

# Algoritmos e Linguagens de Programação

Fabio Galuppo, M.Sc.

<http://fabioaluppo.com>

<http://github.com/fabioaluppo>

fabioaluppo@acm.org

@FabioGaluppo

# O que é Algoritmo?

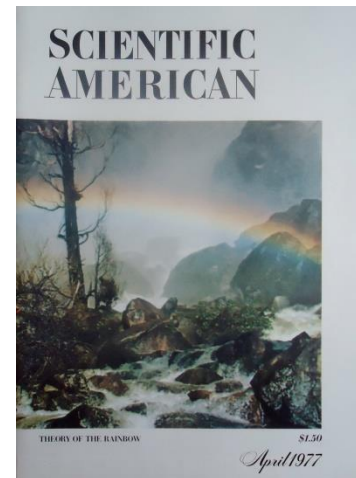
## Algorithms

*An algorithm is a set of rules for getting a specific output from a specific input. Each step must be so precisely defined it can be translated into computer language and executed by machine*

by Donald E. Knuth



An algorithm must be seen to be believed.  
(Donald Knuth)



<https://www.scientificamerican.com/magazine/sa/1977/04-01/#article-algorithms>

# O que é Algoritmo?

Google Translate

Text

Documents

Websites

DETECT LANGUAGE

ENGLISH

JAPANESE

PORTUGUESE



PORTUGUESE

JAPANESE

ENGLISH

An algorithm is a set of rules for getting a specific output from a specific input. Each step must be so precisely defined it can be translated into computer language and executed by machine

Um algoritmo é um conjunto de regras para obter uma saída específica de uma entrada específica. Cada etapa deve ser definida com tanta precisão que possa ser traduzida em linguagem de computador e executada por máquina

## Donald E. Knuth

[https://amturing.acm.org/award\\_winners/knuth\\_1013846.cfm](https://amturing.acm.org/award_winners/knuth_1013846.cfm)



**DONALD ("DON") ERVIN KNUTH**



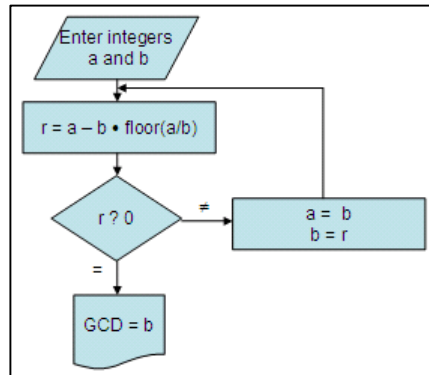
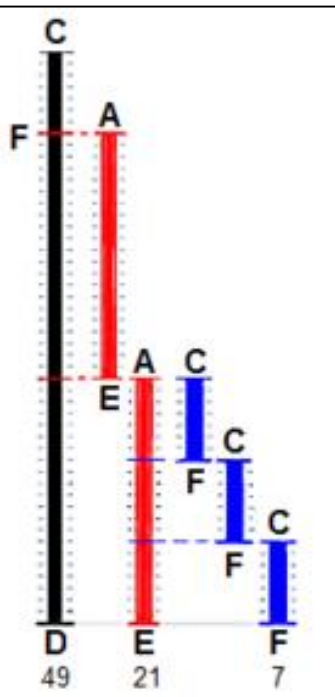
United States – 1974

### CITATION

For his major contributions to the analysis of algorithms and the design of programming languages, and in particular for his contributions to the "art of computer programming" through his well-known books in a continuous series by this title.

# Algoritmos e Linguagens de Programação

- Algoritmo é um conceito mental que existe independentemente de qualquer representação



$$\begin{aligned}7854 &= 1 \cdot 4746 + 3108 \\4746 &= 1 \cdot 3108 + 1638 \\3108 &= 1 \cdot 1638 + 1470 \\1638 &= 1 \cdot 1470 + 168 \\1470 &= 8 \cdot 168 + 126 \\168 &= 1 \cdot 126 + 42 \\126 &= 3 \cdot 42 + 0.\end{aligned}$$

$$\text{GCD}(a, b) = \begin{cases} a & \text{if } b=0 \\ b & \text{if } a=0 \\ \text{GCD}(b, a \bmod b) & \text{otherwise} \end{cases}$$

```
1: procedure EUCLID(a, b)
2:   r ← a mod b
3:   while r ≠ 0 do
4:     a ← b
5:     b ← r
6:     r ← a mod b
7:   end while
8:   return b
9: end procedure
```

