Teoria da Computação

Guilherme Henrique de Souza Nakahata

Universidade Estadual do Paraná - Unespar

22 de Maio de 2024

Avaliação de um algoritmo

- Simplicidade;
  - Facilmente entendido;
  - Implementado;
  - Mantido;
  - Não se conhece técnicas formais para isto.
- Corretude;
  - Toda entrada específica;
  - Saída correta é produzida.

Avaliação de um algoritmo

- Eficiência:
  - Recursos requeridos para o funcionamento;
    - Tempo;
    - Memória;
    - etc...

Onde utilizar?

- Projetar algoritmos mais eficientes;
- Saber se suas implementações são viáveis (Ponto de vista prático);
- Saber qual é o melhor algoritmo;
- Saber o grau de dificuldade de um problema (Teoria da complexidade).

O que é análise de algoritmos?

- Ferramentas matemáticas:
  - Análise Combinatória;
  - Teoria das probabilidades;
  - Destreza matemática:
    - Indução matemática;
    - Séries e produtórios;
    - Potências e Logaritmos;
    - etc...

Invariante de laço

- O que é?
- Onde é utilizada?
- Como utilizar?

Invariante de laço

- O que é?
  - Uma propriedade que se mantém verdadeira antes e após cada iteração de um laço;
- Onde é utilizada?
- Como utilizar?

Invariante de laço

- O que é?
  - Uma propriedade que se mantém verdadeira antes e após cada iteração de um laço;
- Onde é utilizada?
  - Correção de programas;
  - Otimização de códigos;
  - Análise de complexidade.
- Como utilizar?

Invariante de laço

- O que é?
  - Uma propriedade que se mantém verdadeira antes e após cada iteração de um laço;
- Onde é utilizada?
  - Correção de programas;
  - Otimização de códigos;
  - Análise de complexidade.
- Como utilizar?
  - Verificar se a condição é verdadeira:
    - Antes do laço começar;
    - Após cada iteração do laço;
    - Quando o laço termina.

