

“The Hitchhiker’s Guide to Complexity”

Nilton S. Volpato Filho
nilton@ic.unicamp.br

IC/Unicamp

August 29, 2008

Máquina de Turing Determinística (DTM)

- ▶ 3 fitas:
 - ▶ entrada: apenas para leitura
 - ▶ saída: apenas para escrita
 - ▶ trabalho: leitura e escrita
- ▶ **Tempo**: número de passos até parar
- ▶ **Espaço**: número de células usadas na fita de trabalho

Máquina de Turing Não-Determinística (NDTM)

- ▶ Apenas para problemas de decisão
- ▶ Árvore de configurações possíveis
- ▶ **Resposta**: SIM se e somente se existe uma configuração de aceitação
- ▶ **Tempo**: altura da árvore
- ▶ **Espaço**: número de células usadas na fita de trabalho (em qualquer configuração)

Máquina de Turing Alternante (ATM)

- ▶ Apenas para problemas de decisão
- ▶ Árvore de configurações possíveis.
- ▶ Cada configuração é \forall ou \exists
- ▶ **Resposta**: SIM é determinada indutivamente
- ▶ **Tempo**: altura da árvore
- ▶ **Espaço**: número de células usadas na fita de trabalho (em qualquer configuração)

Máquina de Turing Oráculo (OTM)

- ▶ Possui “fita oráculo”
- ▶ Associada a problemas particulares que o oráculo resolve de graça
- ▶ Consulta a oráculo tem custo unitário
- ▶ **Tempo** e **Espaço** dependem do modelo subjacente
- ▶ **Perguntas ao oráculo**: uma função de complexidade possível.

Família de Circuitos Booleanos

- ▶ Coleção de circuitos booleanos acíclicos: $\{B_1, B_2, \dots\}$
- ▶ Aplicável somente para problemas nos quais entradas de mesmo tamanho implicam em saídas de mesmo tamanho
- ▶ B_n resolve instâncias de tamanho n
- ▶ **Uniforme**: dado n constrói-se B_n
- ▶ **Não-Uniforme**: não há uma forma automática de construir B_n dado n
- ▶ **Tempo**(paralelo): profundidade
- ▶ **Tempo**(sequencial): número de portas
- ▶ **Espaço**: largura

Outros critérios de aceitação

- ▶ Máquina de Turing Probabilística: SIM se mais da metade das conf. finais são de aceitação
- ▶ Máquina de Turing Aleatória: SIM se pelo menos metade das conf. de parada são de aceitação, NÃO se nenhuma das conf. de parada é de aceitação, caso contrário não diz nada.
- ▶ Máquina de Contagem: devolve o número de conf. de aceitação
- ▶ PRAM: número de processadores arbitrários, acesso aleatório a registradores

Limitantes de Recursos

- ▶ Constante
- ▶ Logarítmico
- ▶ Polilogarítmico
- ▶ Linear
- ▶ Polinomial
- ▶ Exponencial
- ▶ Ilimitado

TIME($f(n)$) e SPACE($f(n)$)

- ▶ TIME($f(n)$): problemas resolvidos por DTM em tempo $O(f(n))$
- ▶ SPACE($f(n)$): problemas resolvidos por DTM em espaço $O(f(n))$
- ▶ TIME(n^2) \subset TIME(n^3)
 - ▶ Contenção é próprio?
 - ▶ Existe um problema que pode ser resolvido em tempo $O(n^3)$ mas não em $O(n^2)$?
- ▶ “Dada uma MT M , M pára em no máximo $n^{2.5}$ passos?”
 - ▶ podemos resolver em $O(n^3)$ \rightarrow óbvio
 - ▶ suponha que podemos resolver em $O(n^2)$ com MT P . Criamos P' assim:
 1. Roda para sempre se M pára em $n^{2.5}$ passos com seu próprio código como entrada.
 2. Pára em $n^{2.5}$ passos se M roda por mais de $n^{2.5}$ passos com seu próprio código como entrada.
 - ▶ $P'(P')$ \rightarrow contradição!

