# Algoritmos e complexidade: uma introdução

Marcus Ritt

Instituto de Informática
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brasil

# Agenda

1. Parte 1: Modelos computacionais e analise assintótica
2. Parte 2: Analise na prática
3. Parte 3: Complexidade de problemas e classes de complexidade

# Parte I

# Modelos e analise

# Agenda

# Multiplicação

$$3 \times 5 = ?$$

# Multiplicação

$$3 \times 5 = 15$$

# Multiplicação...

$$
\begin{aligned}
&334780716989568987860441698482126908177 \\
&047949837137685689124313889828837938780 \\
&022876147116525317430877378144679994 89 \\
&\times \\
&367460436667995904282446337996279526322 \\
&791581643430876426760322838157396665112 \\
&792333734171433968102700927987363089 17 = ?
\end{aligned}
$$

# Multiplicação...

$$
\begin{aligned}
&334780716989568987860441698482126908177\\
&047949837137685689124313889828837938780\\
&022876147116525317430877378144679994 89\\
&\times\\
&367460436667995904282446337996279526322\\
&791581643430876426760322838157396665112\\
&792333734171433968102700927987363089 17\\
&=\\
&123018668453011775513049495838496272077\\
&285356959533479219732245215172640050726\\
&365751874520219978646938995647494277406\\
&384592519255732630345373154826850791702\\
&612214291346167042921431160222124047927\\
&473779408066535141959745985690214341 3
\end{aligned}
$$

# Versus: Fatoração

$$21 = ? \times ?$$

# Versus: Fatoração

$$21 = 3 \times 7$$

# Versus: Fatoração...

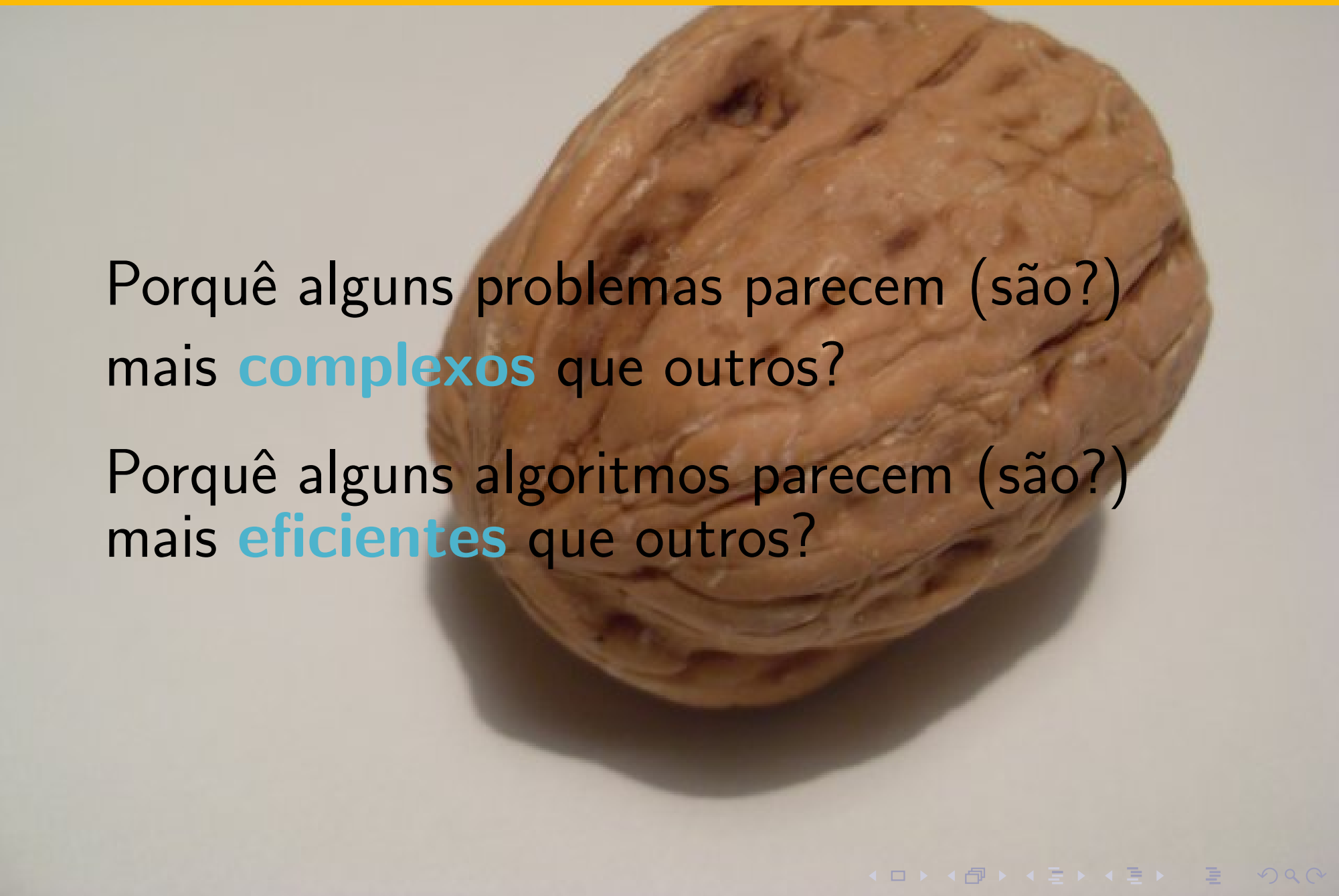
$$
\begin{aligned}
&1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786469389956474942774063845925192557326303453731548268507917026122142913461670429214311602221240479272=?\times ?
\end{aligned}
$$

# Versus: Fatoração...

$$
\begin{aligned}
&1230186684530117755130494958384962720772853569595334792197322452151726400507263657518745202199786469389956474942774063845925192557326303453731548268507917026122142913461670429214311602212404792747377940806653514195974595685690214341 3\\
&= \\
&3347807169895689878604416948212690817704794983713768568912431388982883793878002287614711652531743087737814467999489\\
&\times \\
&3674604366679959042824463379962795263227915816434308764267603228381573966651127923337341714339681027009279873630891 7
\end{aligned}
$$

**Custo**: 2000 2.2GHz-Opteron–anos CPU

# Complexidade?



Porquê alguns problemas parecem (são?) mais **complexos** que outros?

Porquê alguns algoritmos parecem (são?) mais **eficientes** que outros?

# Nossos exemplos

Multiplicação de números inteiros de $O(n)$ bits:

- Método "escola": $O(n^2)$
- Melhor método: $n \log n \, 2^{O(\log^* n)}$ (Fürer, 2007)

Fatoração de um número inteiro com $O(n)$ bits:

- Crivo de Eratóstenes: $O(2^n n \log n)$.
- General prime number sieve: $O(e^{(c+o(1))n^{1/3} \log^{2/3} n})$

(

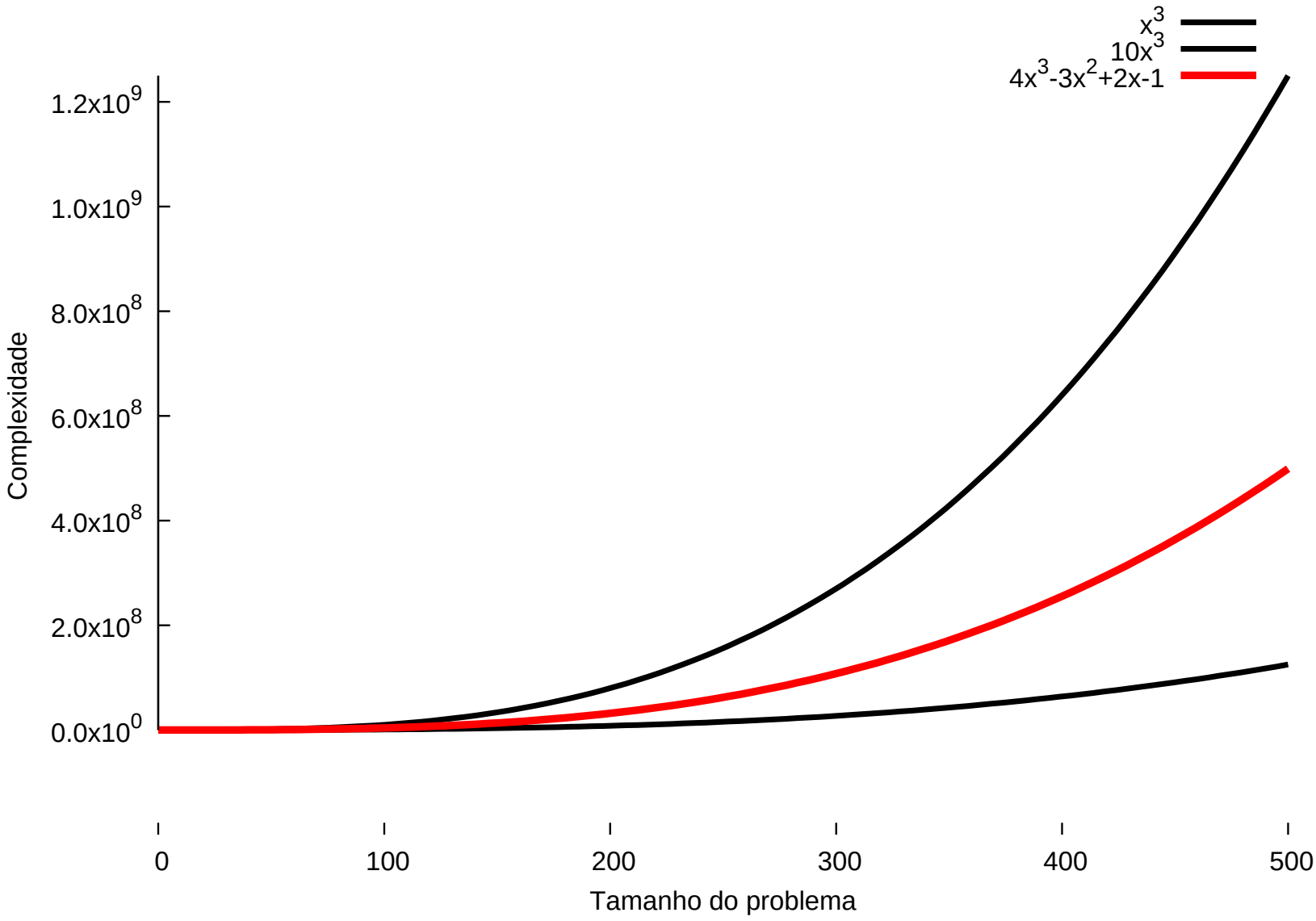
# Agenda

# Notação assintôtica

- Frequentemente desejável: só considerar a **ordem de crescimento** de uma função
- Exemplo: $4n^3 - 3n^3 + 2n - 1$ cresce "como" $n^3$

Notação:

- Limite superior: $4n^3 - 3n^3 + 2n - 1 \in O(n^3)$: cresce "não mais" que $n^3$
- Limite inferior: $4n^3 - 3n^3 + 2n - 1 \in \Omega(n^3)$: cresce "mais" que $n^3$
- $4n^3 - 3n^3 + 2n - 1 \in \Theta(n^3)$: cresce "igual" a $n^3$


$x^3$
$10x^3$
$4x^3-3x^2+2x-1$
1.2x10^9
1.0x10^9
8.0x10^8
6.0x10^8
4.0x10^8
2.0x10^8
0.0x10^0
Complexidade
0
100
200
300
400
500
Tamanho do problema

# Mais formal...

## Classes de crescimento

Para funções $f, g : \mathbb{N} \to \mathbb{R}^+$:

$$O(g(n)) = \{f \mid \exists c > 0 \exists n_0 \forall n > n_0 : \ f(n) \leq cg(n)\}$$
$$\Omega(g(n)) = \{f \mid \exists c > 0 \exists n_0 \forall n > n_0 : \ f(n) \geq cg(n)\}$$
$$\Theta(g(n)) = O(g(n)) \cap \Omega(g(n))$$
$$o(g(n)) = \{f \mid \forall c > 0 \exists n_0 \forall n > n_0 : \ f(n) \leq cg(n)\}$$
$$\omega(g(n)) = \{f \mid \forall c > 0 \exists n_0 \forall n > n_0 : \ f(n) \geq cg(n)\}$$

# Exemplos

- $\operatorname{sen}(n) \in O(1)$
- $n^2 \in O(n^3)$
- $n^{O(1)}$ (tempo polinomial)
- $n! \in \sqrt{2\pi n}\left(\frac{n}{e}\right)^n(1 + O(1/n))$ (Stirling, 1730)
- $\pi(n) \in \Theta(n/\ln n)$ (Hadamard, 1896; de la Vallée Poussin, 1896)

# Notações adicionais

- Logaritmo iterado

$$\log^* n = \begin{cases} 0 & \text{se } n \leq 1 \\ 1 + \log^*(\log n) & \text{caso contrário} \end{cases}$$

- Desconsiderar fatores logarítmicos (soft-$O$, $O$-suave)

$$f \in \tilde{O}(g) \implies \exists k : f \in O(g \log^k g)$$

# Algumas características

$$
\begin{aligned}
f &= O(f) \\
cO(f) &= O(f) \\
O(f) + O(f) &= O(f) \\
O(O(f)) &= O(f) \\
O(f)O(g) &= O(fg) \\
O(fg) &= fO(g) \\
g = O(f) &\implies f + g = \Theta(f)
\end{aligned}
$$

)

# Eficiente: em que?

- Tempo de execução
- Memôria consumida
- Número de operações E/S
- Memôria acessada e número de falhas de cache
- Energia consumida
- Energia dissipada

Problema básico: qual algoritmo é mais eficiente?

# O que é eficiente?

Uma tentativa:

**Definition (Kleinberg and Tardos (2005))**

An algorithm is efficient if, when implemented, it runs quickly on real input instances.

# Abordagem experimental

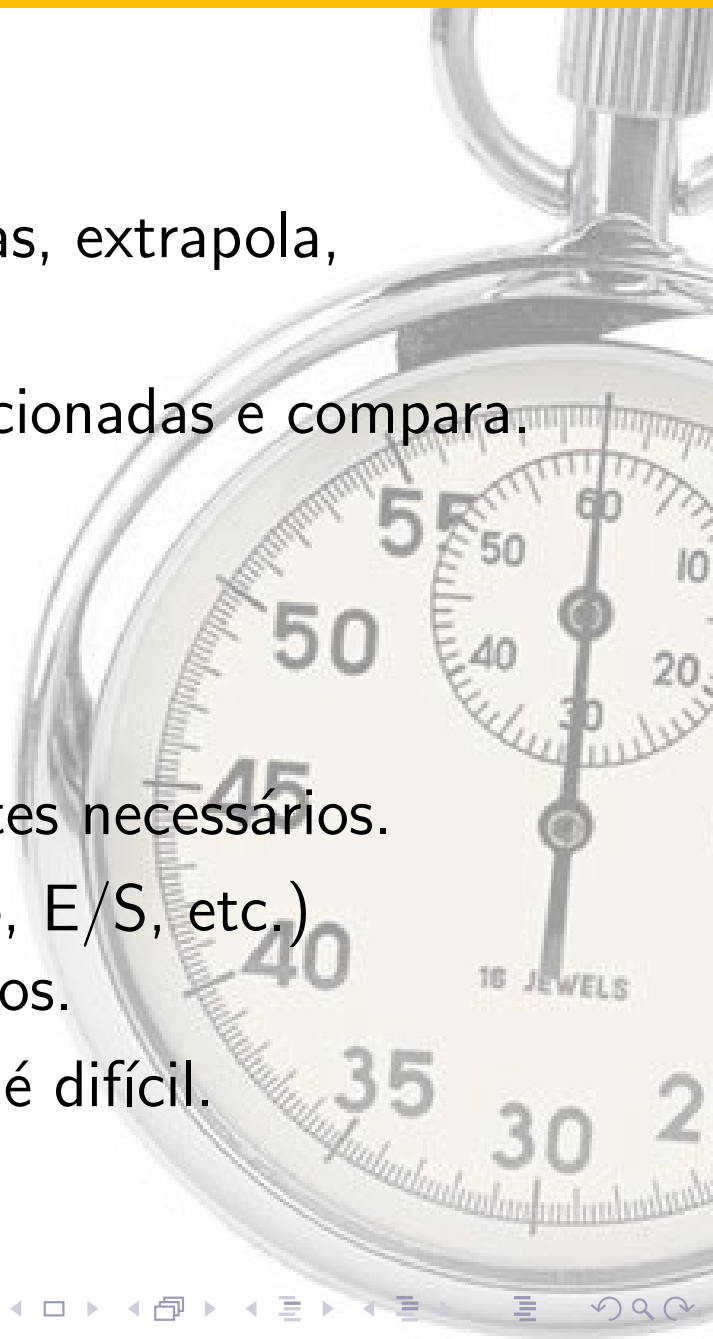
- Mede um grande número de instâncias, extrapola, compara.
- Mede um conjunto de instâncias selecionadas e compara.

Vantagens:

- Relativamente simples

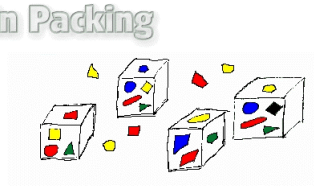
Desvantagens:

- Pode ser inviável pelo número de testes necessários.
- Muitas dependências (máquina, ruído, E/S, etc.) dificultam a comparação dos resultados.
- Escolha de instâncias representativas é difícil.



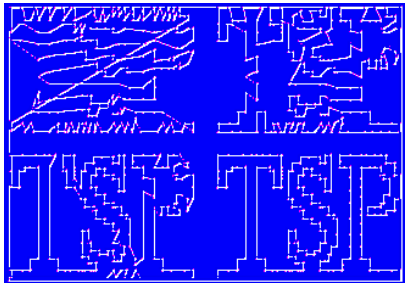
# Alguns exemplos de coleções de instâncias

QAPLIB




SteinLib

VRP web




VRPLIB

SATLIB


MIPLIB

# Exemplo de um fracasso

*It is all to easy to make predictions which are quite at variance with observed performance.*
*It is also easy to assume that relative performance on one computer will apply to another computer.*
*Perhaps even worse, it is possible to optimize away the worst case [...] at the cost of penalizing the usual case.*

**Fenwick, Some perils of performance prediction: a case study in pattern matching.**

# Agenda

1. Complexidade
2. Notação assintôtica
3. Modelos de computação
   - A máquina de Turing
   - A máquina RAM
4. Um fundamento para a análise prática

# Abordagem teórica

Define um modelo teórico, e compara neste modelo.

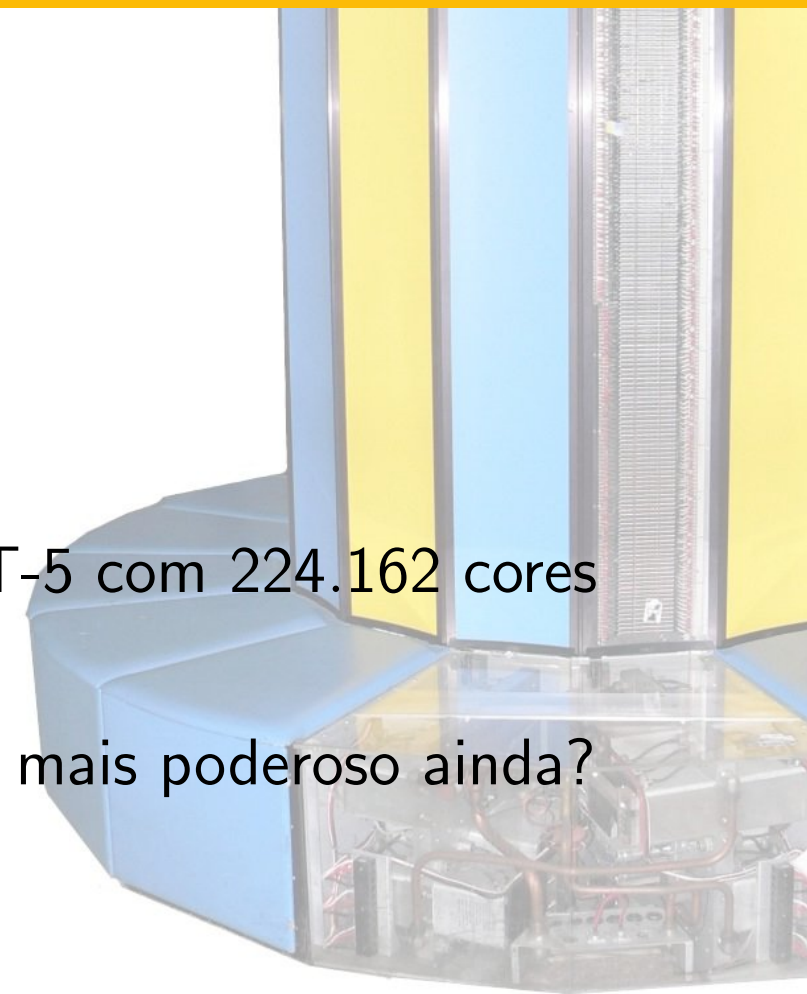
Vantagens:

- Resultados comparáveis.

Desvantagens:

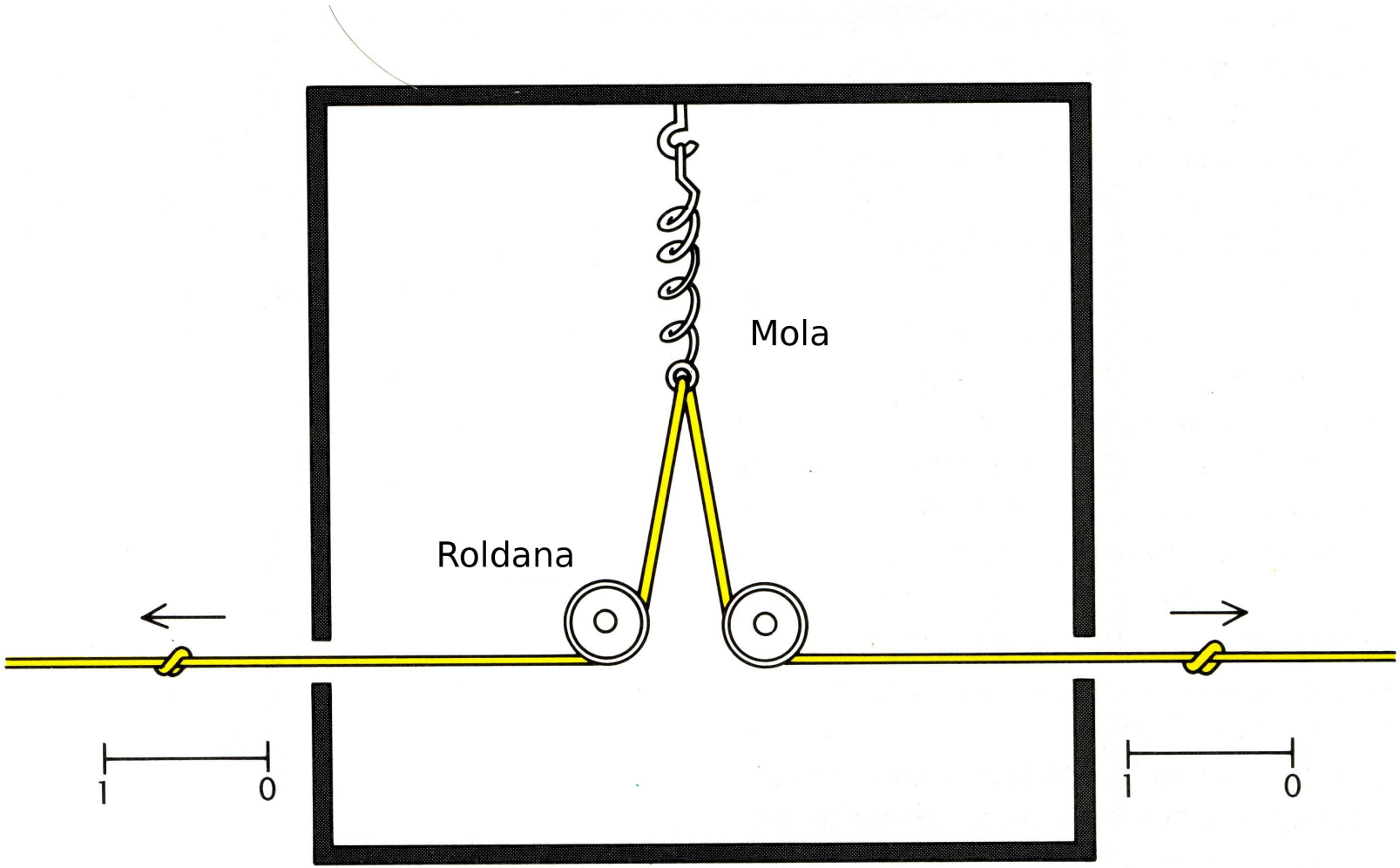
- Modelos podem ser elementares demais para algoritmos práticos (máquina Turing).
- Não é obviou qual o modelo certo (quais instruções?)

