Teoria da Complexidade

Recap: Decisor MTND $M = (\Sigma_0, \Sigma, Q, q_1, F, \Delta)$, $F = \{S, N\}$

- qualquer execução com qualquer $w \in \Sigma_0^*$ de entrada para (em s ou n)
- $w \in \Sigma_0^*$ é <u>aceita</u> se <u>alguma</u> execução para em s
- $w \in \Sigma_0^*$ é <u>rejeitada</u> se <u>todas</u> execuções param em N

↳ implicação para a simulação determinística! Só acaba em rejeição após buscar na árvore inteira.

def: Seja $f: \mathbb{N} \to \mathbb{N}$.

Dizemos que uma M.T. M (determinística ou não) é limitada (em tempo) por f se qualquer execução de M em qualquer entrada w "para após no máximo $f(|w|)$ passos", ou seja, qualquer execução

$$(q_0, w) \vdash (q_1, w_1) \vdash (q_2, w_2) \vdash \cdots \vdash (q_{l-1}, w_{l-1})$$

tem comprimento (número de configurações, l) menor ou igual a $f(|w|)$.

Def:	Tipo de máquina que decide as linguagens	Limitada (em tempo) por f satisfazendo
P	Determinística	f polinômio (em 1 variável com coef. em $\mathbb{N}$)
NP	Não determinística	f polinômio (em 1 variável com coef. em $\mathbb{N}$)
EXPTIME	Determinística	$f(n) = c^{p(n)}$, para $c \in \mathbb{N}$ constante e p polinômio como acima
NEXPTIME	Não determinística	$f(n) = c^{p(n)}$, para $c \in \mathbb{N}$ constante e p polinômio como acima

↑ classes de linguagens

Ex: $L_0 = \{ w \in \Sigma_0^* \; ; \; |w| \text{ é par} \} \in P$

↘ qualquer coisa

$L_1 = \{ w \in \{0,1\}^* \; ; \; w \text{ é um número múltiplo de } 3 \} \in P$

$L_2 = \{ w \in \{0,1\}^* \; ; \; w \text{ é um número composto} \}$ $(\in P, \text{AKS})$

$\subseteq NP$

Proposição: $P \overset{1}{\subseteq} NP \overset{2}{\subseteq} EXPTIME \overset{3}{\subseteq} NEXPTIME$

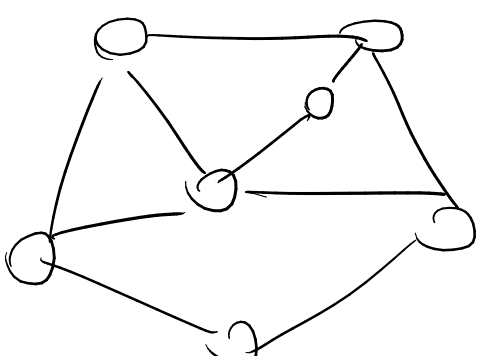
Ideia da prova: (1): ideia é $\Delta(q,x) = \{\delta(q,x)\}$

(3):

(2): Pela simulação; uma MTND M tem "bifurcação máxima" $r \in \mathbb{N}$, e se as execuções de M são limitadas por polinômio $p(n)$, a simulação é limitada por algo como $\approx r^{p(n)}$.

Não sabemos se as inclusões são estritas, mas sabemos que $P \subsetneq EXPTIME$.

Certificados Sucintos & NP



Def: Seja Σ um alfabeto e seja $\# \notin \Sigma$

Dizemos que $L' \subseteq \Sigma^* \# \Sigma^*$ é polinomialmente balanceada se existe polinômio p tal que, para todo $w, v \in \Sigma^*$

se $w \# v \in L'$ então $|v| \leq p(|w|)$

Teorema: Seja $L \subseteq \Sigma^*$, com $\# \notin \Sigma$ & $|\Sigma| \geq 2$.

Então $L \in NP$ sse existe $L' \subseteq \Sigma^* \# \Sigma^*$ polinom. balanc.

tal que $L' \in P$ &

$$L = \{w \in \Sigma^* ; \exists v \in \Sigma^* \, (w \# v \in L')\}$$

Idéia da prova: ($\Leftarrow$) Para decidir se $w \in L$, "chute" v e use o decisor determinístico de L' para verificar se $w \# v \in L'$

↑ certificado sucinto ou testemunha sucinta

($\Rightarrow$) Suponha que a MTND M decida L em tempo polinomial.

