

## Aula 01

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

## *Introdução à Análise de Algoritmos* *(E1, E2)*

**2025**

### Pergunta

- O que é Algoritmo?



### Resposta



- Algoritmo é:
  - ✓ Um procedimento, passo-a-passo, para a execução de alguma tarefa, em uma quantidade finita de tempo.

### Definição



### Algoritmo:

- Corresponde a uma descrição de um padrão de comportamento, expresso em termos de um conjunto finito de ações (*Dijkstra*, 1971 citado por *Ziviani*, 2004);
- Ao se executar a operação  $a + b$  percebe-se um padrão de comportamento, mesmo para valores diferentes de  $a$  e  $b$ ;
- Segundo *Cormen* et al. (2002), informalmente, é um procedimento computacional bem definido que toma um conjunto de valores como entrada e produz algum conjunto de valores como saída.

### Pergunta

- O que é Estrutura de Dados?



### Resposta

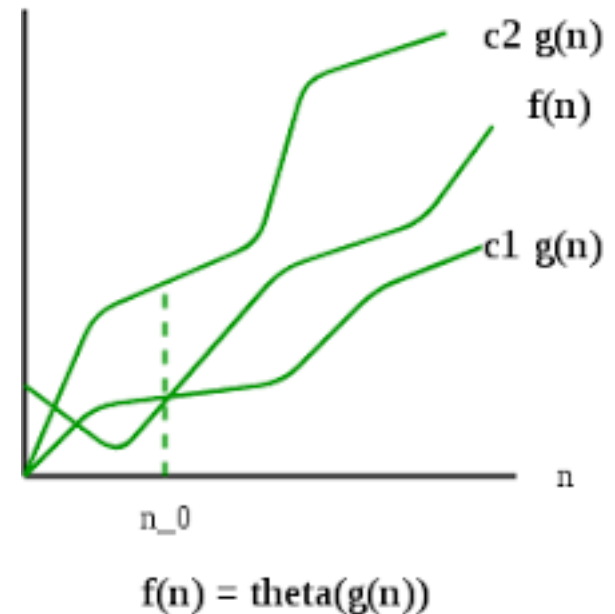


- Estrutura de Dados é:
  - ✓ Uma coleção de dados organizados, de tal forma, para que possam ser convenientemente manuseados pelos algoritmos, em suas operações.

## Introdução à Análise de Algoritmos

### Pergunta

- O que é Análise de Algoritmos?



### Resposta



- Análise de Algoritmos é:
  - ✓ O estudo da estimativa de recursos de tempo, espaço etc., consumidos pelos algoritmos;
  - ✓ Prever os recursos que o algoritmo necessitará;
  - ✓ Estimar o grau de dificuldade dos problemas computacionais.



### Situação 1

- Dado um algoritmo, pode-se executá-lo em uma dada máquina, para um determinado conjunto de dado.



### Reflexão

- Tal conhecimento é restrito e válido apenas para aquela situação?



### Conclusão

- Para se avaliar o comportamento de um algoritmo é necessário analisá-lo nos casos gerais, para várias instâncias.



### Situação 2

- Um desenvolvedor fica feliz, ao terminar a programação de um algoritmo, pelo fato dele estar sendo executado corretamente e com um bom desempenho nos testes.



### Reflexão

- O comportamento desse programa será também satisfatório em outras aplicações, ambientes e/ou computadores, para outras instâncias do problema?