P, PSPACE, EXPTIME

- ▶ $P: \bigcup_{k>0} \text{TIME}(n^k)$
- ▶ $\text{PSPACE}: \bigcup_{k>0} \text{SPACE}(n^k)$
- ▶ $\text{EXPTIME}: \bigcup_{k>0} \text{TIME}(2^{n^k})$
- ▶ $P \subset \text{PSPACE} ?$
- ▶ $\text{PSPACE} \subset \text{EXPTIME} ?$
 - ▶ Uma máquina com n^k bits de memória pode passar por 2^{n^k} configurações diferentes antes de parar ou entrar em loop infinito

PROGRAMAÇÃO LINEAR

Instância vetores de números inteiros $V_i = (v_i[1], \dots, v_i[n])$ para $1 \leq i \leq n$, $D = (d[1], \dots, d[n])$, $C = (c[1], \dots, c[n])$ e um inteiro B .

Resposta SIM se e somente se existe um vetor $X = (x[1], \dots, x[n])$ de números racionais, tal que $C.X \geq B$ e $V_i \leq d[i]$ para todo $1 \leq i \leq m$.

FÓRMULAS BOOLEANAS COM QUANTIFICADORES (QBF)

Instância Uma sentença

$S = (Q_1 x_1)(Q_2 x_2) \dots (Q_j x_j)[F(x_1, x_2, \dots, x_j)]$, onde $Q_i \in \{\exists, \forall\}$ e F uma expressão booleana.

Resposta SIM se S é verdadeira.

GEOGRAFIA GENERALIZADA

Instância Grafo dirigido $G = (V, A)$ e vértice v_0 .

Resposta SIM se o jogador 1 tem uma vitória garantida do seguinte jogo. Jogares alternam escolhendo novas arestas de A . Jogador 1 começa escolhendo uma aresta que sai de v_0 , e depois disso cada jogador escolhe uma aresta cuja origem é o destino da aresta escolhida anteriormente. O primeiro jogador que não puder escolher novas arestas perde.

XADREZ GENERALIZADO DAMAS GENERALIZADA

Instância Um tabuleiro $n \times n$ e posições das peças.

Resposta SIM se e somente se as brancas tem vitória garantida.

NP

- ▶ Problemas resolvidos em tempo polinomial por NDTM
- ▶ Problemas que se a resposta é SIM então existe um **certificado** de tamanho polinomial que pode ser verificado em tempo polinomial.
- ▶ $NP \subset PSPACE$?
 - ▶ idéia: verificar todas as provas possíveis de tamanho n^k bits
- ▶ $P \subset NP$? \rightarrow óbvio
- ▶ **Exercício**: mostrar se $P = NP$ ou não.

E se $P = NP$?

- ▶ O mundo seria uma utopia
- ▶ Matemáticos poderiam ser substituídos por programas provadores de teorema
- ▶ Softwares de AI seriam perfeitos
- ▶ Designers de circuitos VLSI poderiam gastar o mínimo possível de energia
- ▶ Poderia se obter a teoria mais simples para um conjunto de dados experimentais
- ▶ Aleatoriedade seria insignificante
- ▶ Qualquer esquema de criptografia teria um algoritmo de decodificação trivial
- ▶ Em geral descobrir e inventar seria tão simples quanto verificar que funciona ou está correto...

Reduções

- ▶ Uma redução (de Turing) de X para Y é uma OTM que resolve X dado um oráculo para Y .
- ▶ Uma transformação de X para Y é uma função (DTM-computável) $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$ tal que x tem resposta SIM em X sse $f(x)$ tem resposta SIM em Y .

Reduções Compatíveis

- ▶ Se C é uma classe de complexidade e R uma classe de reduções limitadas por algum recurso, dizemos que R é **compatível** com C se para quaisquer problemas X e Y , $X \leq_R Y$ e $Y \in C$ implica que $X \in C$.

Reduções compatíveis com P

- ▶ Reduções de Turing polinomiais (\leq_T)
- ▶ Transformações polinomiais (\leq_P)
- ▶ Transformações log-space ($\leq_{\log\text{-space}}$)

Completeness

- ▶ Se X é um problema, C uma classe de problemas e R uma classe de reduções, então:
- ▶ Se $Y \leq_R X$ para todo $Y \in C$, então X é **difícil** para C (sob R -reduções), ou simplesmente R -difícil para C
- ▶ Se também $X \in C$, então dizemos que X é **completo** para C (sob R -reduções), ou R -completo para C
- ▶ Ex: PROGRAMAÇÃO LINEAR é completo para P sob reduções log-space

NP-completo

- ▶ Um problema é **NP-completo** se ele é completo para NP sob **transformações polinomiais**
- ▶ Não se sabe se reduções de Turing polinomiais seriam mais fortes que transformações polinomiais nesse caso
- ▶ Para classes maiores que NP elas são diferentes
- ▶ Dentro de NP elas só poderiam ser diferentes se $P \neq NP$

SATISFABILIDADE (SAT)

Instância Uma fórmula booleana sobre u_1, \dots, u_n na forma normal conjuntiva.

