Teoria da Computação

2º semestre 2023/2024

Apresentação da cadeira

João Ribeiro

Alguns slides baseados em material de CS-251 “Great Ideas in Theoretical Computer Science” na Carnegie Mellon University

Equipa docente

Frederico Vicente

(práticas)

https://mr-vicente.github.io/

Vladyslav Mikytiv

(práticas)

Vasco Amaral

(práticas)

https://docentes.fct.unl.pt/vma/

João Ribeiro

(regente)

https://sites.google.com/site/joaorib94/

Olá!

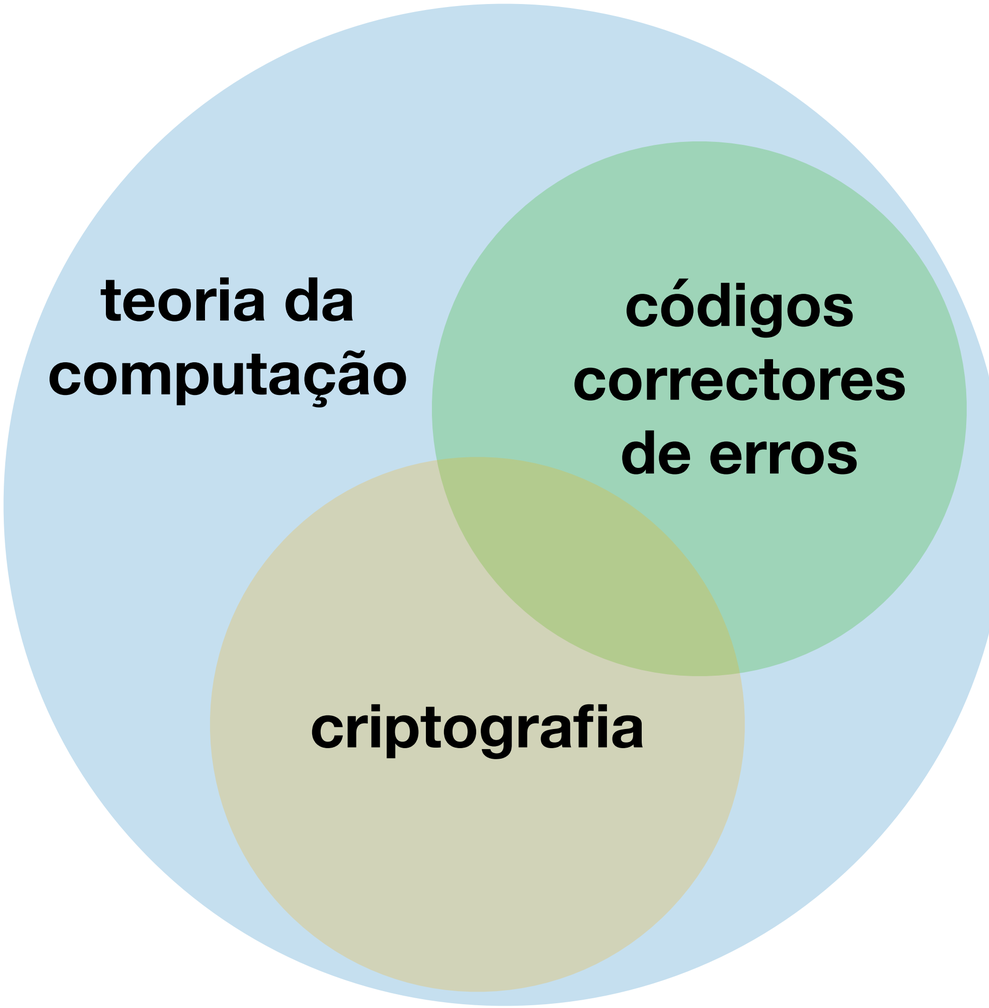
Carnegie Mellon University

Imperial College London

ETH zürich

TÉCNICO LISBOA

NOVA SCHOOL OF SCIENCE & TECHNOLOGY



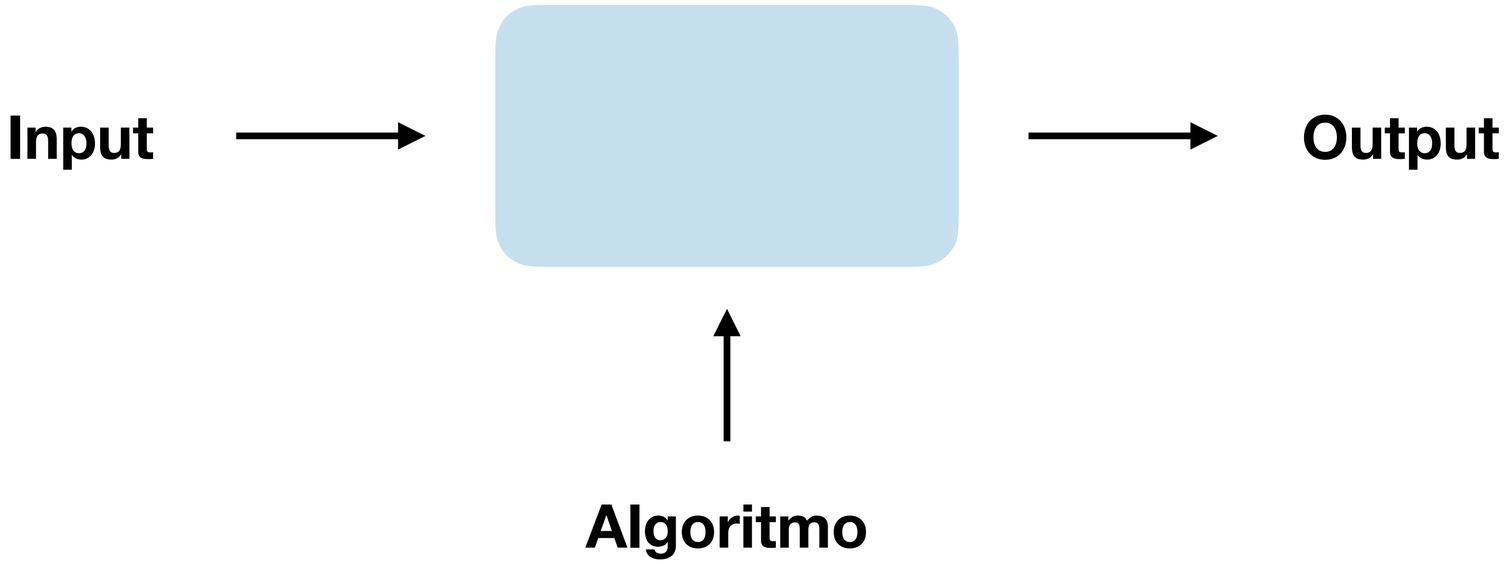
O que vamos estudar nesta cadeira?

*Quais as **capacidades** e **limites** fundamentais da computação?*

O que significa “computar”?

