕ''

- Dada uma CNF-fórmula:

$$\phi' = C_1 \wedge C_2 \wedge C_3 \wedge \cdots \wedge C_n$$

- Substituímos cada cláusula por um conjunto de cláusulas de 3 literais, preservando satisfatibilidade.
- Temos 3 casos:
  - $|C_i| = 1$;
  - $|C_i| = 2$;
  - $|C_i| \geq 4$.

SAT $\xrightarrow{O(n^4)}$ 3SAT

$$|C_i| = 1$$

- Dada uma cláusula, (x_1), a substituímos por:

$$(x_1 \vee y \vee z) \wedge (x_1 \vee \overline{y} \vee z) \wedge (x_1 \vee y \vee \overline{z}) \wedge (x_1 \vee \overline{y} \vee \overline{z})$$

$$|C_i| = 2$$

- Dada uma cláusula, $(x_1 \vee x_2)$, a substituímos por:

$$(x_1 \vee x_2 \vee z) \wedge (x_1 \vee x_2 \vee \overline{z})$$

SAT $\xrightarrow{O(n^4)}$ 3SAT

$$|C_i| \geq 4$$

- Dada uma cláusula, p.ex. $(x_1 \vee \overline{x_2} \vee x_4 \vee \overline{x_5} \vee x_6)$, de n literais, a substituímos por $n-2$ cláusulas de 3 literais:

$$(x_1 \vee \overline{x_2} \vee y_1) \wedge (x_4 \vee \overline{y_1} \vee y_2) \wedge (\overline{x_5} \vee x_6 \vee \overline{y_2})$$

No total, gastamos $O(n^4)$ para construir a fórmula ϕ'' e

ϕ é satisfatível sse ϕ'' também o é

Reduções entre SATs

$$
\begin{tikzcd}
I_{\mathsf{SAT}} \arrow[r, "O(n^4)"] \arrow[d, "\Theta(n2^n)"] & I_{\mathsf{3SAT}} \arrow[r] \arrow[d, "\Theta(n2^n)"] & I_{\mathsf{2SAT}} \arrow[d, "O(n^4)"] \\
S_{\mathsf{SAT}} & S_{\mathsf{3SAT}} \arrow[l, "O(1)"'] & S_{\mathsf{2SAT}} \arrow[l]
\end{tikzcd}
$$

Podemos concluir alguma coisa na relação entre SAT e 3SAT?

Reduções entre SATs

$$
\begin{tikzcd}
I_{\mathsf{SAT}} \arrow[r, "O(n^4)"] \arrow[d, "\Theta(n2^n)"] & I_{\mathsf{3SAT}} \arrow[r] \arrow[d, "\Theta(n2^n)"] & I_{\mathsf{2SAT}} \arrow[d, "O(n^4)"] \\
S_{\mathsf{SAT}} & S_{\mathsf{3SAT}} \arrow[l, "O(1)"'] & S_{\mathsf{2SAT}} \arrow[l]
\end{tikzcd}
$$

Podemos concluir alguma coisa na relação entre SAT e 3SAT?

- Se 3SAT $\in$ P, então SAT $\in$ P
- Contrapositiva: Se SAT $\notin$ P, então 3SAT $\notin$ P

3SAT é pelo menos tão difícil quanto SAT

Reduções entre SATs

$$
\begin{tikzcd}
I_{\mathsf{SAT}} \arrow[r, "O(n^4)"] \arrow[d, "\Theta(n2^n)"] & I_{\mathsf{3SAT}} \arrow[r, "O(n2^n)"] \arrow[d, "\Theta(n2^n)"] & I_{\mathsf{2SAT}} \arrow[d, "O(n^4)"] \\
S_{\mathsf{SAT}} & S_{\mathsf{3SAT}} \arrow[l, "O(1)"'] & S_{\mathsf{2SAT}} \arrow[l, "O(1)"']
\end{tikzcd}
$$

- Por final, se conhecem somente reduções triviais (super-polinomiais) entre 3SAT e SAT;
- Portanto, nenhuma cota polinomial se transfere entre SAT ou 3SAT e 2SAT:
  - O algoritmo polinomial de 2SAT não se transfere para 3SAT e SAT;
  - A relação de dificuldade de 3SAT ou SAT não se transfere para 2SAT.

NP-completude

NP-difícil

Dizemos que um problema B é NP-difícil se todos os problemas da classe NP se reduzem a ele em tempo polinomial.

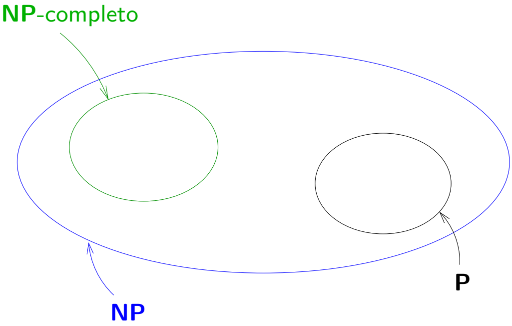
NP-completude

NP-difícil

Dizemos que um problema B é NP-difícil se todos os problemas da classe NP se reduzem a ele em tempo polinomial.