### Conclusão

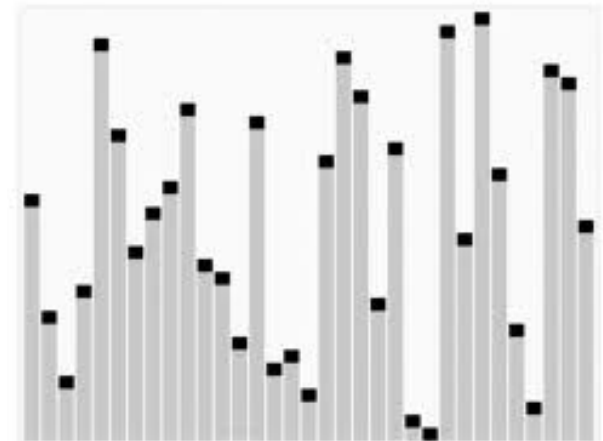
- Apesar de haver várias questões importantes a serem analisadas em um algoritmo, seu desempenho, em geral, é a primeira delas, principalmente em problemas de alta complexidade computacional.



## Introdução à Análise de Algoritmos

### Pergunta

- O que pode ser analisado em um algoritmo?



### Resposta



- Pode ser analisado por:
  - ✓ Desempenho;
  - ✓ Espaço de memória ocupado;
  - ✓ Comprimento total do código;
  - ✓ Corretismo ou Corretude;
  - ✓ Legibilidade;
  - ✓ Robustez;
  - ✓ Etc.



### Definição



### Características importantes de um algoritmo:

1. **Desempenho:** ponto chave de qualquer *software*;
2. **Simplicidade:** um algoritmo simples é mais fácil de ser implementado corretamente e, por consequência, há menor probabilidade de obter erros;
3. **Clareza:** deve ser escrito de forma clara e documentada para facilitar a manutenção;

### Definição



### Características importantes de um algoritmo:

4. **Segurança:** deve ser seguro (*safety & security*);
5. **Funcionalidade:** deve possuir diversas funcionalidades;
6. **Modularidade:** permite melhor manutenção, reuso, etc.;
7. **Interface amigável:** ou, em Inglês, *user friendly*, fundamental para a maior parte dos usuários;

### Definição



### Características importantes de um algoritmo:

- 8. Corretismo ou Corretude:** segundo *Cormen* et al. (2002), um algoritmo é dito correto se, para cada instância de entrada, ele para com a saída correta ou informa que não há solução para aquela entrada, ou seja, deseja-se que um algoritmo termine e seja correto.

Um algoritmo correto sempre termina, mesmo que seu processamento possa levar horas, dias ou anos;

### Definição



### Características importantes de um algoritmo:

9. **Problema Intratável:** não existe um algoritmo que solucione com demanda de recursos e tempo razoável.

Problemas para os quais não se conhece solução eficiente:

- i. N P-difícil; e
- ii. N P-completo.

### Pergunta

- Como avaliar um algoritmo?



### Resposta



- Pode ser avaliado por:
  - ✓ Métodos Experimentais;
  - ✓ Métodos Analíticos.

### Pergunta

- A Medição Direta de um algoritmo é viável em Métodos Experimentais?



### Resposta

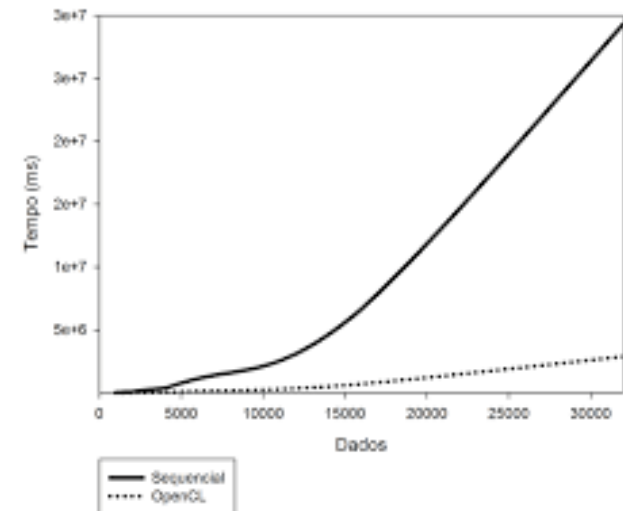


- A Medição Direta de um algoritmo em Métodos Experimentais tem:
  - ✓ Dependência do Compilador/Interpretador;
  - ✓ Dependência de *Hardware*;
  - ✓ Dependência do Sistema Operacional;
  - ✓ Quantidade de Memória disponível;
  - ✓ Espaço disponível em Disco;
  - ✓ Etc.