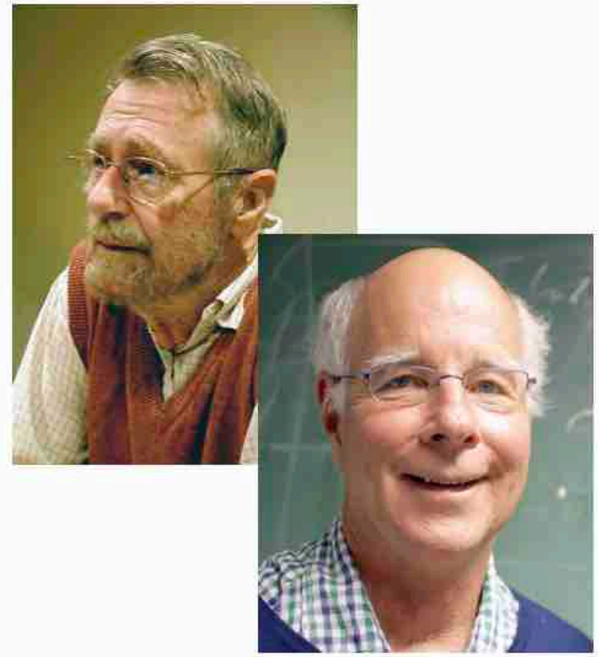
Computação = Manipulação de informação

Algoritmo = Descrição de como manipular informação



“Computer Science is no more about computers than astronomy is about telescopes.”

Edsger Dijkstra
(via Michael Fellows)



Algoritmos

Um conceito que existe há milhares de anos!

Começamos a usá-los quando ainda somos pequenos.

$$\begin{array}{rr} & 1234 \\ + & 6789 \\ \hline & \end{array}$$

Algoritmos

Um conceito que existe há milhares de anos!

Começamos a usá-los quando ainda somos pequenos.

$$\begin{array}{rr} & 1234 \\ + & 6789 \\ \hline & 3 \end{array}$$

Algoritmos

Um conceito que existe há milhares de anos!

Começamos a usá-los quando ainda somos pequenos.

$$\begin{array}{rr} & {\scriptstyle 1} \\ & 1234 \\ + & 6789 \\ \hline & 3 \end{array}$$

Algoritmos

Um conceito que existe há milhares de anos!

Começamos a usá-los quando ainda somos pequenos.

$$\begin{array}{rr} & 1234 \\ + & 6789 \\ \hline & 23 \end{array}$$

Algoritmos

Um conceito que existe há milhares de anos!

Começamos a usá-los quando ainda somos pequenos.

$$\begin{array}{rr} & {\scriptstyle 1} \\ & 1234 \\ + & 6789 \\ \hline & 23 \end{array}$$

Algoritmos

Um conceito que existe há milhares de anos!

Começamos a usá-los quando ainda somos pequenos.

$$\begin{array}{rr} & 1234 \\ + & 6789 \\ \hline & 023 \end{array}$$

Algoritmos

Um conceito que existe há milhares de anos!

Começamos a usá-los quando ainda somos pequenos.

$$\begin{array}{rr} & {\color{red}1} \\ & 1234 \\ + & 6789 \\ \hline & 023 \end{array}$$

Algoritmos

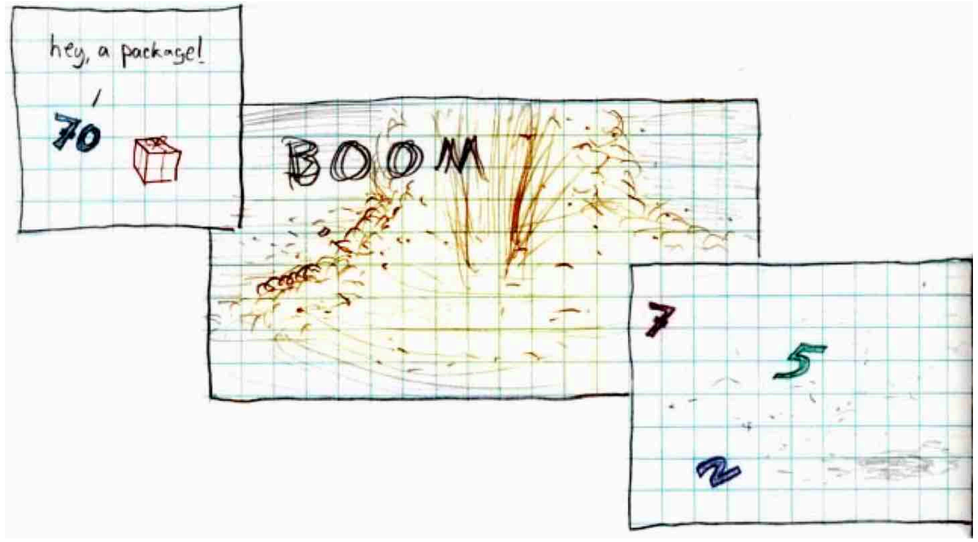
Um conceito que existe há milhares de anos!

Começamos a usá-los quando ainda somos pequenos.

$$\begin{array}{rr} & {\color{red}1} \\ & 1234 \\ + & 6789 \\ \hline & 8023 \end{array}$$

Outro exemplo: Decidir se um número é primo

Input: Um inteiro $n \geq 2$

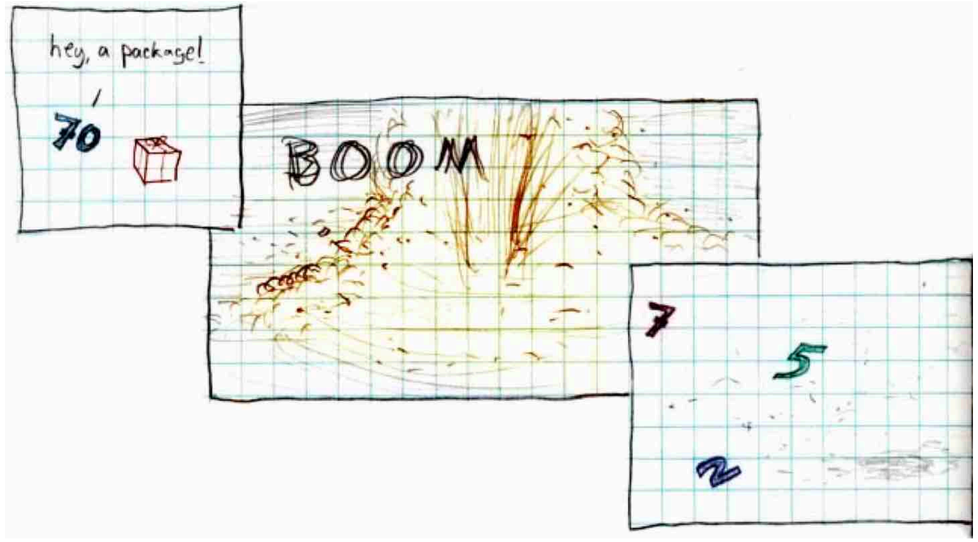


https://xkcd.com/5/

Outro exemplo: Decidir se um número é primo

Input: Um inteiro $n \geq 2$

Um algoritmo simples:



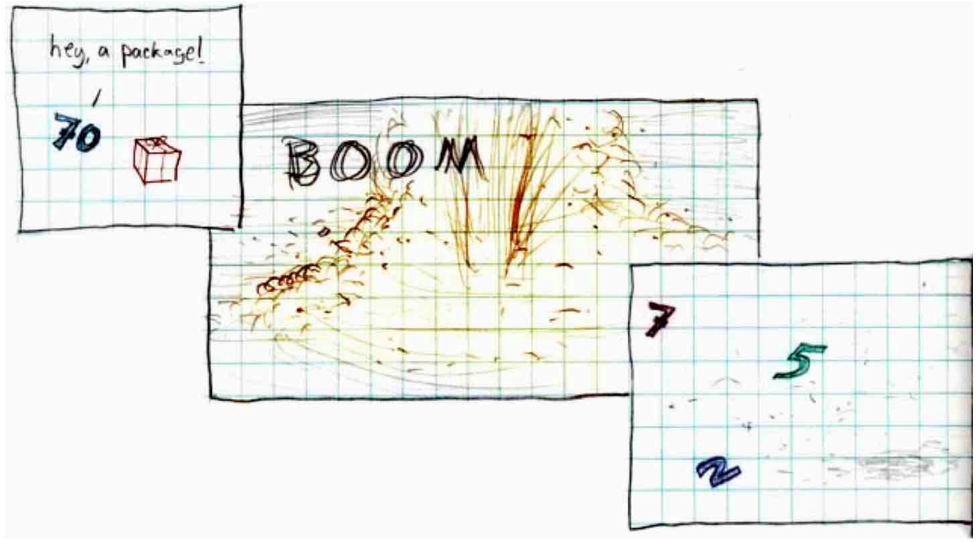
https://xkcd.com/5/

Outro exemplo: Decidir se um número é primo

Input: Um inteiro $n \geq 2$

Um algoritmo simples:

- Verificar se 2 divide n.



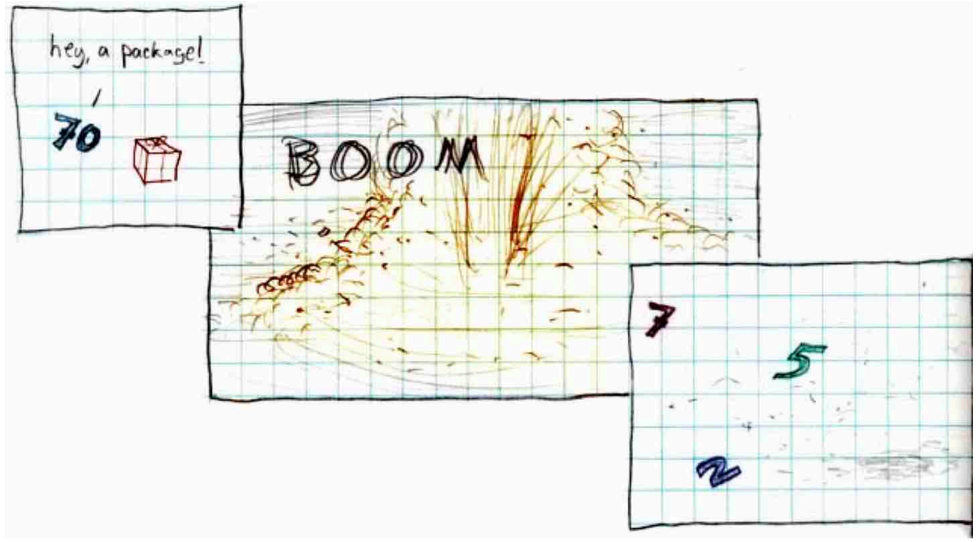
https://xkcd.com/5/

Outro exemplo: Decidir se um número é primo

Input: Um inteiro $n \geq 2$

Um algoritmo simples:

- Verificar se 2 divide n.
- Verificar se 3 divide n.



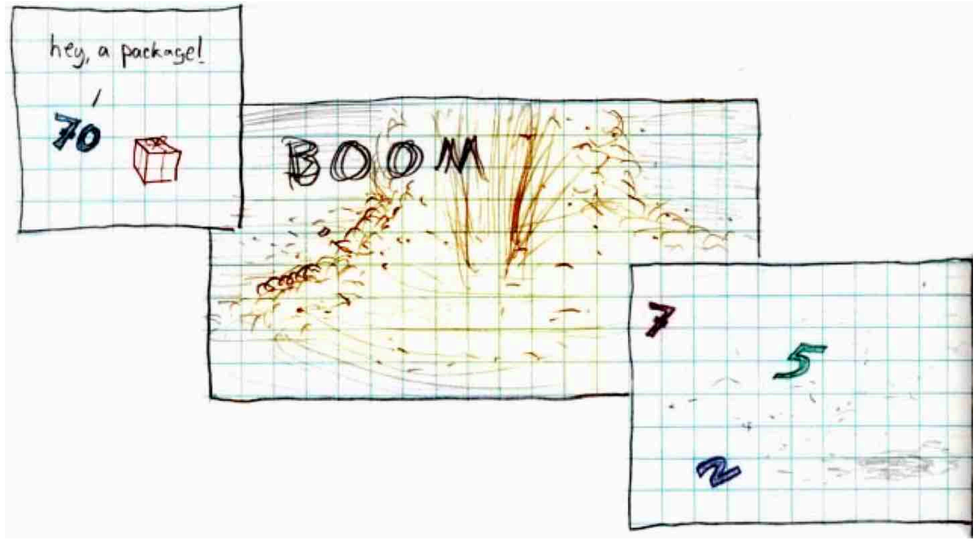
https://xkcd.com/5/

Outro exemplo: Decidir se um número é primo

Input: Um inteiro $n \geq 2$

Um algoritmo simples:

- Verificar se 2 divide n.
- Verificar se 3 divide n.
- …



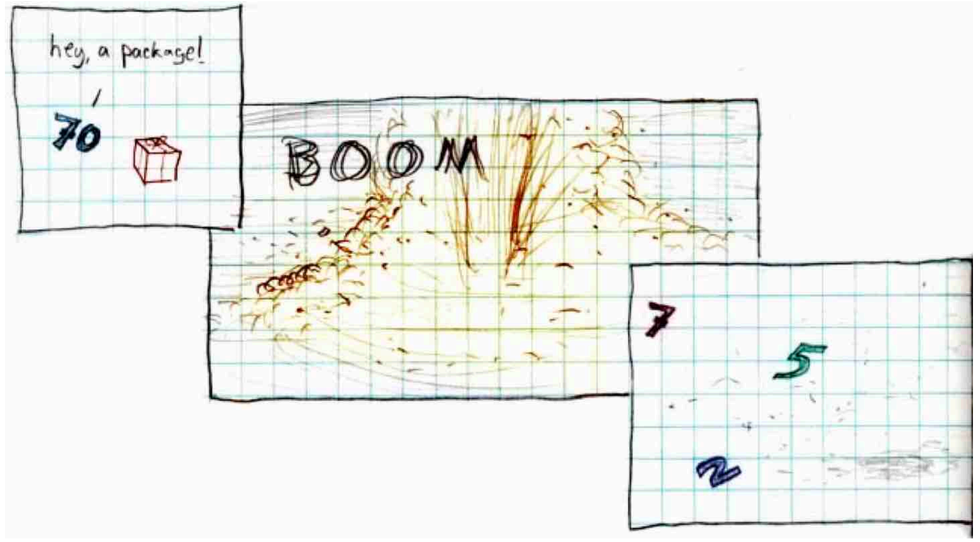
https://xkcd.com/5/

Outro exemplo: Decidir se um número é primo

Input: Um inteiro $n \geq 2$

Um algoritmo simples:

- Verificar se 2 divide n.
- Verificar se 3 divide n.
- …
- Verificar se $n - 1$ divide n.



https://xkcd.com/5/

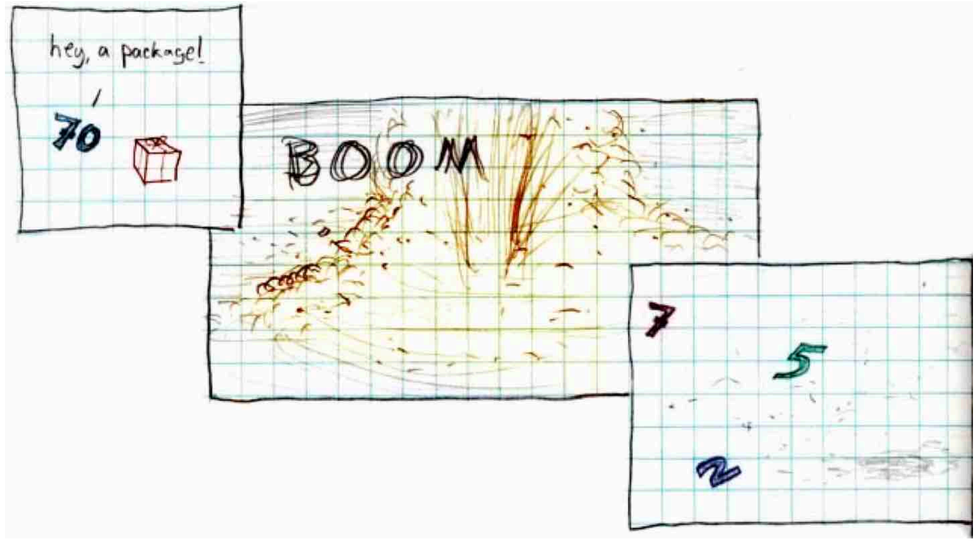
Outro exemplo: Decidir se um número é primo

Input: Um inteiro $n \geq 2$

Um algoritmo simples:

- Verificar se 2 divide n.
- Verificar se 3 divide n.
- …
- Verificar se $n - 1$ divide n.

Podemos melhorar este algoritmo?



https://xkcd.com/5/

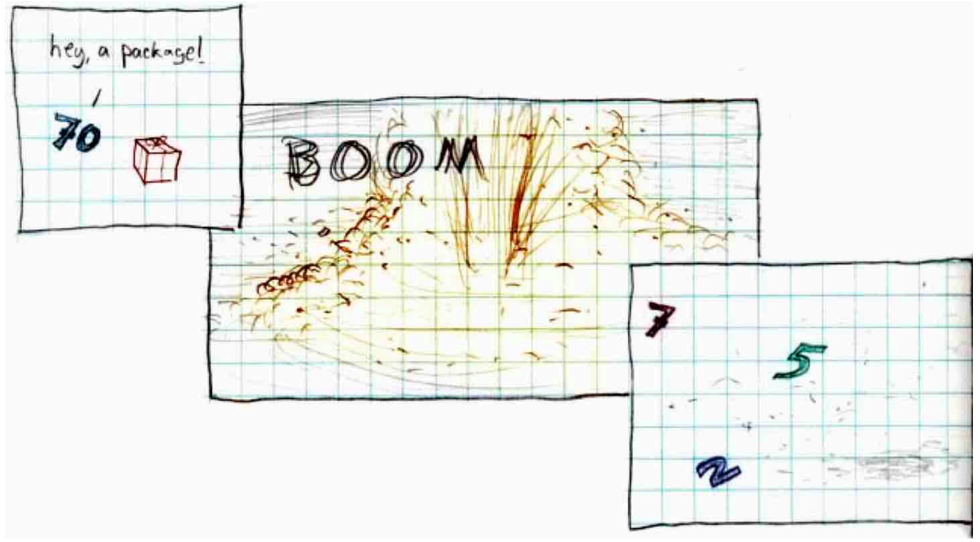
Outro exemplo: Decidir se um número é primo

Input: Um inteiro $n \geq 2$

Um algoritmo simples:

- Verificar se 2 divide n.
- Verificar se 3 divide n.
- …
- Verificar se $n - 1$ divide n.

Podemos melhorar este algoritmo?



https://xkcd.com/5/

só em 2004!

