

### *Engenharia da Computação – 3ª série*

## *Método Analítico e Modelo de Knuth* *(E1, E2)*

**2025**

### Pergunta

- O Método Experimental é suficiente para análise de algoritmos?



### Resposta



- O Método Experimental:
  - ✓ Embora tenha um importante papel em análise de algoritmos, quando tratado de forma isolada não é suficiente.

### Pergunta

- O que é necessário, então, além do Método Experimental?



### Resposta



- É necessário um Método Analítico que:
  - ✓ Considere todas as entradas possíveis;
  - ✓ Seja independente de ambientes de *hardware* e de *software*;
  - ✓ Seja obtido sem a execução do algoritmo.

### Pergunta

- O que é um Método Analítico?



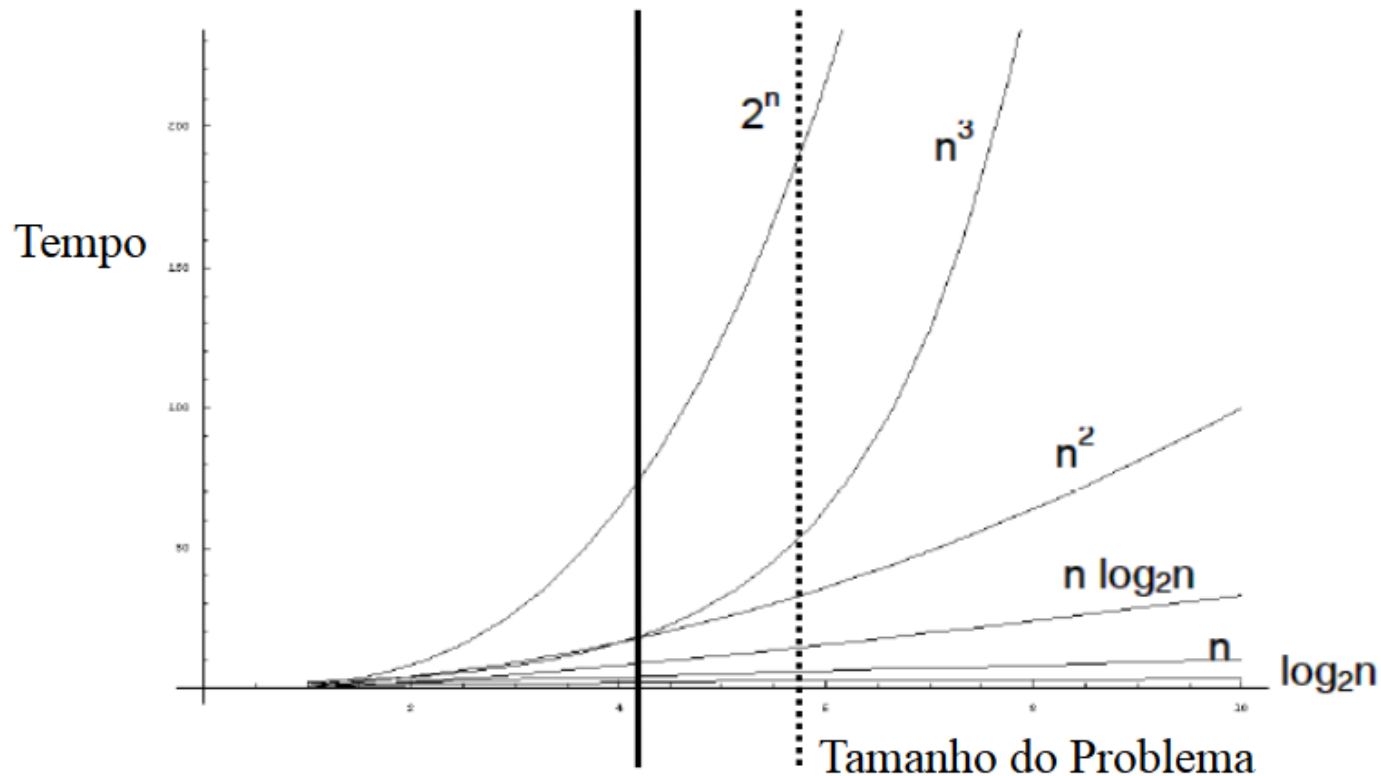
# ECM306 – Tópicos Avançados em Estrutura de Dados

## Método Analítico e Modelo de Knuth

### Resposta



- Método Analítico:



### Pergunta

- O que é o Modelo de Knuth?





