Universidade da Beira Interior

Programação Funcional

Simão Melo de Sousa

Aula 0 - Algoritmos? Estrutura de Dados?

Bem-vindos à UC de Programação Funcional

Programa para hoje

- Como funcionará esta UC?
- Algoritmia e Estrutura de Dados
- Algoritmos, para quê?

Programação Funcional

Programa e Funcionamento

do que vamos falar?

Já sabem programar, mas agora vamos falar

... de uma nova forma de programar

... de algoritmos, de estruturas de dados (AED)

... de formas de avaliar estes últimos

da gramática adquirida, poesia!

docentes:

- Simão Melo de Sousa - desousa@di.ubi.pt (regente)
- Hugo Torres Vieira - hugo.torres.vieira@ubi.pt
- André Pedro - andre.pedro@ubi.pt

site da UC : www.di.ubi.pt/~desousa/PF/pf.html

Microsoft Teams : **PROGRAMAÇÃO FUNCIONAL (20/21)**
este será o nosso **meio de contacto privilegiado**

práticas laboratoriais : na plataforma Learn_OCaml (site)

avaliação contínua : três problemas de programação P_1, P_2 e P_3 (por resolver no mooshak (site))

admissão a exame : pelo menos um problema aceite

exame : um problema de programação P_4, que substituirá a nota de um dos P_1, P_2 e P_3 que maximise a nota final

fraude = reprovação imediata + Não admissão à exame

esta UC organiza-se em torno de uma parte prática importante

as fichas práticas contêm **muitos exercícios**, bem mais dos que se espera que resolvam individualmente, organizados por 3 níveis de dificuldade

grosso modo, por ficha:

- aluno estreante: 2-3 exercícios nível básico, 1-2 exercícios nível intermédio
- aluno intermédio: 1-2 exercícios nível básico, 2-3 exercícios nível intermédio, 0-1 exercício nível consolidado
- aluno familiarizado: 0-1 exercício básico, 1-2 exercícios nível intermédio. 1-3 exercícios nível consolidado

claro, pode (deve ...) resolver mais!

muitos dos exercícios **contam histórias** sobre a programação funcional e sobre a algoritmia

histórias que não estão necessariamente referidas nas aulas teóricas

assim, não são um mero meio de treino, são **complementares** à matéria exposta nas teóricas!

não se iluda, **a responsabilidade** do bom aproveitamento **é sua**!
é ensino centrado em si

se não jogar o jogo, não aprende e não superará a avaliação

se tiver dificuldades, use o **MS-Teams** para contactar a equipa docente ou para colocar a dúvida aos colegas

conte com o apoio da equipa docente!

aprenda com os colegas, com os docentes
peça ajuda mas resolva sozinho, não copie!

Praticar, praticar, praticar... mas praticar sabiamente

a algoritmia, como a programação, domina-se praticando!

saber de um algoritmo e conhecer o algoritmo são coisas bem diferentes

há um perigo não menosprezável em usar um algoritmo ou uma estrutura de dados que não se domina!

não confie na minha exposição, não confie na sua leitura da documentação (etc.), tem de experimentar por si!

enquanto não programar um algoritmo não pode afirmar que o domina!

adquirir esta competência exige prática, empenho e resiliência,

mas está ao alcance de **todos vós**, sem excepção

a diferença de cada um estará no esforço pessoal a aplicar para esta caminhada

a diferença entre conhecer e ... o saber fazer

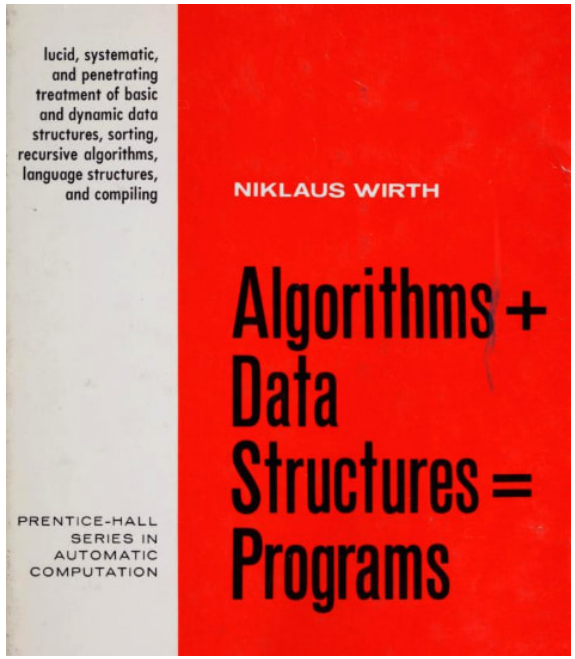
Algoritmia e Estrutura de Dados

A equação central

Programs = Algorithms + Data Structures

Niklaus Wirth

(Pai da linguagem Pascal)



