Complexidade de Algoritmos: Limitações da Máquina, Modelo ou Ser Humano ?

Jayme Luiz Szwarcfiter

INSTITUTO DE MATEMÁTICA, COPPE e NCE

Universidade Federal do Rio de Janeiro“

Universidade Federal do Amazonas

Março 2011

./

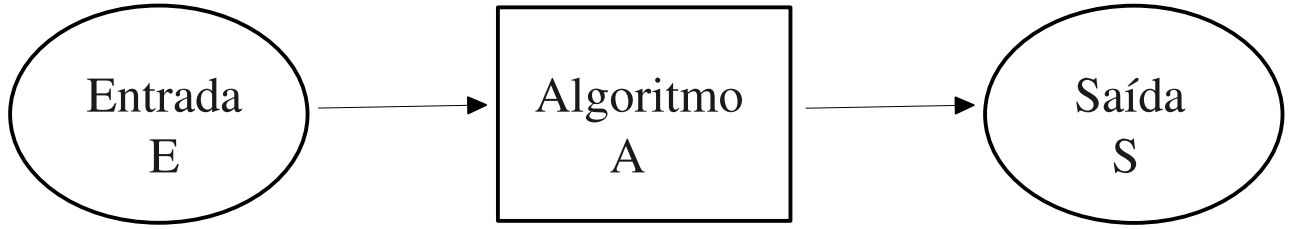
Motivação

- O que a máquina consegue realizar (em tempo hábil)
- O que o homem não consegue realizar

Conteúdo

- Um breve histórico da complexidade de algoritmos
- O conceito de tratabilidade
- Problemas tratáveis
- Problemas possivelmente intratáveis
- Problemas comprovadamente intratáveis
- O conceito de indecidibilidade

Algoritmos



A computa a entrada E

O resultado da computação é S

Um Primeiro Algoritmo

Homem das cavernas

Fonte: Livro didático público.

./

Tempo e Espaço

- Um algoritmo efetua computações.
  Qual o tempo total requerido ?
- Um algoritmo ocupa espaço
  (entrada, saída, formulação, cálculos, etc.)
  Qual o espaço total requirido ?

Questões básicas $\Longrightarrow$ complexidade

Avaliação de Tempo e Espaço

Ao longo do tempo, até mais recentemente:

1. Avaliação de tempo:
   Moderadamente importante
   (ex. velocidade de convergência de séries)
2. Avaliação de espaço:
   Praticamente, sem importância

Século XX

- Conceito teórico de computabilidade
- Surgimento do computador
- A avaliação de tempo e espaço tornam-se fundamentais
- Nos primórdios: avaliação empírica
- Avaliação empírica: depende da entrada, computador, implementação, etc.

Avaliação Analítica

A, algoritmo
$E_1, \ldots, E_t$, entradas de A
t_i, número de passos efetuados por A, sob a entrada E_i

complexidade de pior caso = max$\{t_i\}$
complexidade de melhor caso = min$\{t_i\}$
complexidade de caso médio = $\sum p_i t_i$,

onde p_i é a probabilidade de ocorrência de E_i

Aproximações

- Desconsiderar constantes aditivas ou multiplicativas
- Variável independente para medida da complexidade: tamanho da entrada
- Complexidade (pior caso) de um algoritmos A: Ordem de grandeza do número de passos efetuados por A, para computar a entrada mais desfavorável, de tamanho n

Notação O

Para expressar complexidades de pior caso:

Notação O

$2n^3 + n^2 \log n + 5 = O(n^3)$

$2n \log n + 10^{125} n + 2 = O(n \log n)$

$2^{436} = O(1)$

Soma e Multiplicação de Matrizes

Entrada:

Matrizes A, B, $n \times n$, inteiro k, $0 \leq k \leq 2$

Computação:

$k = 0 \Longrightarrow$ nada fazer

$k = 1 \Longrightarrow$ calcular $A + B$

$k = 2 \Longrightarrow$ calcular $A.B$

Complexidade:

Melhor caso: $O(n^2)$

Pior caso: $O(n^3)$

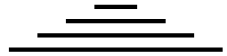
Exemplo: Torre de Hanói

Dados:
3 pilhas A, B, C de discos, num total de n discos

Computação:
No início, todos os discos estão na pilha A, em ordem decrescente de tamanho, de baixo para cima. Transferir todos os discos de A para B, um a um, com a restrição de que nenhum disco seja colocado sobre outro, de tamanho menor. A pilha C é usada como armazenamento temporário de discos.