### Pergunta

- O que é Tamanho da Entrada de um algoritmo?



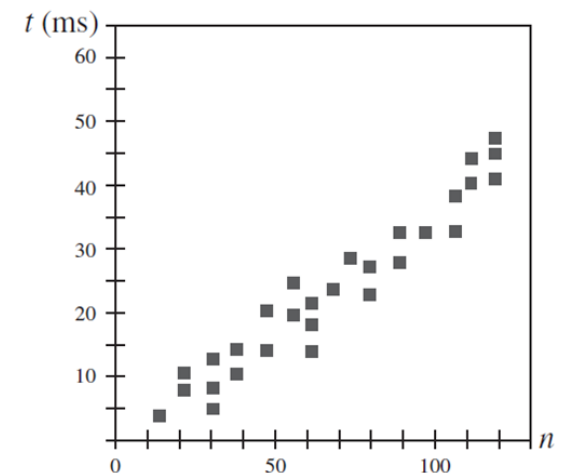
### Resposta



- Tamanho da Entrada de um algoritmo, depende do problema, mas geralmente é relativo ao número de elementos da entrada que são processados pelo algoritmo:
  - ✓ O número de elementos em um arranjo, lista, árvore etc.; e
  - ✓ O tamanho de um inteiro que é passado por parâmetro.

### Pergunta

- O Tempo de Execução de um algoritmo depende do Tamanho da Entrada?



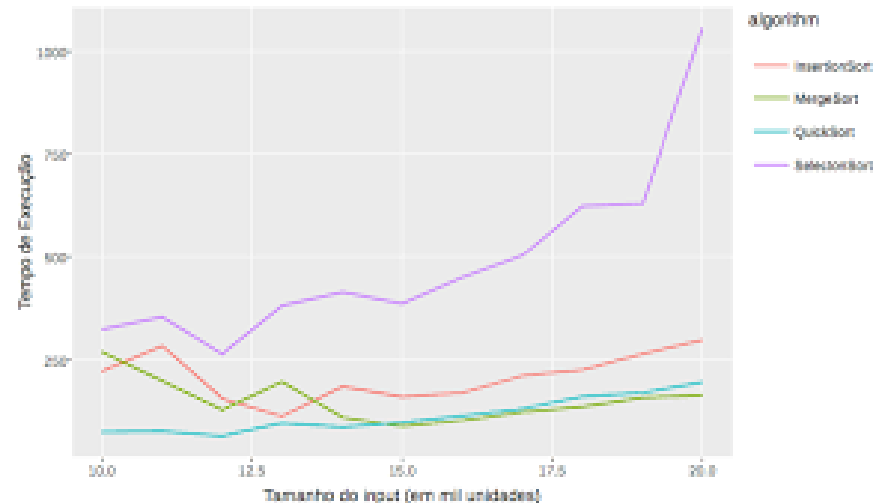
### Resposta



- O Tempo de Execução de um algoritmo:
  - ✓ Depende do tamanho da entrada, ou instância;
  - ✓ Aumenta, em geral, com o aumento da quantidade de suas entradas, mantidas “constantes” as outras dependências.

### Pergunta

- É útil Experimentar algoritmos, executando-os para diversos tamanhos de entrada e medindo os seus tempos?



