Complexidade de Tempo

Esdras Lins Bispo Jr.
bispojr@ufg.br

Teoria da Computação
Bacharelado em Ciência da Computação

02 de junho de 2014

UFG
Campus Jataí

Plano de Aula

1. Pensamento
2. Avisos
3. Revisão
   - Variantes de MT (Cont.)
4. Definição de algoritmo
   - Terminologia para descrever MTs
5. Complexidade de Tempo

Sumário

Pensamento

Pensamento

Frase

Para todo problema complexo existe sempre uma solução simples, elegante e completamente errada.

Quem?

Henry Mencken (1880-1956)
Jornalista e crítico social americano.

Sumário

Avisos

Teste 04

Dia **11 de junho** (Quarta-feira)!!!

UFG
Campus Jataí

Notícias do Santa Cruz

UFG
Campus Jataí

Sumário

UFG
Campus Jataí

Equivalência com outros modelos

- Característica essencial de máquinas de Turing: acesso irrestrito à memória;
- Todos os modelos com essa característica vêm a ser equivalente em poder, desde que satisfaçam requisitos razoáveis;
- Exemplo: qualquer algoritmo escrito em LISP pode ser escrito em Pascal (e vice-versa).

Corolário importante

Embora possamos imaginar muitos modelos computacionais diferentes, a classe de algoritmos que eles descrevem permanece a mesma.

Sumário

UFG
Campus Jataí

Definição de algoritmo

Contribuição

Apresentou uma noção do que seria um algoritmo no Congresso Internacional de Matemáticos em Paris, no ano de 1900.

Quem?

David Hilbert (1862-1943)
Matemático alemão.

Polinômio

Definições

Um **polinômio** é uma soma de termos. Um **termo** é um produto de variáveis e uma constante chamada de **coeficiente**.

Exemplo: Termo

$6 \cdot x \cdot x \cdot y \cdot z \cdot z \cdot z = 6x^2yz^3$

Exemplo: Polinômio

$6x^2yz^3 + 3xy^2 - 10$

Polinômio

Definições

Uma **raiz** de um polinômio é uma atribuição de valores às suas variáveis de modo que o valor do mesmo seja 0. Chamamos de **raiz inteira** aquela em todos os valores atribuídos são valores inteiros.

Exemplo: Raiz

O polinômio $6x^3yz^2 + 3xy^2 - x^3 - 10$ tem uma raiz em $x = 5, y = 3$ e $z = 0$.

Exemplo: Raiz Inteira

A raiz do exemplo acima é uma raiz inteira.

Polinômio

Problema apresentado por Hilbert

É possível conceber um algoritmo que teste se um polinômio tem uma raiz inteira ou não?

Expressão utilizado por Hilbert

"Um processo com o qual ela possa ser determinada por um número finito de operações".

Curioso

Não existe algoritmo que execute esta tarefa.

Definição de algoritmo

Contribuição

Mostrou, em 1970, que não existe algoritmo para se testar se um polinômio tem raízes inteiras.

Quem?

Yuri Matijasevich (1947-)
Cientista da computação e matemático russo.

Definição de Algoritmo

Noção intuitiva de algoritmos	é igual a	*algoritmos de máquina de Turing*

FIGURA 3.22
A Tese de Church–Turing

Conclusão

Existem problemas que são algoritmicamente insolúveis.

Definição de Algoritmo

Contexto

$D = \{p \mid p$ é um polinômio com uma raiz inteira$\}$

Problema

O conjunto D é decidível?

Resposta

Não é decidível. Mas é Turing-reconhecível.

Definição de Algoritmo

Problema análogo

$D_1 = \{p \mid p$ é um polinômio sobre x com uma raiz inteira$\}$

MT M_1 que reconhece D_1

M_1 = "A entrada é um polinômio p sobre a variável x.

1. Calcule o valor de p com x substituída sucessivamente pelos valores $0, 1, -1, 2, -2, 3, -3, \ldots$
   Se em algum ponto o valor do polinômio resulta em 0, *aceite*.

Considerações

M_1 reconhece D_1, mas não a decide.

Definição de Algoritmo

Resultado obtido por Matijasevich

É possível construir um decisor para D_1. Mas não para D.

Justificativa

É possível obter um limitante para polinômios de uma única variável. Porém, Matijasevich provou ser impossível calcular tais limitantes para polinômios multivariáveis.

Limitante para polinômios de uma única variável

$$\pm k \frac{c_{max}}{c_1}$$

em que

- k é o número de termos do polinômio,
- c_{max} é o coeficiente com maior valor absoluto, e
- c_1 é o coeficiente do termo de mais alta ordem.

Terminologia para descrever MTs

Níveis de descrição

- **Descrição formal**: esmiúça todos os elementos da 7-upla, conforme definição;
- **Descrição de implementação**: descreve a forma pela qual a MT move a sua cabeça e a forma como ela armazena os dados na fita;
- **Descrição de alto nível**: neste nível não precisamos mencionar como a máquina administra a sua fita ou sua cabeça de leitura-escrita.