# Algoritmos e Linguagens de Programação

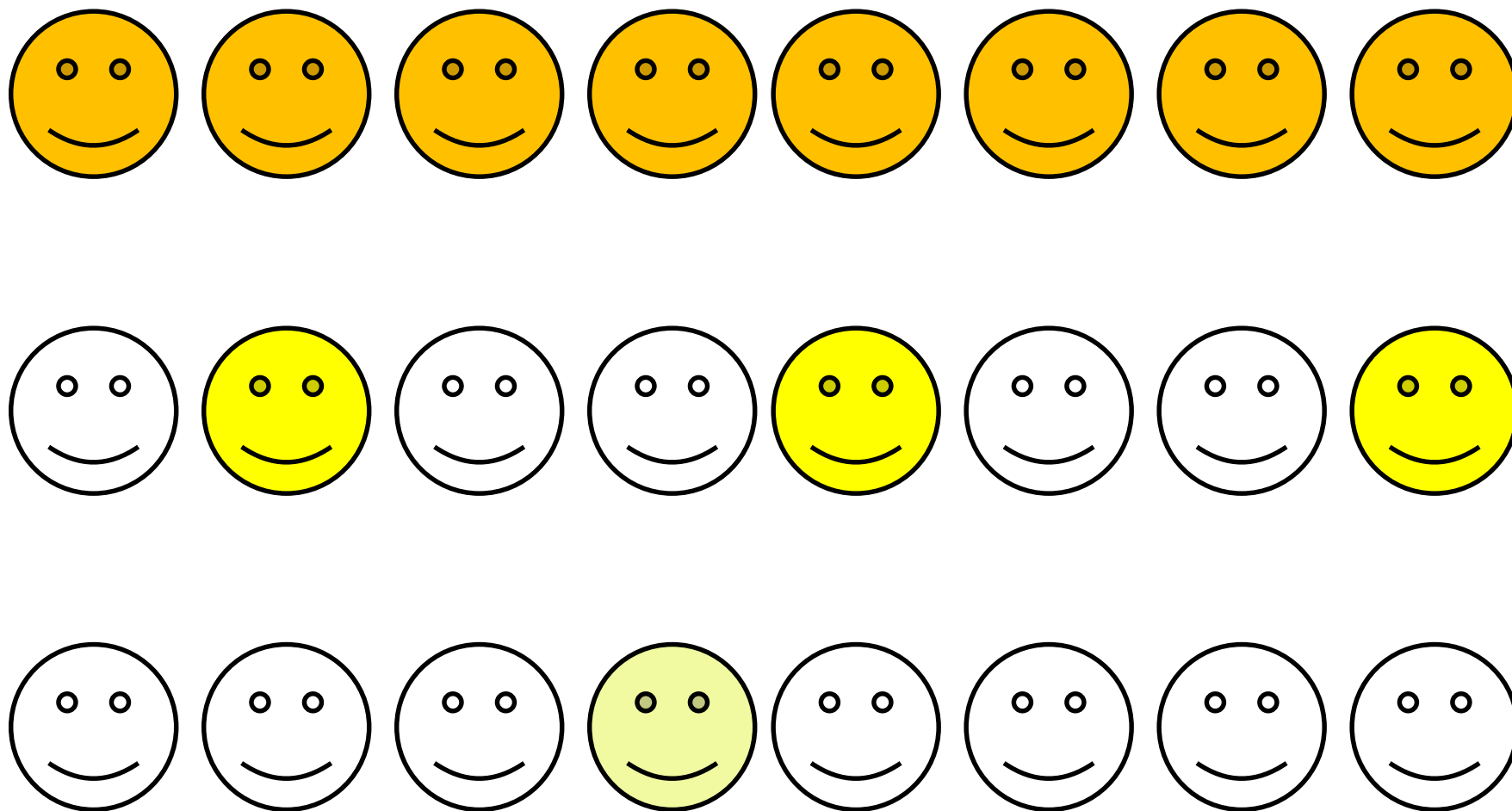
- Greatest Common Divisor (GCD) in C++:

```
template <typename T>
T gcd_iterative(T a, T b)
{
    T temp = a % b;
    while (!(temp == T(0)))
    {
        a = b;
        b = temp;
        temp = a % b;
    }
    return b;
}
```