# Qual o modelo de computação adequado ?

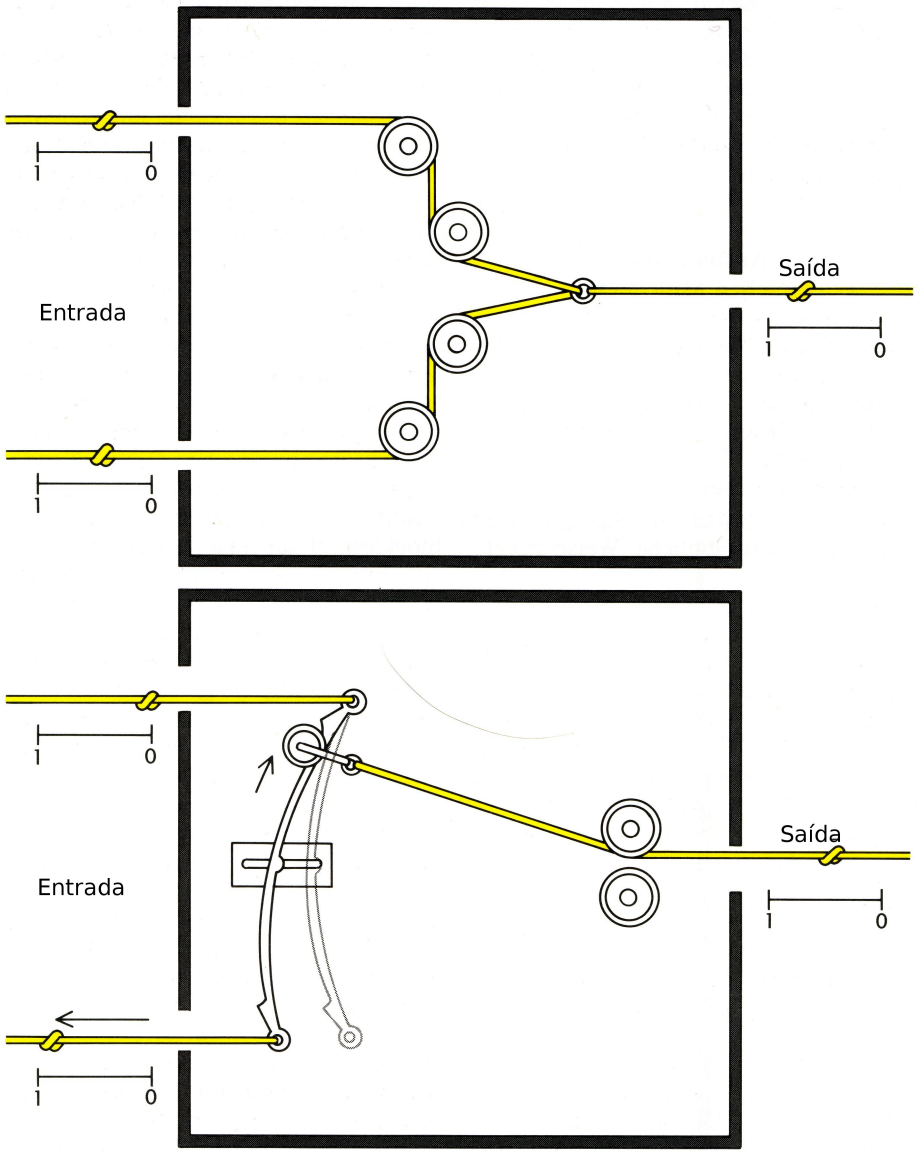


- O computador de von Neumann?
- Um super-computador paralelo?
  Top 500, 9/2009: Jaguar Cray XT-5 com 224.162 cores
- Um computador quântico?
- Um outro modelo de computação mais poderoso ainda?
  O universo?

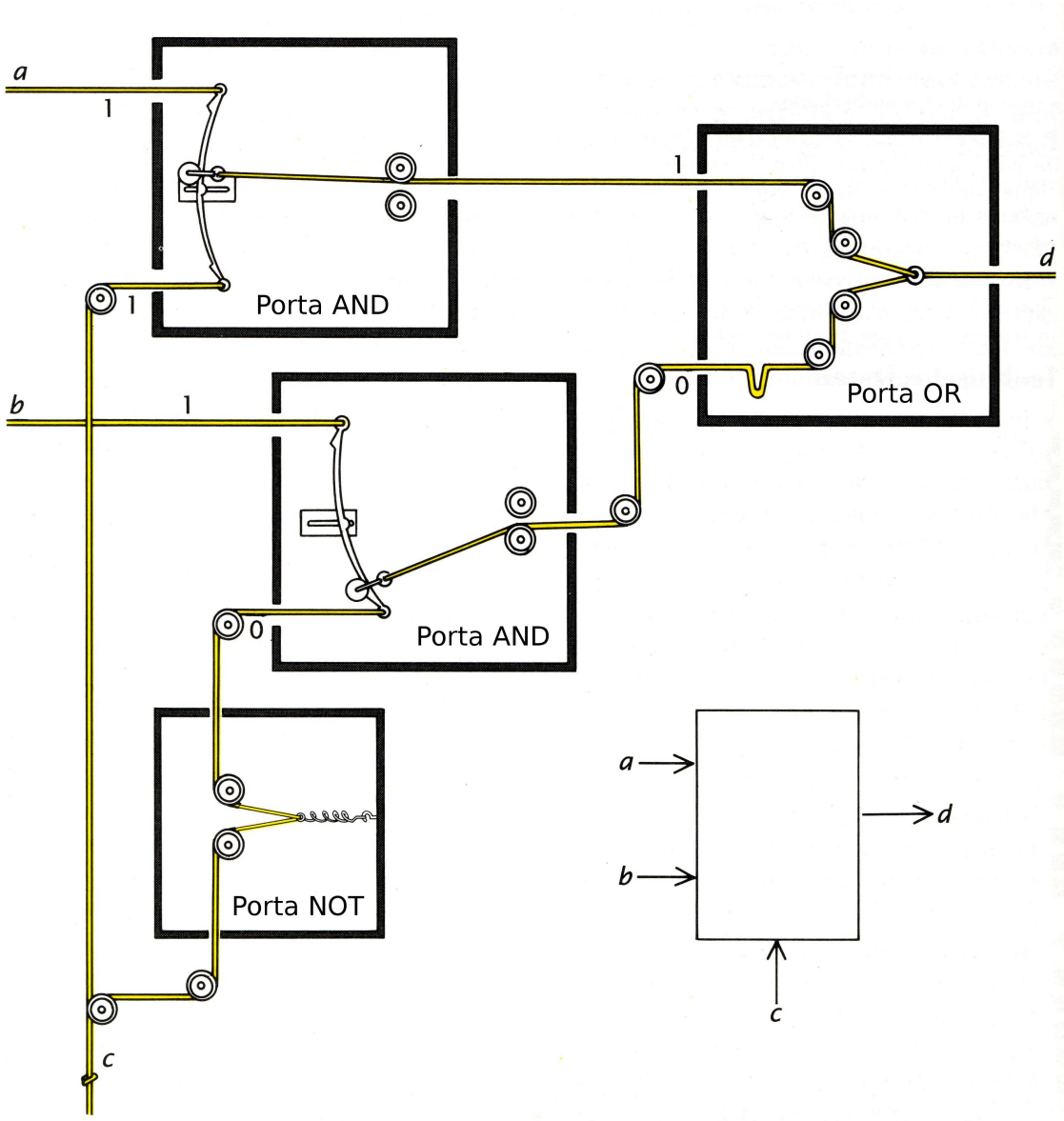
# Qual o modelo certo?



# Qual o modelo certo?

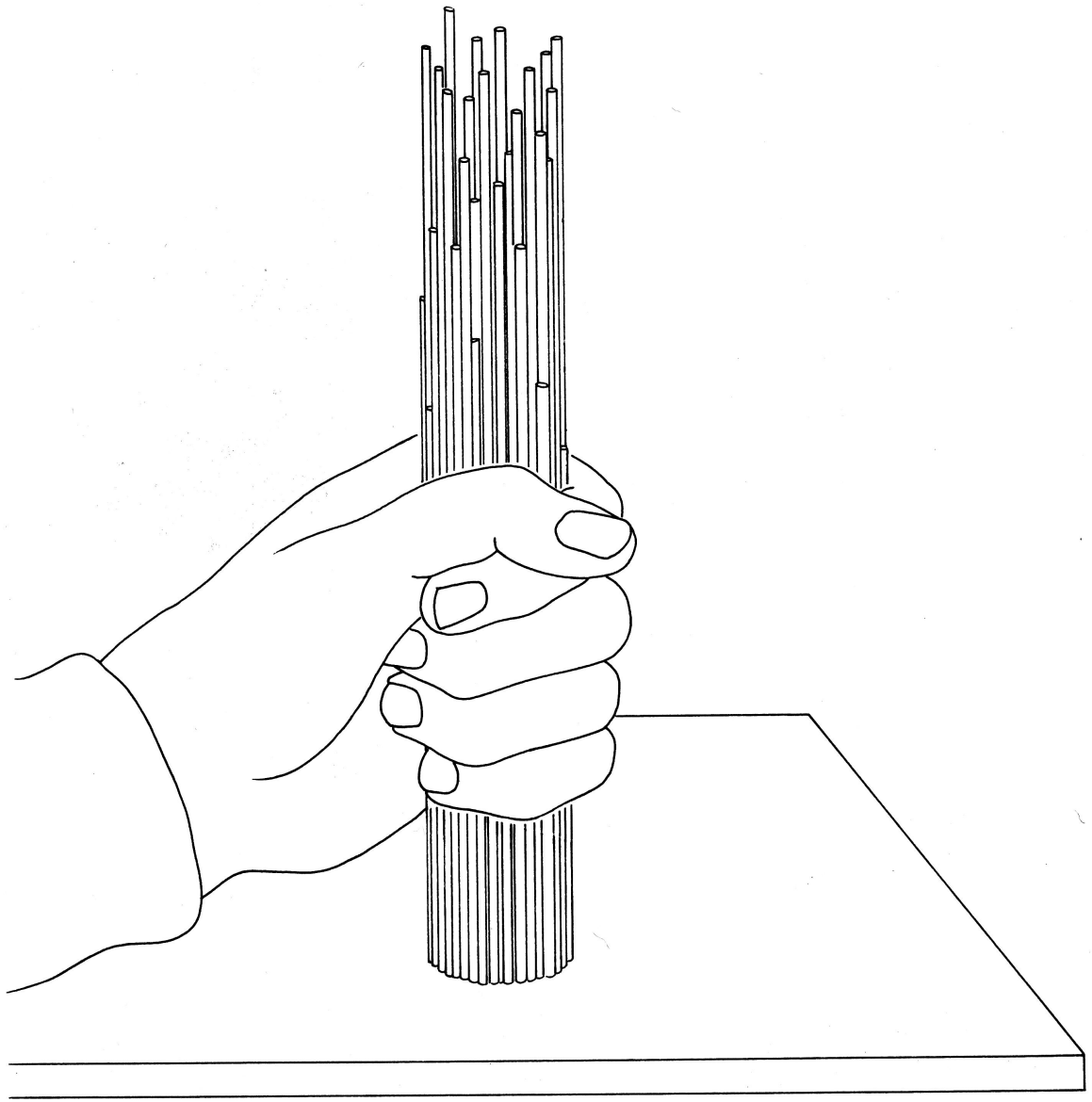


# Qual o modelo certo?



# Qual o modelo certo?

Ordenar $n$ números em tempo linear?



# O universo é o computador? Fatos

| | |
|---|---|
| Idade | $13.7 \pm 0.2 \times 10^9$ anos $\approx 43.5 \times 10^{16}$ s |
| Tamanho | $\geq 78 \times 10^9$ anos-luz |
| Densidade | $9.9 \times 10^{-30} g/cm^3$ |
| Número de átomos | $10^{80}$ |
| Número de bits | $10^{120}$ |
| Operações lógicas elementares até hoje | $10^{120}$ |
| Operações/s | $\approx 2 \times 10^{102}$ |

(Pelo consenso científico atual. Fontes principais: Lloyd (2002); WMA)

# Resgate 1

## Tese de Church-Turing

As funções efetivamente computáveis sobre os números inteiros positivos são precisamente as funções computáveis pela máquina de Turing.

- Verdadeiro para todos modelos conhecidos

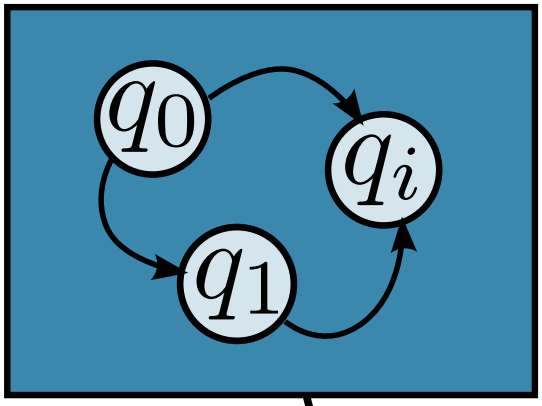
  *Maquina de Turing, cálculo lambda, máquina RAM, máquina pontador, circuitos lógicos, autômatos celulares (Conway), avaliação de templates em C++, computador billiard, computador quântico, ...*

# A máquina de Turing

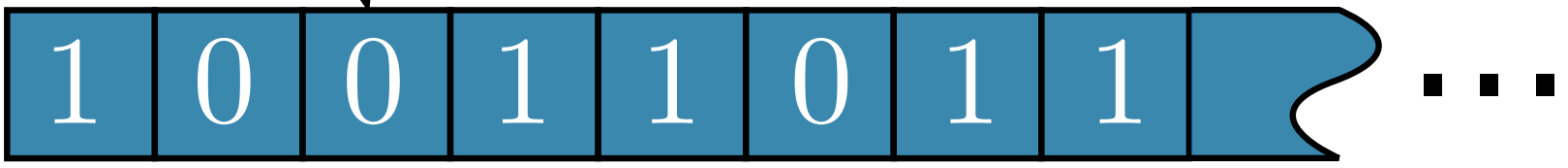
Computing is normally done by writing certain symbols on paper. "We may suppose this paper is divided into squares like a child's arithmetic book. In elementary arithmetic the two-dimensional character of the paper is sometimes used. But such a use is always avoidable, and I think that it will be agreed that the two-dimensional character of paper is no essential of computation. I assume then that the computation is carried out on one-dimensional paper, i.e. on a tape divided into squares. I shall also suppose that the number of symbols which may be printed is finite. If we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent. The effect of this restriction of the number of symbols is not very serious. It is always possible to use sequences of symbols in the place of single symbols. Thus an Arabic numeral such as 17 or 999999999999999 is normally treated as a single symbol. Similarly in any European language words are treated as single symbols (Chinese, however, attempts to have an enumerable infinity of symbols). The differences from our point of view between the single and compound symbols is that the compound symbols, if they are too lengthy, cannot be observed at one glance. This is in accordance with experience. We cannot tell at a glance whether 9999999999999999 and 999999999999999 are the same. (Turing, 1936).

Alan Mathison Turing (*1912, +1954)

# Um exemplo – a máquina de Turing




Cabeça de leitura
e escritura




Fita infinita

# A máquina RAM

A **máquina RAM** (random access machine) é o modelo padrão para análise de algoritmos. Ela possui

- um processador com um ou mais registros, e com apontador de instruções,
- uma memória infinita de números inteiros e
- instruções elementares (controle,transferência inclusive endereçamento indireto,aritmética).

# A máquina RAM

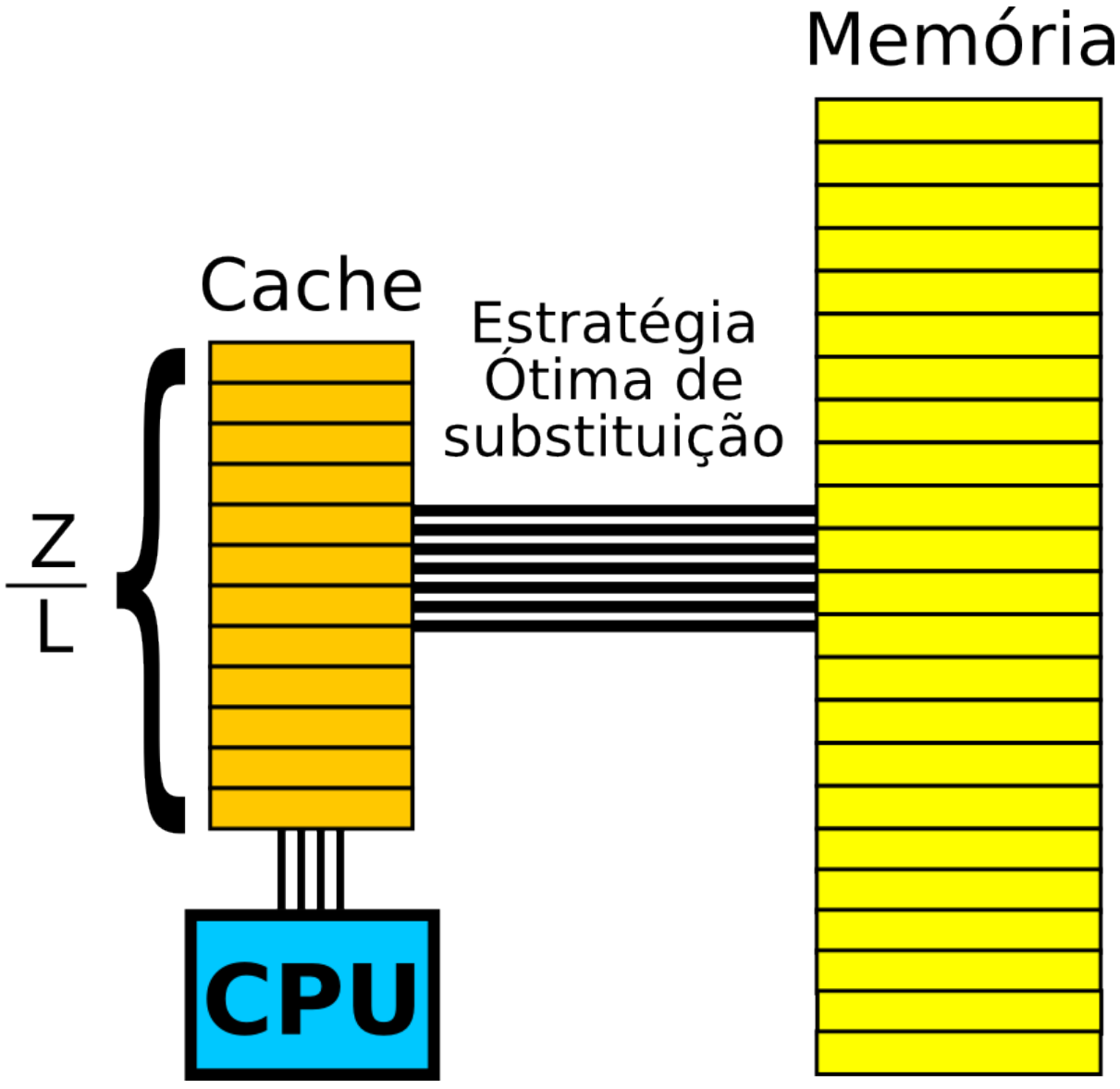
Existem RAMs com diferentes tipos de instruções aritméticas

- **SRAM**: somente sucessor
- **RAM**: adição e subtração
- **MRAM**: multiplicação e divisão

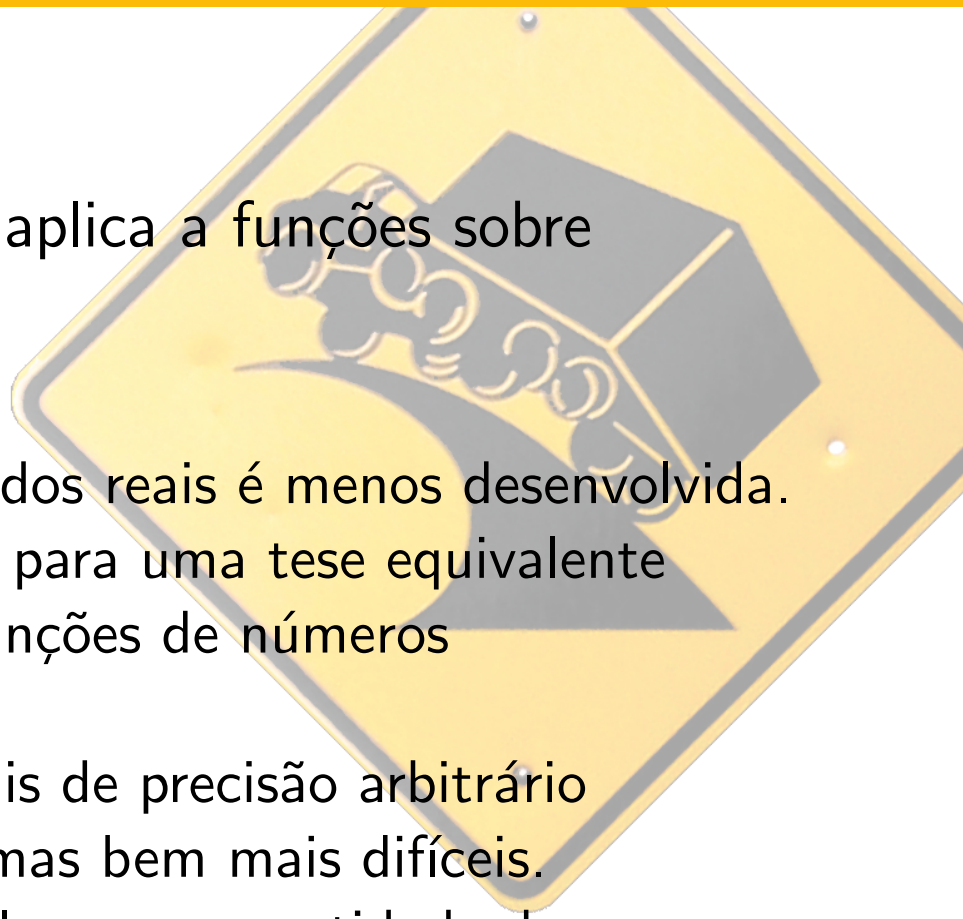
e com diferentes tipos de custos

- Custo **uniforme**: cada operação em $O(1)$
- Custo **logarítmico**: proporcional ao número de bits dos operandos

# Máquina RAM com cache

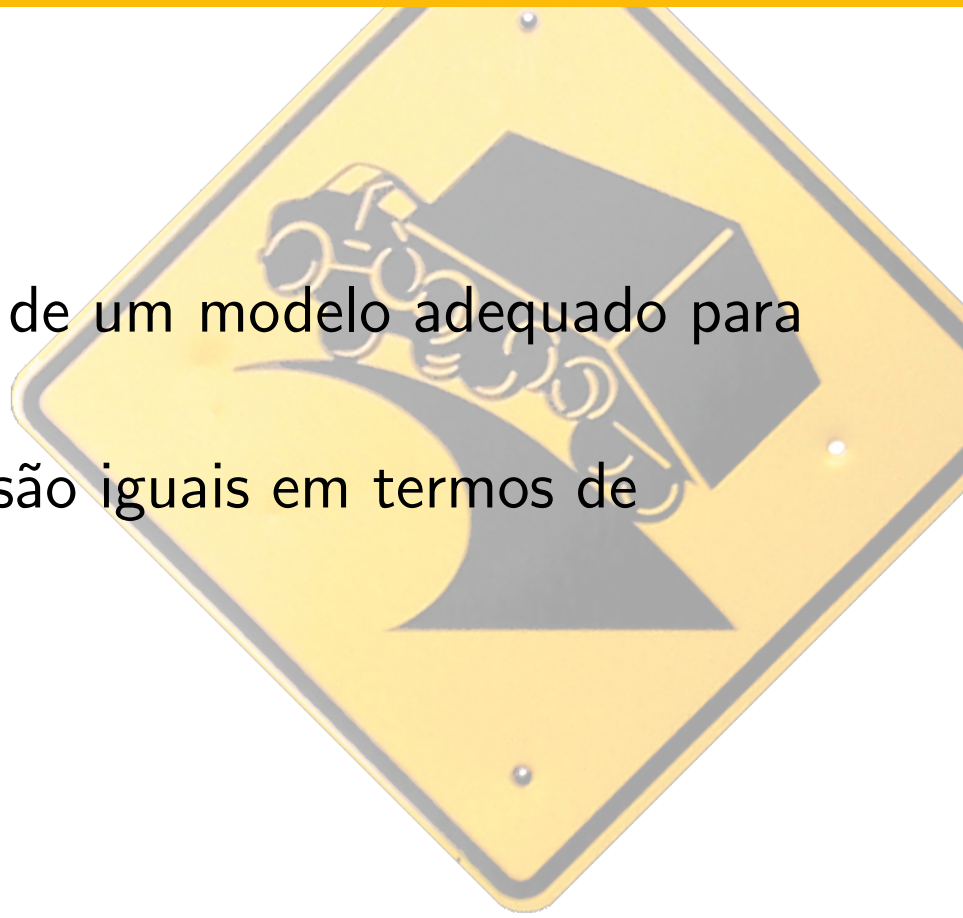


# Perigo?



- A tese de Church-Turing se aplica a funções sobre números inteiros.
- E números reais?
  - A teoria da computação dos reais é menos desenvolvida.
  - Exemplo: falta evidência para uma tese equivalente sobre as funções sobre funções de números inteiros (Mitchell, 1996).
  - Com operações sobre reais de precisão arbitrário podemos resolver problemas bem mais difíceis.
  - Práticamente não é problema: a quantidade de informação por volume (ou energia) é limitada (Bekenstein, 1981).