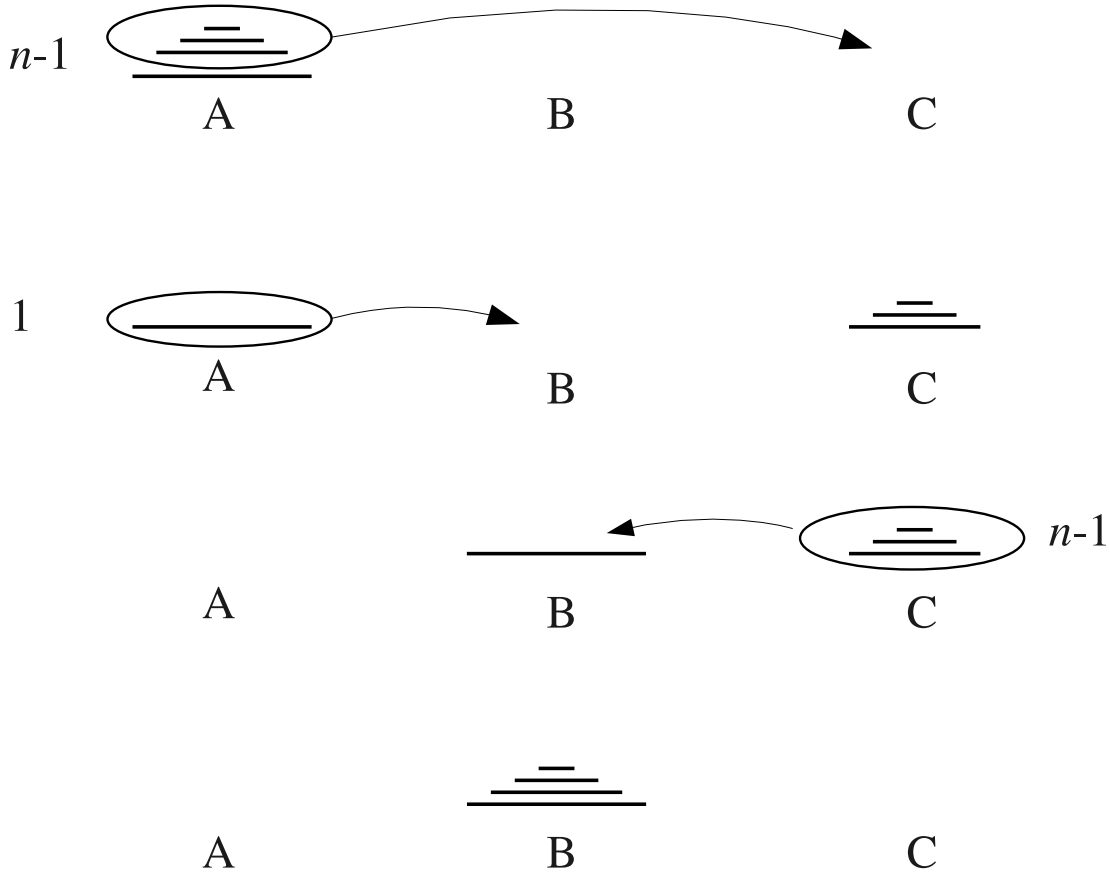
Torre de Hanói

A B C Início

A  B C Fim

Não

Torre de Hanói - Algoritmo



Torre de Hanói - Formulação

proc $HANOI(A, B, C, n)$
$HANOI(A, C, B, n-1)$
mover disco de A para B
$HANOI(C, B, A, n-1)$