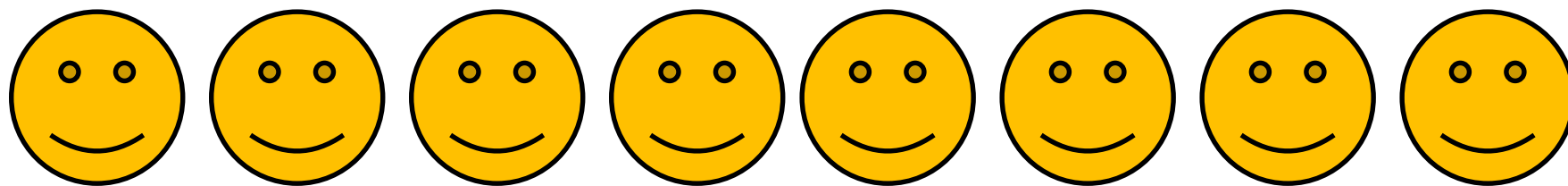
```
template <typename T>
T gcd_recursive(T a, T b)
{
    if (b == T(0)) return a;
    if (a == T(0)) return b;
    return gcd_recursive(b, a % b);
}
```

- Você utiliza um código para dizer ao computador o que ele deve fazer. No entanto, antes você precisa elaborar um algoritmo

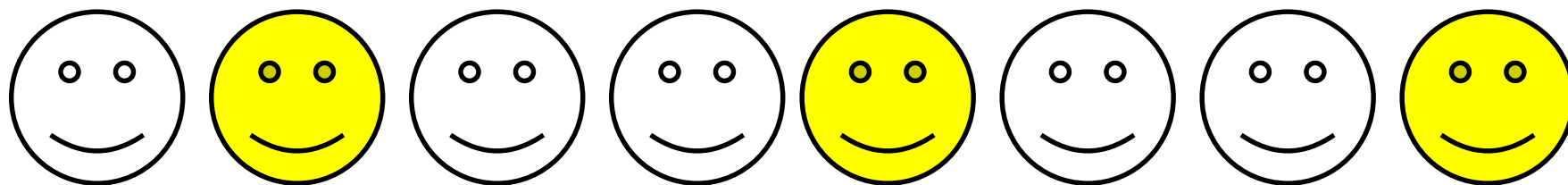
# Sobre esforço computacional



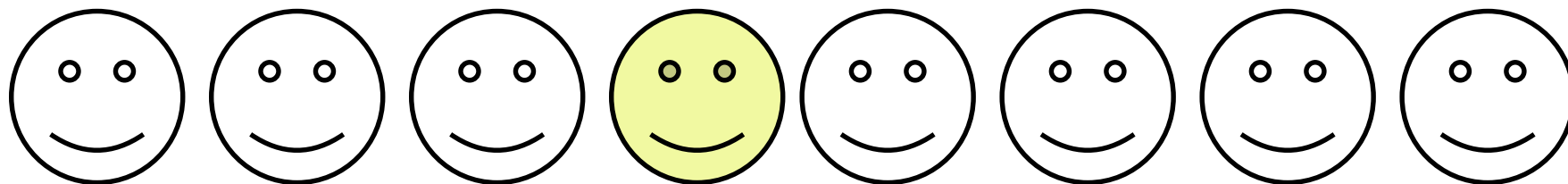
# Sobre esforço computacional



$$N = 8 \rightarrow \textit{Contagem} = 8$$



$$N = 8 \rightarrow \textit{Contagem} = 3 \qquad 3 = \log_2 8$$



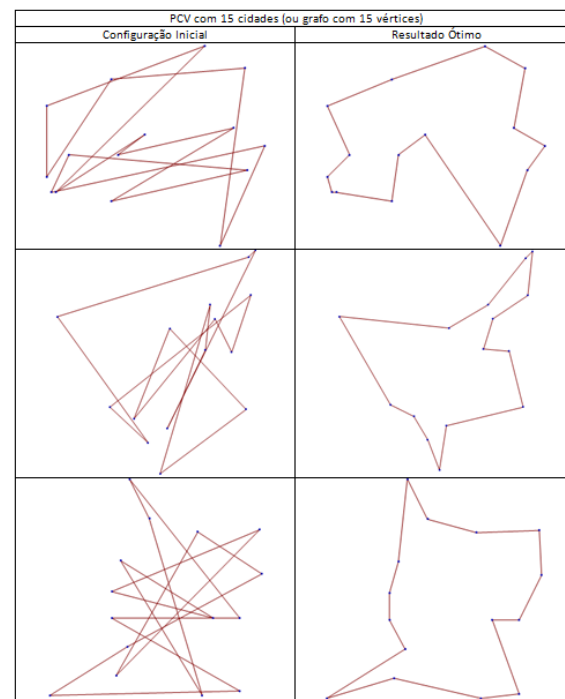
$$N = 8 \rightarrow \textit{Contagem} = 1$$

# Sobre eficiência e utilidade

- Um algoritmo é plenamente útil se e somente se for eficiente!