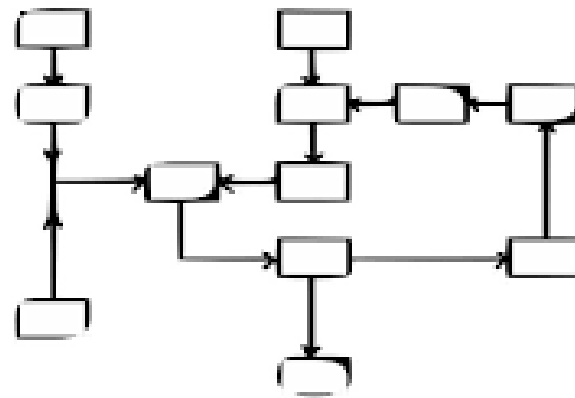
### Resposta



- Experimentar algoritmos, em função do Tempo de Execução, é útil:
  - ✓ Mas pode encontrar limitações;
  - ✓ Mas pode ser feito com um conjunto limitado e não representativo das entradas;
  - ✓ Porém, de difícil comparação e conclusão em ambientes de *software* e *hardware* não idênticos;
  - ✓ Mas sempre necessita da execução e da medição precisa, de forma experimental.

### Pergunta

- O que são Passos Básicos de um algoritmo?



### Resposta



- Passos Básicos de um algoritmo referem-se às operações primitivas, utilizadas pela máquina:
  - ✓ Operações aritméticas;
  - ✓ Comparações;
  - ✓ Chamadas à funções;
  - ✓ Retornos de funções;
  - ✓ Etc.



### Situação 3

- Numa situação hipotética, “desenvolveu-se um novo algoritmo chamado *TripleX*, que leva 14,2 segundos para processar 1.000 números, enquanto o método anterior, o *SimpleX* leva 42,1 segundos”.



### Reflexão

- Com essas informações, seria correto trocar o *SimpleX*, que já é utilizado na sua empresa, pelo *TripleX*?
- Seria o *TripleX* também mais rápido para processar quantidades maiores que 1.000 números?



### Conclusão

- Há vários outros fatores envolvidos que merecem análise;
- O número de passos básicos necessários, em função do tamanho da entrada que o algoritmo recebe:
  - Descorrelaciona a performance da máquina da performance do algoritmo; e
  - Reduz a análise ao desempenho, em função do tamanho da entrada.



### Conclusão

- Assumindo-se que o Tamanho da Entrada como sendo  $n$  e que cada operação leva aproximadamente o mesmo tempo, constante;
- *TripleX*: para uma entrada de tamanho  $n$ , o algoritmo, então, realiza  $n^2 + n$  operações:
  - Em termos de função,  $t(n) = n^2 + n$
- *SimpleX*: para uma entrada de tamanho  $n$ , o algoritmo, então, realiza  $2000 \cdot n$  operações:
  - Em termos de função,  $s(n) = 2000 \cdot n$

### Conclusão

- Calculando o Número de Operações, em função da entrada:

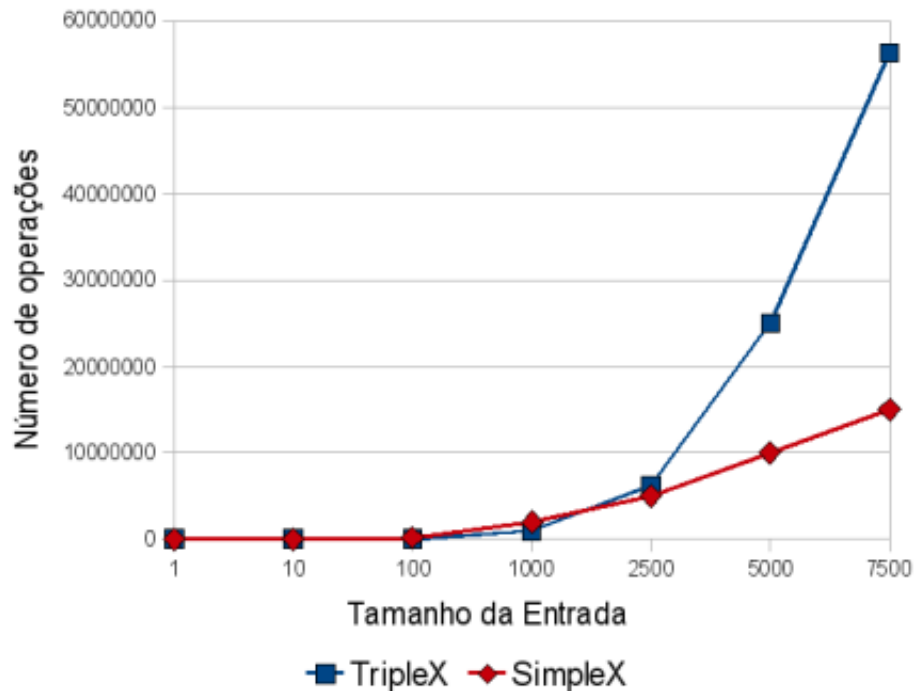
$n$	1	10	100	1.000	10.000
$t(n) = n^2 + n$					
$s(n) = 2000 \cdot n$					