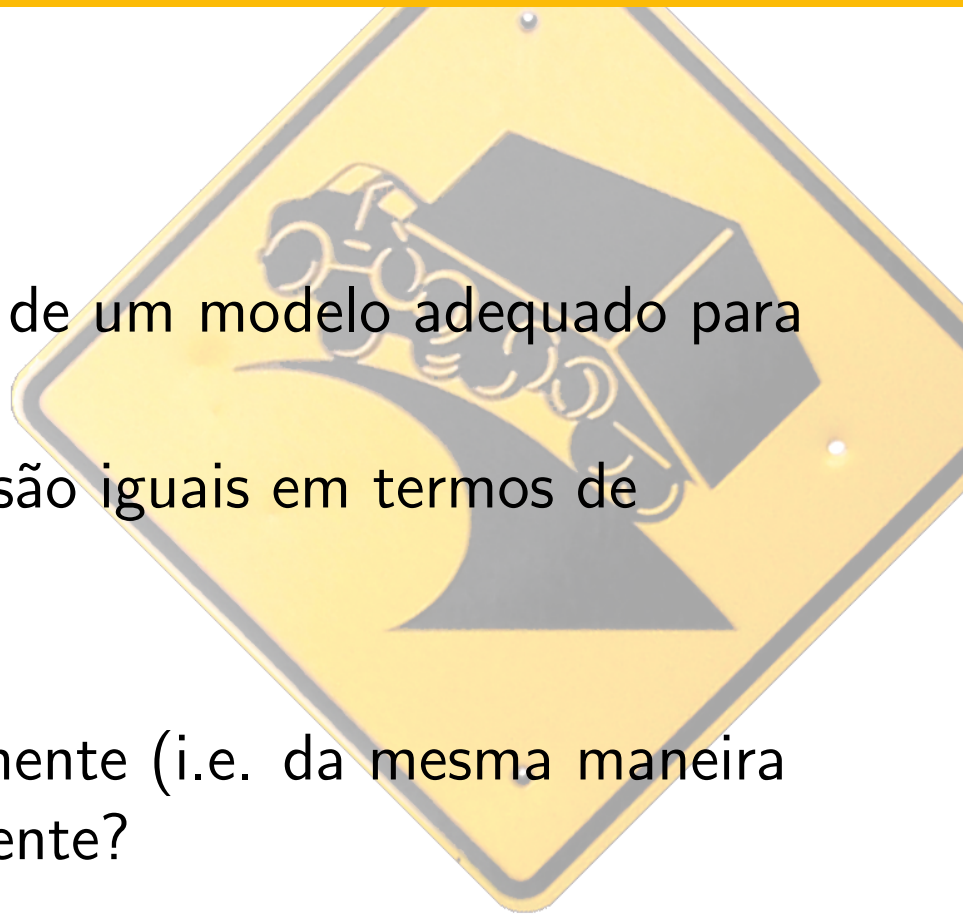
# Perigo?



Isso ainda não garante a escolha de um modelo adequado para analise de eficiência!

- Será que todos os modelos são iguais em termos de eficiência?
- Não!

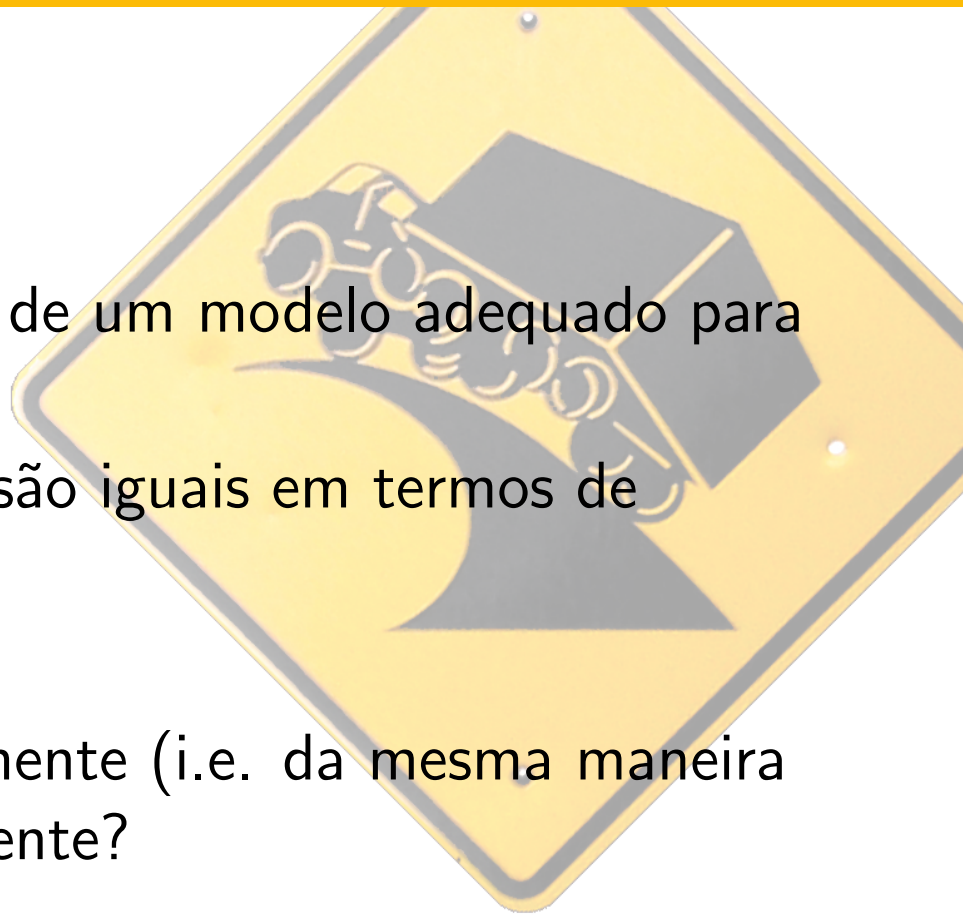
# Perigo?



Isso ainda não garante a escolha de um modelo adequado para analise de eficiência!

- Será que todos os modelos são iguais em termos de eficiência?
- Não!
- Será que eles são uniformemente (i.e. da mesma maneira para todos problemas) diferente?

# Perigo?



Isso ainda não garante a escolha de um modelo adequado para analise de eficiência!

- Será que todos os modelos são iguais em termos de eficiência?
- Não!
- Será que eles são uniformemente (i.e. da mesma maneira para todos problemas) diferente?
- Não!

# Exemplos de simulação

## Theorem (van Leeuwen (1990))

$$\begin{aligned}
\mathrm{m-tapes} &\leq 1-\mathrm{tape}(\textit{time } kn^2 \ \&\ \textit{space } Lin)\\
\mathrm{m-tapes} &\leq 2-\mathrm{tape}(\textit{time } kn\log n \ \&\ \textit{space } Lin)\\
\mathrm{SRAM-UTIME} &\leq T(\textit{time } n^2\log n)\\
\mathrm{RAM-UTIME} &\leq T(\textit{time } n^3)\\
\mathrm{MRAM-UTIME} &\leq T(\textit{time } Exp)\\
\mathrm{SRAM-LTIME} &\leq T(\textit{time } n^2)\\
\mathrm{RAM-LTIME} &\leq T(\textit{time } n^2)\\
\mathrm{MRAM-LTIME} &\leq T(\textit{time } Poly)
\end{aligned}$$

# Resgate 2

## Tese estendida de Church-Turing

Qualquer modelo de computação universal é equivalente à máquina de Turing com

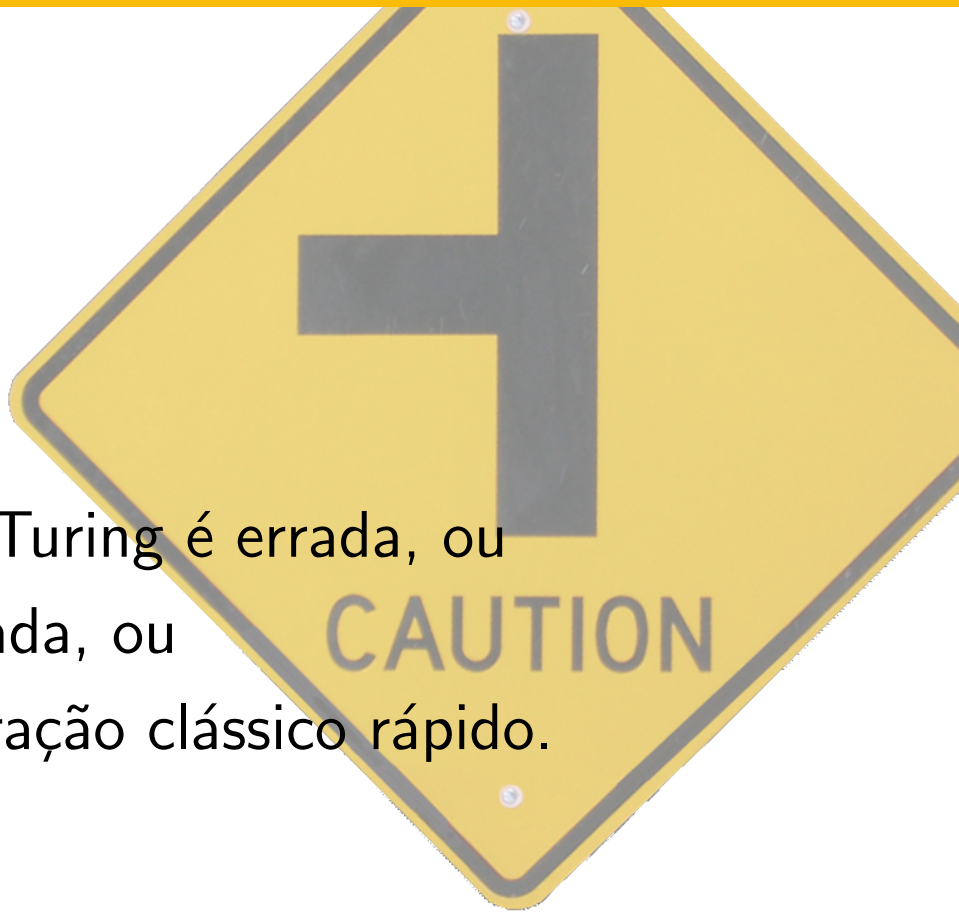
- custo adicional de tempo no máximo polinomial
- custo adicional de espaço no máximo constante

- Equivalência definido por simulação mutual.
- Verdadeiro para **quase** todos modelos conhecidos:

  *Maquina de Turing, cálculo lambda, máquina RAM, máquina pontador, circuitos lógicos, autômatos celulares (Conway), avaliação de templates em C++, computador billiard, ...*

- Computador quântico?

# Consequência: Shor's trilemma




Ou

- a tese estendida de Church-Turing é errada, ou
- a física quântica atual é errada, ou
- existe um algoritmo de fatoração clássico rápido.

# Agenda

# Um fundamento para a análise prática

- A **máquina RAM** (simples) serve como base da análise.
- A complexidade de um algoritmo depende do **tamanho** $n$ da instância
  Queremos classificar o **crescimento** da complexidade com $n$.
- Na análise desconsideramos fatores constantes.

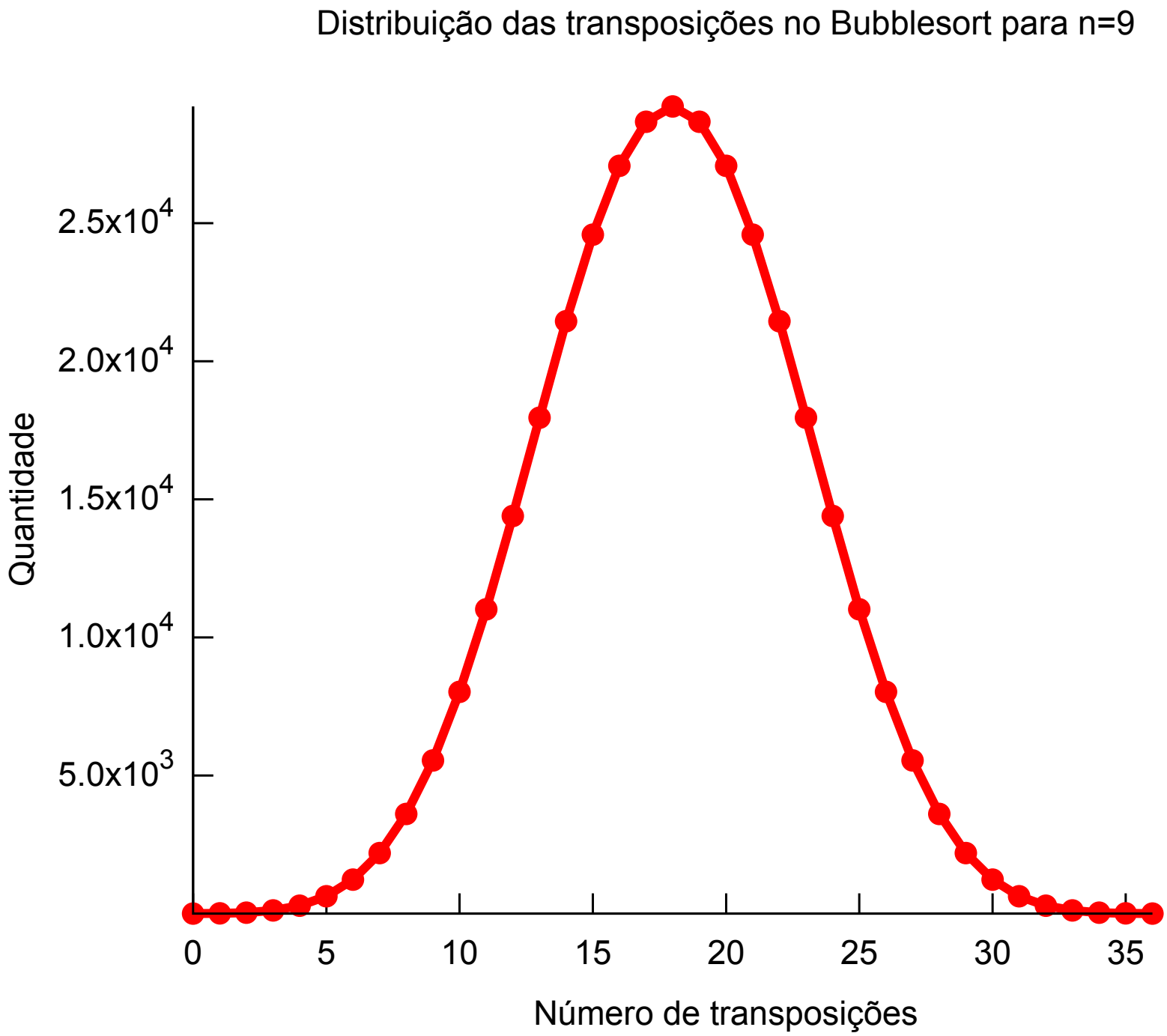
# Tipos de crescimento

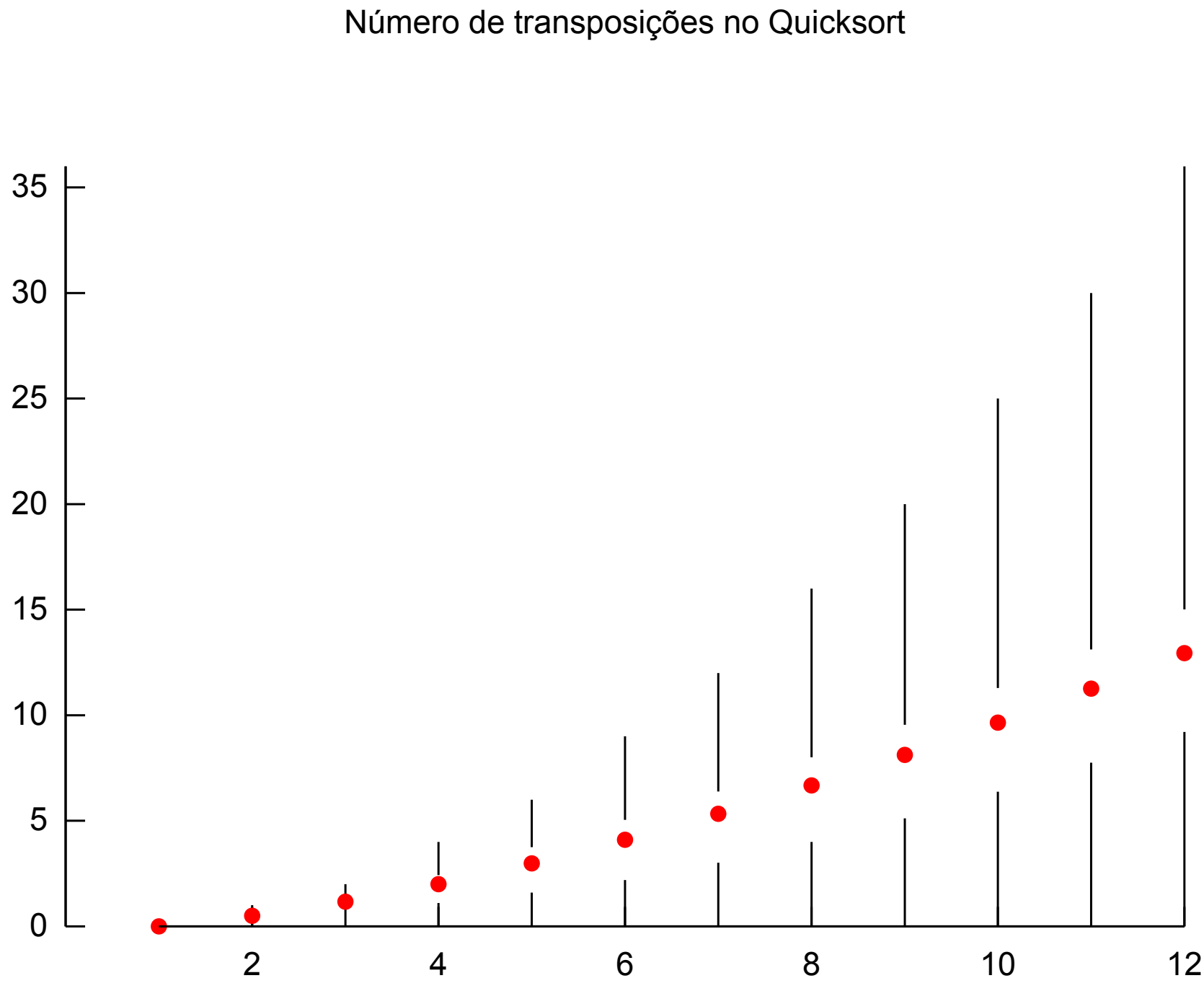
- Tempo sublinear: busca binária
- Tempo linear $O(n)$: máximo de $n$ números
- Tempo $O(n \log n)$: ordenação de $n$ números
- Tempo quadrático $O(n^2)$: menor distância entre $n$ pontos
- Tempo cúbico $O(n^3)$: multiplicação de matrizes

# Tipo de análise

Temos a opção de considerar, entre todos instâncias de tamanho $n$

- O melhor caso (análise otimista): práticamente inútil, ou
- o caso médio, ou
- o pior caso (análise pessimista): o mais comum.


Distribuição das transposições no Bubblesort para n=9
Quantidade
2.5x10^4
2.0x10^4
1.5x10^4
1.0x10^4
5.0x10^3
0
5
10
15
20
25
30
35
Número de transposições


Número de transposições no Quicksort
35
30
25
20
15
10
5
0
2
4
6
8
10
12

# Consequências

- Um problema faz parte da classe DTIME($t(n)$) caso existe um MT que decide-o em tempo $ct(n)$ para alguma constante $c > 0$.
- (A teoria foca em problemas de decisão.)
- Para o estudo da **tratabilidade** de problemas desconsideramos fatores polinomiais.
- A classe de problemas com soluções polinomiais é

$$\mathrm{P} = \bigcup_{k \geq 0} \mathrm{DTIME}(n^k)$$

# Parte II

# Análise prática

# Agenda

# Ainda vale a pena estudar algoritmos?

# O pai dele


Muhammad ibn Mūsā al-Khwārizmī ($\approx 780 - \approx 850$)

محمد بن موسى الخوارزمى

# O matador dele

Technical Report IDSIA-16-00, 3 March 2001
ftp://ftp.idsia.ch/pub/techrep/IDSIA-16-00.ps.gz

## The Fastest and Shortest Algorithm for All Well-Defined Problems[1]

**Marcus Hutter**

## Theorem (Busca de Hutter)

*Para cada problema $P$ formalmente definido e cada algoritmo $p$, que demonstravelmente resolve $P$, e para qual $t_p(x)$ que demonstravelmente é um limite superior para o tempo de execução, podemos construir um algoritmo (de forma genérica) tal que*

$$t_A(x) \leq 5t_p(x) + d_p \, time_{t_p}(x) + c_p$$

*onde $time_{t_p}$ é o tempo necessário para calcular o limite de tempo $t_p(x)$.*

Em outras palavras: $t_A(x) = O(t_p(x))$ caso a função de tempo pode ser calculado em tempo razoável!

# Busca de Hutter: Qual o problema?

- Desvantagem? Os constantes $c_p$ e $d_p$ são grandes...
- Por exemplo: Se a prova de corretude precisa $b$ bits, temos

$$c_p = 40\,2^{b+1}\,O(b^2).$$

- Conseqüência: Temos que procurar algoritmos eficientes **dentro dos limites do universo**!

# Agenda

# Exemplo

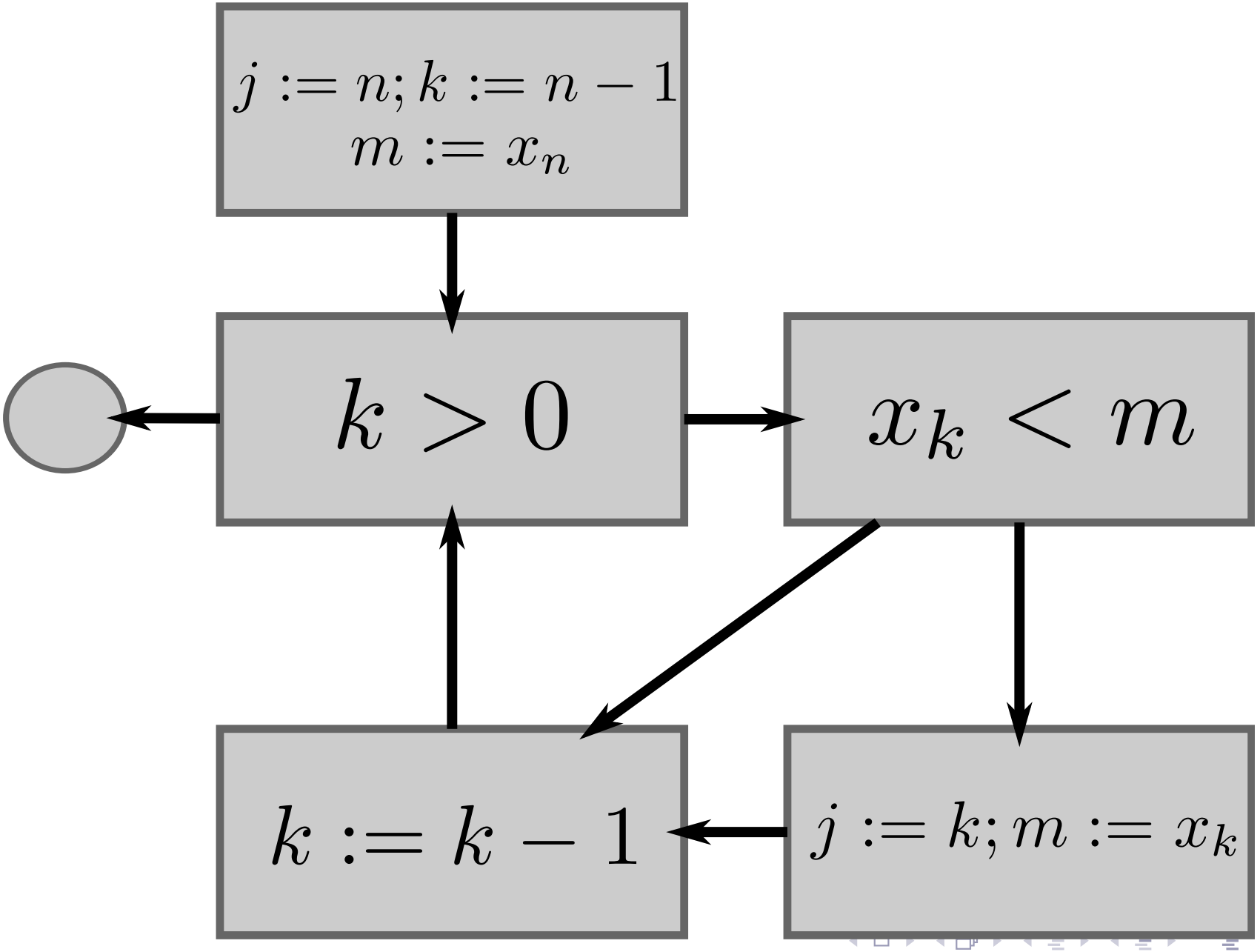
## Máximo

**Entrada** $n$ números $x_1, \ldots, x_n$

**Saída** O máximo $m = \max_{1 \leq i \leq n} x_i$.

```
1  j := n; k := n − 1; m := x_n
2  { Invariante: m = x_j = max_{k<i≤n} x_i }
3  while k > 0 do
4     if x_k < m then
5        j := k; m := x_k
6     end if
7     k := k − 1
8  end while
```

# Fluxograma do algoritmo máximo



# Custo computacional

| Linha | Repetições |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | $n$ |
| 4 | $n-1$ |
| 5 | $A$ |
| 6 | $n-1$ |
| 7 | $n-1$ |
| 8 | $n-1$ |