	Tempo (ms)
Número de Cidades	Força Bruta
13	743691
12	53093
11	4056
10	331
9	39
8	2
7	1

PCV com 15 Cidades		
	Força Bruta	
Solução Ótima	Tempo (ms)	Tempo (h)
359,399	165340592	45,928
317,232	165590540	45,997
368,79	165517424	45,977



15!

 **WolframAlpha** computational intelligence.

Result:

1 307 674 368 000

Number name:

1 trillion 307 billion 674 million 368 thousand

$$47!/2 = 1,29311 \times 10^{59}$$

$$47! = 2,58623 \times 10^{59}$$

$$48! = 1,24139 \times 10^{60}$$

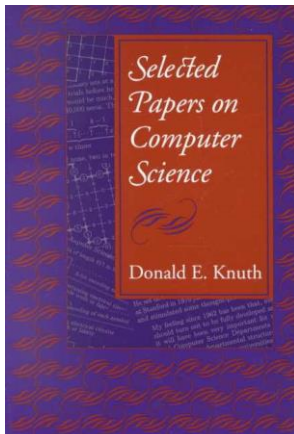


Algorithms

**SHOW ME THE C++ CODE**

# Algorithms

- 1: Searching a Computer's Memory
- 2: The Advantage of Order
- 3: Binary Tree Search
- 4: Hashing (linear probing)
- 5: Improving Unsuccessful Searches



---

**I find that I don't understand things unless I  
try to program them.**

---



**Donald E. Knuth**

Professor Emeritus at Stanford University

<https://www-cs-faculty.stanford.edu/~knuth/cs.html>

# Programação como uma arte

- Computer programming as an art
  - [https://amturing.acm.org/award\\_winners/knuth\\_1013846.cfm](https://amturing.acm.org/award_winners/knuth_1013846.cfm)
- Conclusão

Google Translate

Text Documents Websites

DETECT LANGUAGE ENGLISH JAPANESE PORTUGUESE



PORTUGUESE JAPANESE ENGLISH

We have seen that computer programming is an art, because it applies accumulated knowledge to the world, because it requires skill and ingenuity, and especially because it produces objects of beauty.

A programmer who subconsciously views himself as an artist will enjoy what he does and will do it better.

Therefore we can be glad that people who lecture at computer conferences speak about the state of the Art.



Vimos que a programação de computadores é uma arte, porque aplica o conhecimento acumulado ao mundo, porque requer habilidade e engenhosidade e, principalmente, porque produz objetos de beleza.

Um programador que subconscientemente se vê como um artista vai gostar do que faz e fará melhor.

Portanto, podemos ficar felizes que as pessoas que dão palestras em conferências de computador falam sobre o estado da arte.



[Computer programming as an art - 1974 Turing Award Lecture by Donald E. Knuth](https://amturing.acm.org/award_winners/knuth_1013846.cfm)

# Programas

- Eles são rodados pelo computador para executar tarefas específicas
  - Solucionar problemas

**Internet.** Web search, packet routing, distributed file sharing, ...

**Biology.** Human genome project, protein folding, ...

**Computers.** Circuit layout, file system, compilers, ...

**Computer graphics.** Movies, video games, virtual reality, ...

**Security.** Cell phones, e-commerce, voting machines, ...

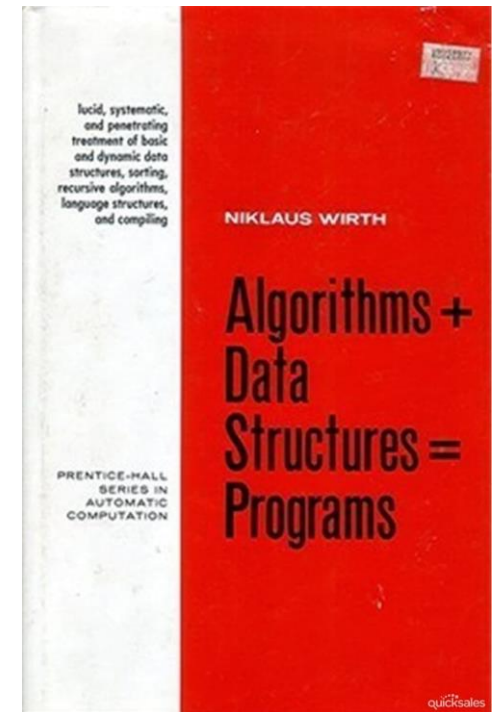
**Multimedia.** MP3, JPG, DivX, HDTV, face recognition, ...