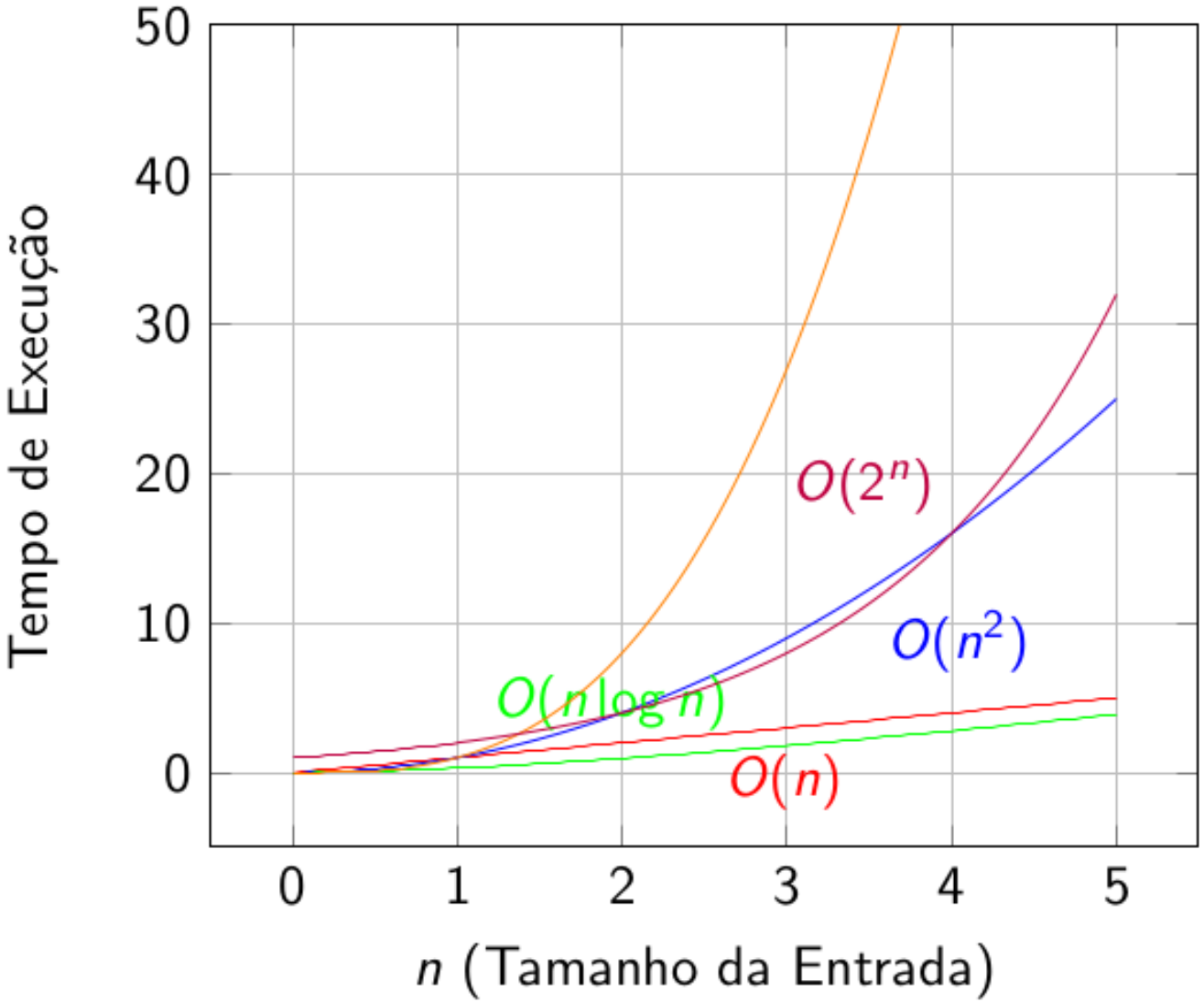
Notação assintótica

- Expressar a complexidade de algoritmos;
- Funções matemáticas;
- Notação Assintótica;
- Descreve o comportamento de funções no limite;
- A notação assintótica descreve o crescimento de funções;
- Foca no que é importante;
- Abstrair os termos de baixa ordem e constantes multiplicativas;
- Análise Assintótica de Algoritmos.

Principais Anotações de Funções Assintóticas

Notação	Descrição
O	$f(n) = O(g(n))$ significa que $g(n)$ é um limite superior assintótico para $f(n)$.
Ω	$f(n) = \Omega(g(n))$ significa que $g(n)$ é um limite inferior assintótico para $f(n)$.
Θ	$f(n) = \Theta(g(n))$ significa que $f(n)$ é limitada assintoticamente superior e inferiormente por $g(n)$.
o	$f(n) = o(g(n))$ significa que $f(n)$ cresce mais lentamente do que $g(n)$ para entradas grandes.
ω	$f(n) = \omega(g(n))$ significa que $f(n)$ cresce mais rapidamente do que $g(n)$ para entradas grandes.

Classes Comuns em Análise Assintótica



Definição

- Seja T(n) e f(n) função dos números inteiros para os reais;
- Dizemos que T(n) é O(f(n)) se:
- Existir constantes positivas c e n_0;
- Tais que T(n) $\leq$ cf(n);
- Para todo $n \geq n_0$.

Exemplo

- $\mathrm{T(n)} = 50n^3 + 5n + n$ é $O(n^3)$;
- $\mathrm{T(n)} = 50n^3 + 5n + n \leq c.n^3$;

Exercícios

- $T(n) = 30n^4 + 20n^2 + 10n$ é $O(n^4)$;
- $T(n) = 15n^5 + 7n^3 + 3n^2$ é $O(n^5)$;
- $T(n) = 40n^6 + 5n^4 + 9n$ é $O(n^6)$;

Notação assintótica

- Notação Ω e Θ;
- $f(n) = \Omega(g(n))$ significa que $g(n)$ é um limite inferior assintótico para $f(n)$;
- $f(n) = \Theta(g(n))$ significa que $f(n)$ é limitada assintoticamente superior e inferiormente por $g(n)$;

Definição

- Seja f(n) e g(n) função dos números inteiros para os reais;
- Dizemos que f (n) é Ω(g(n)) se existirem constantes positivas c e n_0 tais que:
- $f(n) \geq cg(n)$;
- Para todo n $\geq n_0$.