Resposta SIM se existe uma atribuição às variáveis u_1, \dots, u_n que satisfaz todas as cláusulas.

PROBLEMA DO CAIXEIRO VIAJANTE (TSP)

Instância Lista c_1, c_2, \dots, c_n de cidades, uma distância inteira positiva para cada par de cidades, e um inteiro B .

Resposta SIM se existe um *tour* por todas as cidades de comprimento total B ou menos.

Há problemas intermediários entre P e NP-completo?

- ▶ Se $P = NP$ então não há
- ▶ Se $P \neq NP$ então o teorema de Ladner diz que há
- ▶ Idéia: Defina $t(n)$ uma função lentamente crescente. Considere o seguinte problema: dada uma instância do SAT de tamanho n , se $t(n)$ é ímpar resolva o problema SAT, se $t(n)$ é par então devolva NÃO.

coNP

- ▶ Complemento de NP
- ▶ Possui certificado que pode ser verificado em tempo polinomial se a resposta for NÃO.
- ▶ Ex: INSATISFABILIDADE
- ▶ $NP = coNP$? \rightarrow não sabemos
- ▶ $P = NP \implies NP = coNP$, por que?
- ▶ Por outro lado pode ser que $P \neq NP$ mas ainda $NP = coNP$.
- ▶ **Exercício**: mostrar que $NP \neq coNP$

NP \cap coNP

- ▶ Tanto uma resposta SIM como uma resposta NÃO têm um certificado que pode ser verificado em tempo polinomial.
- ▶ Especial interesse para as pessoas de computação quântica.

FATORAÇÃO

Instância Um inteiro N (com n bits).

Resposta Decomposição de N em fatores primos,
 $N = p_1^{k_1} p_2^{k_2} \dots p_m^{k_m}$ (não é necessário fornecer os certificados de primalidade).

- ▶ Não se conhece algoritmo clássico polinomial (acredita-se que não exista).
- ▶ Possui muito mais estrutura do que problemas NP-completos
- ▶ Possui algoritmo $O(2^{n^{1/3}})$ ao invés de $O(2^{n/2})$
- ▶ Algoritmo quântico de Shor: $O(n^2)$
- ▶ Essencialmente diferente dos problemas difíceis em NP
 - ▶ sempre existe uma única fatoração de um inteiro
 - ▶ SAT pode ter ou não uma atribuição satisfável
- ▶ Versão de decisão da FATORAÇÃO
- ▶ FATORAÇÃO \in NP \cap coNP. Por quê?

- ▶ Se FATORAÇÃO fosse NP-completo então $NP = coNP$. Por quê?
- ▶ Não sabemos se $NP = coNP$ (acredita-se que não seja), o que nos dá uma forte evidência que FATORAÇÃO não é NP-completo.
- ▶ Nesse caso, FATORAÇÃO seria P ou um dos problemas intermediários (do teorema de Ladner)
- ▶ Pode ser que $P = NP \cap coNP$ mas ainda $P \neq NP$
- ▶ **Exercício:** mostrar que $P \neq NP \cap coNP$

NP-difícil

- ▶ Um problema de busca X é **NP-difícil** se $Y \leq_T X$ para algum $Y \in \text{NP-completo}$
- ▶ Um problema coNP-completo é NP-difícil, por exemplo
- ▶ Não formam classe de equivalência (como NP-completo)
- ▶ Não há limitante superior na complexidade, por exemplo: o PROBLEMA DA PARADA é NP-difícil