- Complexidade: $c_1 n + c_2 A + c_3$
- Única informação desconhecida: $A$
- Caso otimista: $c_1 n + c_3 = \Theta(n)$

$$x_n > \cdots > x_1 \implies A = 0$$

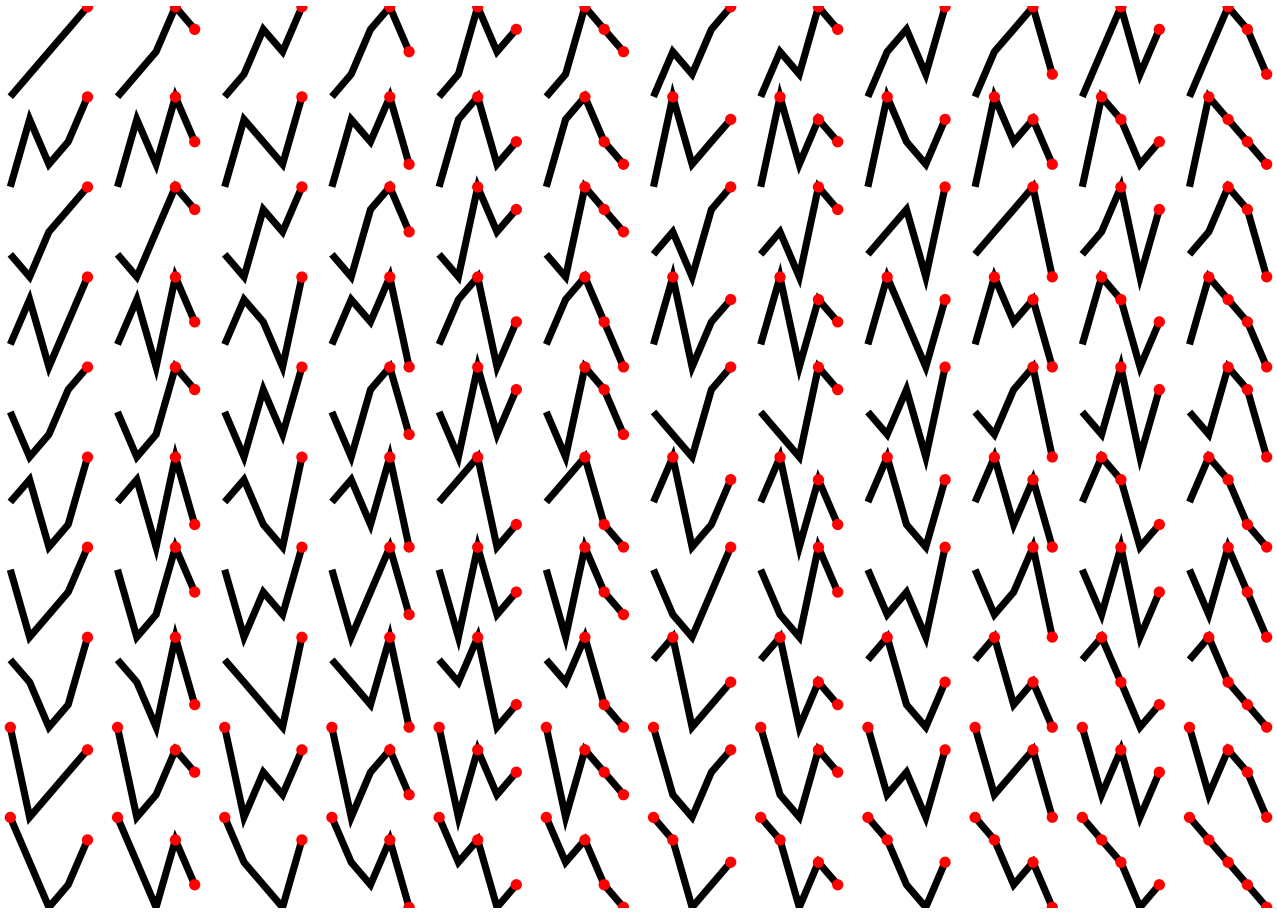
- Caso pessimista $c_1' n + c_3' = \Theta(n)$

$$x_n < \cdots < x_1 \implies A = n - 1$$

- Caso médio? $O(n)$

# E caso queremos saber mesmo?

- $A$ não depende dos valores. A questão é: "Qual o número médio de picos de uma permutação randômica?"



# E caso queremos saber mesmo?

- Para uma permutação $\pi$ considere a **tabela de inversões** $b_1, \ldots, b_n$.
- $b_i$ é o número de elementos na esquerda de $i$ que são maiores que $i$.
- Exemplo: Para 53142

| $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ |
|---|---|---|---|---|
| 2 | 3 | 1 | 1 | 0 |

- Os $b_i$ obedecem $0 \leq b_i \leq n - i$.

# Tabelas de inversões

- Observação: Cada tabela de inversões corresponde com uma permutação e vice versa.
- Exemplo: A permutação correspondente com

| $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ |
|---|---|---|---|---|
| 3 | 1 | 2 | 1 | 0 |

é 52413.

- Vantagem para a análise: Podemos escolher os $b_i$ independentemente.
- Observação: $i$ é máximo local (da esquerda), caso não tem números maiores na esquerda, i.e. para $b_i = 0$.

# Número esperado de atualizações

- Seja $X_i$ a variável aleatória $X_i = [i \text{ é máximo local}]$.
- Temos $\Pr[X_i = 1] = \Pr[b_i = 0] = 1/(n - i + 1)$.
- O número de máximos locais é $X = \sum_{1 \le i \le n} X_i$.
- Portanto, o número esperado de máximos locais é

$$E[X] = E\left[\sum_{1 \le i \le n} X_i\right] = \sum_{1 \le i \le n} E[X_i]$$

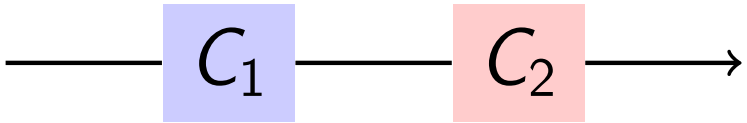
$$= \sum_{1 \le i \le n} \Pr[X_i] = \sum_{1 \le i \le n} \frac{1}{n - i + 1} = \sum_{1 \le i \le n} \frac{1}{i} = H_n$$

- Contando atualizações: tem uma a menos que os máximos locais $H_n - 1$.

# Agenda

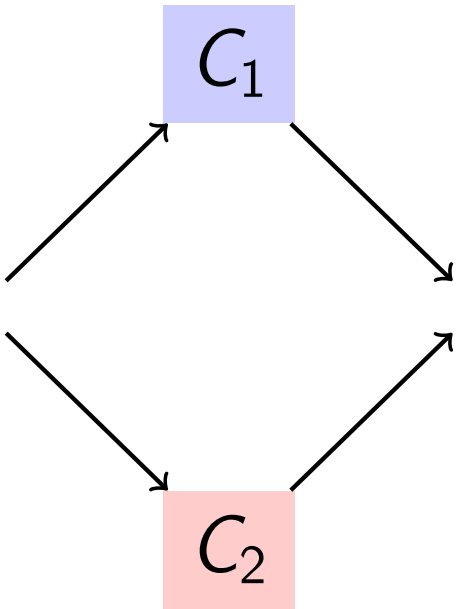
# Componentes básicos

- Componentes conjuntivas



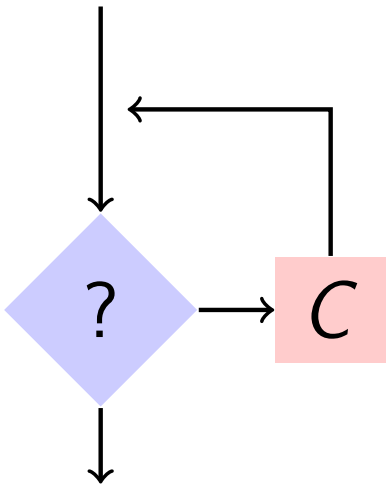

$$O(f) + O(g) = O(f + g)$$

- Componentes disjuntivas




$$O(f) + O(g) = O(\max(f, g)) = O(f + g)$$

# Laços




$$O(k)\,O(f) = O(kf)$$

- Laço definido: Número $k = k(n)$ de iterações conhecido
- Laço indefinido: Número de iterações não conhecido
  - Estimar um limite
  - Problema é indecidível no caso geral

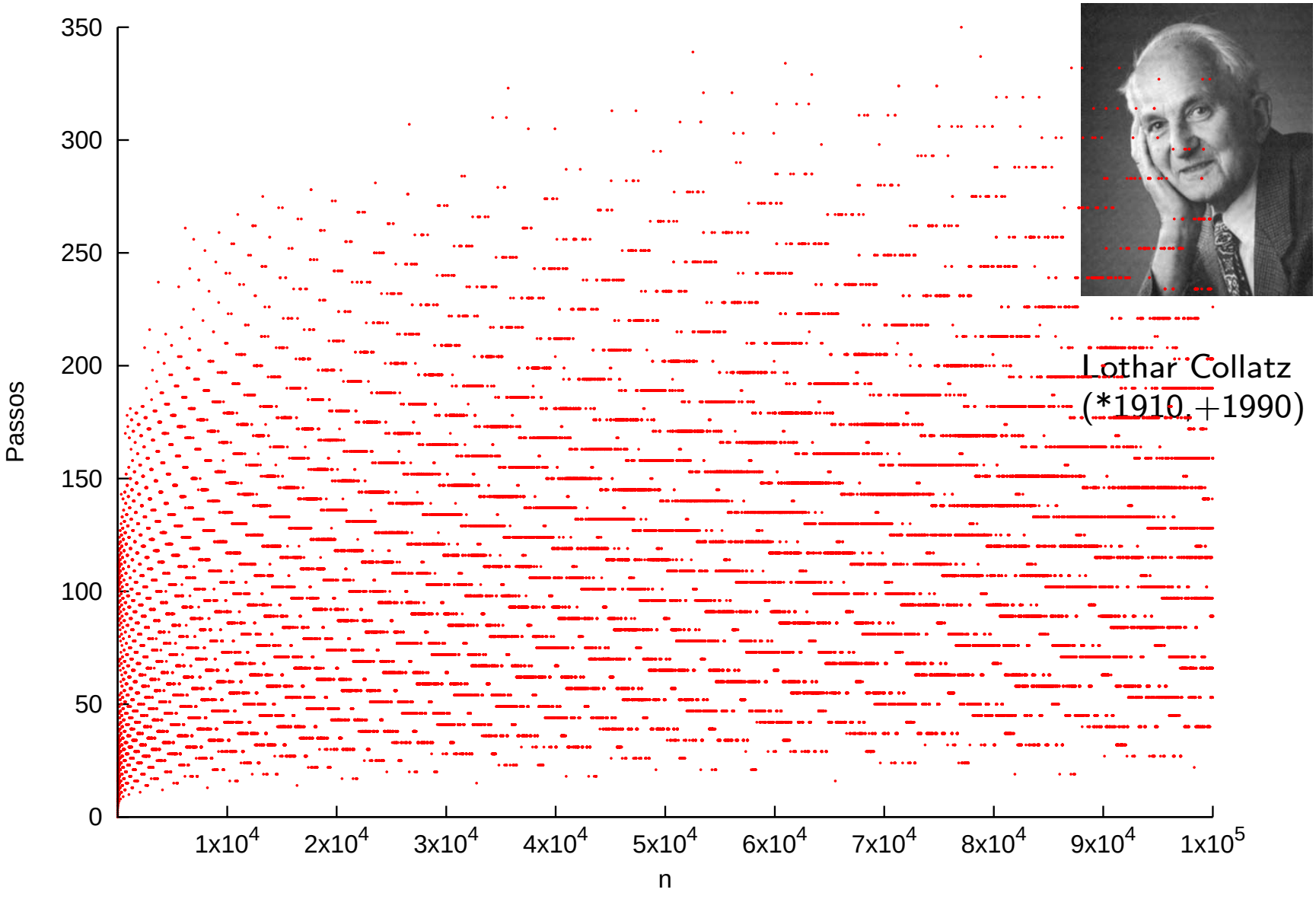
# Número de iterações?

## C

Entrada Número $x \in \mathbb{N}$.

```
1  while x ≠ 1 do
2     if x mod 2 = 0 then
3        x := x/2
4     else
5        x := 3x + 1
6     end if
7  end while
```

# Conjetura de Collatz






Lothar Collatz
(*1910, +1990)

## Bubblesort

**Entrada** Uma seqüência $a_1, \ldots, a_n$ de números inteiros.

**Saída** Uma seqüência $a_{\pi(1)}, \ldots, a_{\pi(n)}$ de números inteiros com $\pi$ uma permutação de $[1, n]$ tal que para $i < j$ temos $a_{\pi(i)} \leq a_{\pi(j)}$.

```
1  for i:=1 to n
2    for j:=1 to n−i
3      if a_j > a_{j+1} then
4        swap a_j , a_{j+1}
5      end if
6    end for
7  end for
```

$$\sum_{1 \leq i \leq n} \sum_{1 \leq j \leq n-i} O(1) = O(n/2(n-1)) = O(n^2)$$

## Busca Binária

**Entrada** Número $x \in \mathbb{Z}$ e seqüência ordenada $S = a_1, \ldots, a_n$

**Saída** Posição $i$ com $a_i = x$, ou $-1$ caso $x \notin S$.

```
1  i := 1; f := n; m := ⌊(f−i)/2⌋ + i
2  while i ≤ f do
3      if a_m = x then return m
4      if a_m < x then f := m − 1
5      else i := m + 1
6      m := ⌊(f−i)/2⌋ + i
7  end while
8  return −1
```

$$\sum_{1 \leq i \leq \log_2 n} O(1) = O(\log_2 n) = O(\log n)$$

# Dependência de valores

## P

Entrada Um número $n \in \mathbb{N}$.

```
1  r := 1
2  for i = 1, ..., n do
3      r := 2 r
4  end for
```

Complexidade?

# Dependência de valores

## P

Entrada Um número $n \in \mathbb{N}$.

```
1  r := 1
2  for i = 1, ..., n do
3      r := 2r
4  end for
```

Complexidade?

- O tamanho da entrada é $m = \Theta(\log n)$.
- Logo: a complexidade é $\Theta(2^m)$ **exponencial**.
- Algoritmo **pseudo-polinomial**: polinomial no valor, não no tamanho da entrada.

# Agenda

# Um contador

- Um contador binário com $k$ bits conta de 0 até $2^k - 1$.
- Um incremento: complexidade $O(k)$ no pior caso.
- Em total: complexidade $O(k2^k)$.
- Difícil: Estimar um limite para o número de 1's (OEIS A001511)

00000
00001
00010
00011
00100
00101
00110
00111
01000
01001
01010
01011
01100
01101
01110
01111
10000

# Solução: análise amortizada

- Pagaremos duas operações por bit setado
  - Uma para setar mesmo
  - Outra vai para conta.
- Zerando bits será pagado com operações da conta, e custa "nada".
- Logo: cada incremento custa $O(1)$, e o total é $O(2^k)$.
- **O custo amortizado é a média sobre o caso pessimista!**

# Solução: análise amortizada

- Temos custos reais $c_1, \ldots, c_n$ da uma série de operações
- Queremos o custo amortizado $\sum_i c_i/n$
- A "conta" é uma **função potencial** $\varphi_i$ do estado atual
  - Normalização $\varphi_1 = 0$ ("conta vazio")
  - $\varphi \geq 0$ ("não emprestar dinheiro")
- Trabalharemos com o custo amortizado

$$a_i = c_i - \varphi_i + \varphi_{i+1}$$

- Com isso

$$\sum a_i = \sum c_i - \varphi_i + \varphi_{i+1} = \varphi_{n+1} - \varphi_1 + \sum c_i \geq \sum c_i$$

# Análise amortizada do contador

- Contador: $\varphi(i)$ é o número de bits na representação de $i$
- Custo de um incremento de

$$e_1 : \quad \underbrace{\cdots}_{q \text{ bits um}} \; 0 \underbrace{11\cdots1}_{p \text{ bits um}}$$

para

$$e_2 : \quad \underbrace{\cdots}_{q} \; 1 \underbrace{00\cdots0}_{0} \quad ?$$

- Custo amortizado

$$c - \varphi(e_1) + \varphi(e_2) = p + 1 - p - q + 1 + q = 2 = O(1).$$

# Aplicações da análise amortizada

- Análise de **operações sobre estruturas de dados**

Dois exemplos:

- Inserção numa tabela dinâmica: $\Theta(n)$ pessimista, $\Theta(1)$ amortizado
- Deleção num heap Fibonacci: $O(\log n)$ pessimista, $O(1)$ amortizado

## Busca em Largura

**Entrada** Grafo direcionado $G = (V, A)$, nó origem $s$.

```
1  for cada vértice u ∈ V \ {s} do
2      c_u := Branco
3  c_s := Cinza
4  Q := {s}
5  while Q ≠ ∅ do
6      seleciona u ∈ Q; Q := Q \ {u}
7      for v ∈ N(u) do
8          if c_v = Branco then
9              c_v = Cinza
10             Q := Q ∪ {v}
11         end if
12     c_u = Preto
```

## Busca em Largura

**Entrada** Grafo direcionado $G = (V, A)$, nó origem $s$.

```
1  for cada vértice u ∈ V \ {s} do
2      c_u := Branco
3  c_s := Cinza
4  Q := {s}
5  while Q ≠ ∅ do
6      seleciona u ∈ Q;  Q := Q \ {u}
7      for v ∈ N(u) do
8          if c_v = Branco then
9              c_v = Cinza
10             Q := Q ∪ {v}
11         end if
12     c_u = Preto
```

Laço exterior: $O(|V|)$

## Busca em Largura

**Entrada** Grafo direcionado $G = (V, A)$, nó origem $s$.

```
1  for cada vértice u ∈ V \ {s} do
2      c_u := Branco
3  c_s := Cinza
4  Q := {s}
5  while Q ≠ ∅ do
6      seleciona u ∈ Q; Q
7      for v ∈ N(u) do
8          if c_v = Branco then
9              c_v = Cinza
10             Q := Q ∪ {v}
11         end if
12     c_u = Preto
```

Laço exterior: $O(|V|)$

Laço interior: $O(|V|)$

## Busca em Largura

**Entrada** Grafo direcionado $G = (V, A)$, nó origem $s$.

```
1  for cada vértice u ∈ V \ {s} do
2      c_u := Branco
3  c_s := Cinza
4  Q := {s}
5  while Q ≠ ∅ do
6      seleciona u ∈ Q; Q
7      for v ∈ N(u) do
8          if c_v = Branco then
9              c_v = Cinza
10             Q := Q ∪ {v}
11         end if
12     c_u = Preto
```

Laço exterior: $O(|V|)$

Laço interior: $O(|V|)$

Total: $O(|V|^2)$?

## Busca em Largura

**Entrada** Grafo direcionado $G = (V, A)$, nó origem $s$.

```
1  for cada vértice u ∈ V \ {s} do
2      c_u := Branco
3  c_s := Cinza
4  Q := {s}
5  while Q ≠ ∅ do
6      seleciona u ∈ Q; Q
7      for v ∈ N(u) do
8          if c_v = Branco then
9              c_v = Cinza
10             Q := Q ∪ {v}
11         end if
12     c_u = Preto
```

Laço exterior: $O(|V|)$

Laço interior: $O(\delta_u)$

Total: $O(|V|^2)$?

## Busca em Largura

**Entrada** Grafo direcionado $G = (V, A)$, nó origem $s$.

```
1  for cada vértice u ∈ V \ {s} do
2      c_u := Branco
3  c_s := Cinza
4  Q := {s}
5  while Q ≠ ∅ do
6      seleciona u ∈ Q; Q
7      for v ∈ N(u) do
8          if c_v = Branco then
9              c_v = Cinza
10             Q := Q ∪ {v}
11         end if
12     c_u = Preto
```

Laço exterior: $O(|V|)$

Laço interior: $O(\delta_u)$

Total: $O(|V| + \sum_u \delta_u) = O(|V| + |E|)$

# Agenda

# Método de Divisão e Conquista

- **Dividir** o problema em subproblemas independentes
- **Conquistar** os subproblemas, resolvendo-os recursivamente
- **Combinar** as soluções dos subproblemas

# Divisão e conquista

## DC

Entrada Uma instância $I$ de tamanho $n$.

```
1  if n = 1 then
2    return Solução direta
3  else
4    Divide I em sub-instâncias
5      I_1, ..., I_k ,  k > 0
6      com tamanhos n_i < n .
7    Resolve recursivamente: I_1, ..., I_k .
8    Resolve I combinando sub-soluções
9      DC(I_1), ..., DC(I_k) .
10 end if
```

# Recursão natural

- Seja $d(n)$ o tempo para a divisão.
- Seja $s(n)$ o tempo para combinar a solução final.
- Podemos somar: $f(n) = d(n) + s(n)$ e obtemos

$$T(n) = \begin{cases} \Theta(1) & \text{para } n < n_0 \\ \sum_{1 \leq i \leq k} T(n_i) + f(n) & \text{caso contrário.} \end{cases}$$

Problema: como achar a solução de uma recorrência?

## Theorem (Akra-Bazzi e Leighton)

*Dado a recorrência*

$$T(x) = \begin{cases} \Theta(1) & \text{se } x \leq x_0 \\ \sum_{1 \leq i \leq k} a_i T(b_i x + h_i(x)) + g(x) & \text{caso contrário} \end{cases}$$

*com constantes* $a_i > 0$, $0 < b_i < 1$ *e funções* $g$, $h$, *tal que*

$$|g'(x)| \in O(x^c); \qquad |h_i(x)| \leq x/\log^{1+\epsilon} x$$

*para um* $\epsilon > 0$ *e a constante* $x_0$ *e suficientemente grande*

$$T(x) \in \Theta\left(x^p \left(1 + \int_1^x \frac{g(u)}{u^{p+1}} du\right)\right)$$

*com* $p$ *tal que* $\sum_{1 \leq i \leq k} a_i b_i^p = 1$.