Exemplo de Prova de Notação Big-Ω

- Prove $T(n) = 10n^3 + 5n^2 + n$ é $\Omega(n^3)$;
- $T(n) = 10n^3 + 5n^2 + n \geq c.n^3$;

Exercícios

- Prove $T(n) = 30n^4 + 20n^2 + 10n$ é $\Omega(n^4)$;
- Prove $T(n) = 15n^5 + 7n^3 + 3n^2$ é $\Omega(n^5)$;
- Prove $T(n) = 40n^6 + 5n^4 + 9n$ é $\Omega(n^6)$;

Exemplo Big-Θ

- Dizemos que f (n) é Θ(g(n)) se existirem constantes positivas c_1 e c_2 e n_0 tais que:
- c1 g(n) $\leq$ f (n) $\leq$ c2 g(n);
- para todo n $\geq$ n_0;
- Dizemos que f (n) = Θ(g(n)) se somente se:
  - f(n) = Ω(g(n))
  - f(n) = O(g(n))

Exemplo Big-Θ

- Prove que $f(n) = 2n^2 + 3n + 5$ é $\Theta(n^2)$;
- $$c_1 \cdot n^2 \leq 2n^2 + 3n + 5 \leq c_2 \cdot n^2$$

.

Exemplo Big-Θ

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$
$f(n) = \Omega(g(n))$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$
$f(n) = \Theta(g(n))$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$
$f(n) = O(g(n))$

Exercícios

- Provar $f(n) = 3n^2 + 4n + 7$ é $\Theta(n^2)$;
- Provar $f(n) = n \log n + n$ é $\Theta(n \log n)$;

Complexidade de Algoritmos Recursivos

- $T(n) = T(n-1) + n$
  - $\Theta(n^2)$
  - Algoritmo recursivo que a cada loop examina a entrada e elimina um item;
- $T(n) = T(n/2) + c$
  - $\Theta(\log n)$
  - Algoritmo recursivo que divide a entrada em cada passo;
- $T(n) = T(n/2) + n$
  - $\Theta(n)$
  - Algoritmo recursivo que divide a entrada, mas precisa examinar cada item na entrada;
- $T(n) = 2T(n/2) + 1$
  - $\Theta(n)$
  - Algoritmo recursivo que divide a entrada em duas metades e executa uma quantidade constante de operações.

Exemplo

Algorithm 1 MergeSort(A, p, r)

if $p < r$ **then**
  $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
  MergeSort(A, p, q)
  MergeSort$(A, q + 1, r)$
  Merge(A, p, q, r)
end if

Exemplo

- O método é expresso na forma $T(n) = aT(n/b) + f(n)$.
- a é o número de subproblemas.
- b é o fator pelo qual o tamanho do problema é reduzido em cada chamada recursiva.
- $f(n)$ é o custo da divisão e conquista.