Algoritmia e Estrutura de Dados

Estrutura de Dados

O que é uma estrutura de dados

a organização **racional, estruturada e adequada** da **informação** por ser processada ou arquivada pelos algoritmos e subsequentes programas

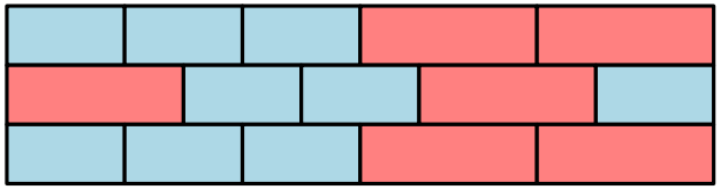
para resolver um **problema** com a ajuda de um computador, é **sempre importante** pensar qual é a melhor representação dos dados que o problema obriga a manipular

uma má estrutura de dados tornará a resolução muito mais dolorosa

uma boa estrutura de dados ajudará em muito a clareza, a simplicidade ou ainda a eficiência da resolução

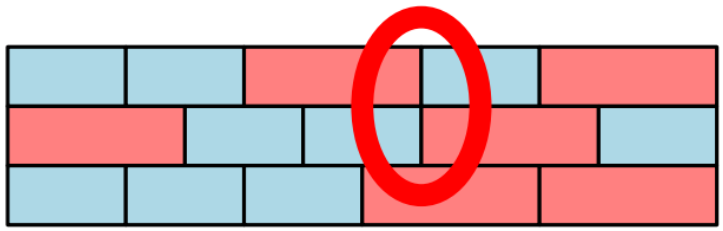
imagine que queiramos construir um muro com dois tipos de tijolos
os de comprimento 2 ( ) e os de comprimento 3 ( )

por exemplo, um muro de comprimento 12 e de altura 3 :



(courtesy of Jean-Christophe Filliâtre)

mas, dos muros que pretendemos construir, queremos **evitar absolutamente** as situações em que duas ou mais linhas de tijolos tenham **uma junção sobreposta**



estas situações fragilizam os muros construídos

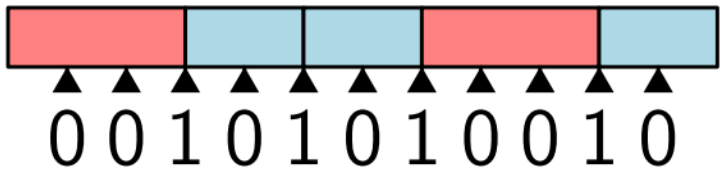
Problema : quantas formas há de construir um muro de comprimento 32 e de altura 10 ?

para resolver o problema por computador temos de propor uma **estrutura de dados** para representar os dados do problema e um **algoritmo** para calcular a solução

que estrutura de dados pode ser útil para representar a informação por processar e para facilitar à expressão do algoritmo ?

$\Longrightarrow$ representar uma linha por inteiros em base 2 em que o bit 1 representa uma junção

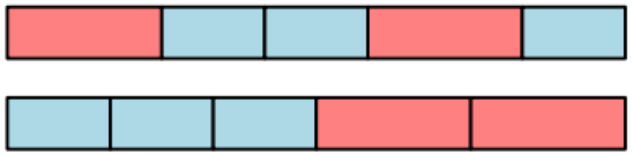
com a excepção das extremidades (porque não são úteis ao problema)



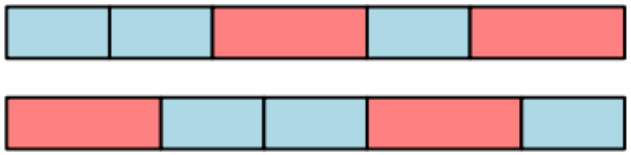
esta linha é representada pelo inteiro 338 ($= 00101010010_2$)

mas é uma boa ideia porquê ?