# Caso particular: pisos e tetos

$$T(x) = \begin{cases} \Theta(1) & \text{se } x \leq x_0 \\ \sum_{1 \leq i \leq k} a_i T(b_i x + h_i(x)) + g(x) & \text{caso contrário} \end{cases}$$

- Os $h_i(x)$ permitem pequenas pertubações.
- Usando $h_i(x) = \lfloor b_i x \rfloor - b_i x$ obtemos

$$T(x) = \begin{cases} \Theta(1) & \text{se } x \leq x_0 \\ \sum_{1 \leq i \leq k} a_i T(\lfloor b_i x \rfloor) + g(x) & \text{caso contrário} \end{cases}$$

# Exemplo 1: Mergesort

## Mergesort

**Entrada** Índices $p, r$ e um vetor $A$ com elementos $A_p, \ldots, A_r$

**Saída** $A$ com elementos em ordem não-decrescente, i.e. para $i < j$ temos $A_i \leq A_j$.

```
1  if  p < r  then
2       q = ⌊(p + r)/2⌋
3       MergeSort(A, p, q);
4       MergeSort(A, q+1, r);
5       Merge(A, p, q, r)
6  end  if
```

# Exemplo 1: Mergesort

$$T(n) = \begin{cases} \Theta(1) & \text{se } n \leq 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n) & \text{caso contrário} \end{cases}$$

e com $2^{-p} + 2^{-p} = 1$ temos $p = 1$ e

$$\begin{aligned} T(n) &= \Theta\left(n\left(1 + \int_1^n \frac{1}{u} du\right)\right) \\ &= \Theta(n \log n) \end{aligned}$$

# Exemplo 2: Multiplicação

$$p = \boxed{p_l}\boxed{p_r} = p_l\, 2^{n/2} + p_r$$
$$q = \boxed{q_l}\boxed{q_r} = q_l\, 2^{n/2} + q_r$$

Logo

$$\begin{aligned} pq &= (p_l\, 2^{n/2} + p_r)(q_l\, 2^{n/2} + q_r) \\ &= 2^n\, p_l q_l + 2^{n/2}\,(\,p_l q_r + p_r q_l\,) + p_r q_r \end{aligned}$$

Observação

$$p_l q_r + p_r q_l = (p_l + p_r)(q_l + q_r) - p_l q_l - p_r q_r$$

# Exemplo 2: Multiplicação

## mult-bin (Karatsuba and Ofman, 1962)

**Entrada** Dois números binários $p, q$ com $n$ bits.

**Saída** O produto $pq$ (com $\leq 2n$ bits).

```
1  if n = 1 then return pq
2  else
3     x1 := mult-bin(pl, ql)
4     x2 := mult-bin(pr, qr)
5     x3 := mult-bin(pl + pr, ql + qr)
6     return x1 2^n + (x3 − x2 − x1) 2^(n/2) + x2
7  end if
```

# Exemplo 2: Multiplicação

$$T(n) = \begin{cases} \Theta(1) & \text{se } n \leq 1 \\ 2T(\lceil n/2 \rceil) + T(\lceil n/2 \rceil + 1) + \Theta(n) & \text{caso contrário} \end{cases}$$

e com $3\,2^{-p} = 1$ temos $p = \log_2 3 \approx 1.58$ e

$$\begin{aligned} T(n) &= \Theta\left(n^p \left(1 + \int_1^n u^{-p} du\right)\right) \\ &= \Theta(n^p(1 + (n^{1-p}/(1-p) - 1/(1-p)))) \\ &= \Theta(c_1 n^p + c_2 n) \\ &= \Theta(n^{1.58}) \end{aligned}$$

# Exemplo 3: Multiplicação de matrizes

$$\left(\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right) \times \left(\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right) = \left(\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right).$$

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

# Exemplo 3: Multiplicação de matrizes

Observação de Strassen (1969):

$$
\begin{aligned}
M_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22})B_{11} \\
M_3 &= A_{11}(B_{12} - B_{22}) \\
M_4 &= A_{22}(B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12})B_{22} \\
M_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \\
C_{11} &= M_1 + M_4 - M_5 + M_7 \\
C_{12} &= M_3 + M_5 \\
C_{21} &= M_2 + M_4 \\
C_{22} &= M_1 - M_2 + M_3 + M_6
\end{aligned}
$$

# Exemplo 3: Multiplicação de matrizes

$$T(n) = \begin{cases} \Theta(1) & \text{se } n \leq 1 \\ 7T(n/2) + \Theta(n^2) & \text{caso contrário} \end{cases}$$

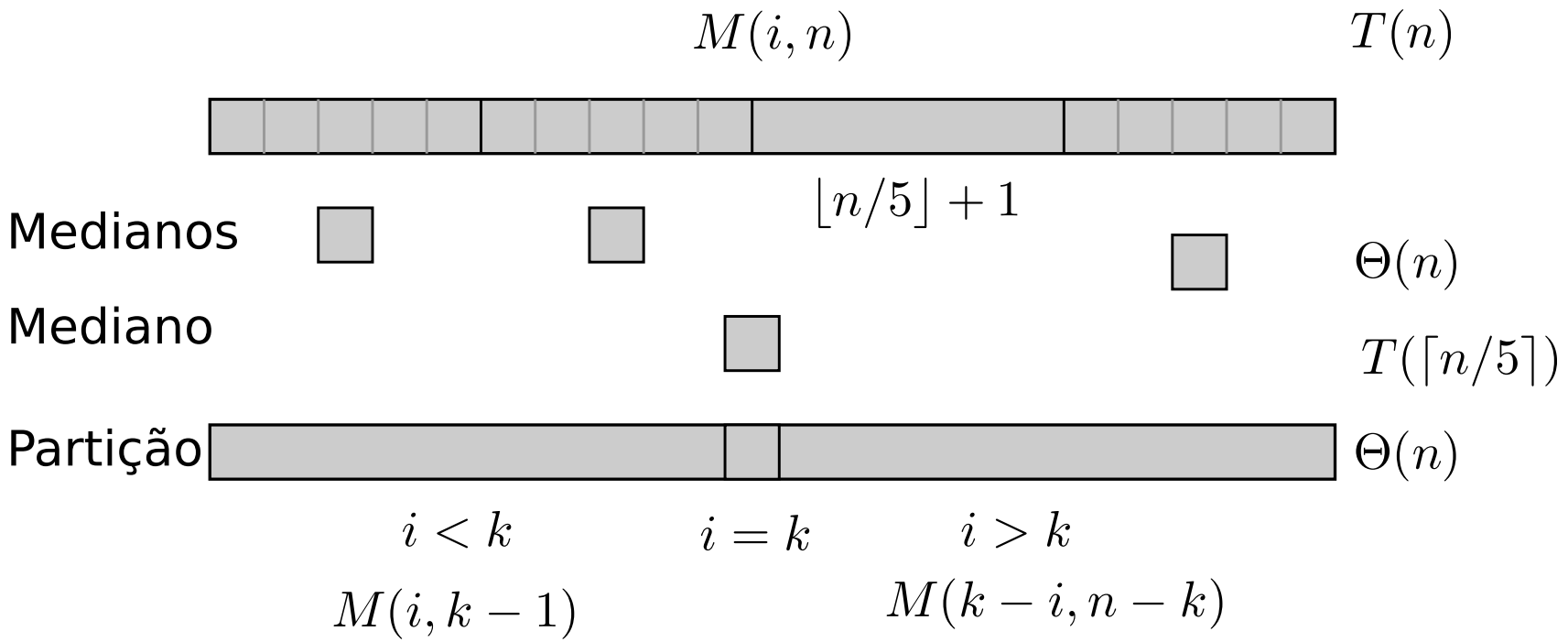
e com $7\,2^{-p} = 1$ temos $p = \log_2 7 \approx 2.81$ e

$$\begin{aligned} T(n) &= \Theta\left(n^p \left(1 + \int_1^n u^{1-p} du\right)\right) \\ &= \Theta(n^p(1 + (n^{2-p}/(2-p) - 1/(2-p)))) \\ &= \Theta(c_1 n^p + c_2 n^2) \\ &= \Theta(n^{2.81}) \end{aligned}$$

# Exemplo 4: Seleção

- Tarefa: Seleciona o $k$-ésimo elemento de $n$ elementos não ordenados.
- Com ordenação: $O(n \log n)$.

# Exemplo 4: Seleção



# Exemplo 4: Seleção

- Número de medianos: maior que $n/5 - 1$
- Número de medianos menor que posição $k$: maior que $n/10 - 2$
- Número de elementos menor que posição $k$: maior que $3n/10 - 6$
- Número de elementos maior que posição $k$: menor que $7n/10 + 6$
- Portanto

$$T(n) = \begin{cases} \Theta(1) & \text{se } n \leq 5 \\ T(\lceil n/2 \rceil) + \Theta(7n/10 + 6) + \Theta(n) & \text{caso contrário} \end{cases}$$

# Exemplo 4: Seleção

$$T(n) = \begin{cases} \Theta(1) & \text{se } n \leq 5 \\ T(\lceil n/2 \rceil) + \Theta(7n/10 + 6) + \Theta(n) & \text{caso contrário} \end{cases}$$

e com $5^{-p} + (7/10)^p = 1$ temos $p = \log_2 7 \approx 0.84$ e

$$\begin{aligned} T(n) &= \Theta\left(n^p \left(1 + \int_1^n u^{-p} du\right)\right) \\ &= \Theta(n^p(1 + (n^{1-p}/(1-p) - 1/(1-p)))) \\ &= \Theta(c_1 n^p + c_2 n) \\ &= \Theta(n) \end{aligned}$$

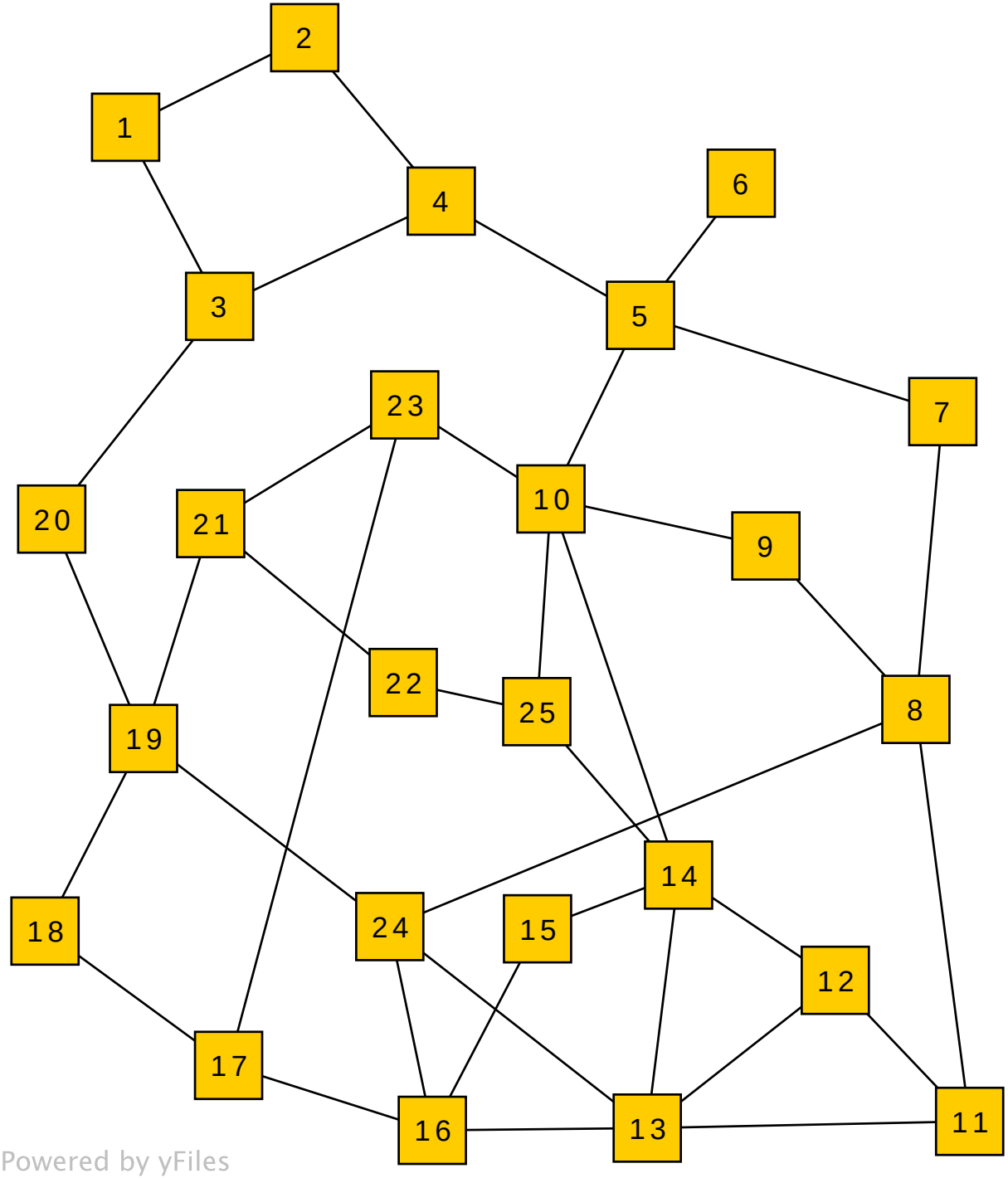
# Exemplo

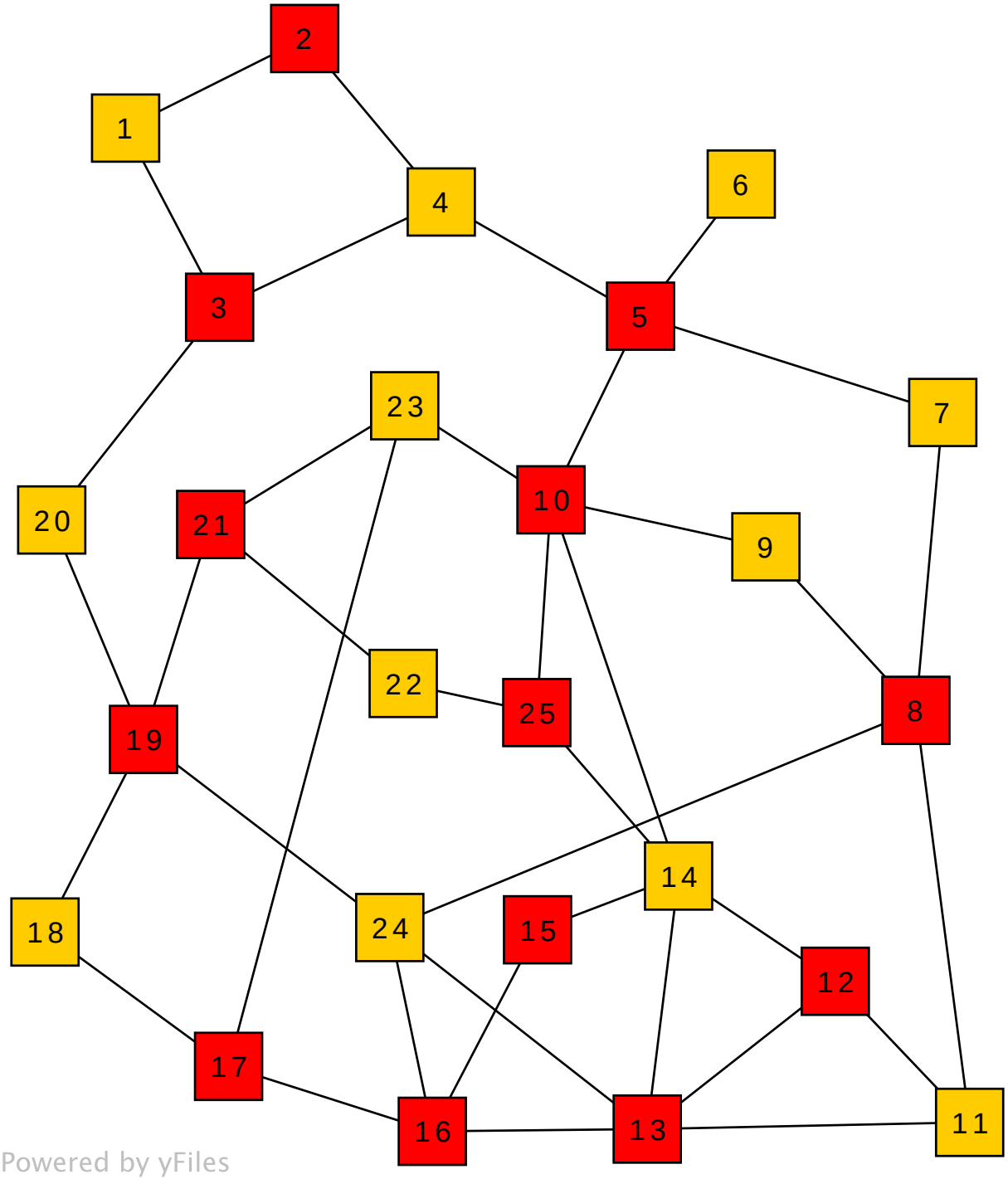
## Cobertura por vértices

**Instância** Um grafo não-direcionado $G = (V, A)$.

**Solução** Uma cobertura $C$, i.e. um conjunto $C \subseteq V$ tal que $\forall a \in A : \ a \cap C \neq \emptyset$.

**Objetivo** Minimizar $|C|$.





# Força bruta

## mvc

**Entrada** Um grafo não-direcionado $G = (V, A)$.

**Saída** A menor cobertura.

```
1  if A = ∅ return ∅
2  seleciona {u, v} ∈ A não coberta
3  C1 := {u} ∪ mvc(G \ {u})
4  C2 := {v} ∪ mvc(G \ {v})
5  return a menor cobertura C1 ou C2
```

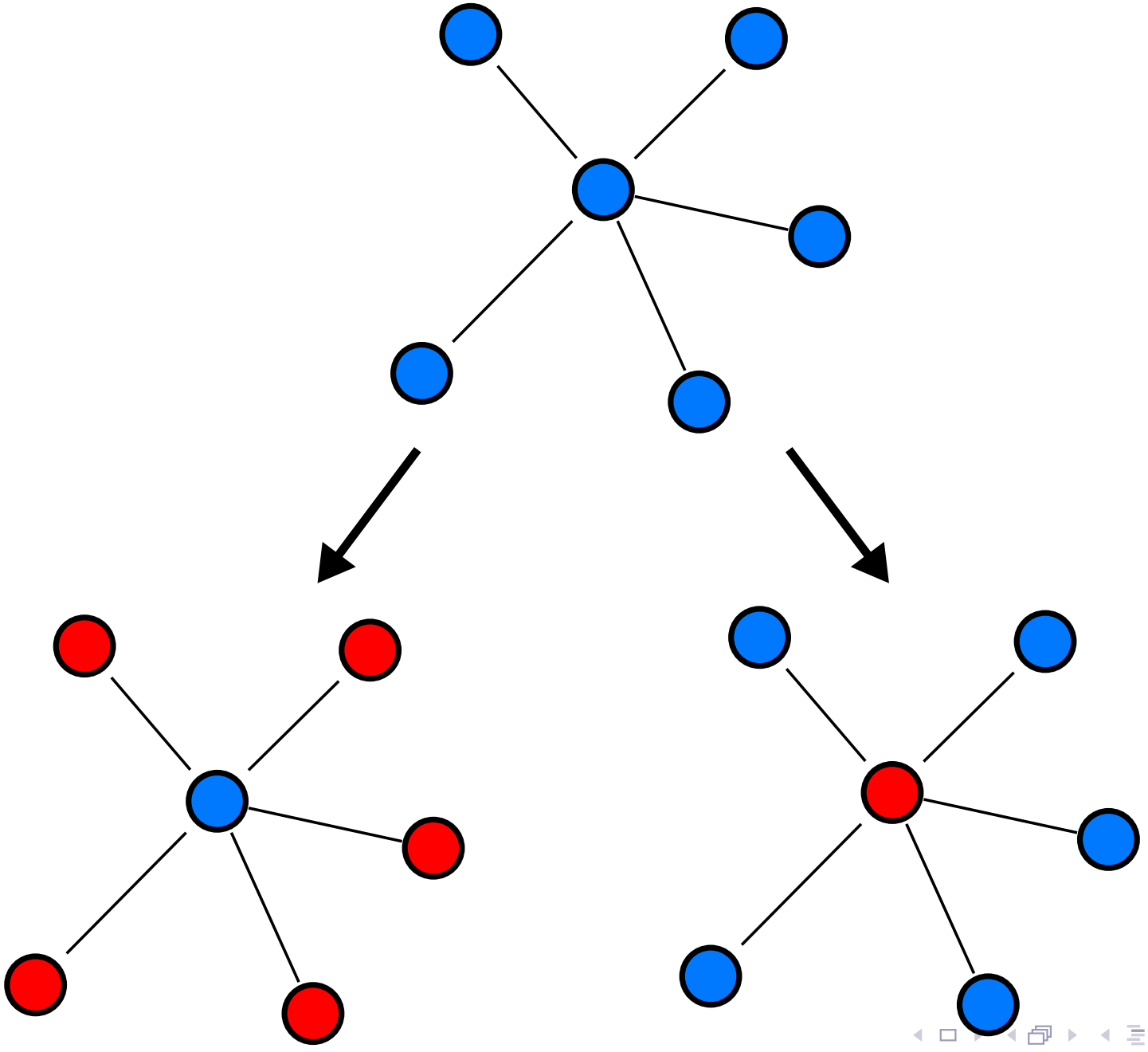
$$T_n = 2T_{n-1} + \Theta(n) = \Theta(2^n)$$

# Um algoritmo melhor para cobertura

Observação:

- Caso o grau máximo $\Delta$ de $G$ é 2, o problema pode ser resolvido em tempo $O(n)$, porque $G$ é uma coleção de caminhos simples e ciclos.
- Caso contrário, temos ao menos um vértice $v$ de grau $\delta_v \geq 3$. Ou esse vértice faz parte da cobertura mínima, ou todos seus vizinhos $N(v)$

$$\delta_v \geq 3$$



# Um algoritmo melhor para cobertura

```
1  mvc'(G) :=
2     if Δ(G) ≤ 2 then
3        determina a cobertura mínima C em tempo O(n)
4        return C
5     end if
6     seleciona um vértice v com grau δ_v ≥ 3
7     C_1 := {v} ∪ mvc'(G \ {v})
8     C_2 := N(v) ∪ mvc'(G \ N(v))
9     return a menor cobertura entre C_1 e C_2
```

# Análise do algoritmo

- Existem mais folhas que nós internos
- Para simplificar vamos só contar folhas.

O número de folhas obedece

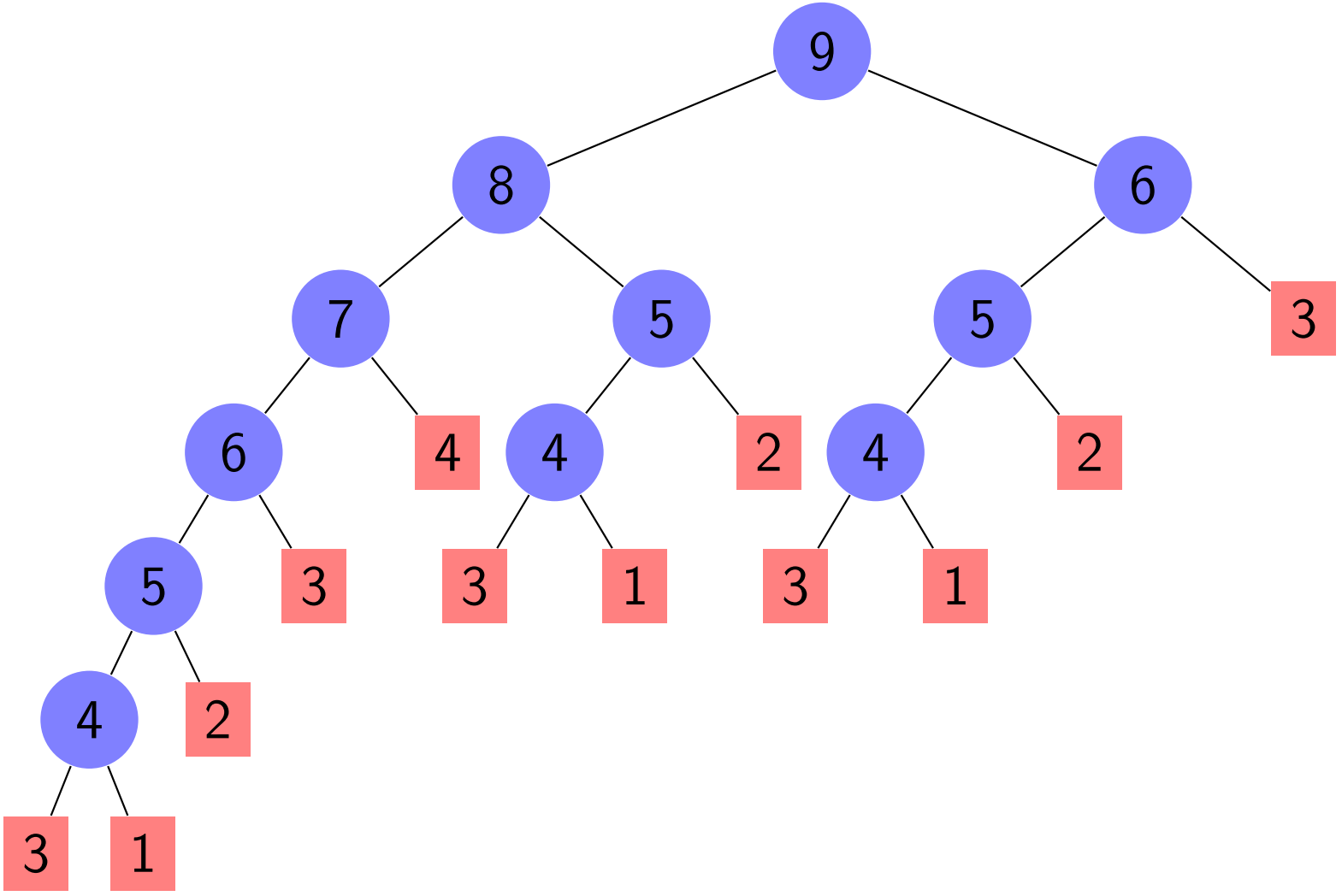
$$T(n) \leq T(n-1) + T(n-3)$$

No caso geral queremos resolver

$$T(n) = T(n-d_1) + T(n-d_2) + \cdots + T(n-d_k)$$

$(d_1, \ldots, d_k)$ é o **vetor de bifurcação**.

# Árvore de busca para $n = 9$



# Analisar árvores de busca

## Theorem (Graham et al. (1988))

*Dado a recorrência*

$$T(n) = \begin{cases} \Theta(1) & n \leq \max_{1 \leq i \leq k} d_i \\ \sum_i \alpha_i T(n - d_i) & \textit{caso contrário} \end{cases}$$

*seja* $\alpha$ *a raiz com a maior valor absoluto com multiplicidade* $l$ *do* **polinômio característico**

$$z^d - \alpha_1 z^{d-d_1} - \cdots - \alpha_k z^{d-d_k}$$

*com* $d = \max_k d_k$. *Então*

$$T(n) = \Theta(n^{l-1}\alpha^n) = \Theta^*(\alpha^n).$$

# Analise do algoritmo para cobertura por vértices

- Vetor de bifurcação $(1, 3)$
- Polinômio característico $z^3 = z^2 + 1$
- Maior raiz $\alpha \approx 1.47$
- Número de folhas

$$\Theta(\alpha^n).$$

- Complexidade

$$O(\alpha^n n).$$

# 3-SAT

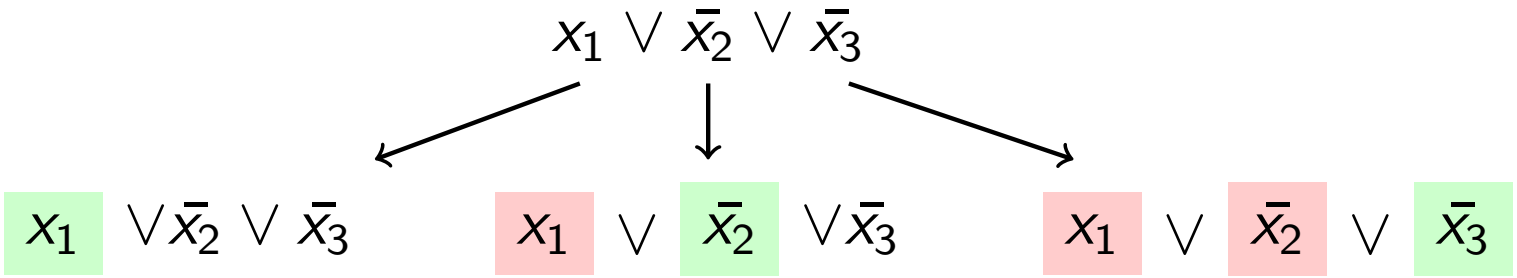
- Fórmula em **forma normal conjuntiva** com 3 literais por claúsula

$$\varphi(x_1, \ldots, x_n) = \bigwedge_i l_{i1} \vee l_{i2} \vee l_{i3}$$

- Problema: Existe **atribuição** $a : x_i \mapsto \{F, V\}$ que satisfaz $\varphi$?
- Força bruta: $\Theta(2^n)$ testes.