Usando o fato de que $|\Sigma| \geq 2$, podemos codificar as execuções "aceitadoras" de M usando o próprio alfabeto Σ, de forma que o tamanho da descrição é polinomialmente

limitado em função do tamanho da entrada da execução. Tais codificações são os certificados; verificável em tempo polinomial simplesmente conferindo se codificam execuções válidas de M. ∎

Reduções Polinomiais

def: Uma função $f: \Sigma_0^* \to \Gamma_0^*$ é computável em tempo polinomial se existe MT M polinomialmente limitada (em tempo) que computa f.

Def: Dizemos que $L_1 \subseteq \Sigma_0^*$ é **polinomialmente redutível** a $L_2 \subseteq \Gamma_0^*$, $L_1 \leq_p L_2$, se existe $f: \Sigma_0^* \to \Gamma_0^*$ computável em tempo polinomial, tal que $\forall w \in \Sigma_0^*$

$$w \in L_1 \iff f(w) \in L_2$$

Propriedade: Se $L_1 \leq_p L_2$ & $L_2 \in P$ então $L_1 \in P$.

Prova: "Concatena" as máquinas que computa f em tempo p & decide L_2 em tempo q resulta em máquina que decide L_1 em tempo $q \circ p$ (polinômio!) ▨

Def: Seja $\mathcal{C}$ uma classe de linguagens.

Dizemos que uma linguagem L é:

a) $\mathcal{C}$-hard ou $\mathcal{C}$-difícil se toda linguagem de $\mathcal{C}$ se reduz polinomialmente a L, ou seja,

$$\forall L' \in \mathcal{C} \left(L' \leq_p L \right)$$

b) $\mathcal{C}$-completa se L é $\mathcal{C}$-difícil e $L \in \mathcal{C}$.

Proposição: Qualquer linguagem $L \neq \emptyset, \bar{L} \neq \emptyset$ $L \in P$ é P-difícil.

Prova: Como $L \neq \emptyset$ & $\overline{L} \neq \emptyset$, sejam $w_S \in L$, $w_N \in \overline{L}$

Dada $L' \in P$, definimos f que reduz L' a L em tempo polinomial.

Dado w, fazemos $f(w) = \begin{cases} w_S, \text{ se } w \in L' \\ w_N, \text{ se } w \notin L' \end{cases}$

(usando o decisor polinomial de L') ∎

Teorema: Existem linguagens NP-completas

Teorema: Suponha que $\mathcal{C}$ seja classe de linguagens e $P \subseteq \mathcal{C}$. Então, dada L que seja $\mathcal{C}$-completa, temos $P = \mathcal{C}$ sse $L \in P$.

Prova: ($\Rightarrow$) $L \in \mathcal{C}$ (pois é $\mathcal{C}$-completa) e $P = \mathcal{C}$ por hipótese

($\Leftarrow$) Como $P \subseteq \mathcal{C}$ por hipótese, basta mostrarmos $\mathcal{C} \subseteq P$

Mas se $L \in P$, como L é $\mathcal{C}$-difícil, qualquer $L' \in \mathcal{C}$ satisfaz $L' \leq_p L$, logo $L' \in P$. ▨

Ex: Satisfabilidade Booleana, SAT

Dada uma fórmula φ da lógica proposicional, pergunta-se: φ é satisfatível?

Ou seja, existe atribuição de valores V ou F às variáveis de φ que tornem φ V?

Ex: $(p \vee q) \wedge (\neg p) \wedge (\neg q)$

SAT $\in$ NP: o certificado, (sucinto) é a atribuição

Teorema (Cook-Levin): SAT é NP-completo.
1971 1973

Karp (1972): Provou que 21 problemas são NP-completos, começando de redução polinomial de SAT para alguem deles, etc.

Garey & Johnson: Livro trazendo problemas NP-completo

Johnson: publicação de coluna mensal listando novos problemas NP-completos