porque é assim muito fácil perceber se duas linhas são compatíveis a custa de uma simples operação de E lógico (o operador **land** em OCaml)



$$\begin{array}{rl} & 00101010010_2 \\ \texttt{land} & 01010100100_2 \\ = & 00000000000_2 = 0 \end{array}$$



$$\begin{array}{rl} & 01010010100_2 \\ \texttt{land} & 00101010010_2 \\ = & 00000\mathbf{1}0000_2 \neq 0 \end{array}$$

isto é um exemplo em que a estrutura de dados ajuda *crucialmente* o algoritmo

Algoritmia e Estrutura de Dados

Algoritmos

dado um número positivo S, para encontrar uma aproximação (até a j décima) da raiz quadrada de S procedemos da seguinte forma :

1. propor uma estimativa inicial : x_0
2. melhorar a estimativa : $x_1 = \frac{x_0 + \frac{S}{x_0}}{2}$
3. iterar até convergência, isto é : aplicar repetidamente $x_{n+1} = \frac{x_n + \frac{S}{x_n}}{2}$ até x_n e x_{n+1} coincidirem até a j-ésima decimal.
4. **x_{n+1} até a j-ésima décima é a aproximação de $\sqrt{S}$**

(método de Babilónia, um algoritmo com mais de 3000 anos!)

O que é um algoritmo?

1. guess x_0
2. repeat $x_{n+1} = \frac{x_n + \frac{S}{x_n}}{2}$ until convergence

é um processo geral para resolver um **problema**

funciona **para todas** as entradas admissíveis

um caso particular é uma **instância** do problema (e.g. $S = 49$, $j = 3$)

pode ser **executado cegamente**

é constituído por um número **finito de passos** elementares.

é **determinista**
(em contexto igual devolve sempre o mesmo resultado)

funciona sempre (é **correcto**)

termina sempre (é **finito**)
(preferencialmente em **tempo razoável**)

programa $\neq$ *algoritmo*

um programa é a tradução de um algoritmo (uma implementação) para uma forma que tome em conta todas as contingências do meio que o executará

por exemplo... a linguagem de programação, as estruturas de dados de suporte, o estilo, o paradigma...

não é menos nobre! Há todo um saber técnico exigido para fazê-lo com maestria

$$\begin{array}{rcccl} \mathit{average} & : & \mathbb{N} \times \mathbb{N} & \rightarrow & \mathbb{N} \\ & & (\mathit{mark}1, \mathit{mark}2) & \mapsto & \lfloor \frac{\mathit{mark}_1 + \mathit{mark}_2}{2} \rfloor \end{array}$$

versus

```
int average (int mark1, int mark2)
{
  return mark1 + (mark2 - mark1) / 2 ;
}
```

Algoritmia e Estrutura de Dados

Algoritmia, para quê?

Para quê estudar algoritmos e estruturas de dados?

para quê estudar AED?

em resumo, porque

- é **a fibra** de que é feito um verdadeiro informático
- resolve problemas e faz de si um melhor programador
- os AED estão em todo o lado, mesmo onde não parecem estar
- é divertido e é intelectualmente estimulante!
- lhe dará um melhor emprego

Porque os informáticos medem-se pela algoritmia

porque não há bom informático sem o domínio desta matéria!

um engenheiro informático não só sabe programar como **domina a programação**

sabe avaliar as **dimensões de um problema**

conhece as ferramentas algorítmicas adequadas, e **sabe construir** as suas

sabe **aplicar a melhor delas** e sabe **porquê**

$\Longrightarrow$ é o que o distingue do programador amador

porque a sociedade moderna é alimentada por software!

o programador assegura assim um papel vital no funcionamento da sociedade

se o software que produz falha ou não esta adaptado para o ambiente em que deve funcionar, há ***problemas***

ora, no **coração do software estão os algoritmos e as estruturas de dados**

chegará o tempo em que se juntará aos ministérios para a saúde, infra-estruturas, agricultura, educação (etc.) o ministério da sociedade digital

provavelmente com a partilha da secretaria de estado às catástrofes naturais

Nova empresa resolveu colocação de professores em seis dias. Base de dados foi a mesma. A partir da mesma base de dados do ministério que contém todos os docentes por colocar, a ATX Software criou um novo algoritmo, ou seja, uma solução informática, "pensado na íntegra durante seis dias e baseado em princípios matemáticos muito sólidos", afirmou ontem o engenheiro informático e autor da solução, Luís Andrade, durante uma conferência de imprensa, em Lisboa.