### Resposta



- Modelo de Knuth é:
  - ✓ Modelo matemático, desenvolvido pelo cientista computacional e professor Donald Knuth, da Stanford University, em 1968;
  - ✓ Baseado na contabilização do conjunto de operações executadas pelo algoritmo;
  - ✓ Uma associação de custo para cada operação executada, ignorando-o, em geral, em algumas operações e contabilizando-o, apenas, nas mais significativas, denominadas **Operações Básicas**.

### Pergunta

- O que acontece no Modelo Detalhado?



### Resposta



- No Modelo Detalhado:
  - ✓ O tempo de execução de um algoritmo será calculado pela somatória do tempo necessário para a execução das operações básicas;
  - ✓ O tempo de processamento das operações básicas é definido por um conjunto de **Axiomas**.

### Pergunta

- O que propõe o ***Axioma 1***?

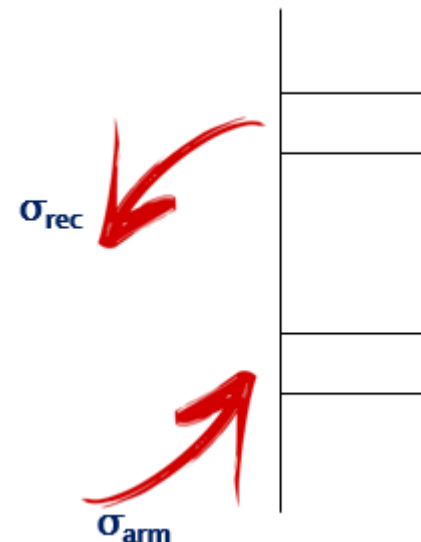


### Resposta



- O **Axioma 1** propõe que:
  - ✓ Os tempos requeridos para recuperar um operando da memória e para armazenar o resultado na memória são constantes, respectivamente:

$\sigma_{rec}$  e  $\sigma_{arm}$



### Exemplo 1



- **Axioma 1:**
  - Qual o tempo de processamento da atribuição?

**$y = x;$**

### Exemplo 1



- **Axioma 1:**

➤ Qual o tempo de processamento da atribuição?

$$y = x;$$

- ✓ Será necessário recuperar-se em memória o conteúdo da variável  $x$ , sendo esse tempo o  $\sigma_{rec}$
- ✓ O tempo requerido para se armazenar o valor na variável  $y$  é  $\sigma_{arm}$

### Exemplo 1



- **Axioma 1:**

➤ Qual o tempo de processamento da atribuição?

$$y = x;$$

- ✓ Será necessário recuperar-se em memória o conteúdo da variável  $x$ , sendo esse tempo o  $\sigma_{rec}$
- ✓ O tempo requerido para se armazenar o valor na variável  $y$  é  $\sigma_{arm}$

Resposta:  $\sigma_{rec} + \sigma_{arm}$



### Exemplo 2



- **Axioma 1:**

- Qual o tempo de processamento da atribuição?

$$y = 1;$$

### Exemplo 2



- **Axioma 1:**

➤ Qual o tempo de processamento da atribuição?

$$y = 1;$$

- ✓ Constantes, chamadas de literais numéricos, também são armazenadas em memória (tabela de literais gerada pelo computador);

### Exemplo 2



- **Axioma 1:**

➤ Qual o tempo de processamento da atribuição?