# 3-SAT

- Árvore de busca (Monien and Speckenmeyer, 1985)




$$T(n) = T(n\text{-}1) + T(n\text{-}2) + T(n\text{-}3)$$

- Vetor de bifurcação $(1, 2, 3)$.
- Polinômio característico: $z^3 - z^2 - z^1 - 1$.
- Maior raiz: $\alpha \approx 1.84$.
- Número de folhas: $\Theta(\alpha^n)$.
- Complexidade: $O(\alpha^n)$.

# Outro exemplo

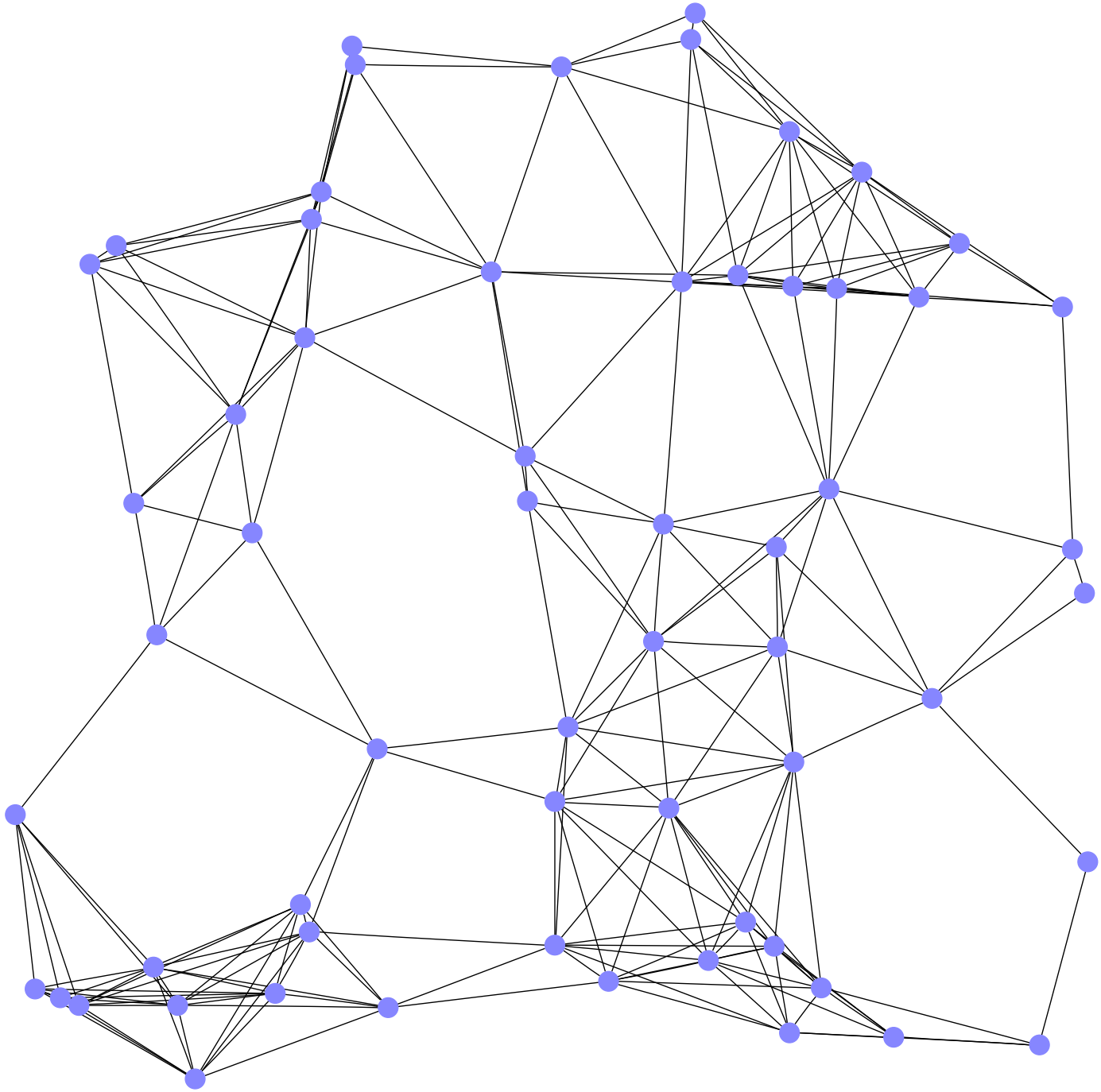
## Conjunto independente máximo

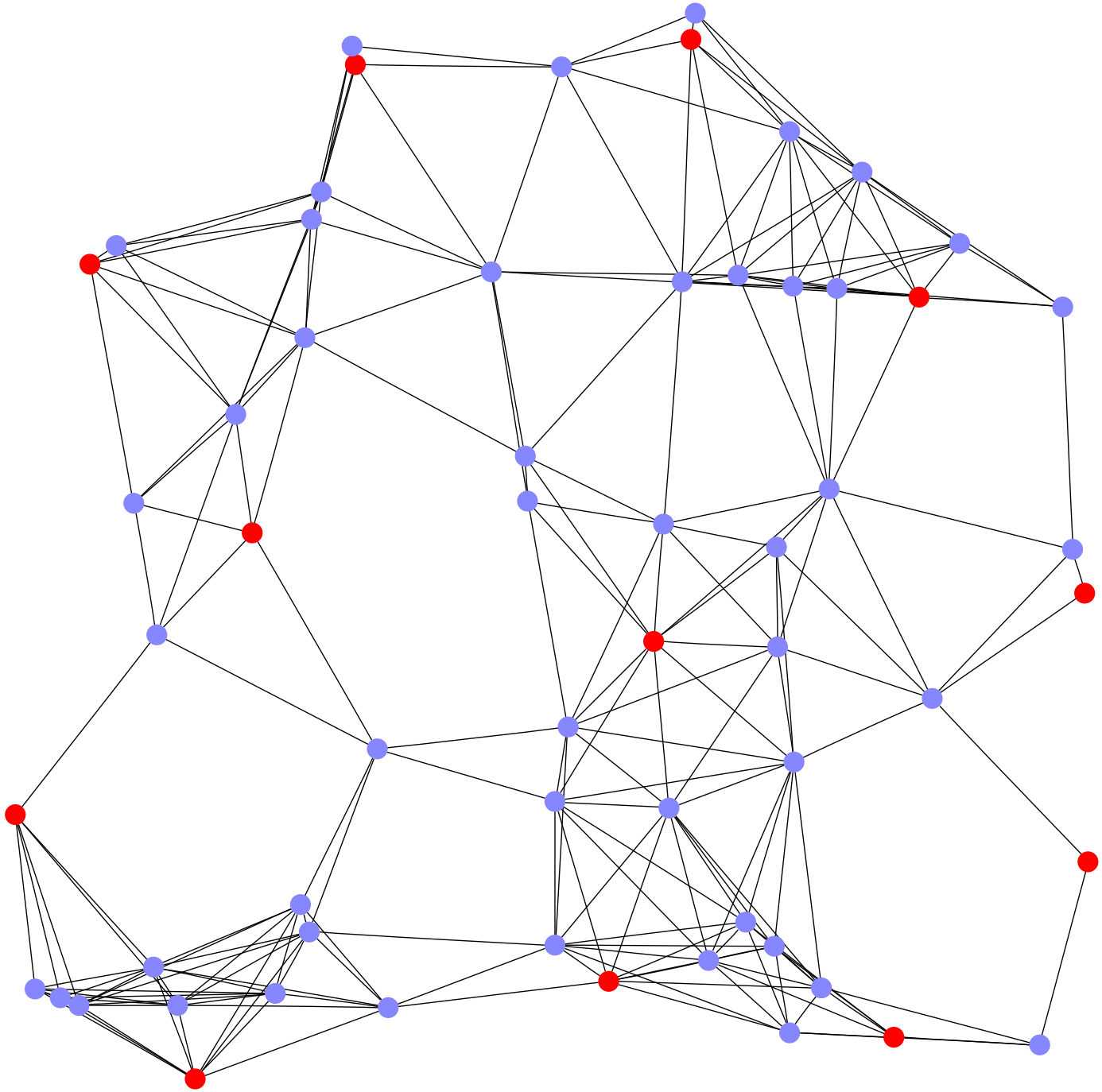
**Instância** Um grafo não-direcionado $G = (V, E)$.

**Solução** Um conjunto **independente** $M \subseteq V$, i.e. todo $m_1, m_2 \in V$ temos $\{m_1, m_2\} \notin E$.

**Objetivo** Maximiza a cardinalidade $|M|$.

- Força bruta: $\Theta(2^n)$ testes.

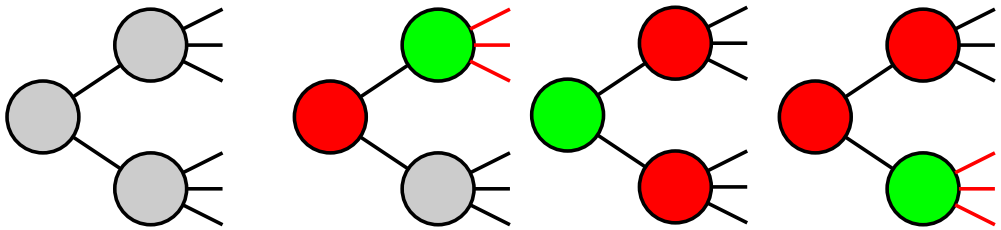


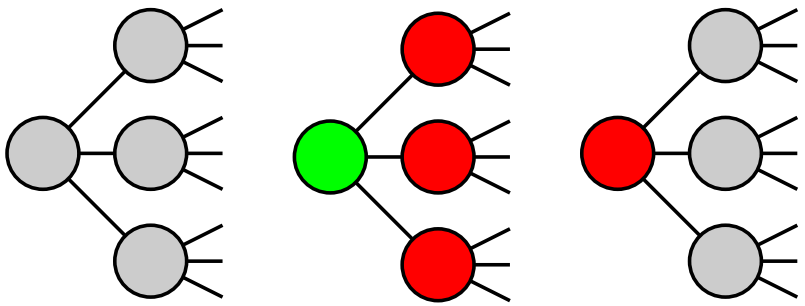


# Uma abordagem por análise de casos

$\delta_v = 0$  $T_n = T_{n-1}$

$\delta_v = 1$  $T_n = T_{n-2}$

$\delta_v = 2$  $T_n = 3T_{n-3}$

$\delta_v \geq 3$  $T_n = T_{n-4} + T_{n-1}$

# Análise do algoritmo

- Os primeiros dois casos não aumentam o número de folhas.
- Caso $\delta_v = 2$: Vetor de bifurcação $(3, 3, 3)$ com solução $O(1.45^n)$.
- Caso $\delta_v \geq 3$: Vetor de bifurcação $(4, 1)$ com solução $O(1.39^n)$.
- Em soma: Complexidade pessimista $O(1.45^n)$.

# Parte III

# Teoria de complexidade

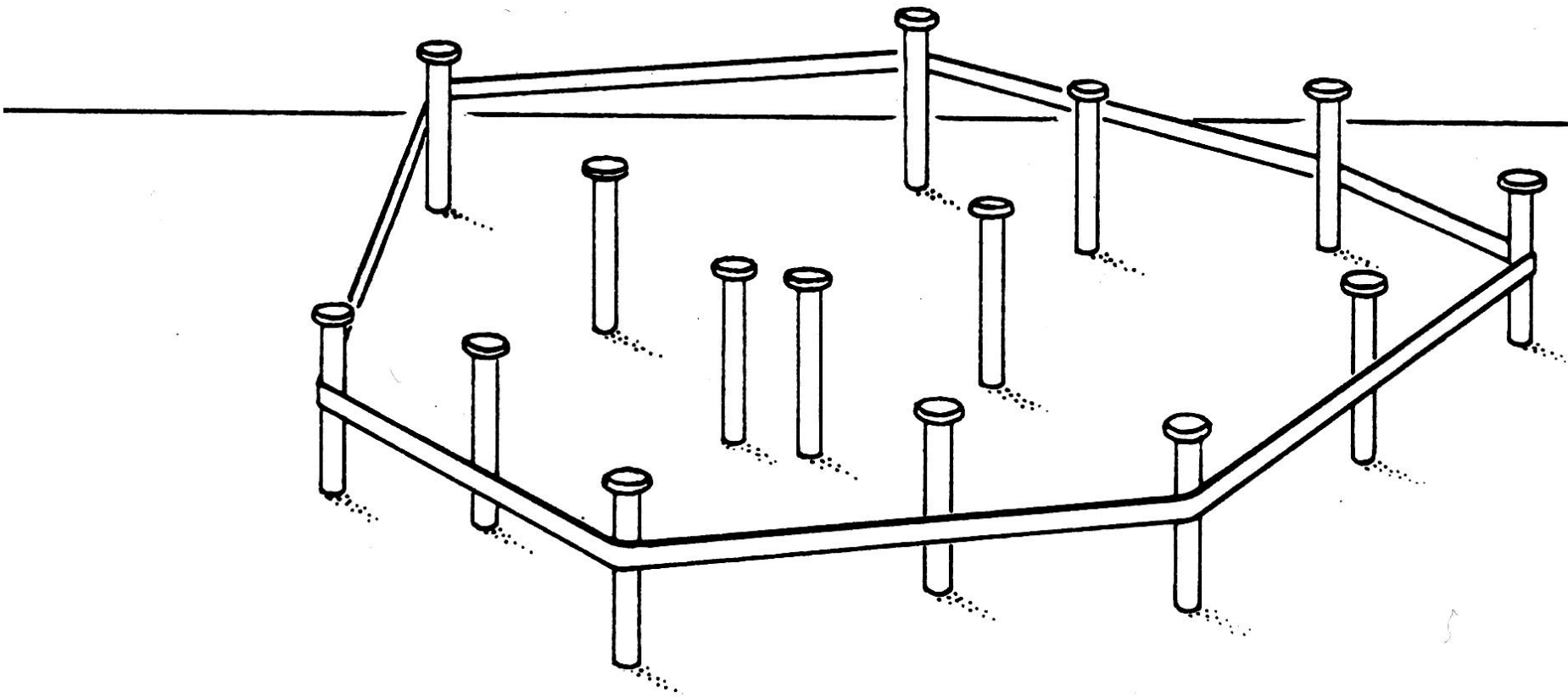
# Agenda

# Envoltória convexa de $n$ pontos em $O(n)$?

Envoltória convexa de $n$ pontos em $O(n)$?

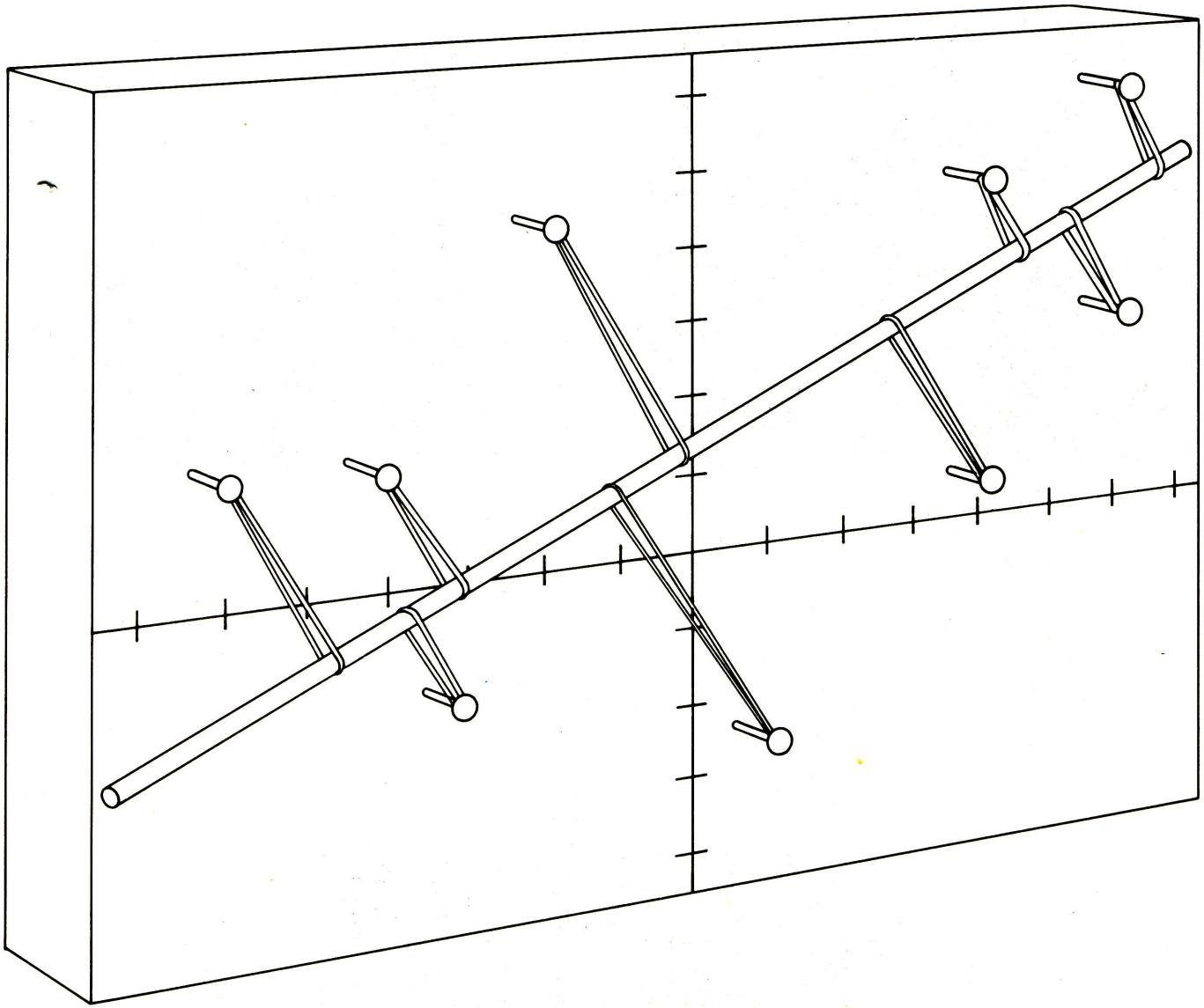
Solução tradicional: $O(n \log n)$.

# Usa pregos!



# Regressão linear?

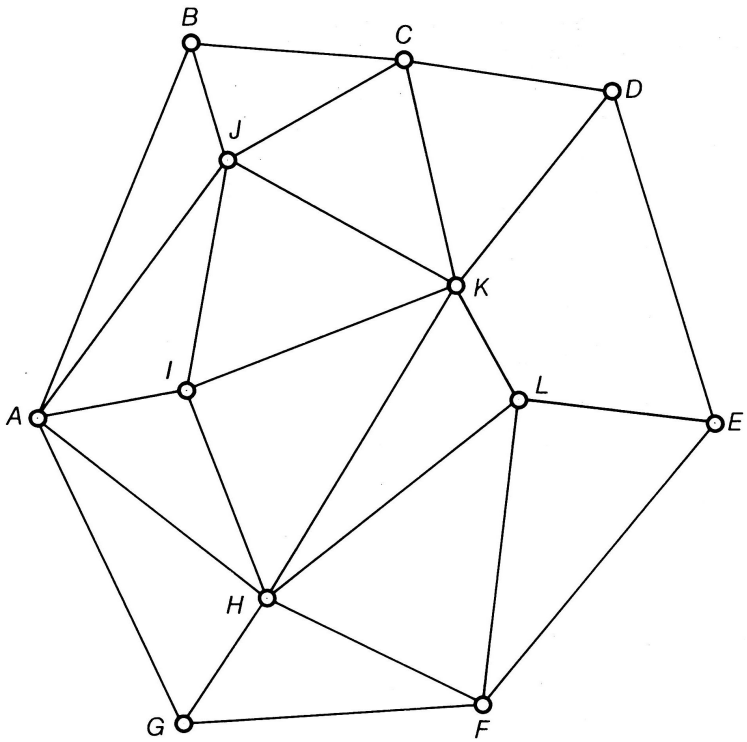
# A mesma coisa!



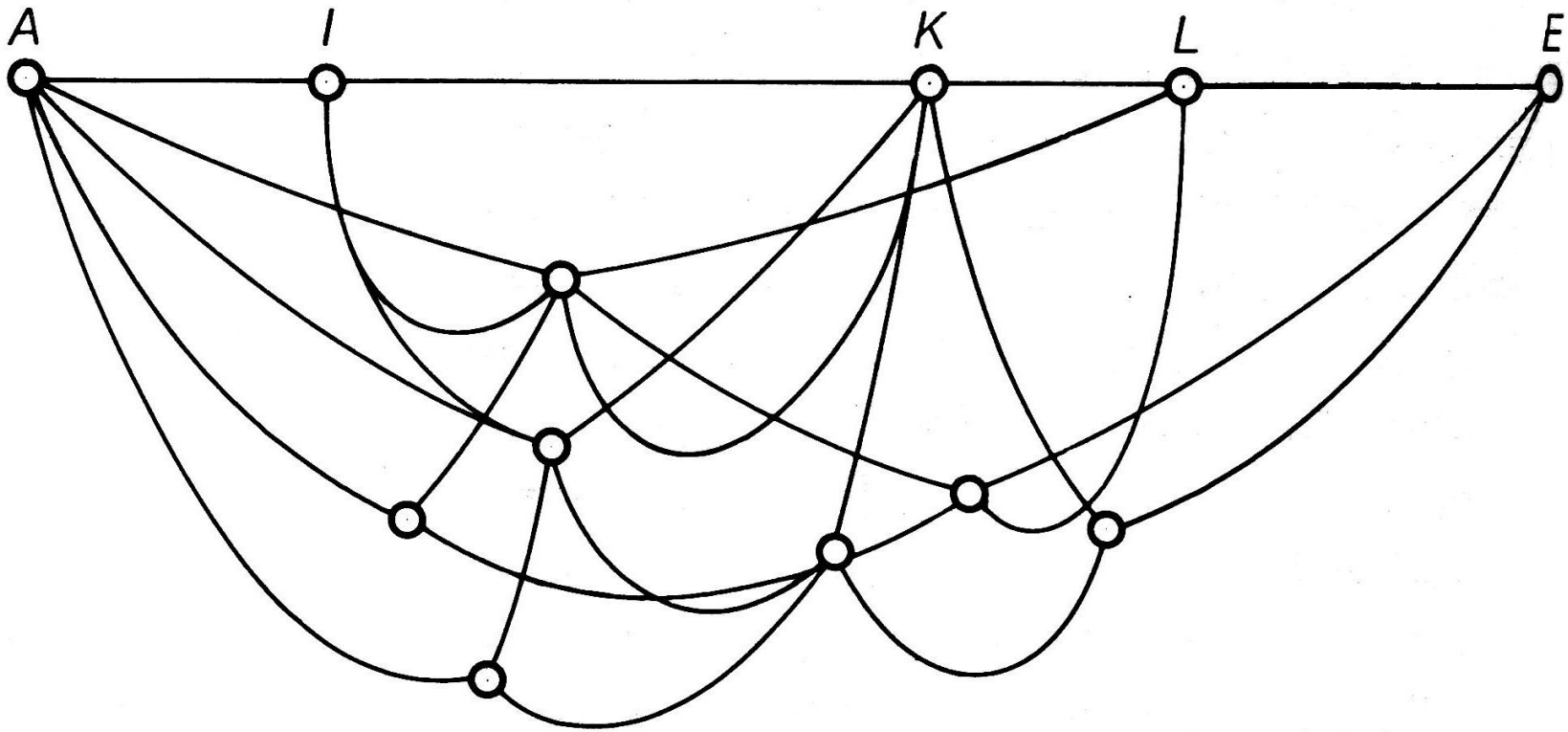
# Caminho mais curto entre um par de vértices em $O(m)$?

# Caminho mais curto entre um par de vértices em $O(m)$?

Solução tradicional: $O(n + m \log n)$.