Sofia Rodrigues, Jornal Público - 30 de Setembro de 2004

para saber pensar algoritmicamente

não há matéria na informática em que os algoritmos e as estruturas de dados não tenham um papel importante

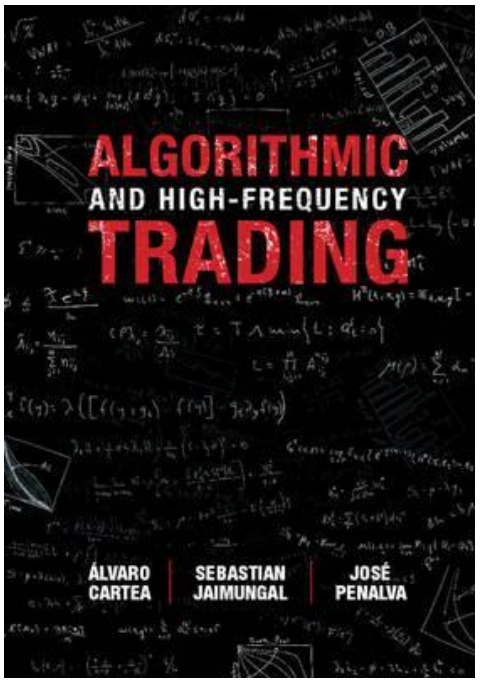
- as ciências informáticas, claro, mas também
- redes
- bases de dados
- segurança
- inteligência artificial
- ciência dos dados
- etc.

para saber pensar algoritmicamente

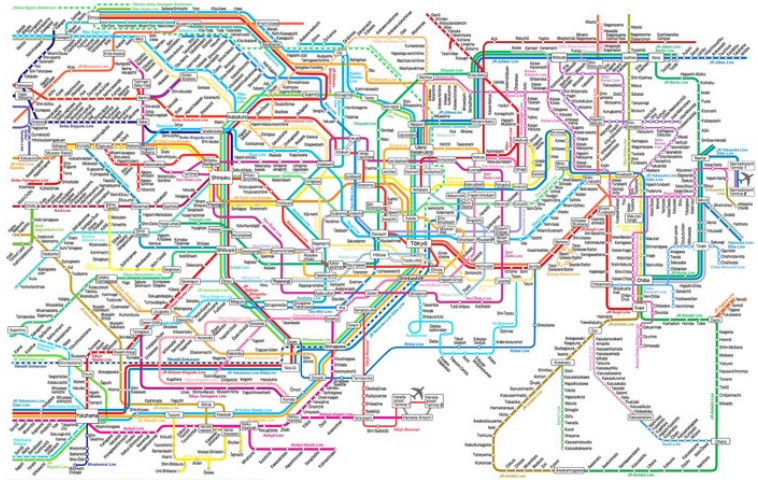
mesmo fora do contexto da programação

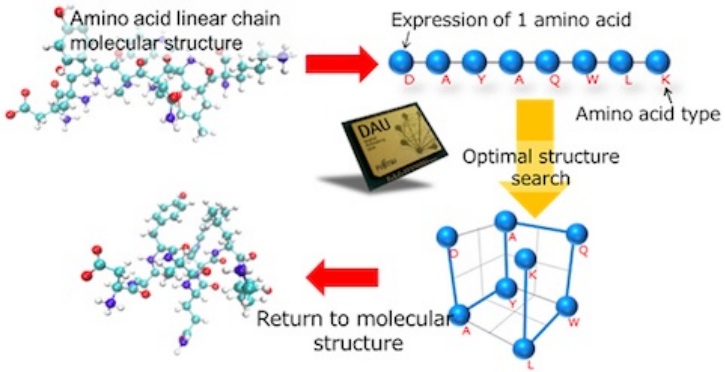
os algoritmos são vetores de inovação

- na finança
- na biologia
- na gestão
- na matemática
- na física
- no planeamento, etc.