Exemplo

Seja A a linguagem consistindo em todas as cadeias representando grafos não-direcionados que são conexos. Logo:

$$A = \{\langle G \rangle | G \text{ é um grafo não-direcionado conexo}\}$$

Descrição de alto nível

M = "Sobre a entrada $\langle G \rangle$, a codificacão de um grafo G:

1. Selecione o primeiro nó de G e marque-o.
2. Repita o seguinte estágio até que nenhum novo nó seja marcado:
    1. Para cada nó em G, marque-o se ele está ligado por uma aresta a um nó que já está marcado.
3. Faça uma varredura em todos os nós de G para determinar se eles estão todos marcados. Se eles estão, *aceite*; caso contrário, *rejeite*".

UFG
Campus Jataí

Exemplo

Pergunta

Como seria a descrição de M no nível de implementação?

Sumário

Complexidade

Por que estudar complexidade?

Um problema pode ser até decidível, mas pode levar uma quantidade de tempo ou memória bastante elevada.

UFG Campus Jataí

Complexidade

Por que estudar complexidade?

Um problema pode ser até decidível, mas pode levar uma quantidade de tempo ou memória bastante elevada.

Questões do estudo de complexidade

- Quanto tempo[espaço] leva[ocupa] um determinado algoritmo?
- O que faz um algoritmo gastar[ocupar] mais tempo[espaço] do que um outro?
- É possível classificar os algoritmos em termos de complexidade?

Complexidade de Tempo

Problema

Seja a linguagem $A = \{0^k 1^k \mid k \geq 0\}$. Quanto tempo uma máquina de Turing simples precisa para decidir A?

Complexidade de Tempo

Problema

Seja a linguagem $A = \{0^k1^k \mid k \geq 0\}$. Quanto tempo uma máquina de Turing simples precisa para decidir A?

Descrição de uma possível MT simples

M_1 = "Sobre a cadeia de entrada ω:

1. Faça uma varredura na fita e *rejeite* se um 0 for encontrado à direita de um 1.
2. Repita se ambos 0s e 1s permanecem sobre a fita:
    1. Faça uma varredura na fita, cortando um único 0 e um único 1.
3. Se 0s ainda permanecerem após todos os 1s tiverem sido cortados, ou se 1s ainda permanecerem após todos os 0s tiverem sido cortados, *rejeite*. Caso contrário, se nem 0s nem 1s permanecerem sobre a fita, *aceite*.

UFG
Campus Jataí

Complexidade de Tempo

Analisando a entrada

- Grafo: número de nós, número de arestas;
- Estrutura de dados: tamanho do vetor, altura da árvore;
- Cadeia: tamanho da cadeia de entrada.

Complexidade de Tempo

Analisando a entrada

- Grafo: número de nós, número de arestas;
- Estrutura de dados: tamanho do vetor, altura da árvore;
- Cadeia: tamanho da cadeia de entrada.

Tipos de Análise

- Análise do pior caso;
- Análise do caso médio;
- Análise do melhor caso.

Complexidade de Tempo

Analisando a entrada

- Grafo: número de nós, número de arestas;
- Estrutura de dados: tamanho do vetor, altura da árvore;
- Cadeia: tamanho da cadeia de entrada.

Tipos de Análise

- Análise do pior caso;
- Análise do caso médio;
- Análise do melhor caso.

Utilizaremos aqui...

O tamanho da cadeia de entrada e a análise de pior caso.

Complexidade de Tempo

Definição 7.1

Seja M uma máquina de Turing determinística que pára sobre todas as entradas. O tempo de execução ou **complexidade de tempo** de M é a função $f : \mathbb{N} \to \mathbb{N}$, em que $f(n)$ é o número máximo de passos que M usa sobre qualquer entrada de comprimento n.

Se $f(n)$ for o tempo de execução de M, dizemos que M *roda* em tempo $f(n)$ e que M é uma máquina de Turing *de tempo* $f(n)$. Costumeiramente usamos n para representar o comprimento da entrada.

Complexidade de Tempo

Notação O-Grande

Sejam f e g funções $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$.
Vamos dizer que $f(n) = O(g(n))$ se inteiros positivos c e n_0 existem tais que para todo inteiro $n \geq n_0$ em que

$$f(n) \leq c.g(n)$$

Quando $f(n) = O(g(n))$, dizemos que $g(n)$ é um **limitante superior** para $f(n)$, ou mais precisamente, que $g(n)$ é um **limitante superior assintótico** para $f(n)$, para enfatizar que estamos suprimindo fatores constantes.