**Social networks.** Recommendations, news feeds, advertisements, ...

**Physics.** N-body simulation, particle collision simulation, ...

⋮

- Algoritmos + Estrutura de Dados





# #include <algorithm>

Table 100 — Algorithms library summary

	Subclause	Header(s)
<a href="#">28.5</a>	Non-modifying sequence operations	
<a href="#">28.6</a>	Mutating sequence operations	<code>&lt;algorithm&gt;</code>
<a href="#">28.7</a>	Sorting and related operations	
<a href="#">28.8</a>	C library algorithms	<code>&lt;cstdlib&gt;</code>

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2017/n4713.pdf>

- Existem mais de 100 algoritmos padrões bem testados
- Serve como base para novos algoritmos
- Compreensível e mais simples do que um *raw loop*
- Mantém *side-effects* dentro de uma interface bem definida
- Facilita o raciocínio sobre o problema
- Atua em conjunto com *iterators* ou *left-closed interval* [begin, end)

# Algorithms

```
#include <algorithm>
```

- Searching a Computer's Memory  
`std::find`
- The Advantage of Order  
`std::binary_search` (`std::lower_bound`)
- Binary Tree Search  
`std::map` (Red-Black Tree → Balanced BST)
- Hashing  
`std::unordered_map`
  - Boost.Unordered
    - [https://www.boost.org/doc/libs/1\\_79\\_0/libs/unordered/doc/html/unordered.html](https://www.boost.org/doc/libs/1_79_0/libs/unordered/doc/html/unordered.html)
- Improving Unsuccessful Searches

# On the Shoulders of Giants



If I have seen further than others, it is by standing  
upon the shoulders of giants.

(Isaac Newton)

# Stepanov on the Shoulders of Knuth

- Generic Programming is about abstracting and classifying algorithms and data structures
- It gets its inspiration from Knuth

STL is only a limited success. While it became a widely used library, its central intuition did not get across. People confuse generic programming with using (and abusing) C++ templates. Generic programming is about abstracting and classifying algorithms and data structures. It gets its inspiration from Knuth and not from type theory. Its goal is the incremental construction of systematic catalogs of useful, efficient and abstract algorithms and data structures. Such an undertaking is still a dream.