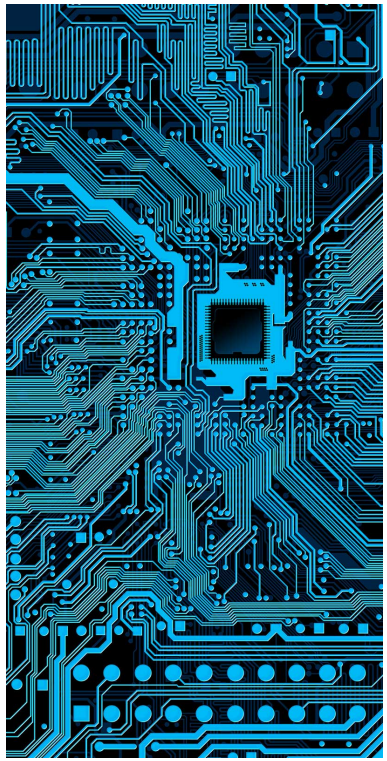
porque permite resolver problemas difíceis mas interessantes

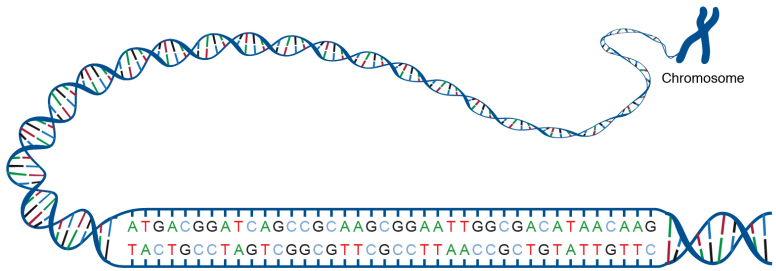




Foi no domingo de pascôa que se soibe em Leiria, que o pároco da Sè, José Miguéis, tinha morido de madrugade com uma apoplexia. O parroco era um homem sangíneo e nutrido, que pasava entre o clero diocesano pelo comillão dos comilões. Contavam-se hestórias singulares da sua vuracidade. O Carlos da Botica — que o detestava — costumava dizer, sempre que o via sair depois da sesta, com a face afogueada de sangue, muito enfartado:

— Lá vai a jibóia esmoer. Um dia estoura!





porque permite-lhe melhorar as suas capacidades de análise

permite-lhe antecipar dificuldades, ver problemas sob outros ângulos

não menospreze o poder do pensamento computacional

é arma secreta que lhe dá um poder de resolução incrível e insuspeitável

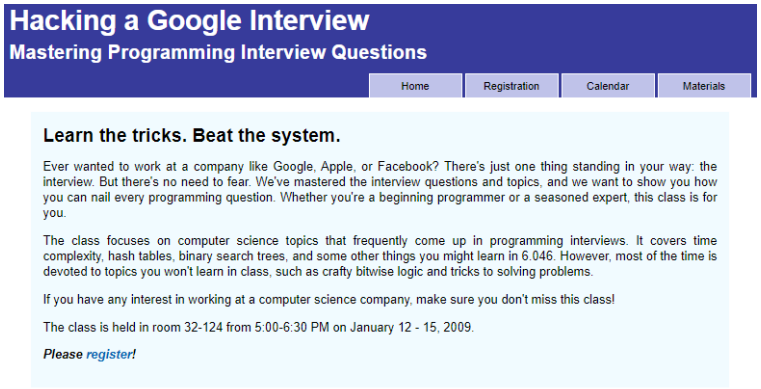
é a competência central que vos definirá como profissional e que pode resumir o vosso percurso na vossa formação académica

é o que o distinguirá dos demais

anedota ubiana

um ex aluno anónimo - o Henrique reconhecer-se-á - relatou-me que a empresa em que trabalhava tinha um problema de gestão de recurso para o qual esta não encontrava solução satisfatória. Até ele ter uma epifania e afirmar "há um grafo ali!" e propor um algoritmo salutar.
Brilhou e deu-me a conhecer como as aulas de algoritmia lhe permitiram tal feito.

Para um melhor emprego, para brilhar em entrevistas



(curso no MIT)