Complexidade de Tempo

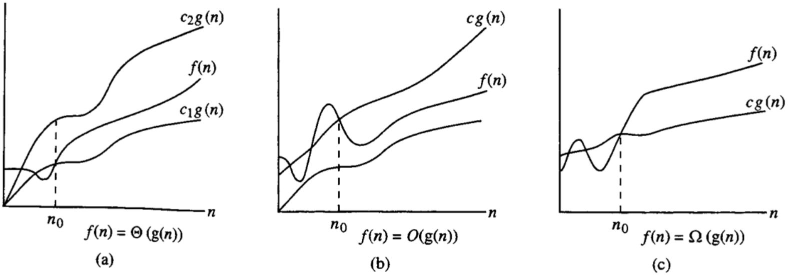

Figura: Comportamento das notações Θ, O e Ω.

Complexidade de Tempo

$f_1(n) = 5n^3 + 2n^2 + 22n + 6$

$$
\begin{aligned}
O(f_1(n)) &= O(5n^3 + 2n^2 + 22n + 6) && (1)\\
&= O(5n^3) && (2)\\
&= O(n^3) && (3)
\end{aligned}
$$

Complexidade de Tempo

$f_1(n) = 5n^3 + 2n^2 + 22n + 6$

$$
\begin{aligned}
O(f_1(n)) &= O(5n^3 + 2n^2 + 22n + 6) && (1)\\
&= O(5n^3) && (2)\\
&= O(n^3) && (3)
\end{aligned}
$$

É verdade porque...

Basta admitir $c = 6$, e $n_0 = 10$. Logo

$$5n^3 + 2n^2 + 22n + 6 \leq 6n^3$$

para todo $n \geq 10$.

Complexidade de Tempo

$f_1(n) = 5n^3 + 2n^2 + 22n + 6$

$$\begin{aligned} O(f_1(n)) &= O(5n^3 + 2n^2 + 22n + 6) && (4)\\ &= O(5n^3) && (5)\\ &= O(n^3) && (6) \end{aligned}$$

Complexidade de Tempo

$f_1(n) = 5n^3 + 2n^2 + 22n + 6$

$$
\begin{aligned}
O(f_1(n)) &= O(5n^3 + 2n^2 + 22n + 6) && (4)\\
&= O(5n^3) && (5)\\
&= O(n^3) && (6)
\end{aligned}
$$

Também é verdade dizer que...

$f_1(n) = O(n^4)$, pois n^4 é maior que n^3 e portanto é ainda um limitante assintótico superior sobre f_1.

UFG
Campus Jataí

Complexidade de Tempo

$f_1(n) = 5n^3 + 2n^2 + 22n + 6$

$$
\begin{aligned}
O(f_1(n)) &= O(5n^3 + 2n^2 + 22n + 6) && (4)\\
&= O(5n^3) && (5)\\
&= O(n^3) && (6)
\end{aligned}
$$

Também é verdade dizer que...

$f_1(n) = O(n^4)$, pois n^4 é maior que n^3 e portanto é ainda um limitante assintótico superior sobre f_1.

Mas...

$f_1(n) \neq O(n^2)$.

Complexidade de Tempo

$f_2(n) = \log_{13} n + 5$

Complexidade de Tempo

$f_2(n) = \log_{13} n + 5$

$$O(f_2(n)) = O(\log_{13} n + 5) \quad (7)$$
$$= O(\log_{13} n) \quad (8)$$
$$= O(\log n) \quad (9)$$

UFG
Campus Jataí

Complexidade de Tempo

$f_2(n) = \log_{13} n + 5$

$$\begin{aligned} O(f_2(n)) &= O(\log_{13} n + 5) && (7)\\ &= O(\log_{13} n) && (8)\\ &= O(\log n) && (9) \end{aligned}$$

Porque...

$$\log n = \log_{10} n = \frac{\log_{13} n}{\log_{13} 10}$$

Complexidade de Tempo

$f_3(n) = 3n\log_2 n + 5n\log_2\log_2 n + 2$

Complexidade de Tempo

$f_3(n) = 3n\log_2 n + 5n\log_2\log_2 n + 2$

$$
\begin{aligned}
O(f_3(n)) &= O(3n\log_2 n + 5n\log_2\log_2 n + 2) && (10)\\
&= O(3n\log_2 n) && (11)\\
&= O(n\log n) && (12)
\end{aligned}
$$

UFG
Campus Jataí

Complexidade de Tempo

$f_3(n) = 3n\log_2 n + 5n\log_2\log_2 n + 2$

$$
\begin{aligned}
O(f_3(n)) &= O(3n\log_2 n + 5n\log_2\log_2 n + 2) && (10)\\
&= O(3n\log_2 n) && (11)\\
&= O(n\log n) && (12)
\end{aligned}
$$

Porque...

$n\log n$ domina sobre $\log\log n$.

UFG Campus Jataí

Lista de Exercícios 05

Livro

SIPSER, M. **Introdução à Teoria da Computação**, 2a Edição, Editora Thomson Learning, 2011. **Código Bib.: [004 SIP/int]**.

Exercícios

- 7.1;
- 7.2;
- 7.6.

UFG
Campus Jataí

Complexidade de Tempo

Esdras Lins Bispo Jr.
bispojr@ufg.br

Teoria da Computação
Bacharelado em Ciência da Computação

02 de junho de 2014

UFG
Campus Jataí