$$y = 1;$$

- ✓ Assim, o custo para se recuperar em memória a constante **1** também será  $\sigma_{rec}$
- ✓ O tempo requerido para se armazenar o valor na variável  $y$  é  $\sigma_{arm}$

### Exemplo 2



- **Axioma 1:**

➤ Qual o tempo de processamento da atribuição?

$$y = 1;$$

- ✓ Assim, o custo para se recuperar em memória a constante **1** também será  $\sigma_{rec}$
- ✓ O tempo requerido para se armazenar o valor na variável **y** é  $\sigma_{arm}$

Resposta:  $\sigma_{rec} + \sigma_{arm}$

### Pergunta

- O que propõe o **Axioma 2**?



### Resposta



- O **Axioma 2** propõe que:
  - ✓ Os tempos necessários para se realizar operações aritméticas elementares, tais como adição, subtração, multiplicação, divisão e comparação são todos **constantes**.
  - ✓ Estes tempos são denotados, respectivamente, por:

$$\sigma_+, \sigma_-, \sigma_x, \sigma_{/} \text{ e } \sigma_{\leq}$$

### Exemplo 3



- **Axioma 2:**

- Qual o tempo de processamento da atribuição?

$$y = y + 1;$$

### Exemplo 3



- **Axioma 2:**

➤ Qual o tempo de processamento da atribuição?

$$y = y + 1;$$

- ✓ Deve-se recuperar dois valores em memória:  $y$  e  $1$ ;
- ✓ O tempo para se recuperar esses valores é  $2\sigma_{rec}$
- ✓ O tempo para se efetuar a soma é  $\sigma_+$
- ✓ O tempo requerido para se armazenar o resultado na variável  $y$  é  $\sigma_{arm}$



### Exemplo 3



- **Axioma 2:**

➤ Qual o tempo de processamento da atribuição?

$$y = y + 1;$$

- ✓ Deve-se recuperar dois valores em memória:  $y$  e  $1$ ;
- ✓ O tempo para se recuperar esses valores é  $2\sigma_{rec}$
- ✓ O tempo para se efetuar a soma é  $\sigma_+$
- ✓ O tempo requerido para se armazenar o resultado na variável  $y$  é  $\sigma_{arm}$

Resposta:  $2\sigma_{rec} + \sigma_+ + \sigma_{arm}$

### Pergunta

- O que propõe o **Axioma 3**?



### Resposta



- O **Axioma 3** propõe que:
  - ✓ O tempo necessário para se chamar uma função é **constante**:  $\sigma_{chamada}$  e o tempo necessário para se retornar de uma função é **constante**:  $\sigma_{retorno}$
  - ✓ Quando um método é chamado, algumas operações de bastidores são necessárias: *save* de endereços de retorno, chaveamento de contexto etc.
  - ✓ Estas operações são desfeitas no momento de retorno.

### Pergunta

- O que propõe o **Axioma 4**?



### Resposta



- O **Axioma 4** propõe que:
  - ✓ O tempo necessário para se passar um parâmetro, ou argumento, a um método é o mesmo tempo para se armazenar um valor em memória:  $\sigma_{arm}$
  - ✓ Conceitualmente, o esforço computacional necessário para o tratamento da passagem de parâmetros é o mesmo que se atribuir ao parâmetro formal do método o valor do argumento.

### Pergunta

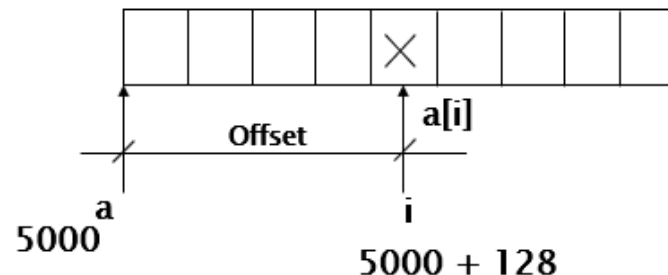
- Como ficam as Operações com índices de *Arrays*?