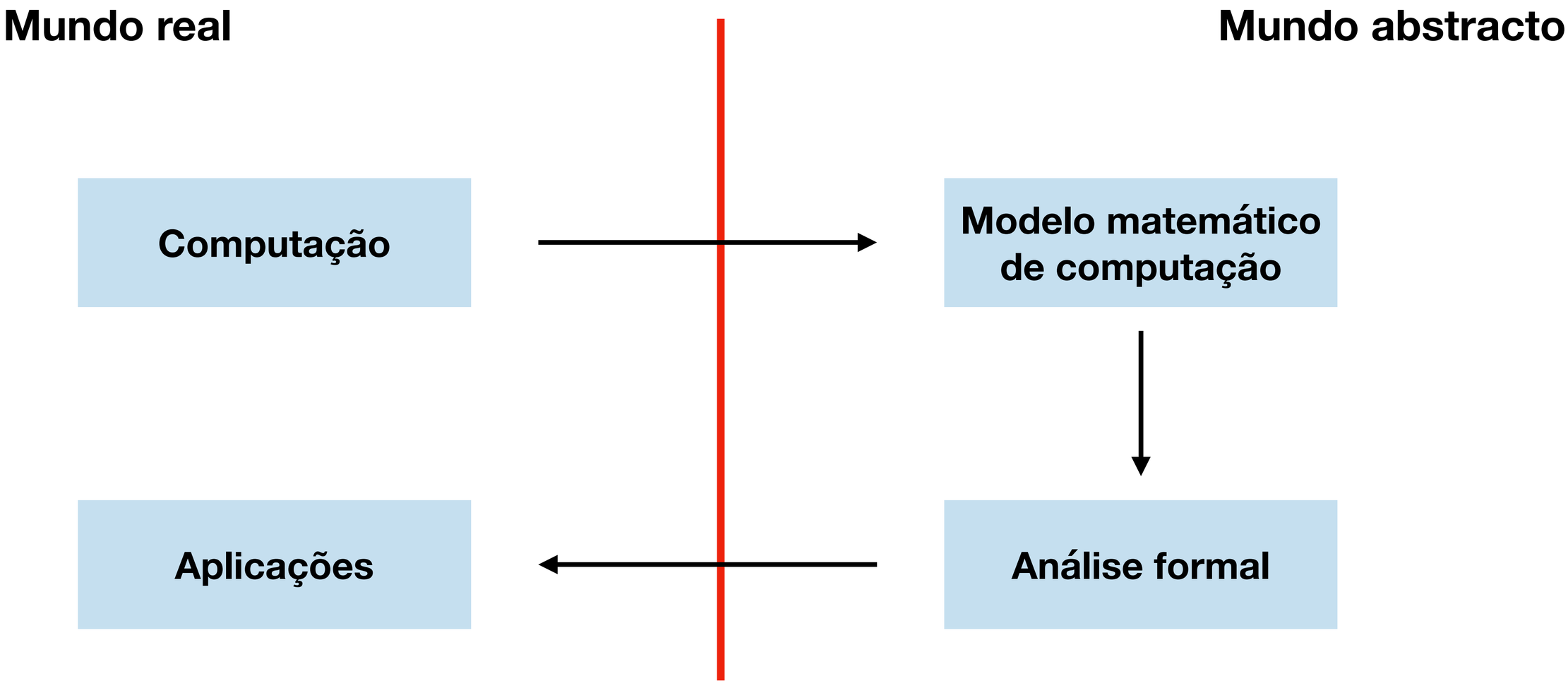
PRIMES is in P

By Manindra Agrawal, Neeraj Kayal, and Nitin Saxena*

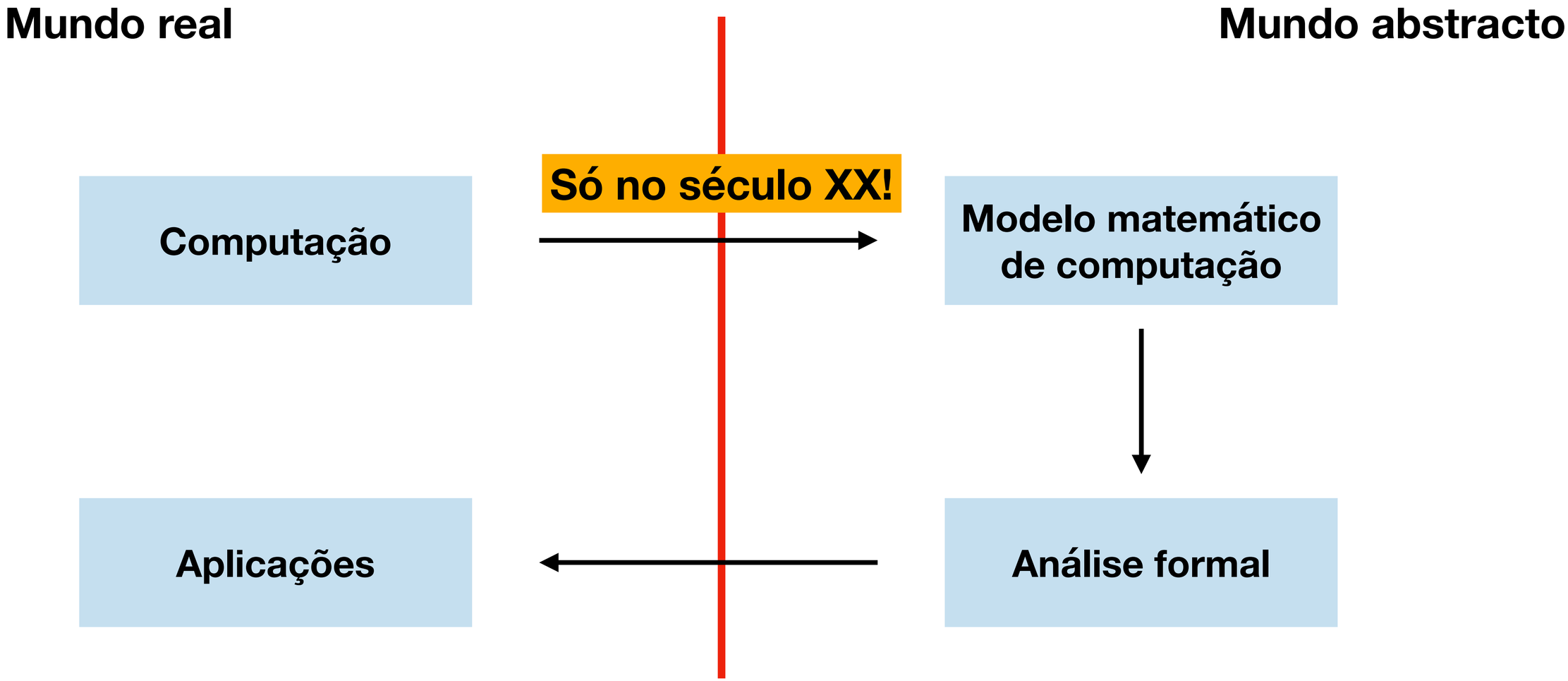
Abstract

We present an unconditional deterministic polynomial-time algorithm that determines whether an input number is prime or composite.

Como estudar a computação?



Como estudar a computação?



Como formalizar a noção de “algoritmo”?

As principais vertentes da teoria da computação

Computabilidade:

Existe algum algoritmo que resolva o problema?

Complexidade:

Existe algum algoritmo **eficiente** que resolva o problema?

As principais vertentes da teoria da computação

Computabilidade:

Existe algum algoritmo que resolva o problema?

Complexidade:

Existe algum algoritmo **eficiente** que resolva o problema?

tempo

memória

aleatoriedade

...

As principais vertentes da teoria da computação

Computabilidade:

Existe algum algoritmo que resolva o problema?