NP-fácil

- ▶ Um problema de busca X é **NP-fácil** se $X \leq_T Y$ para algum $Y \in \text{NP}$
- ▶ Todo problema NP-fácil seria resolvido em tempo polinomial se $P = \text{NP}$
- ▶ A versão de otimização do TSP é NP-fácil e também NP-difícil. Por quê?
 - ▶ Dica: considere o problema em NP: dada um seqüência de cidades c_1, c_2, \dots, c_n e distâncias entre cidades, um limitante B e um inteiro k , existe um *tour* de comprimento B ou menos cujas primeiras k cidades são c_1 até c_k nessa ordem?
- ▶ $P^{\text{NP}} = \Delta_2^{\text{P}}$: conjunto de problemas de decisão que são NP-fáceis

NP-equivalente

- ▶ Um problema de busca é NP-equivalente se é NP-difícil e NP-fácil.
- ▶ Forma uma classe de equivalência (sob reduções polinomiais de Turing)
- ▶ Extensão natural da classe NP-completo para problemas de busca em geral
- ▶ NP-equivalente = Δ_2^P -completo (sob reduções de Turing de tempo polinomial)

Hierarquia Polinomial

- ▶ Δ_2^P recebe esse nome por pertencer à hierarquia polinomial
- ▶ Consiste das classes: Δ_k^P , Σ_k^P e Π_k^P , $k \geq 0$:
 - ▶ $\Delta_0^P = \Sigma_0^P = \Pi_0^P = P$
 - ▶ e para $i \geq 0$:
 $\Delta_{k+1}^P = P^{\Sigma_k^P}$, $\Sigma_{k+1}^P = NP^{\Sigma_k^P}$, $\Pi_{k+1}^P = \text{coNP}^{\Sigma_k^P}$.
- ▶ $PH = \bigcup_{k \geq 0} \Sigma_k^P$.

$$\Delta_0^P = P$$

$$\Sigma_0^P = P$$

$$\Pi_0^P = P$$

$$\Delta_1^P = P^P = P$$

$$\Sigma_1^P = NP^P = NP$$

$$\Pi_1^P = \text{coNP}^P = \text{coNP}$$

$$\Delta_2^P = P^{NP}$$

$$\Sigma_2^P = NP^{NP}$$

$$\Pi_2^P = \text{coNP}^{NP}$$

$$\Delta_3^P = P^{NP^{NP}}$$

$$\Sigma_3^P = NP^{NP^{NP}}$$

$$\Pi_3^P = \text{coNP}^{NP^{NP}}$$

$$\vdots$$
$$\vdots$$
$$\vdots$$

QBF_{k,Q_1}

Instância $S = (Q_1 x_1)(Q_2 x_2) \dots (Q_j x_j)[F(x_1, x_2, \dots, x_j)]$, onde $Q_i \in \{\exists, \forall\}$ e F uma expressão booleana. Com k alternâncias nos Q_i .

Resposta SIM se a sentença é satisfável.

- ▶ $QBF_{k,\exists}$ é completo para Σ_k^P sob transformações polinomiais
- ▶ $QBF_{k,\forall}$ é completo para Π_k^P sob transformações polinomiais

- ▶ Se $P = NP$ então toda a hierarquia polinomial colapsa em P . Por quê?
- ▶ Também se $P = NP$ ou $NP = coNP$ a hierarquia colapsa em NP .

PH

- ▶ Se existir um problema PH-completo, então a hierarquia colapsa em algum nível finito. Por quê?
 - ▶ Se X é PH-completo então $X \in \Sigma_i^P$ para algum $i \geq 0$, mas $Y \in \Sigma_{i+1}$ se reduz a X .
- ▶ $PH \subseteq PSPACE$.
- ▶ $PH = PSPACE$? Não sei, por quê?
 - ▶ PSPACE tem problemas completos, PH não tem a menos que a hierarquia colapse
- ▶ PH é a classe de propriedades em grafos que podem ser expressas usando lógica de segunda ordem.

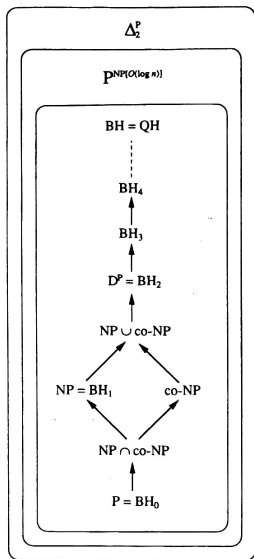


Fig. 1. The Boolean hierarchy.