### Resposta



- Operações com índices de *Arrays*:
  - ✓ Em geral, os elementos de um *array*, ou vetor, são armazenados em locais contíguos de memórias;
  - ✓ Assim, dado o endereço do primeiro elemento do *array*, uma simples operação de adição é suficiente para se determinar o endereço de um elemento arbitrário do *array*.



### Pergunta

- O que propõe o ***Axioma 5***?





### Resposta



- O **Axioma 5** propõe que:
  - ✓ O tempo requerido para o cálculo do **endereço** advindo de uma operação de índice de um *array*, por exemplo,  **$a[i]$**  é constante:  $\mathcal{O}.$
  - ✓ Esse tempo não inclui o tempo para calcular a expressão do índice, nem inclui o tempo de acesso, ou seja, o tempo de recuperação ou armazenamento, ao elemento do *array*;
  - ✓ Exemplo:  **$y = a[i];$**

### Resposta



- O **Axioma 5** propõe que:

✓ Exemplo:  $y = a[i];$

Tempo:  $3\sigma_{rec} + \sigma_{.} + \sigma_{arm}$

- ✓ Serão necessários **três recuperações**:
  - A primeira para recuperar  $a$ , que é o endereço base do *array*;
  - A segunda para recuperar  $i$ ; e
  - A terceira para recuperar o elemento  $a[i]$ .

### Pergunta

- O que ocorre no Modelo Simplificado de Knuth?



### Resposta



- No Modelo Simplificado de Knuth:
  - ✓ É eliminada a dependência de tempo, assumindo-se um tempo constante e igual ao ciclo do processador  **$T = 1$** ;
  - ✓ Assim, contabiliza-se apenas a quantidade de operações efetuadas pelo algoritmo;
  - ✓ O modelo detalhado fornece uma boa previsão do desempenho de execução de um algoritmo, porém, tal modelo é oneroso e trabalhoso, comparado ao modelo simplificado.

### Pergunta

- O que propõe o ***Axioma 1*** do Modelo Simplificado?



### Resposta



- O **Axioma 1** do Modelo Simplificado propõe que:
  - ✓ O total de ciclos de processador requeridos para recuperar um operando da memória e para armazenar o resultado na memória são constantes:  
*1 ciclo = 1 operação, para recuperar; e*  
*1 ciclo = 1 operação para armazenar.*
  - ✓ Exemplo: Qual o total de operações para executar a atribuição?

$$y = x; \quad (\sigma_{rec} + \sigma_{arm})$$

### Resposta



- O **Axioma 1** do Modelo Simplificado propõe que:
  - ✓ O total de ciclos de processador requeridos para recuperar um operando da memória e para armazenar o resultado na memória são constantes:  
*1 ciclo = 1 operação, para recuperar; e*  
*1 ciclo = 1 operação para armazenar.*
  - ✓ Exemplo: Qual o total de operações para executar a atribuição?

$$y = x; \quad (\sigma_{rec} + \sigma_{arm})$$

Resposta:  $1 + 1 = 2$  operações

### Pergunta

- O que propõe o ***Axioma 2*** do Modelo Simplificado?





### Resposta



- O **Axioma 2** do Modelo Simplificado propõe que:
  - ✓ As operações necessárias para se realizar operações aritméticas elementares, tais como adição, subtração, multiplicação, divisão e comparação são todas constantes e iguais a **1 ciclo cada** ou **1 operação**;
  - ✓ Exemplo: Qual o total de operações para executar a atribuição?

$$y = y + 1; \quad (2\sigma_{rec} + \sigma_+ + \sigma_{arm})$$

### Resposta



- O **Axioma 2** do Modelo Simplificado propõe que:
  - ✓ As operações necessárias para se realizar operações aritméticas elementares, tais como adição, subtração, multiplicação, divisão e comparação são todas constantes e iguais a **1 ciclo cada** ou **1 operação**;
  - ✓ Exemplo: Qual o total de operações para executar a atribuição?

$$y = y + 1; \quad (2\sigma_{rec} + \sigma_{+} + \sigma_{arm})$$

Resposta: 4 operações

### Pergunta

- O que propõe o **Axioma 3** do Modelo Simplificado?