Complexidade:

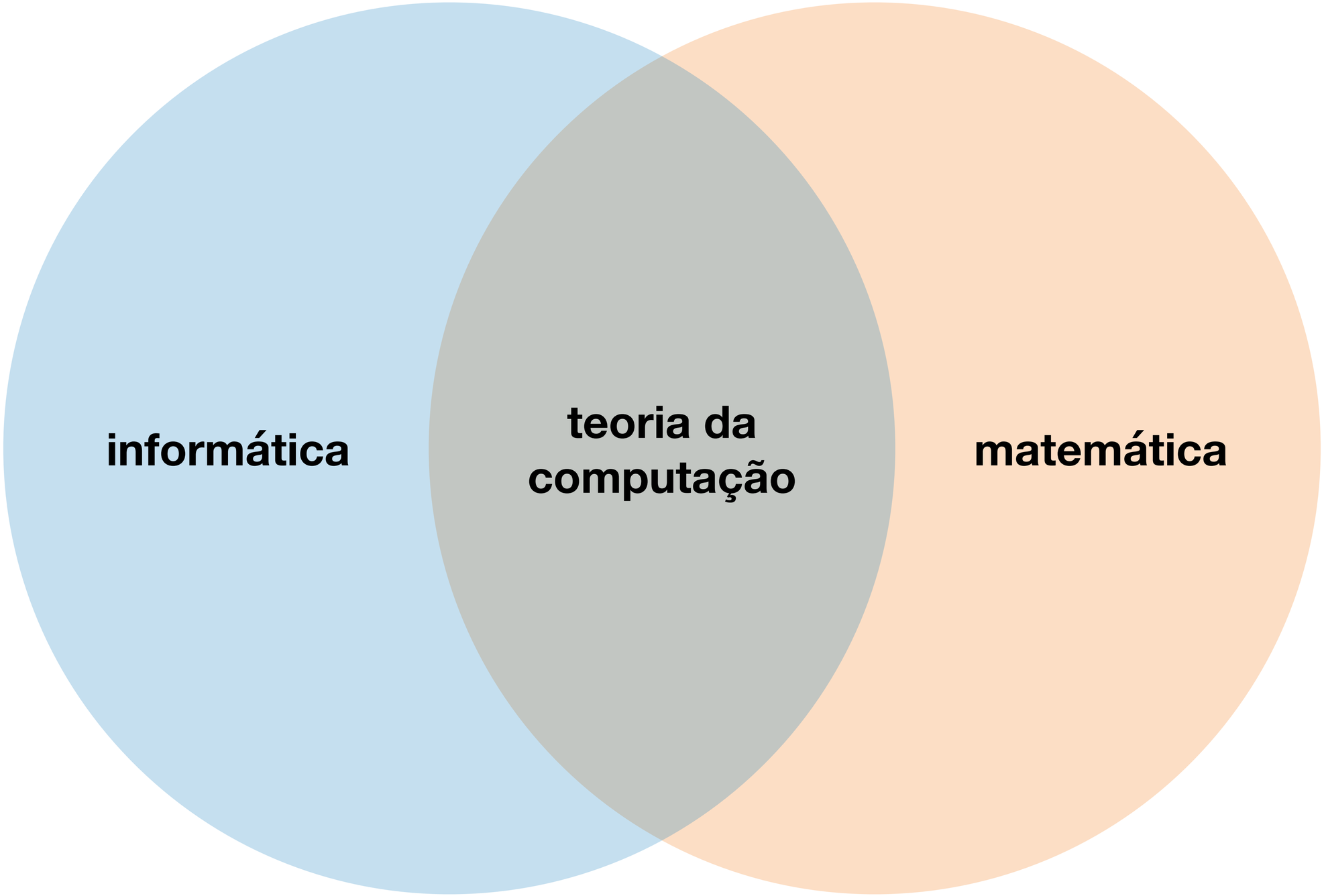
Existe algum algoritmo **eficiente** que resolva o problema?

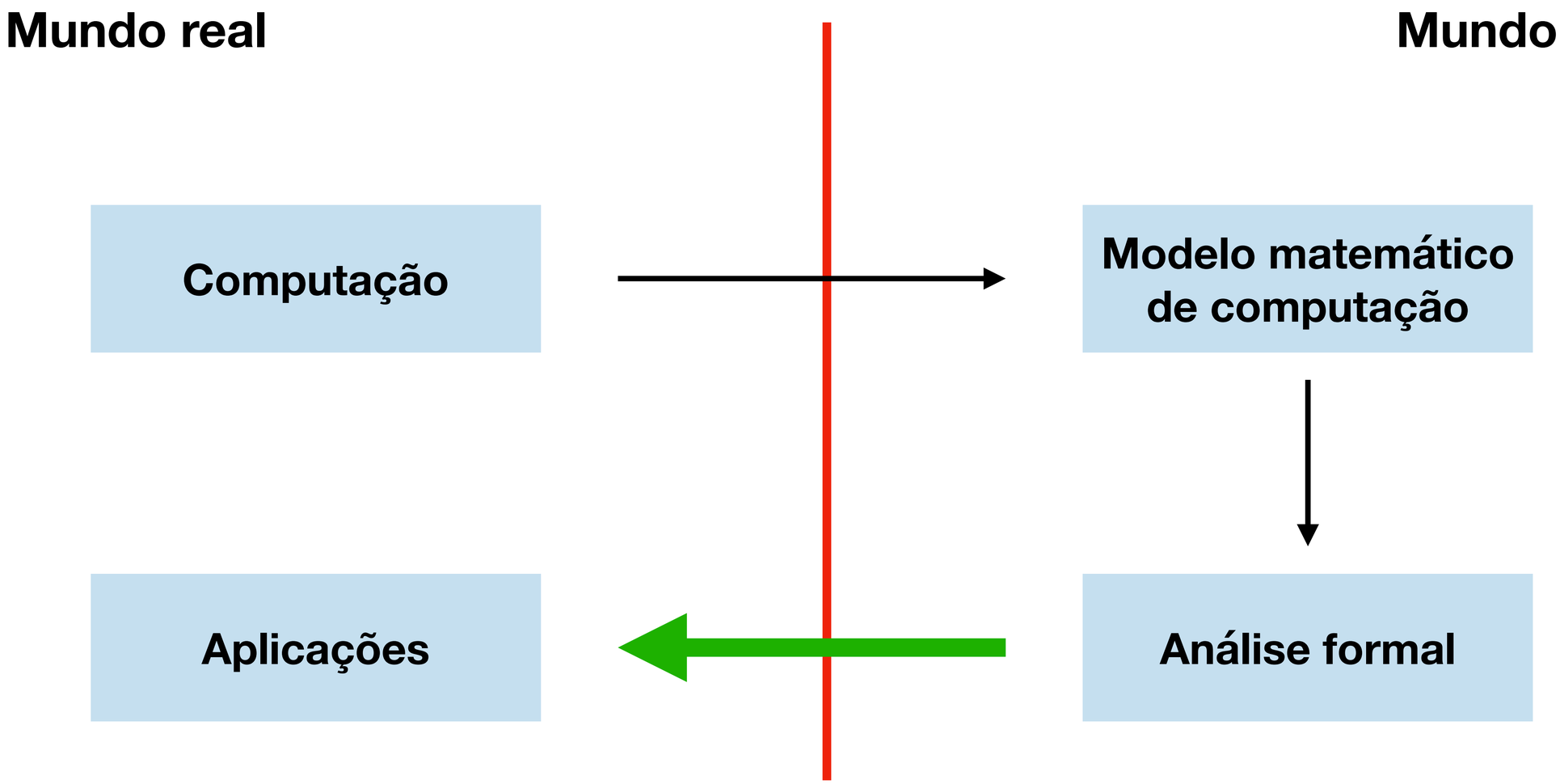
tempo

memória

aleatoriedade

...