Número de movimentos de discos: $2^n - 1$

Complexidade: $O(2^n)$

Generalização

Torre de Hanói com $p \geq 3$ pilhas

Dados:
p pilhas $A_1, \ldots, A_p$ de discos, num total de n discos

Computação: No início, todos os discos estão na pilha A_1, em ordem decrescente de tamanho, de baixo para cima. Transferir todos os discos de A_1 para A_2, um a um, com a restrição de que nenhum disco seja colocado sobre outro, de tamanho menor. As pilhas $A_3, \ldots, A_p$ são usadas como armazenamento temporário de discos.

Algoritmo

- Transferir k discos de A_1 para A_3, usando p pilhas
- Transferir os $n - k$ discos que restaram em A_1 para A_2, usando $p - 1$ pilhas
- Transferir os k discos de A_1 para A_2, usando p pilhas

Perguntas

- Quantos movimentos ?
  $T(n, p) =$ # movimentos para transferir n discos usando p pilhas

  Total: $2T(k, p) + T(n - k, p - 1)$
- Provar minimalidade
  Problema em aberto !
- Determinar o valor de k minimizante
  Possivel

Polynomial $\times$ Exponencial

Algoritmo	Complexidade	1 seg	1 min	1 hor
A_1	n	1000	60000	3600000
A_2	$n \log n$	140	4893	200000
A_3	n^2	31	244	1897
A_4	n^3	10	39	153
A_5	2^n	9	15	21

Supor:
Complexidade $=$ # u. t. ; 1 u. t. $=$ 1 ms

Ref: Aho, Hopcroft and Ullman, The Design and Analysis of Computer Algorithms, 1974

Tratabilidade

Algoritmo *tratável*: complexidade polinomial
Algoritmo *intratável*: caso contrário

Problema *tratável*: admite algoritmo tratável
Problema *intratável*: caso contrário

Multiplicação de matrizes: tratável

Torre de Hanói: intratável

As Classes $\mathcal{P}$ e $\mathcal{NP}$

Problemas de decisão: Resposta SIM ou Não

$\mathcal{P}$: Tempo polinomial para obter solução

$\mathcal{NP}$: Tempo polinomial para certificar resposta SIM

$\mathcal{P} = \mathcal{NP}$?

$\mathcal{P} \subseteq \mathcal{NP}$.

Mas $\mathcal{P} = \mathcal{NP}$?

Isto é, *certificar* seria tão difícil quanto resolver ?

Conjectura: $\mathcal{P} \neq \mathcal{NP}$

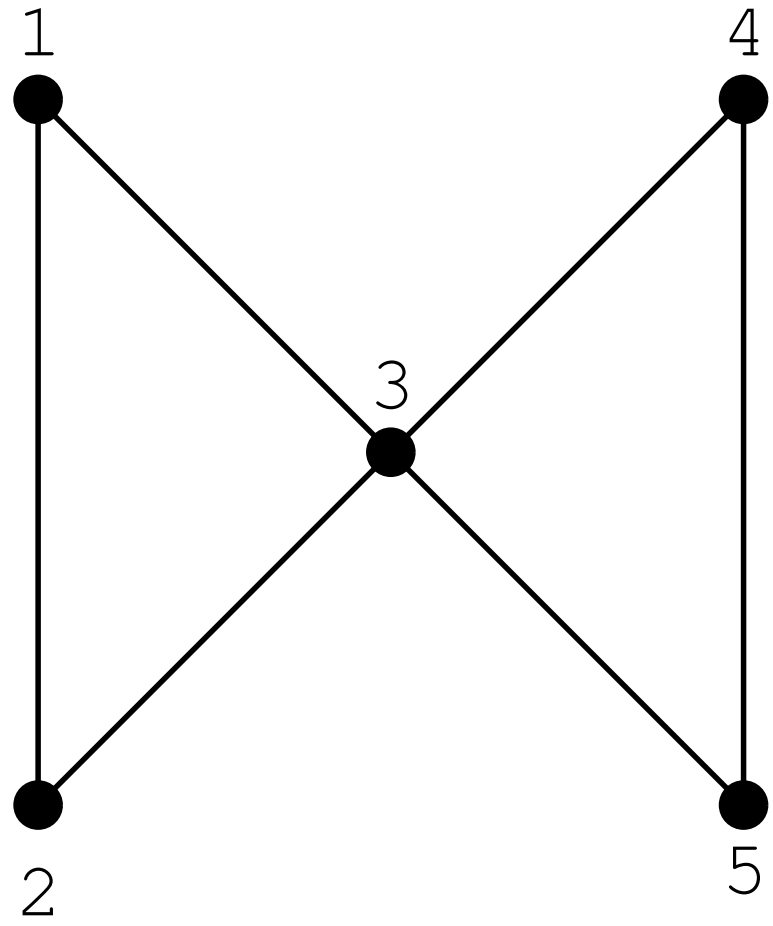
Porque $\mathcal{N}P$?