### Resposta



- O **Axioma 3** do Modelo Simplificado propõe que:
  - ✓ Gasta-se **1 ciclo** de processador, ou **1 operação**, para se chamar um método e **1 ciclo**, ou **1 operação**, para se providenciar o retorno do método;

### Resposta



- O **Axioma 3** do Modelo Simplificado propõe que:
  - ✓ Gasta-se **1 ciclo** de processador, ou **1 operação**, para se chamar um método e **1 ciclo**, ou **1 operação**, para se providenciar o retorno do método;

Resposta: **2 operações**

### Pergunta

- O que propõe o ***Axioma 4*** do Modelo Simplificado?



### Resposta



- O **Axioma 4** do Modelo Simplificado propõe que:
  - ✓ Gasta-se **1 ciclo** de processador, ou **1 operação**, para se passar um parâmetro a um método;

### Resposta



- O **Axioma 4** do Modelo Simplificado propõe que:
  - ✓ Gasta-se **1 ciclo** de processador, ou **1 operação**, para se passar um parâmetro a um método;

Resposta: **1 operação**



### Pergunta

- O que propõe o ***Axioma 5*** do Modelo Simplificado?



### Resposta



- O **Axioma 5** do Modelo Simplificado propõe que:
  - ✓ Serão necessárias três recuperações:
    - A primeira para recuperar ***a***, o endereço base do *array*;
    - A segunda para recuperar ***i***; e
    - A terceira para recuperar o elemento ***a[i]***;

$$y = a[i]; \quad \text{Tempo: } (3\sigma_{rec} + \sigma_{.} + \sigma_{arm})$$

### Resposta



- O **Axioma 5** do Modelo Simplificado propõe que:
  - ✓ Serão necessárias três recuperações:
    - A primeira para recuperar  $a$ , o endereço base do *array*;
    - A segunda para recuperar  $i$ ; e
    - A terceira para recuperar o elemento  $a[i]$ ;

$$y = a[i]; \quad \text{Tempo: } (3\sigma_{rec} + \sigma_{.} + \sigma_{arm})$$

Resposta: 5 operações básicas

### Pergunta

- Qual a utilidade da Função de Complexidade?



### Resposta



- A Função Complexidade é útil:
  - ✓ Para medir o custo de execução de um algoritmo é comum definir uma função de custo ou função de complexidade  $f$ ;
  - ✓ A Função de Complexidade na realidade não representa tempo diretamente, mas o número de vezes que determinada operação, considerada relevante, é executada.

### Pergunta

- Quais os tipos de Função de Complexidade?