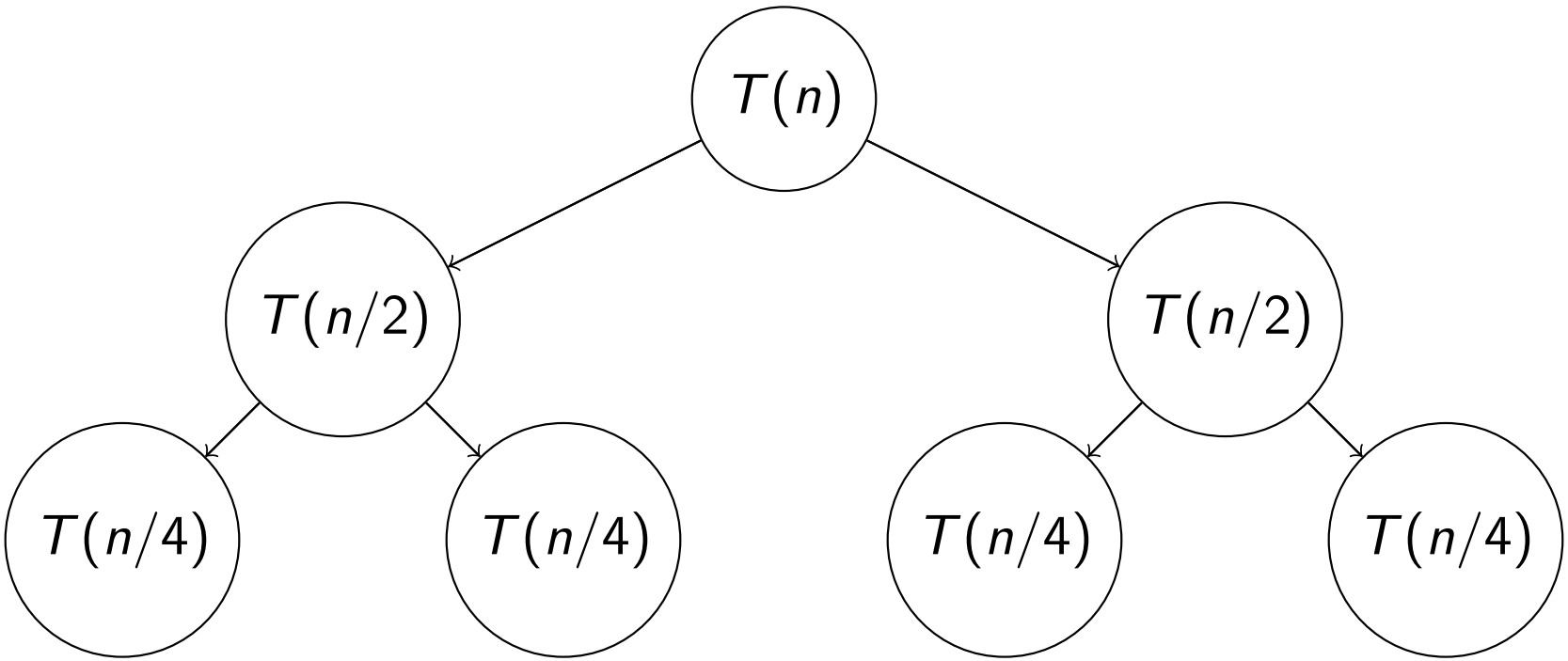
Método de substituição

- O método consiste de duas etapas:
  - Pressupor uma função como solução;
  - Usar **indução matemática** para encontrar as contantes da definição assintótica;
- Provar no passo da indução examente o que foi proposto;
- Provar que $T(n) \leq f(n)$.

Árvore de recursão

- Cada nó representa o custo de um único subproblema;
- Durante o processo de chamadas de funções recursivas;
- Permite somar o custo de cada nível da árvore;
- Determinar o custo total pela soma de todos os níveis.

Árvore de Recursão para $T(n) = 2T(n/2) + cn$



Exemplo

- Cada nível da árvore tem custo de cn;
- Altura da árvore é lg n;
- O(n lg n);

Método mestre

Caso 1: $f(n) = O(n^{\log_b a - \varepsilon})$

$T(n) = \Theta(n^{\log_b a})$.

Caso 2: $f(n) = \Theta(n^{\log_b a})$

$T(n) = \Theta(n^{\log_b a} \log n)$.

Caso 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$ e $af(n/b) \leq kf(n)$ para algum $k < 1$ e n suficientemente grande

$T(n) = \Theta(f(n))$.

Método mestre

- Caso 1: Se $f(n)$ é assintoticamente menor que $n^{\log_b a}$ para algum $\varepsilon > 0$, então a solução é $T(n) = \Theta(n^{\log_b a})$;
- Caso 2: Se $f(n)$ é assintoticamente igual a $n^{\log_b a}$, então a solução é $T(n) = \Theta(n^{\log_b a} \log n)$;
- Caso 3: Se $f(n)$ é assintoticamente maior que $n^{\log_b a}$ e $af(n/b) \leq kf(n)$ para algum $k < 1$ e n suficientemente grande, então a solução é $T(n) = \Theta(f(n))$.

Exercícios

- $T(n) = 2T(n/2) + n$;
- $\mathrm{T(n)} = 16\mathrm{T(n/4)} + \mathrm{n}$;
- $T(n) = 2T(n/2) + n^2$.

Bibliografia Básica

- LEWIS, H. R.; PAPADIMITRIOU, C. H. **Elementos de Teoria da Computação**. 2 ed. Porto Alegre: Bookman, 2000.
- VIEIRA, N. J. **Introdução aos Fundamentos da Computação**. Editora Pioneira Thomson Learning, 2006.
- DIVERIO, T. A.; MENEZES, P. B. **Teoria da Computação: Máquinas Universais e Computabilidade**. Série Livros Didáticos Número 5, Instituto de Informática da UFRGS, Editora Sagra Luzzato, 1 ed. 1999.

Obrigado!
Dúvidas?

Guilherme Henrique de Souza Nakahata

guilhermenakahata@gmail.com

https://github.com/GuilhermeNakahata/UNESPAR-2024