Existe um grande número de problemas que se situam nesta classe:

- Teoria de grafos
- Teoria de números
- Lógica
- Otimização combinatória
- etc

Grafos

Um grafo $G(V, E)$ é um conjunto V de elementos, denominados *vértices* e um conjunto E de pares não ordenados de vértices, denominados *arestas*



./

Cliques

$(v_i, v_j) \in E$: v_i, v_j adjacentes
clique: subconjunto de vértices, 2 a 2 adjacentes

PROBLEMA: CLIQUE
DADOS: Grafo G, inteiro k
QUESTÃO: G possui clique de tamanho $\geq k$?

CERTIFICADO SIM: vertices da clique

CERTIFICADO NÃO: ?

Expressões Booleanas

Variável boolena: x, V ou F
$x, \overline{x}$: literais
x é V $\Longleftrightarrow$ $\overline{x}$ é F
conjunção: $\wedge$
disjunção: $\vee$

Expressão booleana:
Ex.: $B = (x \wedge \overline{y}) \vee y$
x é F, y é V $\Longrightarrow$ B é V
x é F, y é F $\Longrightarrow$ B é F

B satisfatível: $\exists$ atribuições V,F às variáveis que tornam B igual a V

Expressões Booleanas

Forma Normal Conjuntiva:
Conjunção de disjunções

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

SATISFATIBILIDADE

PROBLEMA: SATISFATIBILIDADE
DADOS: Expressáo booleana B na forma normal conjuntiva
QUESTÃO: B é satisfatível ?

$B = (x \vee y \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{y} \vee z)$
RESPOSTA: SIM

CERTIFICADO: x é V, y é F, z é F

$\Longrightarrow$ SATISFATIBILIDADE $\in \mathcal{N}P$

Redução Polinomial

π_1, π_2, problemas $\in \mathcal{N}P$
π_1 *se reduz* π_2:

Dado π_1, construir π_2, t.q. solução π_2 implica solução π_1

REDUÇÃO POLINOMIAL:

construção polinomial de π_2, e solução polinomial de π_1, a partir da solução de π_2

NP-COMPLETO

π é *NP-completo*:

1. $\pi \in \mathcal{NP}$
2. $\pi' \in \mathcal{NP} \Longrightarrow \pi'$ se reduz polinomialmente a π

Se 2 verdadeiro $\Longrightarrow \pi$ é *NP-difícil*

NP-COMPLETO

- Os problemas mais difíceis de $\mathcal{NP}$
- Se um problema NP-completo $\in \mathcal{NP} \Longrightarrow$ todos pertencem
- todos problemas NP-completos são "equivalentes"
- Grande quantidade de problemas de diversas áreas
- Questão $\mathcal{P} = \mathcal{NP}$

Prova de NP-completude

Para provar que π é NP-completo:

- Náo é necessário mostrar que todo problema $\pi' \in \mathcal{N}P$ se reduz polinomialmente a π
- Basta mostrar que **um** problema NP-completo se reduz polinomialmente a π

Primeiro problema NP-completo

Teorema (Cook 1971):
SATISFATIBILIDADE é NP-completo.