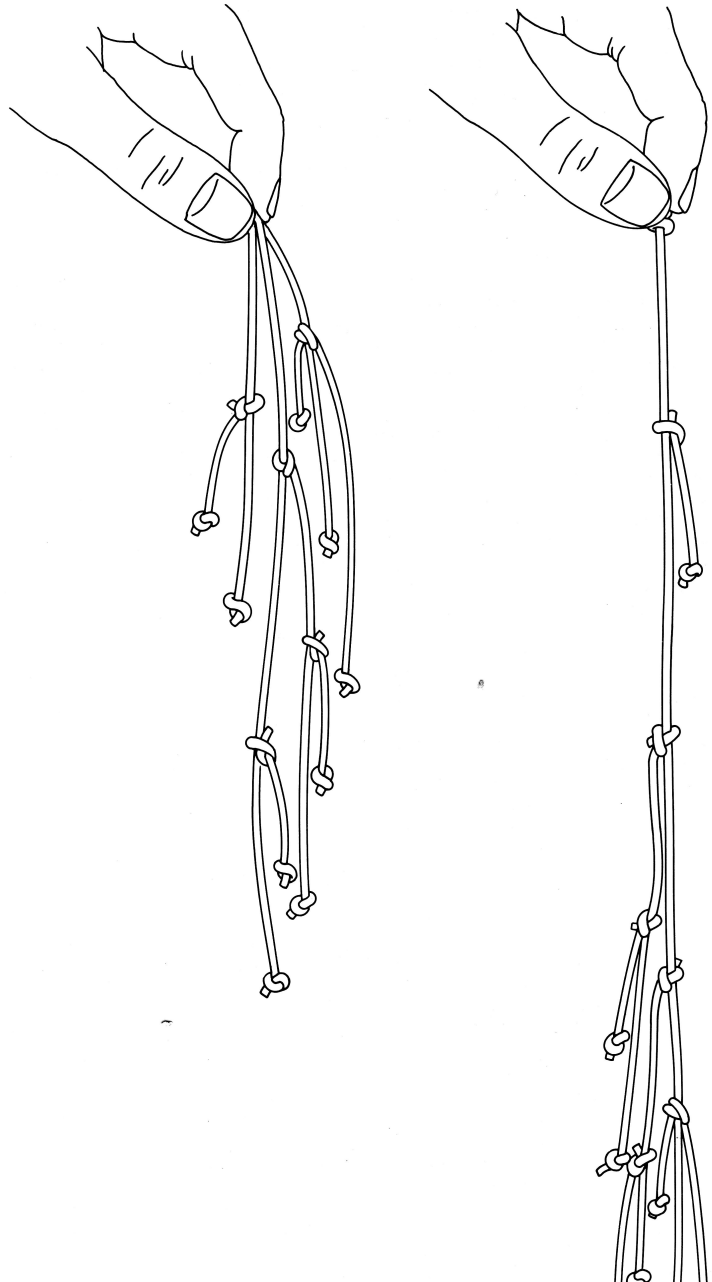


# Usa fios!



Caminho mais **longo** entre um par de vértices de uma árvore em $O(m)$?

# Fios novamente!



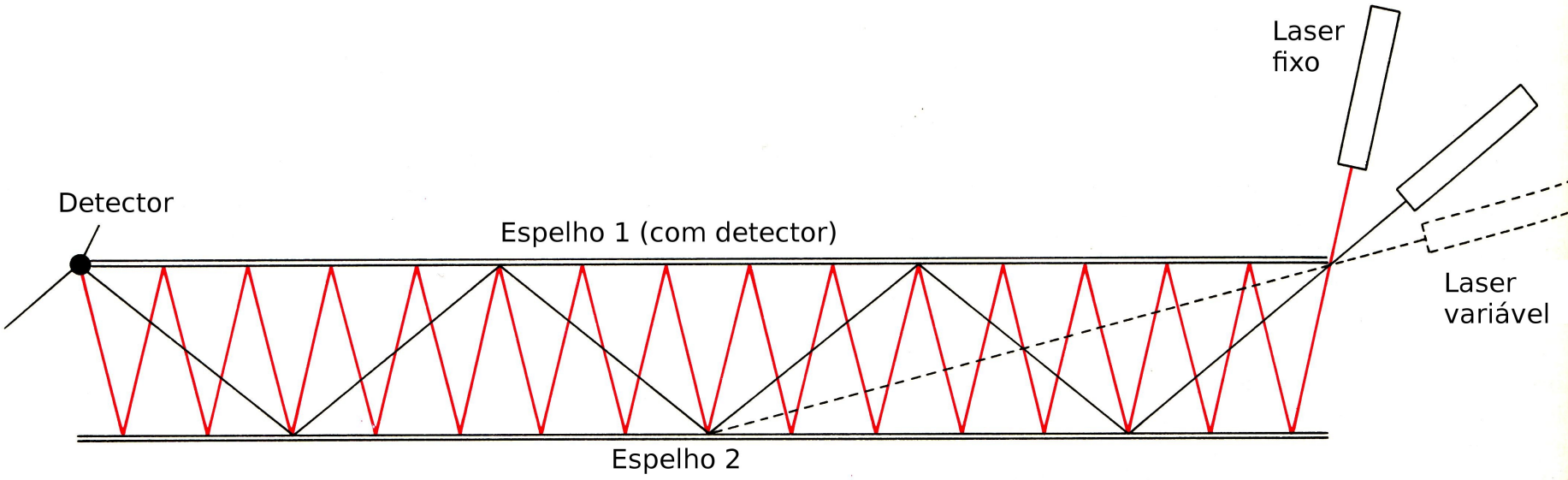
# Teste de primalidade?

# Teste de primalidade?

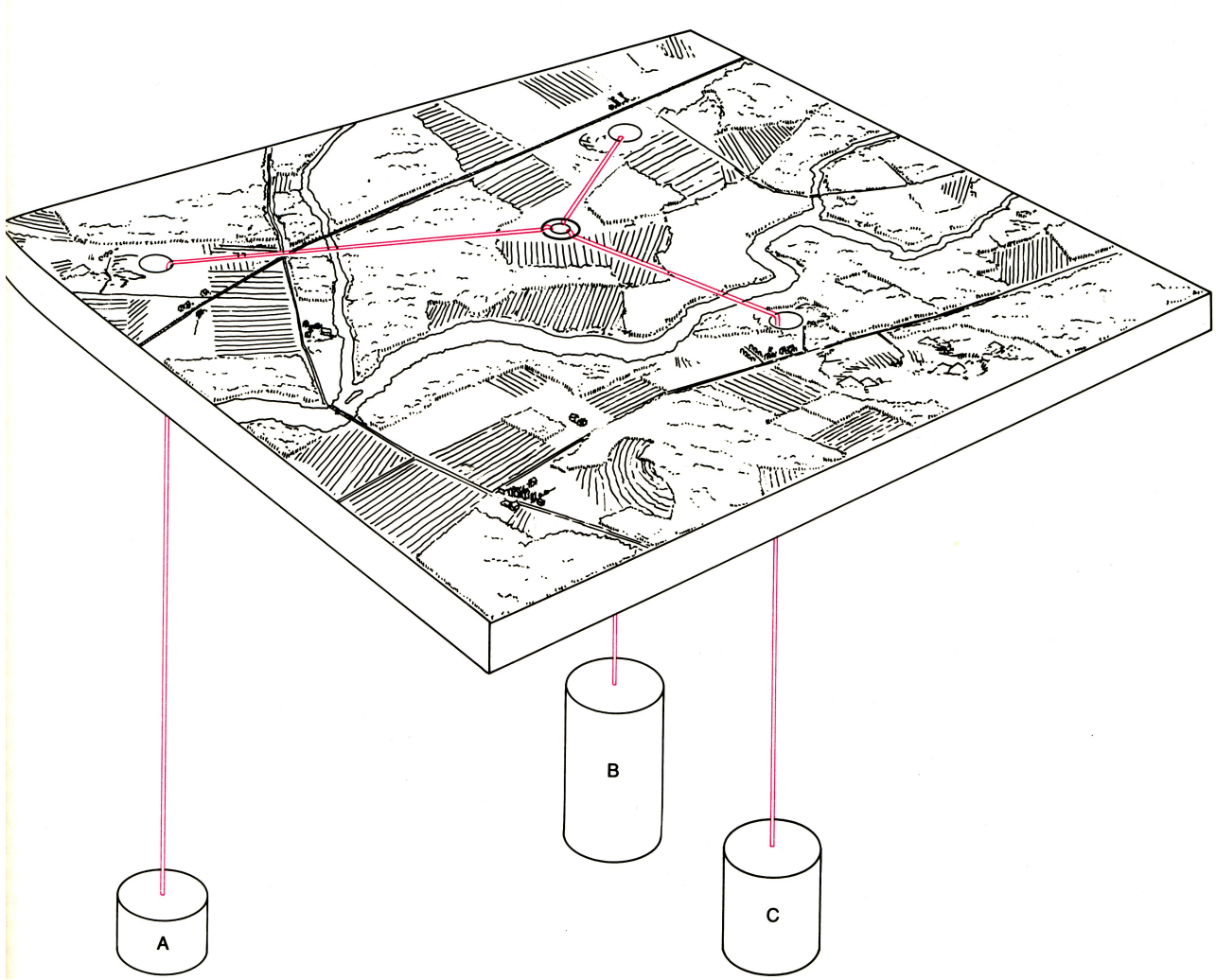
Soluções tradicionais:

- Randomizado: $\tilde{O}(k \log^2 n)$
  (Miller, Rabin 1980)
- Determinístico: $\tilde{O}(\log^6 n)$
  (Agrawal, Kayal, and Saxena, 2004)

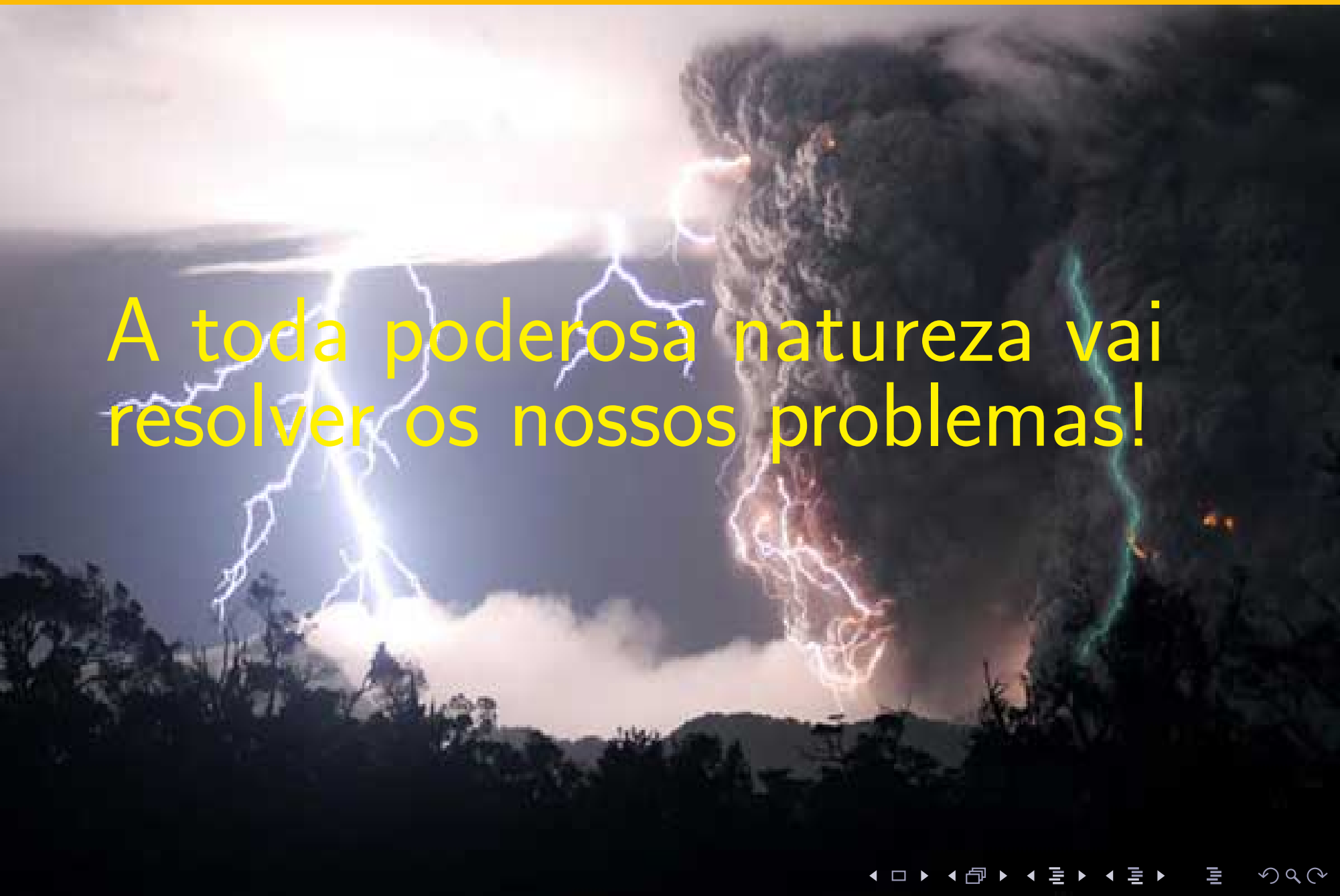
# Usa um laser!


Laser
fixo
Detector
Espelho 1 (com detector)
Laser
variável
Espelho 2

# E ainda localização de facilidades !



# A primeira resposta



# Agenda

# Agora um desafio para mamãe natureza

A questão que vale US$ 1.000.000:

$$\mathrm{P} = \mathrm{NP}?$$

# Agora um desafio para mamãe natureza

A questão que vale US$ 1.000.000:

$$P = NP?$$

**Solução simples: dividir por P, se possível**

$$P = 0 \vee N = 1$$

# Lembrança

## Classe de complexidade P

Problemas de decisão com solução em tempo polinomial.

## Classe de complexidade NP

Problemas de decisão que possuem um **certificado** para a resposta "sim" que pode ser verificado em tempo polinomial.

(NP não é não-polinomial.)

## Classe de complexidade co-NP

Problemas de decisão que possuem um **certificado** para a resposta "não" que pode ser verificado em tempo polinomial.

# Formalmente

Um problema pertence a

- DTIME($t(n)$) se existe uma MT determinística tal que decide-o em tempo $ct(n)$.
- NTIME($t(n)$) se existe uma MT não-determinística que decide-o em tempo $ct(n)$.
- DSPACE($s(n)$) se existe uma MT determinística que decide-o em espaço $cs(n)$.
- NSPACE($s(n)$) se existe uma MT não-determinística que decide-o de espaço $cs(n)$.

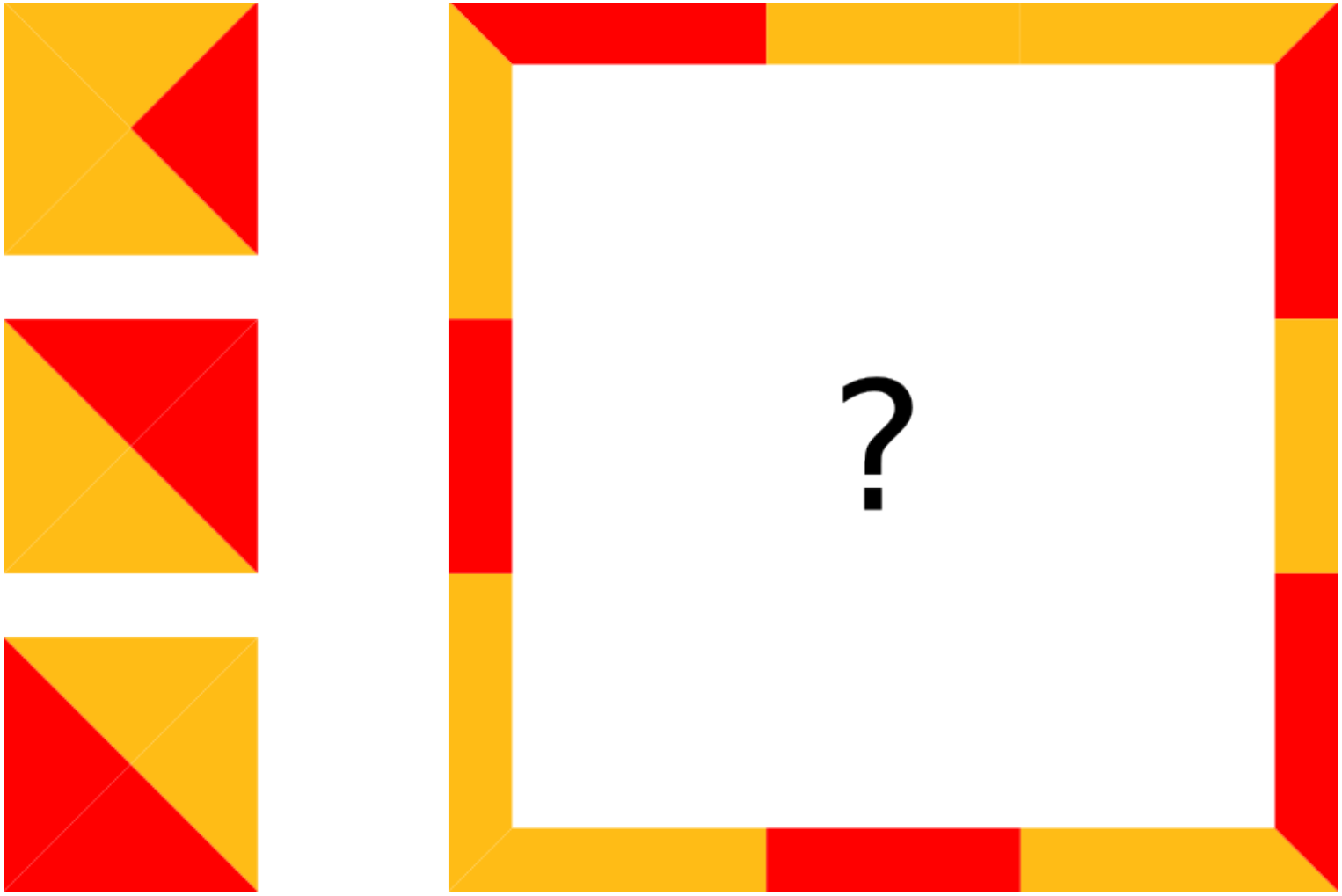
# Lembrança...

## Problemas NP-completos

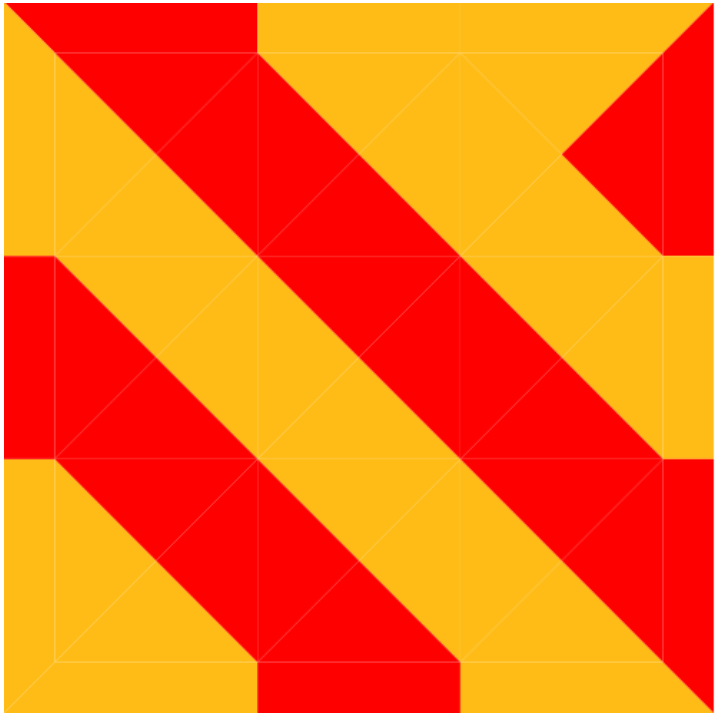
Os mais difíceis da classe NP: todo problema em NP pode ser **reduzido** para um problema NP-completo em tempo polinomial.

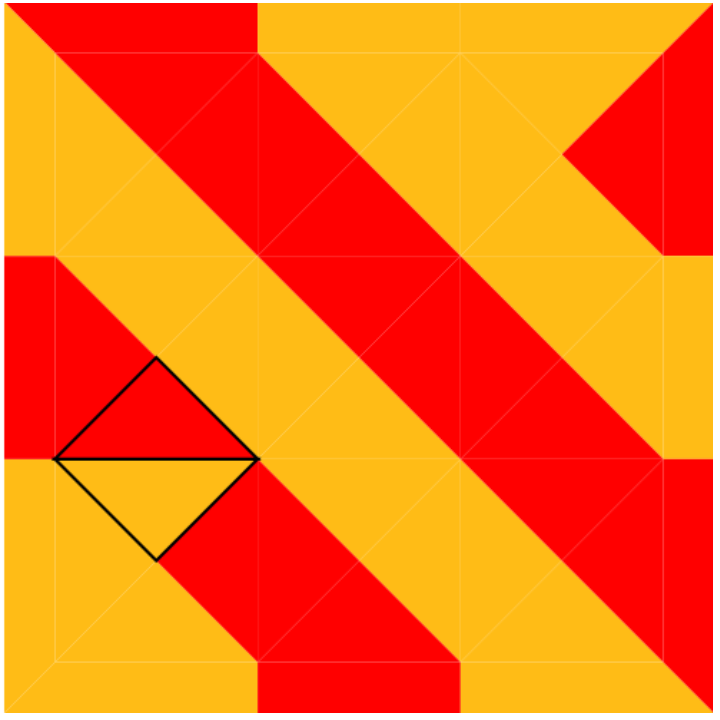
Em particular: resolvendo um problema NP-completo em tempo polinomial permite resolver **todos** em tempo polinomial.

# Ladrilhar: Um exemplo

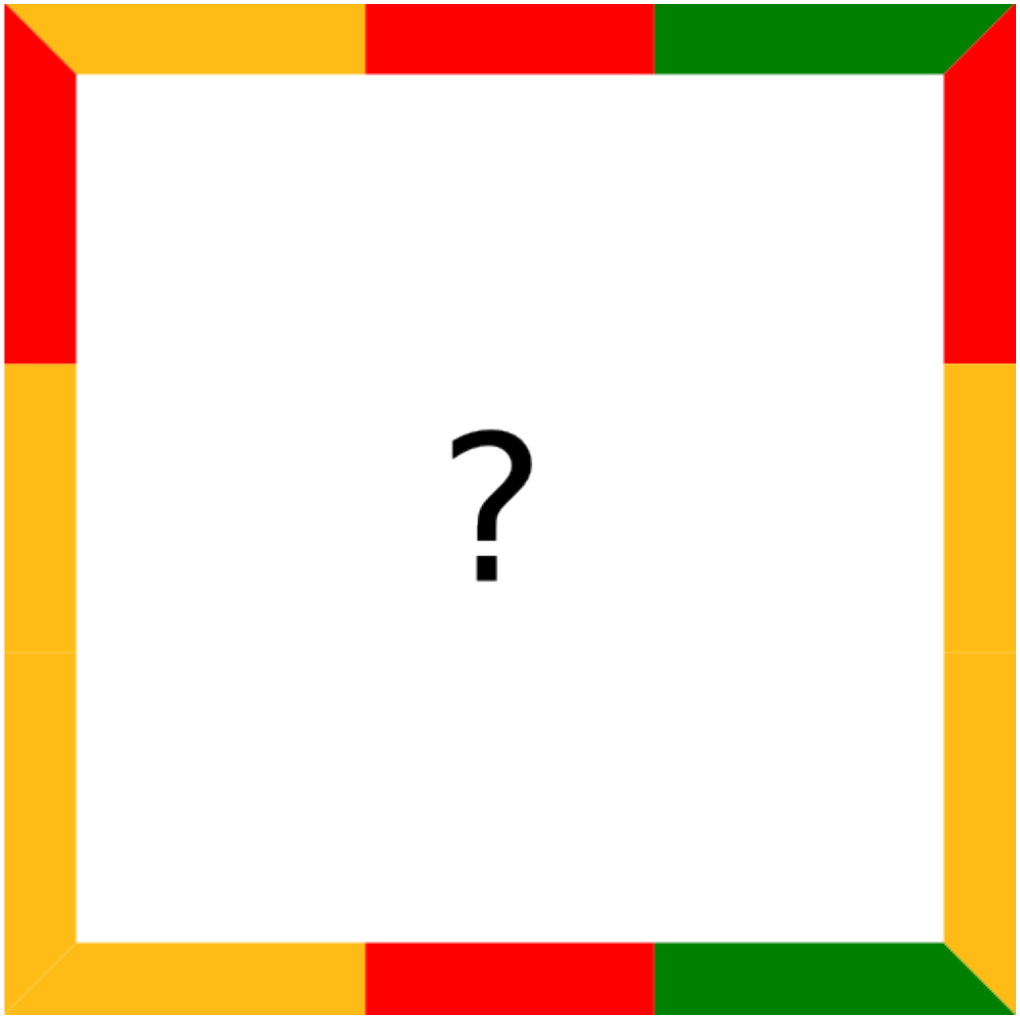


# Ladrilhar: Solução





# Ladrilhar: Exemplo



# Ladrilhar: O problema

- Para um conjunto finito de cores $C$, o **tipo** de um ladrilho é uma função $t : \{N, W, S, E\} \to C$.