### Resposta

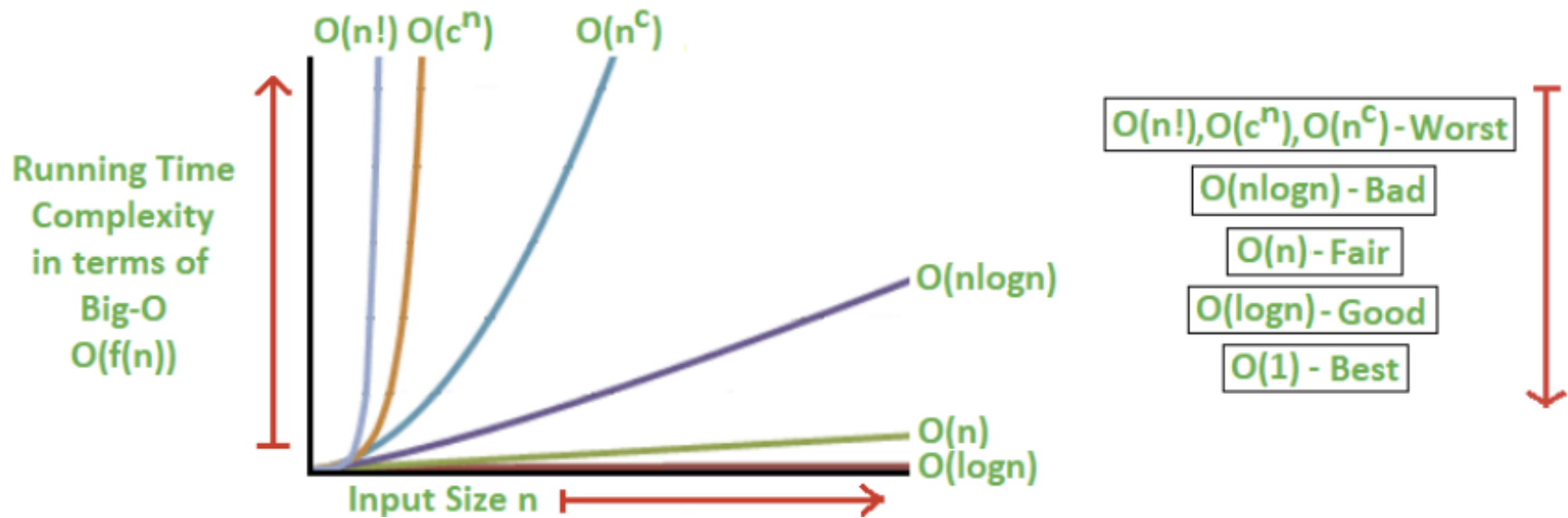


- A Função Complexidade pode ser:
  - ✓ Função de complexidade de **tempo**:  $f(n)$  mede o tempo necessário para executar um algoritmo em um problema de tamanho  $n$ ;
  - ✓ Função de complexidade de **espaço**:  $f(n)$  mede a memória necessária para executar um algoritmo em um problema de tamanho  $n$ .

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Método Analítico e Modelo de Knuth

### Função de Complexidade





# ECM306 – Tópicos Avançados em Estrutura de Dados

## Referências bibliográficas

- CORMEN, T.H. et al. Algoritmos: Teoria e Prática (Caps. 13). Campus. 2002.
- ZIVIANI, N. Projeto de algoritmos: com implementações em Pascal e C (Cap. 1). 2.ed. Thomson, 2004.
- FEOFILOFF, P. Minicurso de Análise de Algoritmos, 2010. Disponível em:  
<http://www.ime.usp.br/~pf/livrinho-AA/>
- DOWNEY, A.B. *Analysis of algorithms* (Cap. 2), Em: *Computational Modeling and Complexity Science*. Disponível em:  
<http://www.greenteapress.com/compmo/html/book003.html>
- ROSA, J.L. Notas de Aula de Introdução a Ciência de Computação II. Universidade de São Paulo. Disponível em:  
<http://coteia.icmc.usp.br/mostra.php?ident=639>

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Referências bibliográficas

- GOODRICH, Michael T. et al: *Algorithm Design and Applications*. Wiley, 2015.
- LEVITIN, Anany. *Introduction to the Design and Analysis of Algorithms*. Pearson, 2012.
- SKIENA, Steven S. *The Algorithm Design Manual*. Springer, 2008.
- Série de Livros Didáticos. *Complexidade de Algoritmos*. UFRGS.
- BHASIN, Harsh. *Algorithms – Design and Analysis*. Oxford University Press, 2015.
- FREITAS, Aparecido V. de – 2022 – Estruturas de Dados: Notas de Aula.
- CALVETTI, Robson - 2015 – Estruturas de Dados: Notas de Aula.

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Aula 02

FIM