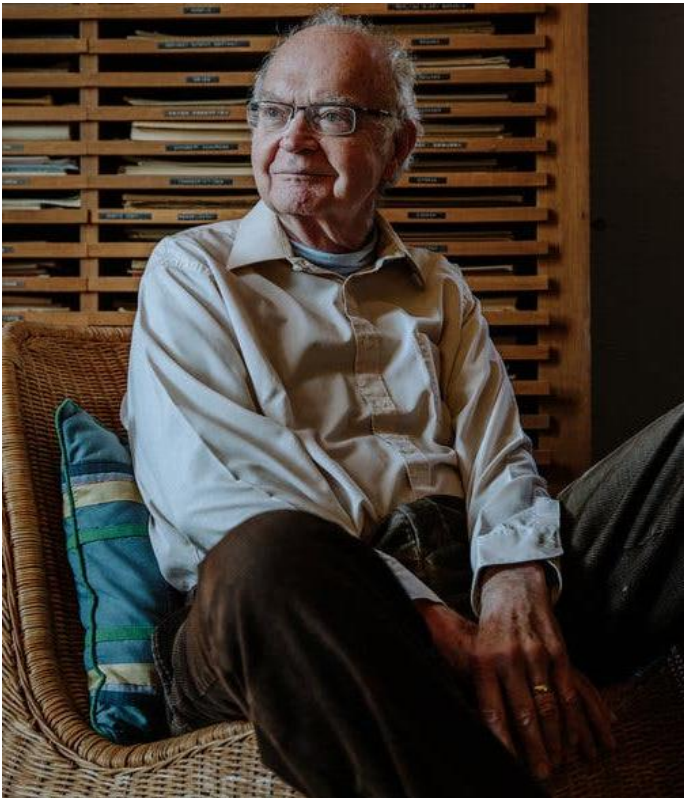
Short History of STL

<http://stepanovpapers.com/history%20of%20STL.pdf>



# On the Shoulders of Giants

Donald Knuth

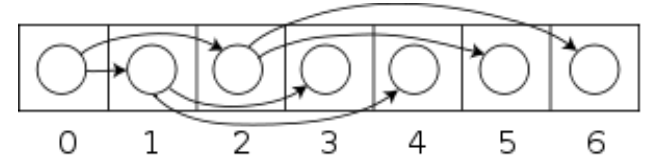
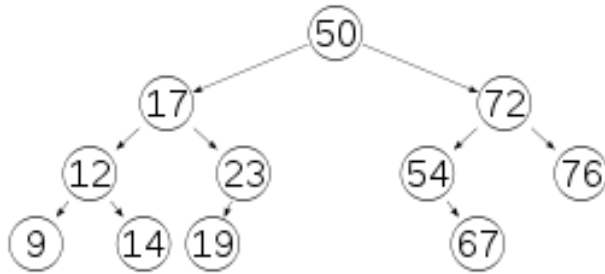


[https://en.wikipedia.org/wiki/Donald\\_Knuth](https://en.wikipedia.org/wiki/Donald_Knuth)

Alex Stepanov



[https://en.wikipedia.org/wiki/Alexander\\_Stepanov](https://en.wikipedia.org/wiki/Alexander_Stepanov)



# Algoritmos e Linguagens de Programação

## Recomendações para Leitura

Fabio Galuppo, M.Sc.

<http://fabioaluppo.com>

<http://github.com/fabioaluppo>


fabioaluppo@acm.org

@FabioGaluppo

# Recomendações

(Curso: Algoritmos com C++ em parceria com a Agit)

CursosEnsinoDesenvolvimentoQuem somosContatoBlog



## Algoritmos com C++

Aprenda um requisito fundamental para escrever programas sólidos e amplie suas opções no mercado de trabalho. Esteja apto a trabalhar em empresas de ponta, nas áreas de alta performance e concepção de soluções eficientes.

Empresas de diferentes portes utilizam Algoritmos com C++, alguns exemplos: Google, Microsoft, Facebook, Amazon, Apple, Spotify, Bolsas de Valores, Bloomberg, Eletronic Arts, Universidades como MIT, Stanford e Princeton.

[Inscreva-se](#)





Aulas Online - Ao vivo



24 horas de conteúdo

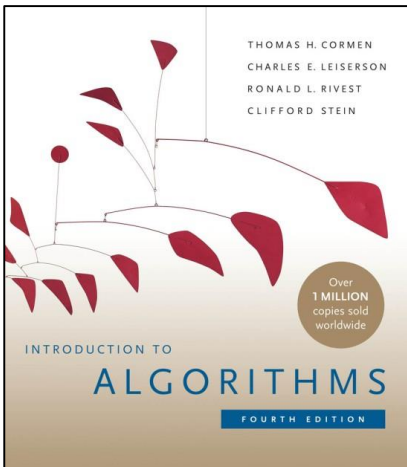
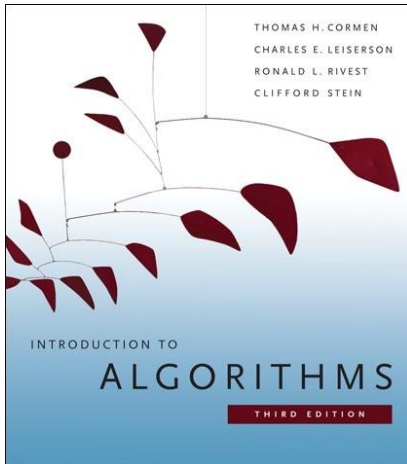


Certificado de participação

<https://www.agit.com.br/cursoalgoritmos.php>

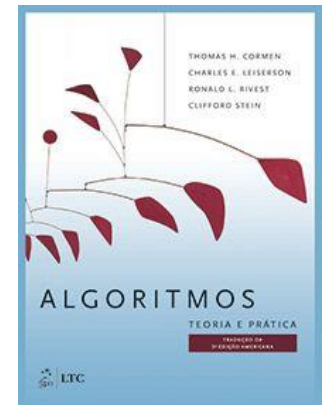
# Recomendações

(livros)



## Introduction to Algorithms, Third Edition Introduction to Algorithms, Fourth Edition

- Um clássico! Conhecido como livro do MIT, ou CLRS, ou livro do Cormen
- Ele aborda algoritmos, estrutura de dados e análise de algoritmos
- Pseudo-código
- 3a edição traduzida para língua portuguesa
- 4a edição lançada em 2022