informática
teoria da computação
matemática


Mundo real
Mundo abstracto
Computação
Modelo matemático de computação
Aplicações
Análise formal

Frutos da teoria da computação

Teoria da computabilidade/ autómatos

Compiladores

Design de hardware

Verificação de software

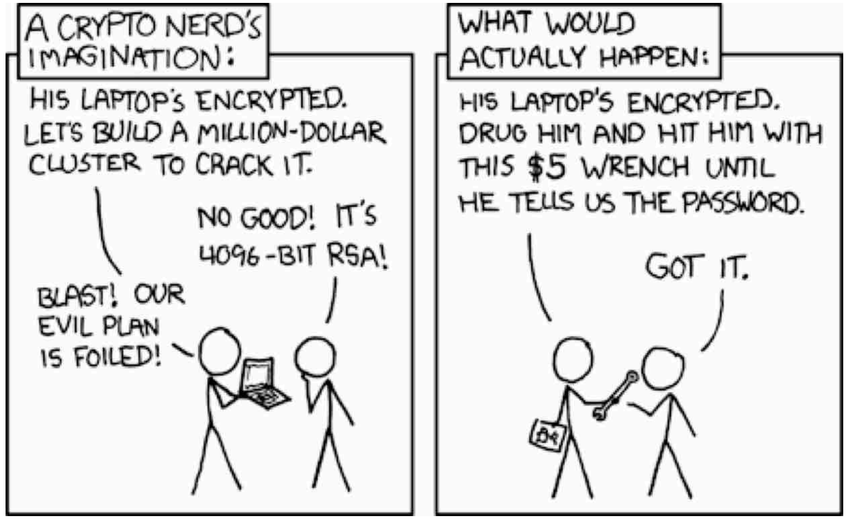
Inteligência artificial

Teoria da complexidade

Criptografia

P vs. NP: Um dos problemas em aberto mais importantes da actualidade!

https://www.claymath.org/millennium-problems/p-vs-np-problem



https://xkcd.com/538/

\+ ligações elegantes e surpreendentes a muitas outras áreas da matemática!

Objectivos desta cadeira, revamped

*Quais as **capacidades** e **limites** fundamentais da computação?*

1. Ensinar-vos a pensar rigorosamente sobre a computação.

2. Explorar questões de computabilidade em vários “modelos de computação” fundamentais.

3. Aguçar a vossa curiosidade pela teoria da computação!

4. Mudar a vossa perspectiva da informática? :)

Teoria da computação pelo mundo

Great Ideas in Theoretical Computer Science @ CMU

Aulas disponíveis no YouTube (https://www.youtube.com/@RyanODonnellTeaching)

http://www.cs.cmu.edu/~15251/

https://www.cs251.com/

Automata, Computability, and Complexity Theory @ MIT

https://people.csail.mit.edu/rrw/6.045-2020/

Introduction to Theoretical Computer Science @ Harvard

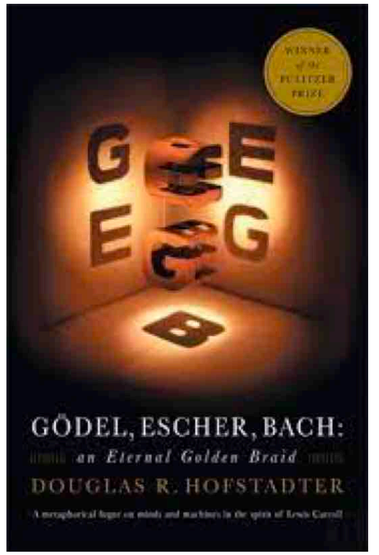
https://cs121.boazbarak.org/

Bibliografia recomendada

- Michael Sipser, *Introduction to the Theory of Computation.*
- Harry Lewis & Christos Papadimitriou, *Elements of the Theory of Computation.*
- As notas e exercícios baseiam-se maioritariamente nestes livros.
- É muito bom ler várias perspectivas sobre o mesmo tópico!

Leitura complementar

Em modo “pop-sci”!



Gödel, Escher, Bach: An Eternal Golden Braid

D. Hofstadter

Logicomix

A. Doxiadis and C. Papadimitriou

Quanta Magazine

https://xkcd.com/917/

Para os mais entusiastas

Se estiverem interessados em aprofundar o vosso conhecimento sobre teoria da computação:

Introduction to the Theory of Computation
M. Sipser

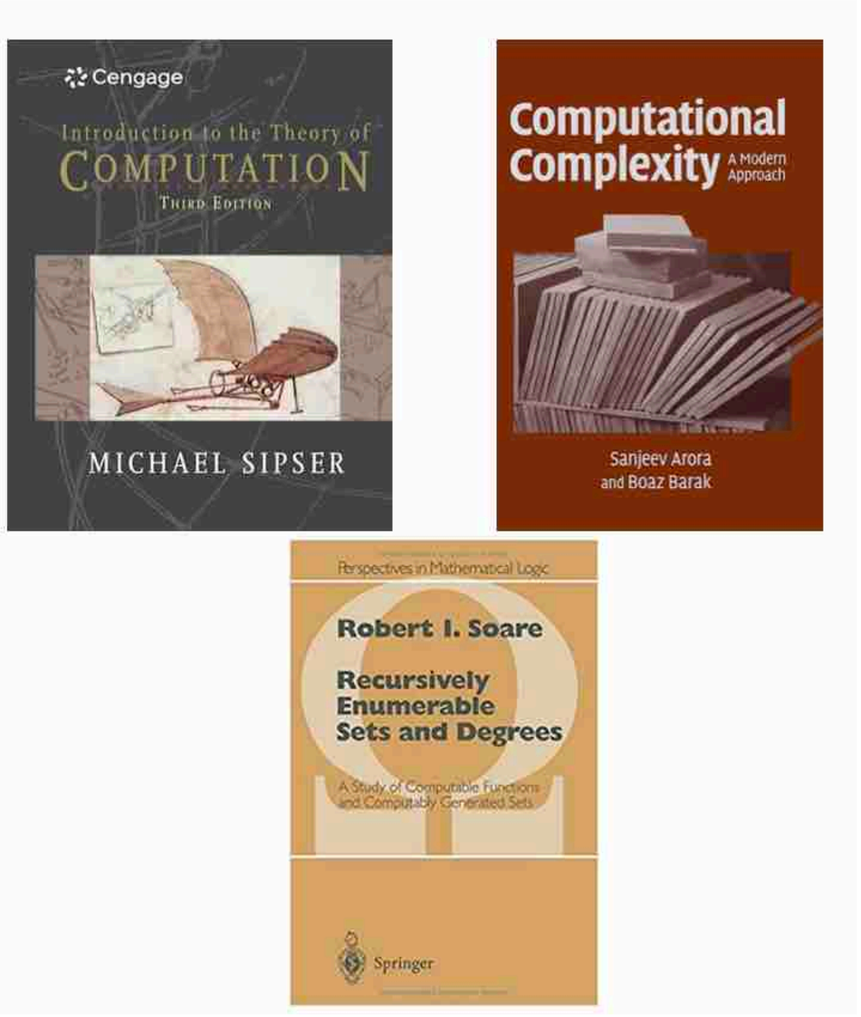
Computational Complexity: A Modern Approach
S. Arora and B. Barak