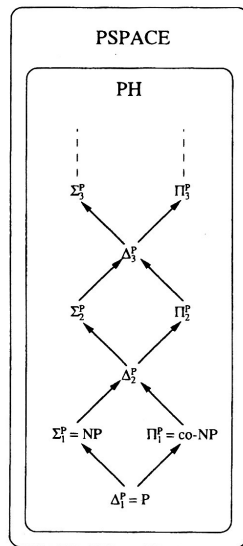


Fig. 2. The polynomial hierarchy.

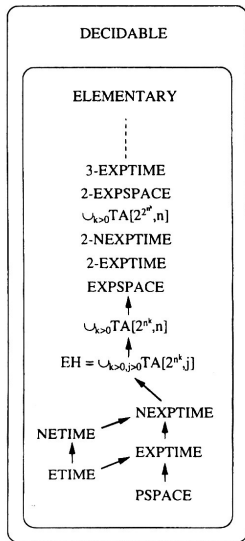


Fig. 3. Provably intractable problems.

- ▶ EXPSPACE-completo: inequivalência de expressões regulares sobre $(\cup, \cdot, ^2, *)$
- ▶ $\bigcup_{k>0} TA[2^{n^k}, n]$: TEORIA DA ADIÇÃO REAL: satisfabilidade de sentenças de primeira ordem com $<, =$, números inteiros constantes, e variáveis representando números reais.
- ▶ $\bigcup_{k>0} TA[2^{2^{n^k}}, n]$: ARITMÉTICA DE PRESBURGER: análogo à teoria da adição real mas com variáveis inteiras
- ▶ ELEMENTARY: união de $TIME(2^n)$, $TIME(2^{2^n})$, $TIME(2^{2^{2^n}})$, ...
- ▶ E além? Ex: Inequivalência de expressões regulares sobre (\cup, \cdot, \neg)

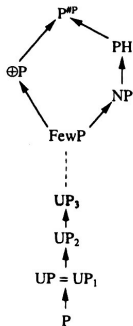


Fig. 4. Counting classes.

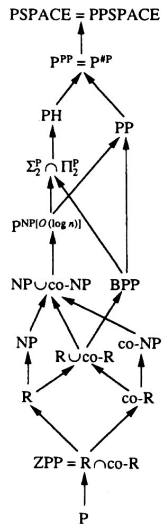


Fig. 5. Randomized complexity classes.

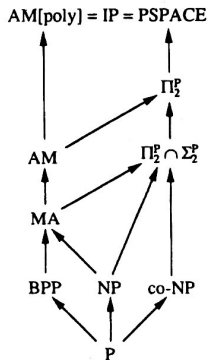


Fig. 6. Interactive complexity classes.

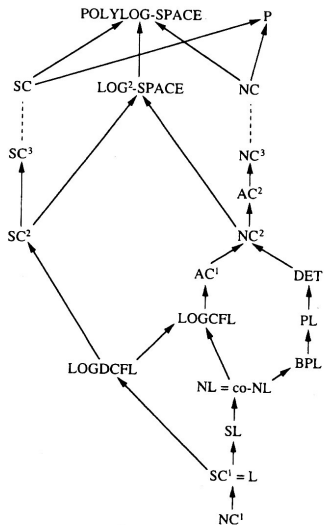
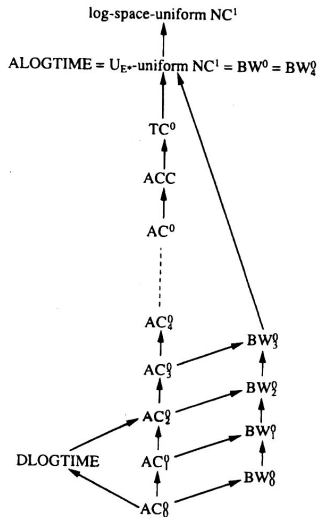


Fig. 7. Classes inside P.

Fig. 8. Classes inside NC^1 .

- ▶ “Computational Complexity” (C. Papadimitriou)
- ▶ “A Catalog of Complexity Classes” (D. S. Johnson)
- ▶ “Quantum Computing Since Democritus Lectures” (S. Aaronson)
- ▶ “Computational Complexity: A Modern Approach” (S. Arora, B. Barak)