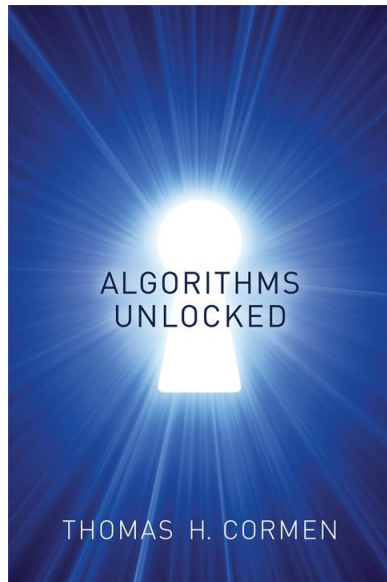
<https://mitpress.mit.edu/books/introduction-algorithms-third-edition>

<https://mitpress.mit.edu/books/introduction-algorithms-fourth-edition>

<https://www.grupogen.com.br/algoritmos>

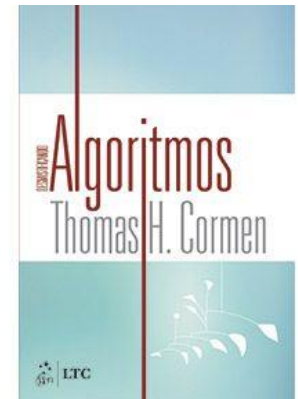
# Recomendações

## (livros)



### Algorithms Unlocked

- Uma “breve” introdução sobre algoritmos, estrutura de dados e análise de algoritmos
- Escrito por Thomas Cormen (C do CLRS)
- Pode ser considerado uma prelúdio ao livro do MIT
- Pseudo-código
- Tem traduzido em língua portuguesa
- Recomendo ler antes do livro do MIT



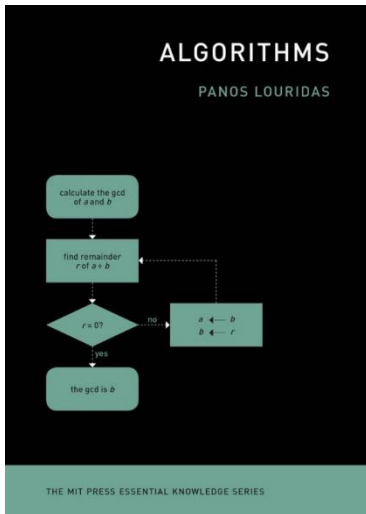
<https://mitpress.mit.edu/books/algorithms-unlocked>

<https://www.grupogen.com.br/e-book-desmistificando-algoritmos>

# Recomendações

## (livros)

### Algorithms

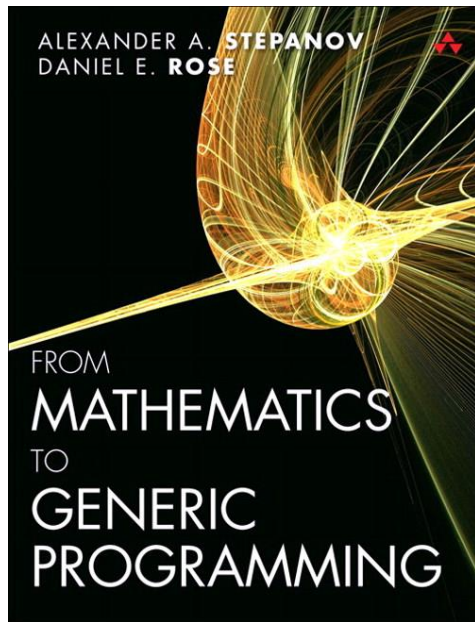


- Um *pocket book* da série The MIT Press Essential Knowledge
- Ele é sobre algoritmos e suas aplicações
  - Inclui tópicos sobre *Deep Learning* e *PageRank* (que deu origem ao Google)
- Praticamente um livro no estilo *popular science*
- Teoricamente, acessível para o público em geral. Isso não quer dizer que é um livro básico
- Logo no primeiro capítulo, ele trás exemplos emblemáticos que conectam música e algoritmos
- Não há código, mas tem diversas ilustrações como apoio

<https://mitpress.mit.edu/books/algorithms>

# Recomendações

(livros)