# ECM306 – Tópicos Avançados em Estrutura de Dados

## Introdução à Análise de Algoritmos

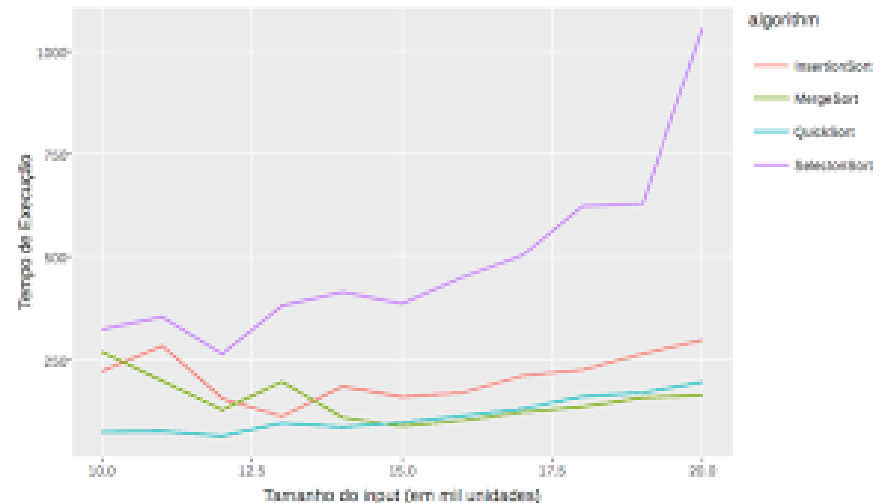
### Conclusão

$n$	1	10	100	1.000	10.000
$t(n) = n^2 + n$	2	110	10.100	1.001.000	<b>100.010.000</b>
$s(n) = 2000 \cdot n$	2.000	20.000	20.000	2.000.000	20.000.000



### Pergunta

- Seria útil Analisar algoritmos, considerando todas as entradas possíveis de serem aplicadas a eles?