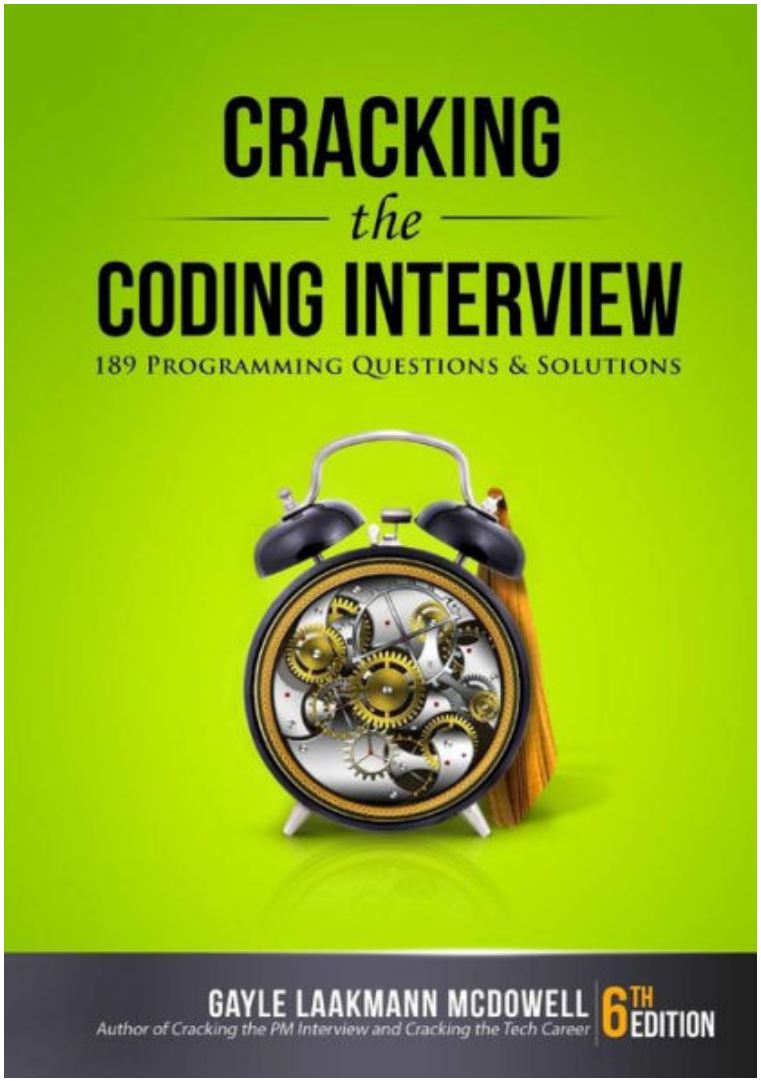
The 21 Best Coding Interview Courses and Books

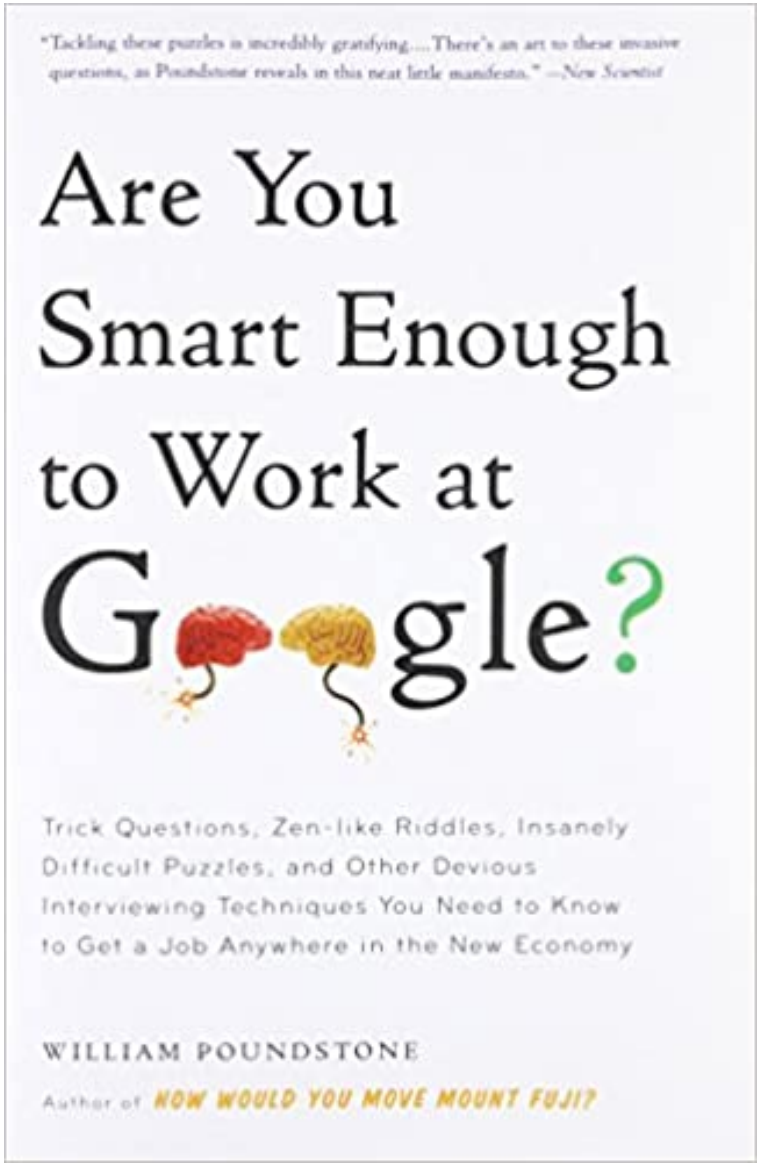
January 15th, 2021 | Get awesome (and free) stuff here