Recursively Enumerable Sets and Degrees
R. Soare

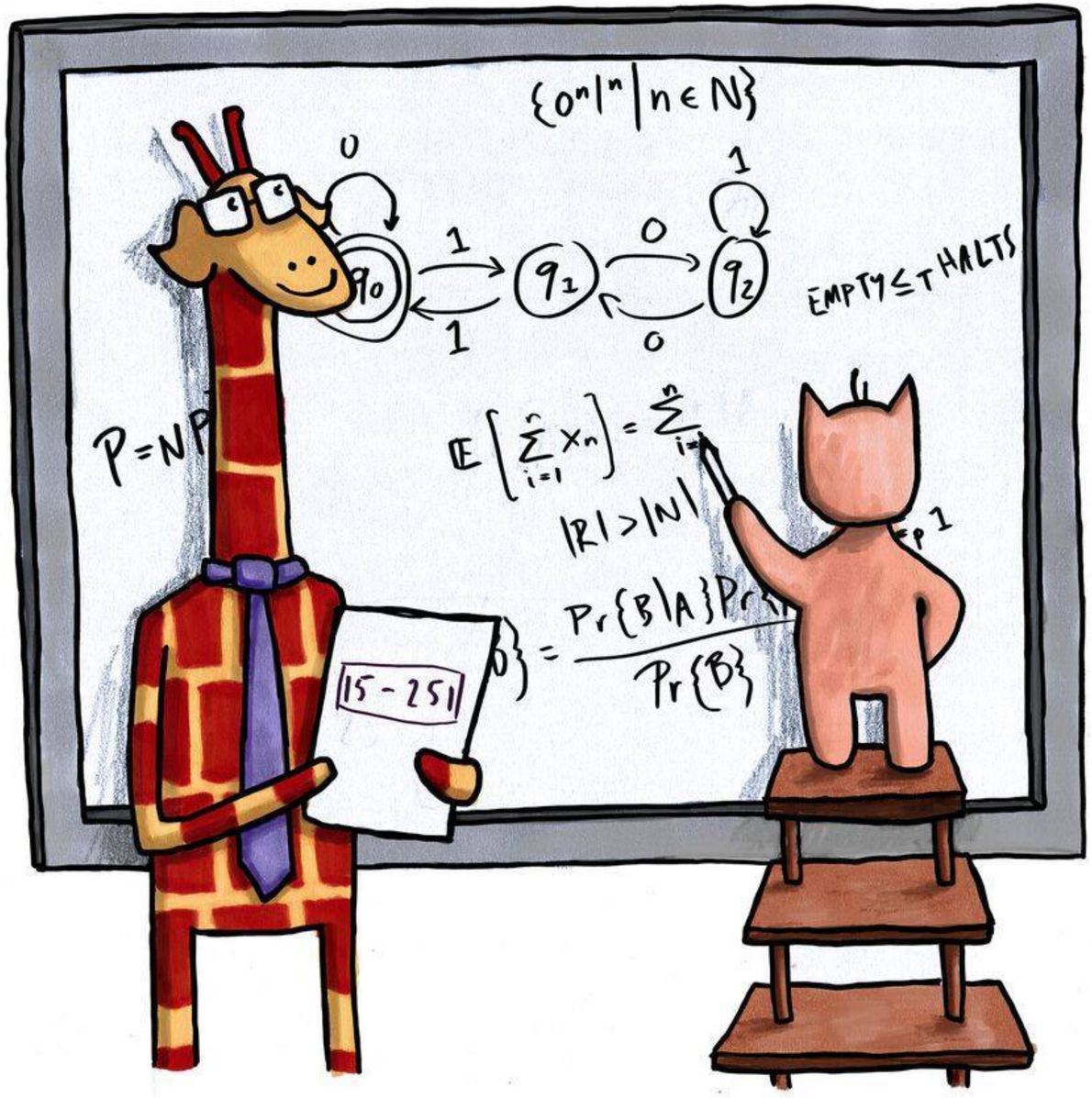
E… venham falar comigo!

joao.ribeiro@fct.unl.pt

https://sites.google.com/site/joaorib94/



Teoria da Computação = Queijo



math is hard, but you don't have to do it alone!

Créditos:
http://www.cs.cmu.edu/~15251/

Piazza

- Local para colocarem dúvidas e discussões em grupo. Moderado pela equipa docente.
- Questões e comentários podem ser publicados de forma anónima.
- Signup link: https://piazza.com/fct.unl.pt/spring2024/2468/home
- Access code: h38f257av3h

Horários de atendimento (+ info no CLIP)

- Frederico Vicente: Segundas, 8h30-10h.
- Vasco Amaral: Segundas, 16h30-18h.
- Vladyslav Mikytiv: Terças, 11h-12h30.
- João Ribeiro: Sextas, 9h-10h30.

Feedback

- O vosso feedback sobre o funcionamento da cadeira ao longo do semestre é muito importante!

- Google form anónima: https://shorturl.at/gnOPS

- Permite-nos adaptar as aulas às vossas necessidades.

Moodle

https://moodle.fct.unl.pt/course/view.php?id=8904

Avaliação

- 2 testes (ou exame). Pelo menos 9.5 de média.
- 2 mini-testes (0.5 valores cada).
- Participação activa nas aulas práticas (1 valor).
- **Bónus (1 valor):** Discussão individual com o regente sobre um tópico de TC não coberto na cadeira.
- Descrição detalhada no CLIP.

Recomendações

- Estudo **contínuo** acompanhando as aulas.
  Importante: Notas e exercícios são disponibilizados com antecedência. Devem tentar **pensar** sobre os exercícios **antes da respectiva aula prática.**

- Resolver **muitos** exercícios. Ajuda a cimentar conceitos formais e técnicas standard.

- Tomar partido do horário de atendimento e Piazza.