## From Mathematics to Generic Programming

- Livro do Alexander Stepanov, criador da STL
- Ele aborda algoritmos (em maior parte, os numéricos)
  - O *rotate* é um algoritmos que se destacam no livro
- Não aborda STL
  - Aborda o racional por trás, por exemplo: *Iterators*
- Aborda o que inspirou a Programação Genérica: Álgebra Abstrata e suas estrutura algébricas
- Muitos exemplos envolvendo Teoria dos Números
- Conecta fortemente Programação e Matemática
- Código em C++

<https://www.informit.com/store/from-mathematics-to-generic-programming-9780321942043>

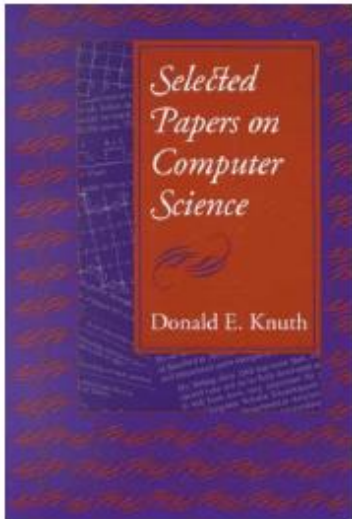
<https://www.fm2gp.com/>



# Recomendações

(livros)

## Selected Papers on Computer Science



- Livro do lendário Donald Knuth, *The Art of Computer Programming (TAOCP)*
- Seleção de *papers* e artigos sobre algoritmos, estrutura de dados e análise de algoritmos
- Segundo livro da coleção *Selected Papers* que contém 8 livros na coleção
- A reimpressão do artigo *Algorithms* que saiu na revista *Scientific American* em Abril de 1977 vale a compra deste livro!
  - Apesar da data, o artigo se tornou atemporal

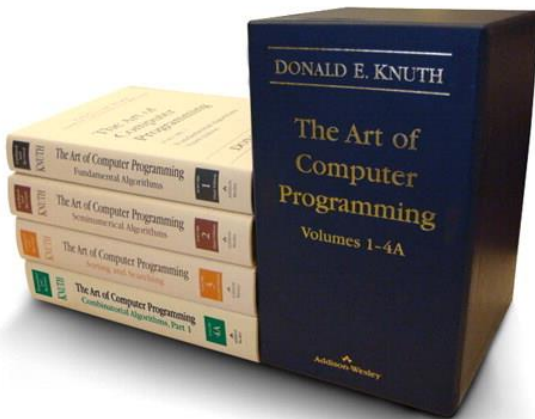
<https://www-cs-faculty.stanford.edu/~knuth/cs.html>



# Recomendação Especial

(The Art of Computer Programming)

## The Art of Computer Programming (TAOCP)



- Obra-prima do lendário Donald Knuth conhecido, entre outras coisas, por ter “criado” a disciplina Análise de Algoritmos
- Os livros abordam algoritmos, estrutura de dados e análise de algoritmos de uma forma rigorosa com muita “Matemática Concreta”. A bíblia do assunto! Obra ainda incompleta.
- Compre pelo menos para deixar na estante e impressionar seus amigos e familiares 😊

---

**I find that I don't understand things unless I  
try to program them.**

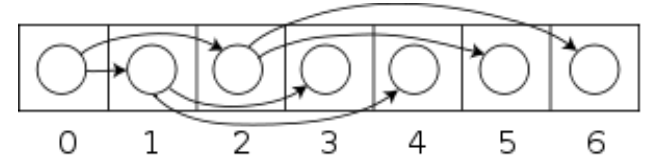
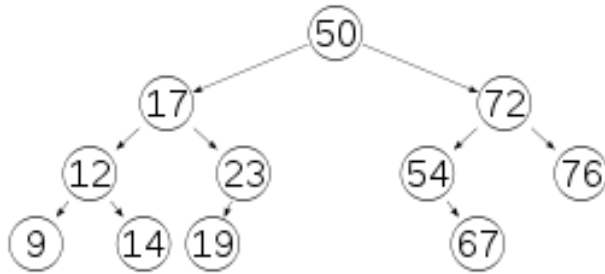
---



**Donald E. Knuth**  
Professor Emeritus at Stanford University

<https://www.informit.com/store/art-of-computer-programming-volumes-1-4a-boxed-set-9780321751041>

<https://www-cs-faculty.stanford.edu/~knuth/taocp.html>



# Algoritmos e Linguagens de Programação

Fabio Galuppo, M.Sc.

<http://fabiojaluppo.com>

<http://github.com/fabiojaluppo>

fabiojaluppo@acm.org

@FabioGaluppo