Tech interviews can be intimidating. They often have

37. Given string str, How do you find the longest palindromic substring in str? (solution)
38. How do you check if a string contains only digits? (solution)
39. How to remove Nth Node from the end of a linked list? (solution)
40. How to merge two sorted linked list? (solution)
41. How to convert a sorted list to a binary search tree? (solution)
42. How do you find duplicate characters in a given string? (solution)
43. How do you count a number of vowels and consonants in a given string? (solution)
44. How do you reverse words in a given sentence without using any library method? (solution)
45. How do you check if two strings are a rotation of each other? (solution)
46. How to convert a byte array to String? (solution)
47. How do you remove a given character from String? (solution)
48. How do you find the middle element of a singly linked list in one pass? (solution)
49. How do you check if a given linked list contains a cycle? How do you find the starting node of the cycle? (solution)
50. How do you reverse a linked list? (solution)

(site de treino para entrevistas)


CRACKING
the
CODING INTERVIEW
189 Programming Questions & Solutions
GAYLE LAAKMANN MCDOWELL
Author of Cracking the PM Interview and Cracking the Tech Career
6TH EDITION


"Tackling these puzzles is incredibly gratifying....There's an art to these invasive questions, as Poundstone reveals in this neat little manifesto." —New Scientist
Are You Smart Enough to Work at Google?
Trick Questions, Zen-like Riddles, Insanely Difficult Puzzles, and Other Devious Interviewing Techniques You Need to Know to Get a Job Anywhere in the New Economy
WILLIAM POUNDSTONE
Author of HOW WOULD YOU MOVE MOUNT FUJI?

Porque é uma atividade intelectualmente estimulante

o exercício da algoritmia raramente é assim

(logicomix)

... é mais assim

(logicomix)