Na mesma ocasião,

Teorema (Cook 1971):
CLIQUE é NP-completo.

Ideía da prova

CLIQUE $\in \mathcal{N}P$, claro.

Redução polinomial de SATISFATIBILIDADE a CLIQUE:

B, expressão booleana dada na forma normal conjuntiva.
$C_1, \ldots, C_n$, cláusula de B,
onde C_i é uma disjunção de literais ℓ_{ip}

Ideía da prova

Construir grafo G:

Cada $\ell_{ip} \Longrightarrow$ vértice v_{ip}
v_{ip}, v_{jq} adjacentes $\Longleftrightarrow i \neq j$ e $\ell_{ip} \neq \overline{\ell_{jq}}$

Definir $k = n$

Então: n variáveis podem assumir o valor V simultâneamente, em cláusulas distintas de B

$\Longleftrightarrow$

existirem n vértices mutuamente adjacentes em G.

B satisfatível $\Longleftrightarrow$ G contém clique tamanho n.

Problemas NP-Completos: Exemplos

- Coloração de vértices
- Ciclo hamiltoniano
- Caixeiro Viajante
- Mochila
- Escalonamento de tarefas (2 processadores)
- Isomorfismo de subgrafos
- Coloração de arestas

São NP-completos ?

- Escalonamento de tarefas de tempos unitários, com restrições de precedência (3 processadores)
- Isomorfismo de grafos

Além da NP-completude

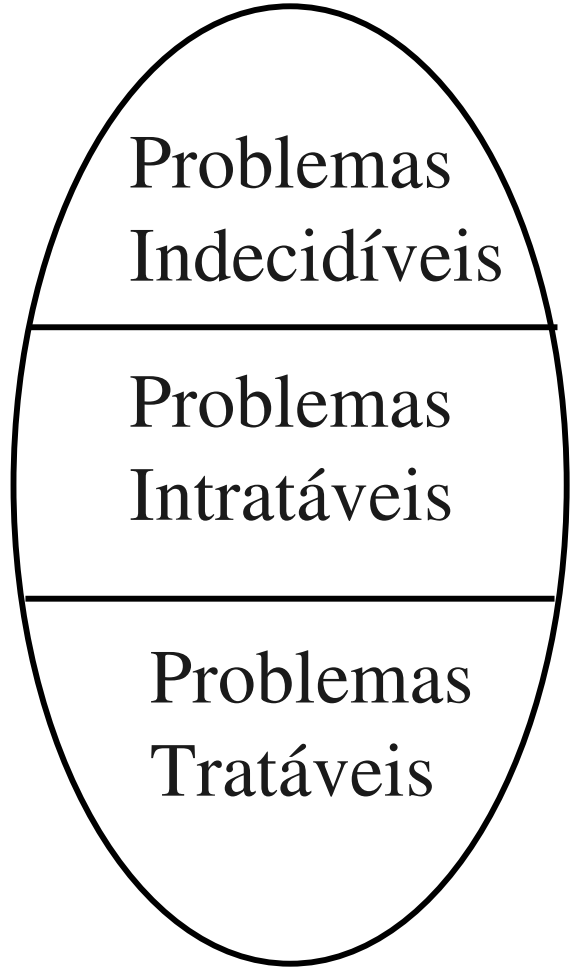
- Problemas comprovadamente exponenciais
- Problemas comprovadamente duplamente, triplamente (ou mais) exponenciais

$$2^n, 2^{2^n}, 2^{2^{2^n}}, 2^{2^{\cdot^{\cdot^{\cdot^{2^n}}}}}$$

Exemplo: Aritmética de Presburguer: 2^{2^n}

E mais além ?