NP-completo

Dizemos que um problema B é NP-completo se $B \in \text{NP}$ e B é NP-difícil.



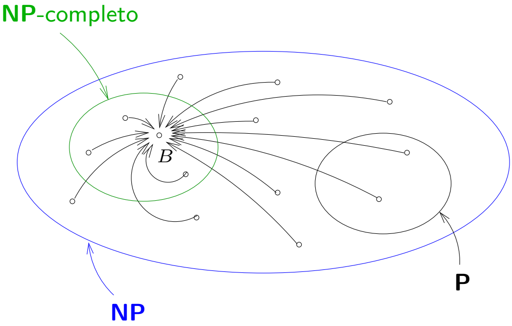
NP-completude

NP-difícil

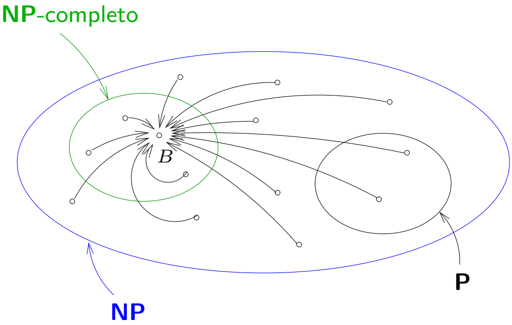
Dizemos que um problema B é NP-difícil se todos os problemas da classe NP se reduzem a ele em tempo polinomial.

NP-completo

Dizemos que um problema B é NP-completo se $B \in \mathrm{NP}$ e B é NP-difícil.



NP-completude



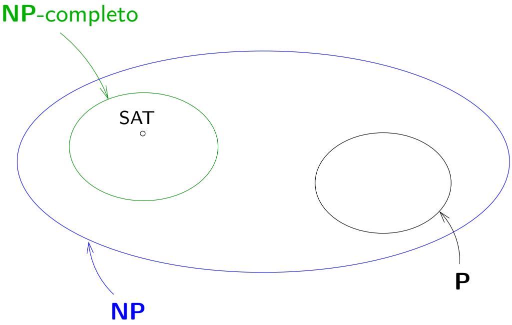
- B é pelo menos tão difícil quanto qualquer outro problema em NP;
- A teoria de NP-completude mostra justamente que os problemas NP-completos são os mais difíceis, mais complexos da classe NP;
  - Mesmo sem que saibamos se P $=$ NP ou P $\neq$ NP!

NP-completude

Teorema de Cook

SAT é NP-completo.

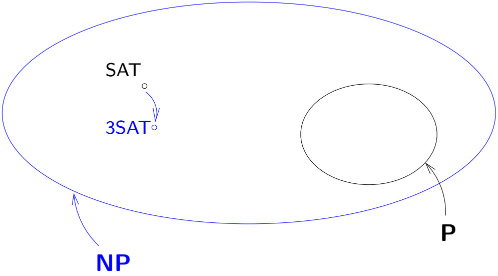
Ver detalhes no Sipser...



SAT e 3SAT

Mas, espera aí!!!

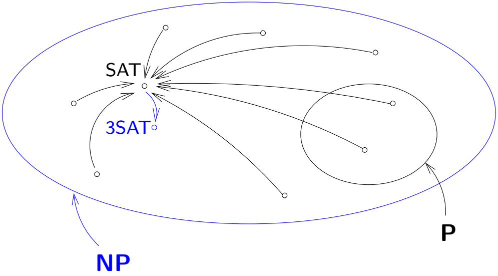
Se SAT é NP-completo, então 3SAT também é NP-completo



SAT e 3SAT

Mas, espera aí!!!

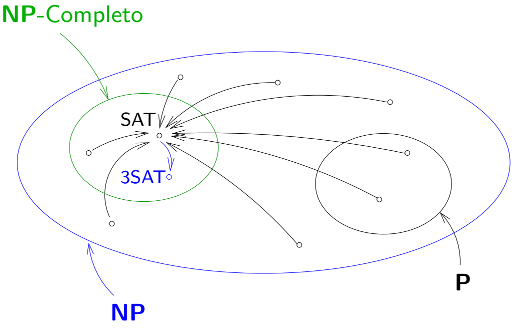
Se SAT é NP-completo, então 3SAT também é NP-completo



SAT e 3SAT

Mas, espera aí!!!

Se SAT é NP-completo, então 3SAT também é NP-completo



NP-completude

- Como mostrar que um dado problema C é NP-completo?
- Mostre que $C \in \mathsf{NP}$;
- Mostre que todos os problemas em NP se reduzem polinomialmente a C. Isso é o difícil!

NP-completude

- Como mostrar que um dado problema C é NP-completo?
- Mostre que $C \in \mathsf{NP}$;
- Mostre que todos os problemas em NP se reduzem polinomialmente a C. Isso é o difícil!

OU

- Mostre que $C \in \mathsf{NP}$;
- Mostre que algum problema P, que já sabemos ser NP-completo, se reduz polinomialmente a C.