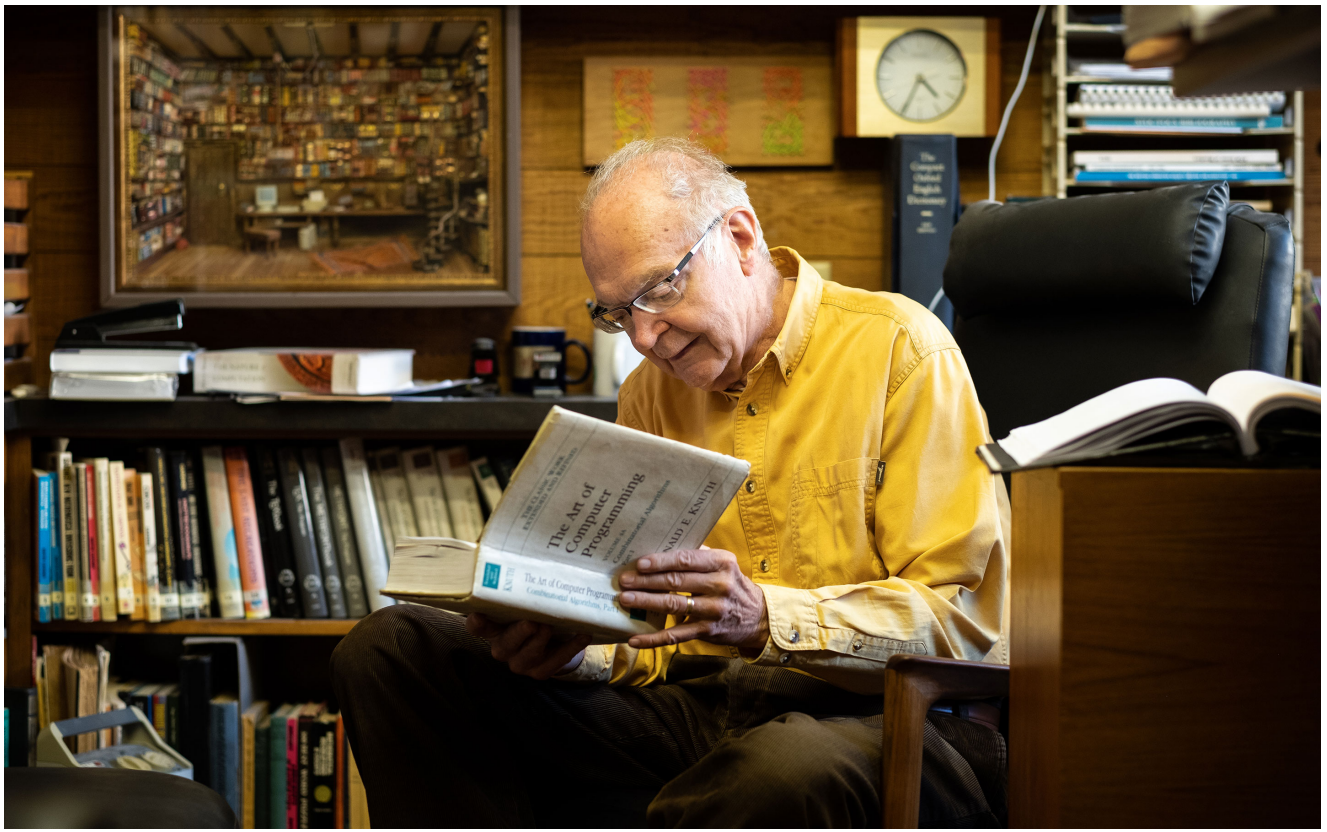
pratique, melhore, insista, não desista, tente novamente!

Computer programming is tremendous fun. Like music, it is a skill that derives from an unknown blend of innate talent and constant practice. Like drawing, it can be shaped to a variety of ends commercial, artistic, and pure entertainment. Programmers have a well-deserved reputation for working long hours, but are rarely credited with being driven by creative fevers. Programmers talk about software development on weekends, vacations, and over meals not because they lack imagination, but because their imagination reveals worlds that others cannot see.

Larry O'Brien and Bruce Eckel in Thinking in C#

Science is knowledge which we understand so well that we can teach it to a computer, and if we don't fully understand something, it is an art to deal with it. In this sense, we should continually be striving to transform every art into a science: in the process, we advance the art.

Donald Knuth



Conclusão. Quer saber mais?

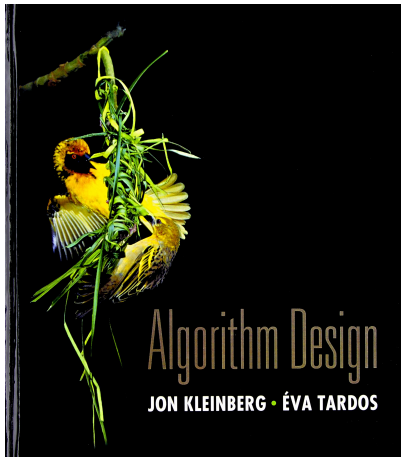
A matéria apresentada nesta aula baseia-se na fonte seguinte:

- **Apprendre à Programmer avec OCaml**
  Tradução para português disponível.

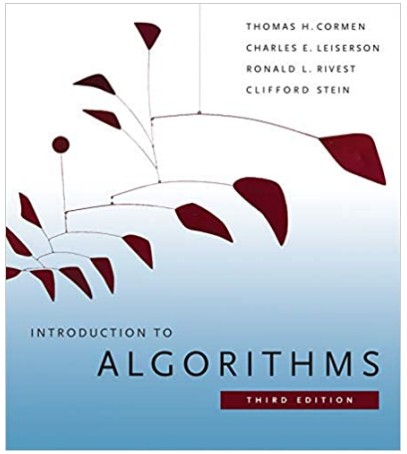
Leitura aconselhada

Adicionalmente ou alternativamente, as referências seguintes introduzem a matéria desta aula de forma completa e detalhada:

- **Algorithm Design** (site aqui)



- **Introduction to Algorithms** (site aqui)



Leitura aconselhada

Competitive programming 4, (site aqui)

(desafios de programação, com soluções em OCaml)