Problema **indecidível**: Não admite algoritmo

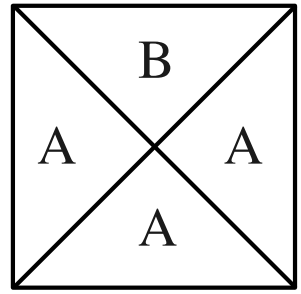


./

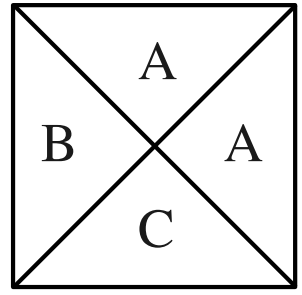
Ladrilhamento

Ladrilho: quadrado 1×1, bordas coloridas

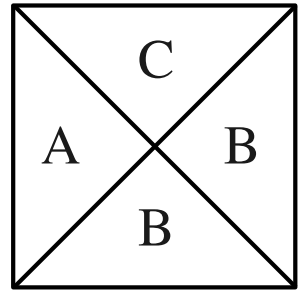
Tipo 1



Tipo 2



Tipo 3



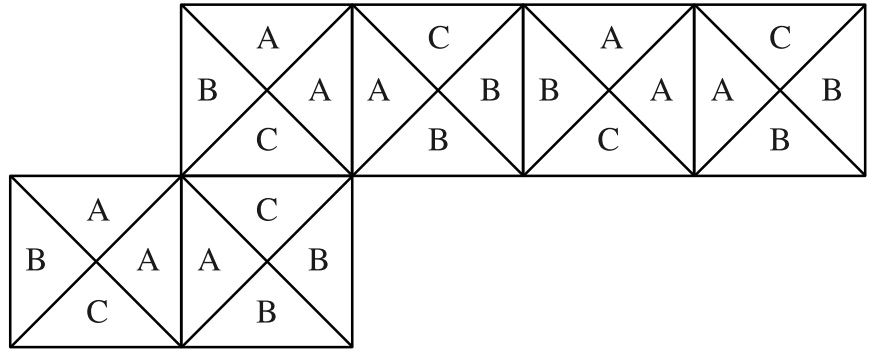
Ladrilhamento

PROBLEMA: LADRILHAMENTO
DADOS: Área X do plano, ladrilhos tipos 1, 2, 3

QUESTÃO: É possivel ladrilhar X com os ladrilhos dados, de modo que as bordas de ladrilhos vizinhos tenham sempre a mesma cor ?

Ladrilhos em linha

PROBLEMA: LADRILHOS EM LINHA
DADOS: Quadrados Q_1, Q_2
QUESTÃO: Existe uma linha de ladrilhos entre Q_1 e Q_2 ?



LADRILHOS EM LINHA é indecidível

Correspondência de palavras

PROBLEMA: CORRESPONDÊNCIA DE PALAVRAS
DADOS: Sequências $X = X_1, \ldots, X_n$ e $Y = Y_1, \ldots, Y_n$ de palavras
QUESTÃO: $\exists$ sequência finita índices, t.q. as concatenações em X e Y, segundo esses índices conduzem a palavras idênticas ?

Correspondência de palavras

	1	2	3	4	5
X	abc	bc	c	bc	a
Y	ac	a	c	b	ca

RESPOSTA: SIM

Sequência 4, 3, 5

Término de algoritmos

algoritmo
  $k :=$ inteiro > 0
  enquanto $k \neq 1$ **faça**
    $k := k - 2$

O algoritmo acima termina ?
RESPOSTA: NÃO necessariamente

algoritmo
  $k :=$ inteiro > 0
  enquanto $k \neq 1$ **faça**
    se k par **então** $k := k/2$ **senão** $k := 3k + 1$

O algoritmo acima termina ?
RESPOSTA: ???

O problema da Parada

PROBLEMA: PARADA
DADOS: Algoritmo A e um conjunto de dados D para A
QUESTÃO: A termina com a entrada D ?

O problema PARADA é indecidível !

FIM

OBRIGADO PELA ATENÇÃO