### Resposta

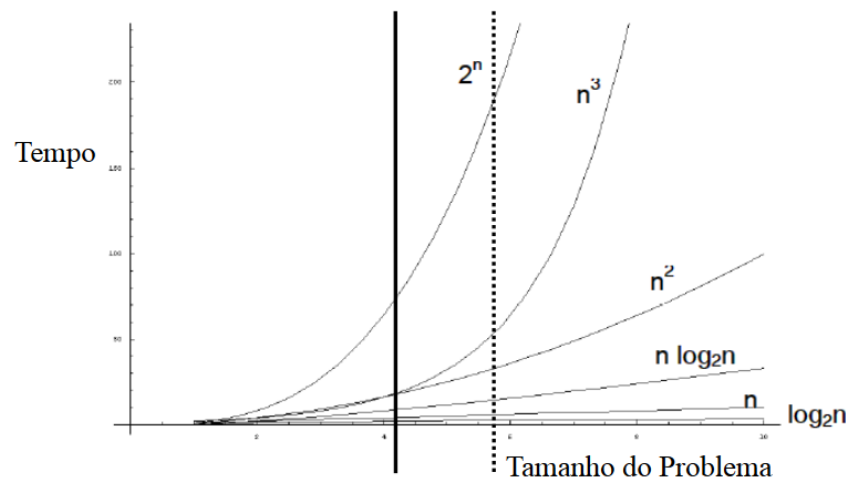


- Analisar algoritmos, considerando todas as entradas possíveis, seria útil:
  - ✓ Desde que seja independente de ambientes de *hardware* e *software*;
  - ✓ Desde que seja obtido sem a execução do algoritmo.



### Conclusão

- A Análise de Algoritmos deve, sempre que possível, ser realizada utilizando-se ambos, o Método Experimental e o Método Analítico, para melhor avaliação, conclusão e tomada de decisões, sobre o Desempenho de um Algoritmo.



### Exemplo 1

- Experimentar algoritmos em Java, em função do Tempo de Execução:



# ECM306 – Tópicos Avançados em Estrutura de Dados

## Introdução à Análise de Algoritmos

### Exemplo 1

```
1 import java.util.concurrent.TimeUnit;
2 /* Programa para medir o tempo decorrido de um algoritmo em Java */
3 class MedeNanoseg
4 {
5     public static void main(String args[]) throws InterruptedException
6     {
7         long startTime = System.nanoTime(); // Captura inicial do "relogio"
8         /* Início do código a ser medido */
9         | // dorme por 5 segundos
10         TimeUnit.SECONDS.sleep(5);
11         /* Termina o código a ser medido */
12         long endTime = System.nanoTime(); // Captura final do "relogio"
13         long timeElapsed = endTime - startTime; // Diferença em nanosegundos
14         /* Apresenta valores */
15         System.out.println("Tempo de Execucao em Nanosegundos.: " + timeElapsed);
16         System.out.println("Tempo de Execucao em Microsegundos: " + timeElapsed / 1000);
17         System.out.println("Tempo de Execucao em Milisegundos.: " + timeElapsed / 1000 / 1000);
18         System.out.println("Tempo de Execucao em Segundos.....: " + timeElapsed / 1000 / 1000 / 1000);
19     }
20 }
```

### Exemplo 2

- Experimentar algoritmos em Java, em função do Número de Comparações e do Número de Operações Aritméticas executadas:



# ECM306 – Tópicos Avançados em Estrutura de Dados

## Introdução à Análise de Algoritmos

### Exemplo 2

```
1  /* Programa para medir a quantidade de comparações e operações aritméticas em Java */
2  class MedeAcoes
3  {
4      public static void main(String args[])
5      {
6          long compar = 0; // Inicia contador de comparacoes com 0
7          long aritOp = 0; // Inicia contador de operacoes aritmeticas com 0
8          /* Início do código a ser medido */
9          int i = 0, multip10 = 0;
10         do
11         {
12             compar = compar + 1; // incrementa contador da proxima comparacao do if()
13             if(i % 10 == 0)
14             {
15                 aritOp = aritOp + 1; // incrementa contador da proxima soma
16                 multip10 = multip10 + 1; // contador e multiplos de 10
17             }
18             aritOp = aritOp + 1; // incrementa contador da proxima soma
19             i = i + 1;
20             compar = compar + 1; // incrementa contador da proxima comparacao do while()
21         }while(i < 1000);
22         /* Termina o código a ser medido */
23         /* Apresenta valores */
24         System.out.println("Quantidade de Ciclos Executados.....: " + i);
25         System.out.println("Quantidade de Multiplos de 10.....: " + multip10);
26         System.out.println("Quantidade de Comparacoes.....: " + compar);
27         System.out.println("Quantidade de Operacoes Aritmeticas: " + aritOp);
28     }
29 }
```

# 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 01

FIM