## Ladrilhamento

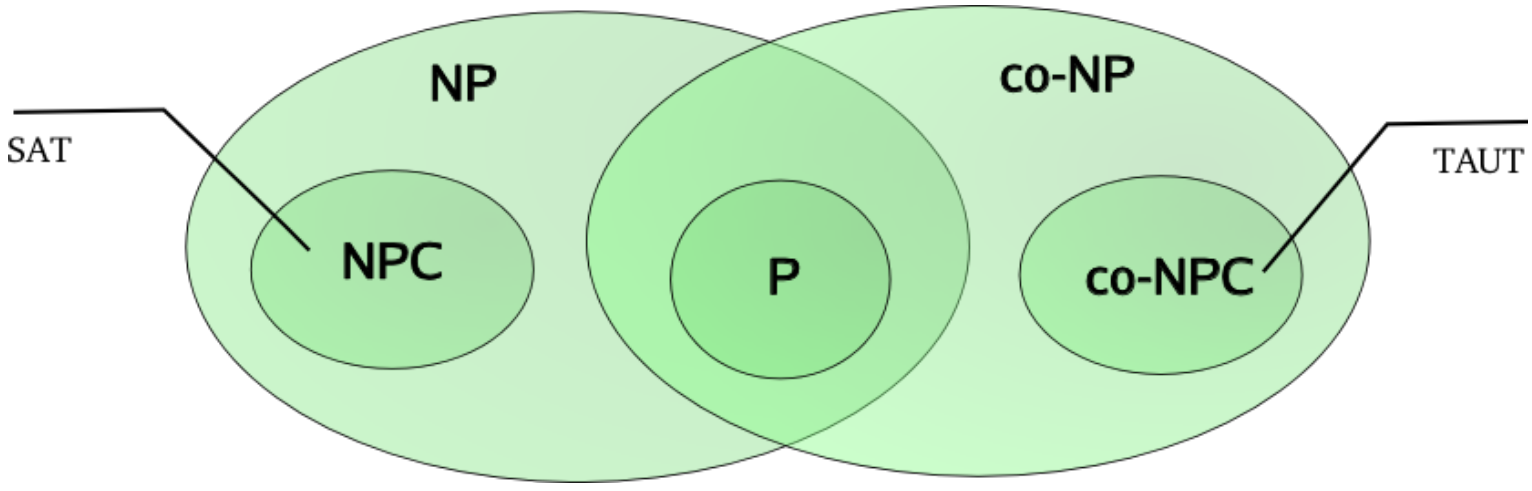
**Instância** Tipos de ladrilhos $t_1, \ldots, t_k$ e uma grade de tamanho $n \times n$ com bordas coloridas.

**Questão** Existe um ladrilhamento correto da grade (sem girar os ladrilhos)?

## Theorem (Levin)

Ladrilhamento *é* NP-*completo.*

# O mundo de complexidade ao redor de P

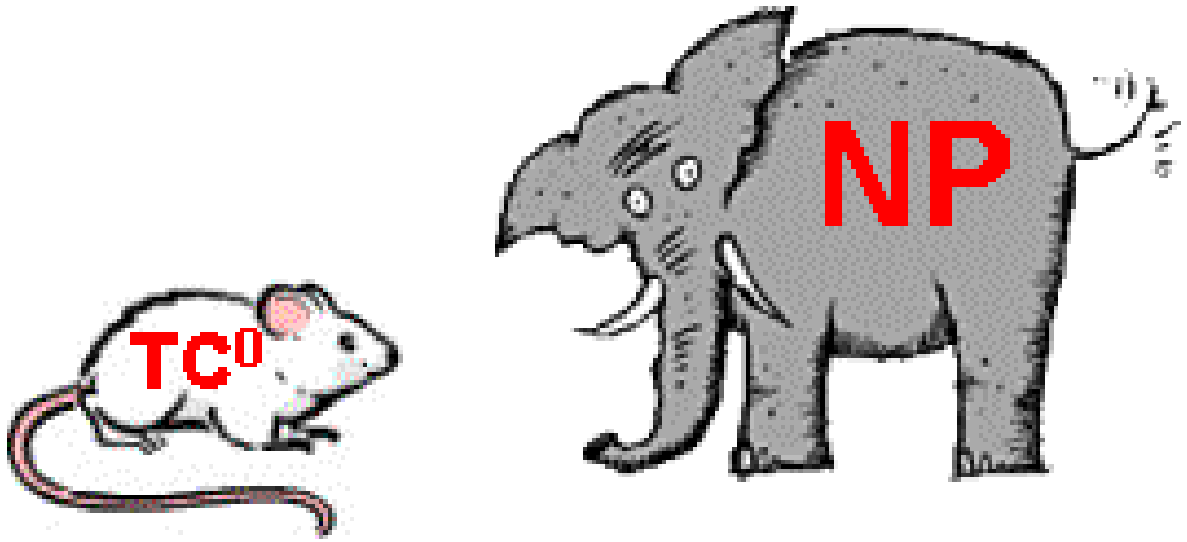


# Classes de complexidade

*I'm sure I'm not the only one who gets rapidly lost when theoretical computer scientists drop names of complexity classes all over the place. I don't know how they do it, but they seem to know what they all are, and how they are related to each other...*

*(Tim Gowers on polymathprojects.org)*

# Excurso: Uma visita no zoológico de complexidade

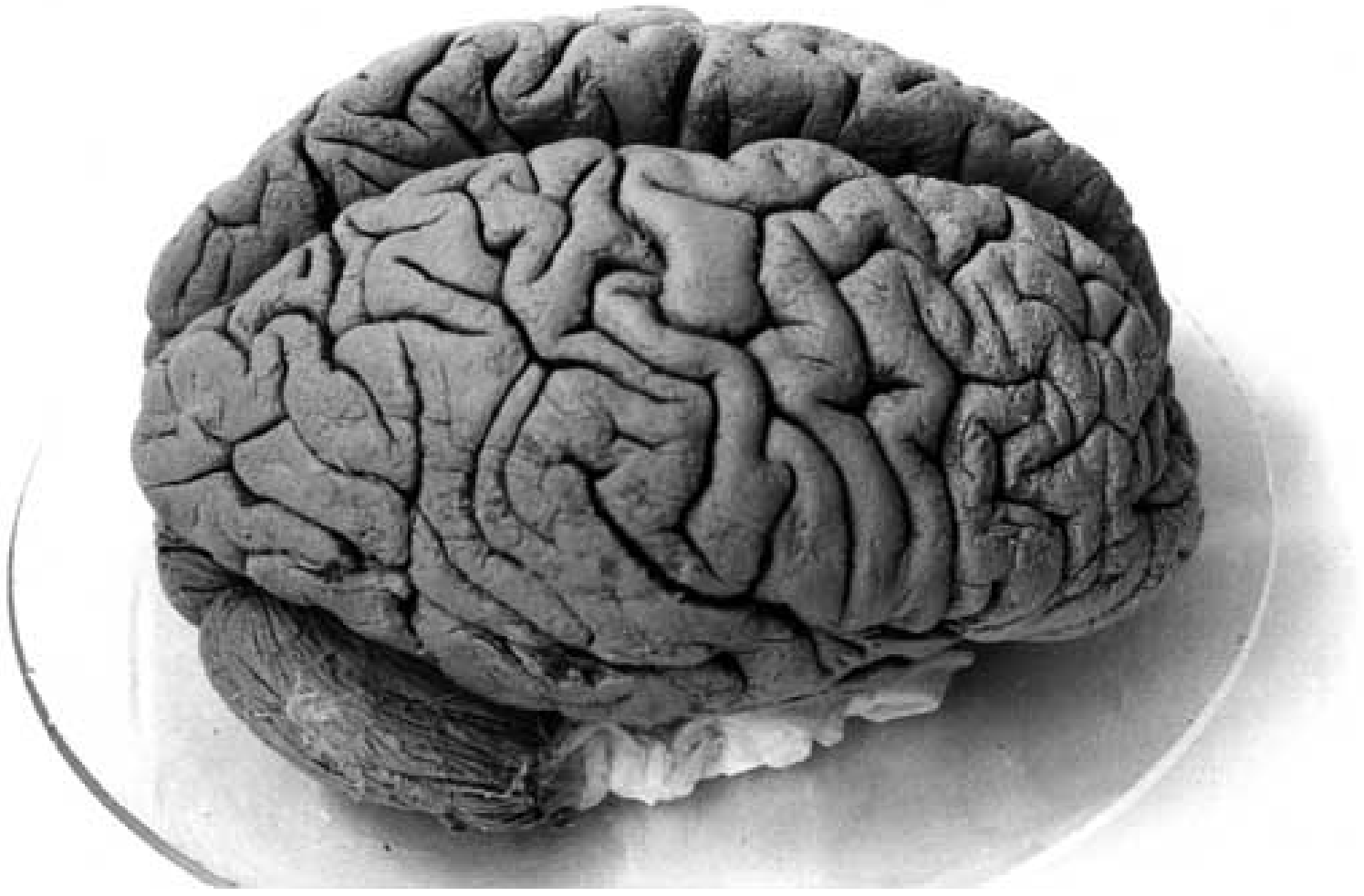



http://qwiki.stanford.edu/wiki/Complexity_Zoo

# De volta ao desafio: Um problema de Gauss


(Fonte: `http://img.photobucket.com/albums/v405/btl/Gauss_sketch.jpg`)

# Excurso: O cerebro de Gauss



O cerebro está em Göttingen.
Resultado da R.M. em 1998: tudo OK.

(Fonte: Häenike, Frahm, Wittmannm, Magnetresonanz-Tomografie des Gehirns von Carl Friedrich Gauß)

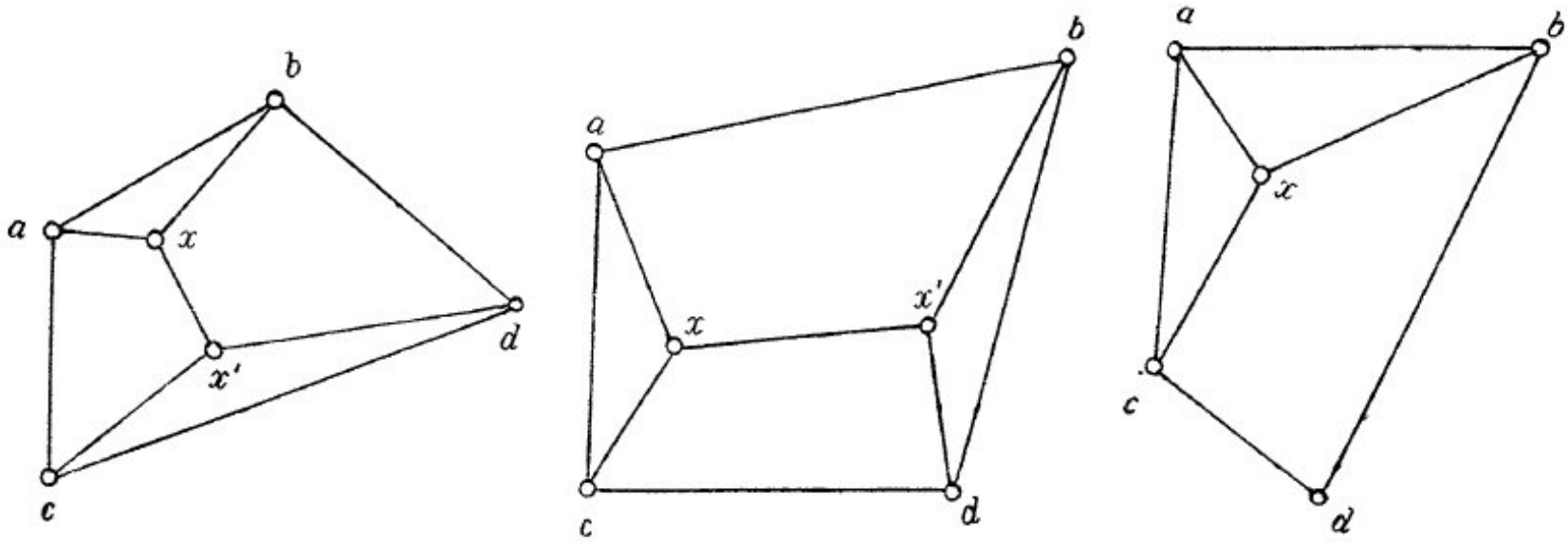
# Gauss para Schumacher, 21 de março 1836

Was ihr Viereck betrifft, so heisst doch die Aufgabe so: Vier Punkte $a$, $b$, $c$, $d$ sind gegeben, man soll einen 5ten $x$ finden, so dass $ax + bx + cx + dx$ ein Minimum wird, und das ist von den 3 Durchschnittspunkten $ab$ mit $cd$, $ac$ mit $bd$, $ad$ mit $bc$ der eine, wo man für die Auswahl die Bedingung entweder leicht auf Anschauung reducieren, oder analytisch einkleiden könnte. Lassen Sie nun $a$, $b$, $c$ fest sein und $d$ dem $c$ immer näher rücken, so bleibt diese Auflösung noch immer so lange richtig, als Sie nicht $c$ mit $d$ zusammenfallen lassen. Fällt aber $c$ mit $d$ zusammen, so erfordert Geist und Buchstabe der mathematischen Aufgabe, als solcher, dass Sie dann $c$ zweimahl zählen, also in dem Dreieck $abc$ $ax + bx + 2cx$ zu einem Minimum machen, wo sich die allgemeine Auflösung noch immer als richtig ausweiset.

# Gauss para Schumacher, 21 de março 1836

Ist bei einem 4 Eck nicht von der stricten mathematischen Aufgabe, wie sie oben ausgesprochen ist, sondern von dem kürzesten Verbindungssystem die Rede, so werden mehrere einzelne Fälle von einander unterschieden werden müssen, und es bildet sich so eine recht interessante mathematische Aufgabe, die mir nicht fremd ist, vielmehr habe ich Gelegenheit einer Eisenbahnverbindung zwischen Harburg, Bremen, Hannover, Braunschweig sie in Erwägung genommen und bin selbst auf den Gedanken gekommen, dass sie eine ganz schickliche Preisfrage für unsere Studenten bei Gelegenheit abgeben könnte. Die Möglichkeiten verschiedener Fälle erläutern wohl hinreichend folgende Figuren:

# Gauss para Schumacher, 21 de março 1836

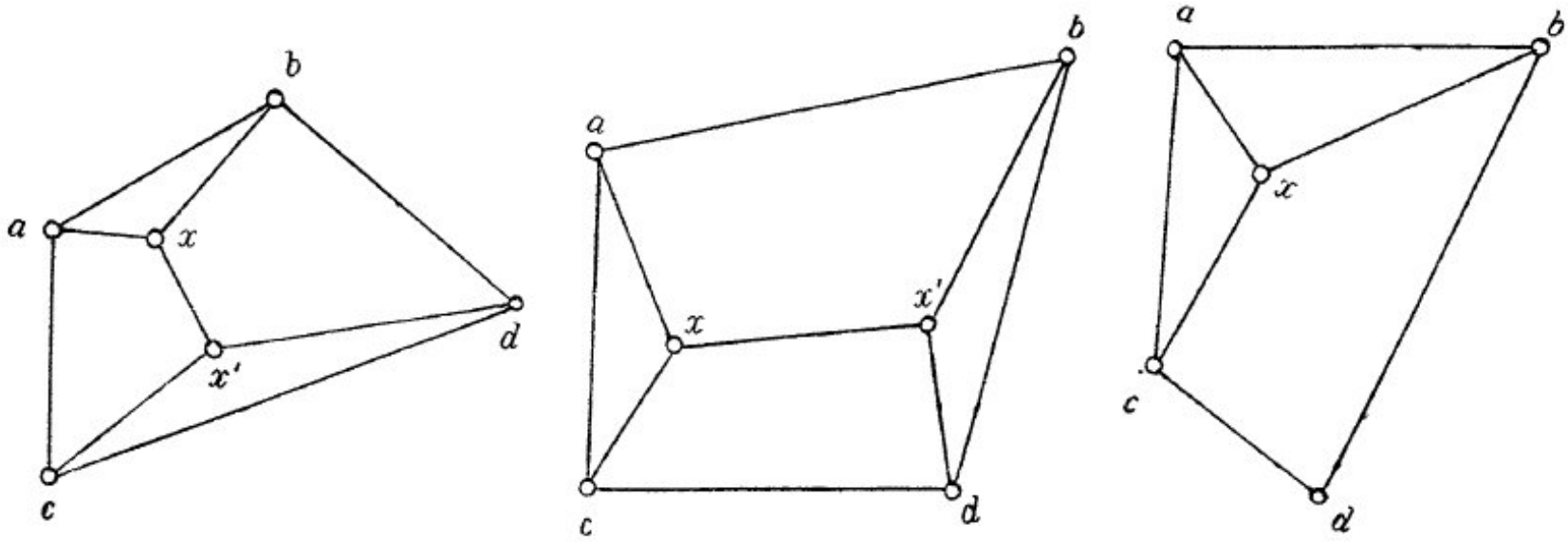



wo in der dritten Figure die Verbindung von *c* nach *d* direct gehen muss (was wirklich bei obigem Beispiel der Fall wird). Doch die Zeit drängt, also heute hiervon nicht mehr ...

# Gauss para Schumacher, 21 de março 1836

Se falamos, no caso de um quadrilátero, não do problema matemático estrito, como definido acima, mas do menor sistema de conexões, temos que separar diversos casos particulares, e obtemos um problema matemático bem interessante, que não é novo para mim, mas que tenho ponderado na ocasião de uma linha de trem entre Harburg, Bremen, Hannover, Braunschweig e cheguei a conclusão, que seria uma questão respeitável para um concurso entre os nossos estudantes em alguma ocasião. As seguintes figuras devem explicar os diferentes casos possíveis suficientemente:

# Gauss para Schumacher, 21 de março 1836




onde na terceira figura a conexão de $c$ para $d$ tem que ser direta (que é o caso do exemplo acima). Mas é tarde, então nada mais disso hoje...

# Árvores de Steiner

## Árvore Steiner Euclidiano

Dado pontos $x_1, x_2, \ldots, x_n$ acha a menor rede de interconexões.

SIAM J. APPL. MATH.
Vol. 32, No. 4, June 1977

**THE COMPLEXITY OF COMPUTING STEINER MINIMAL TREES***

M. R. GAREY, R. L. GRAHAM AND D. S. JOHNSON†

**Abstract.** It is shown that the problem of computing Steiner minimal trees for general planar point sets is inherently at least as difficult as any of the *NP*-complete problems (a well known class of computationally intractable problems). This effectively destroys any hope for finding an efficient algorithm for this problem.

# Árvores de Steiner

## Árvore Steiner em grafos

Dado um grafo $G = (V, E)$, distâncias $d_{uv} \in \mathbb{Q}$ entre vértices $u, v \in V$ e **terminais** $T \subseteq V$:

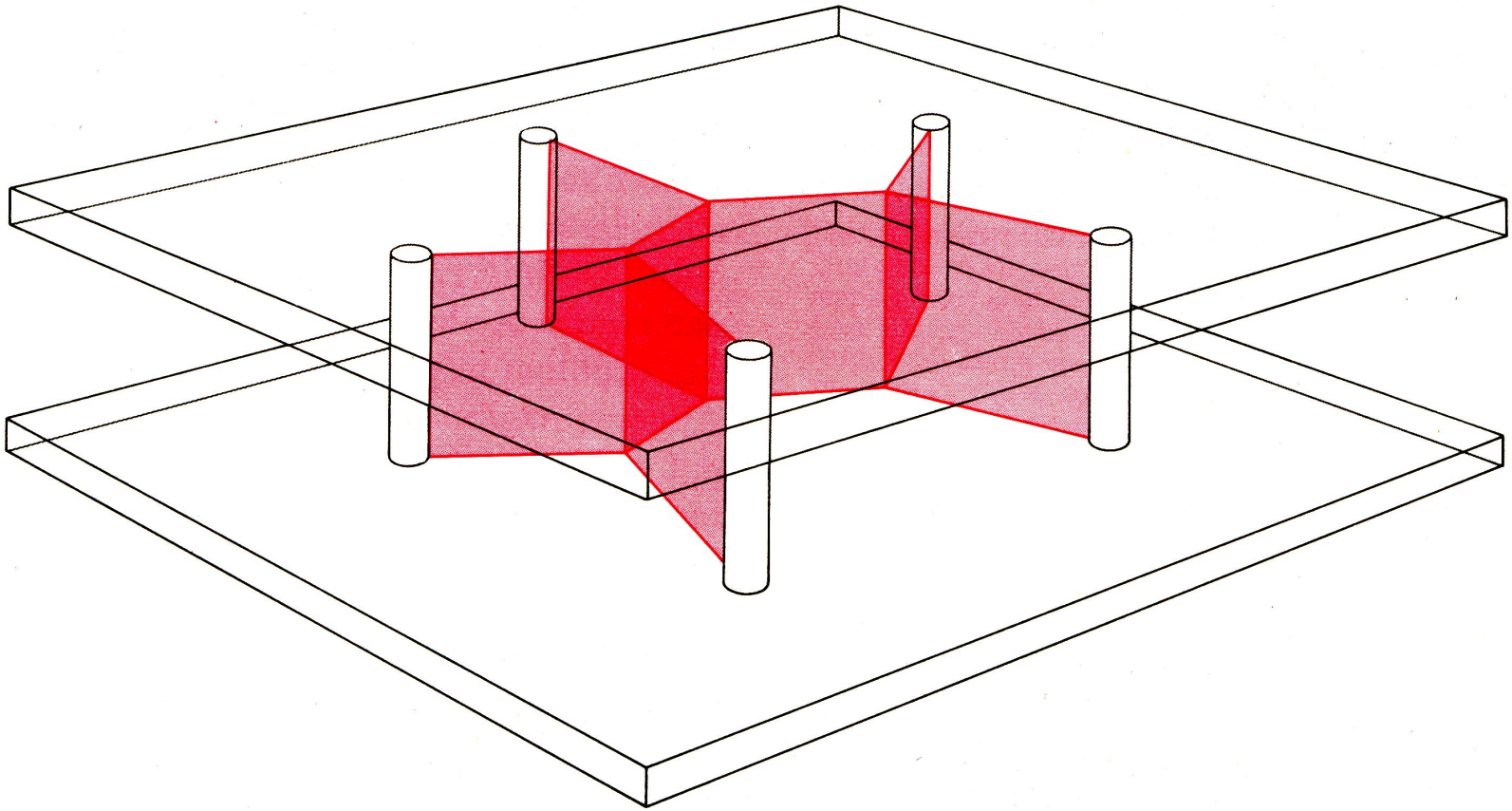
Acha a menor rede de interconexões entre os terminais.

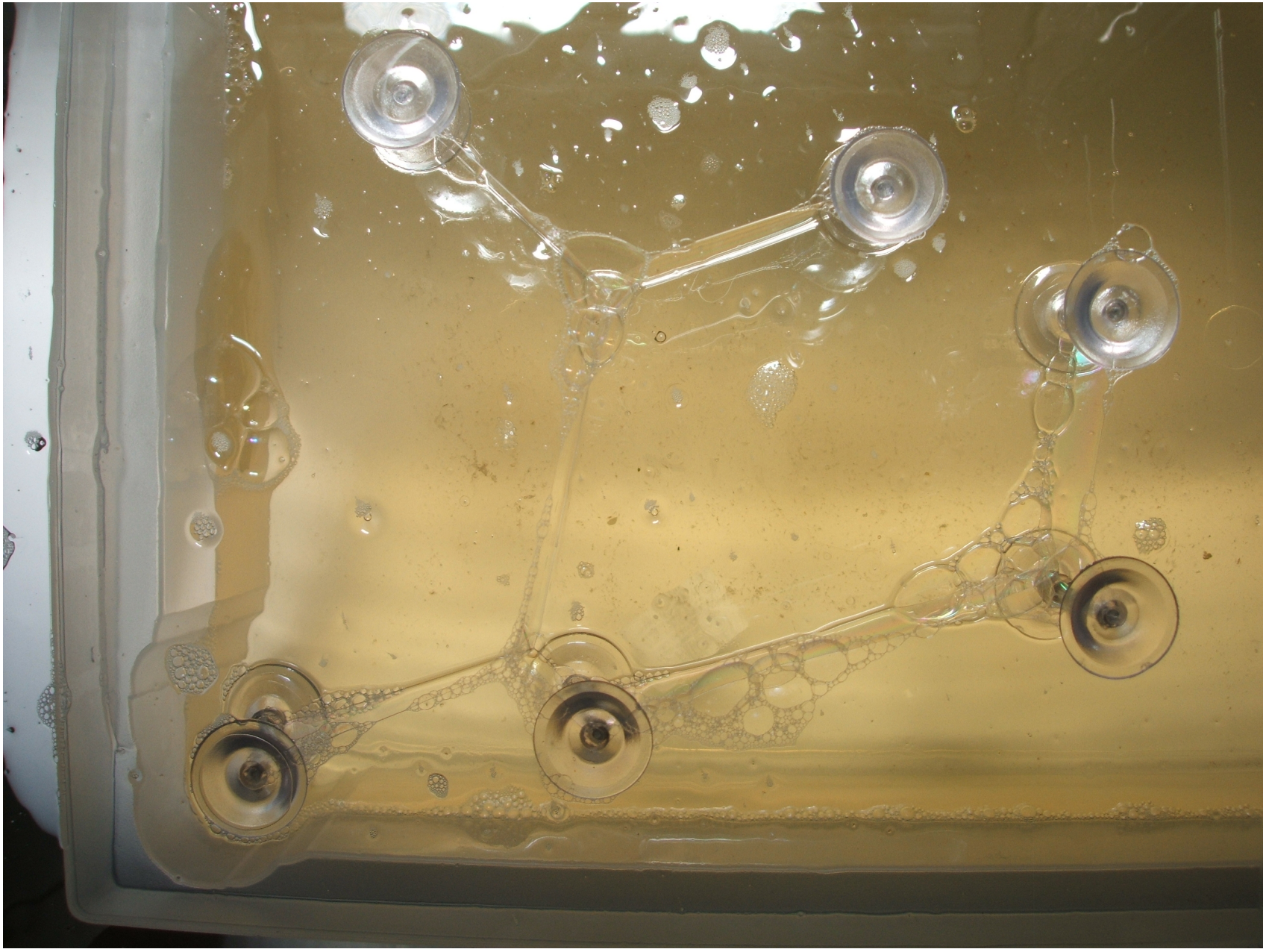
# Será...

Selmer Bringsjord, Joshua Taylor, P = NP
(arXiv:cs/040605v1)

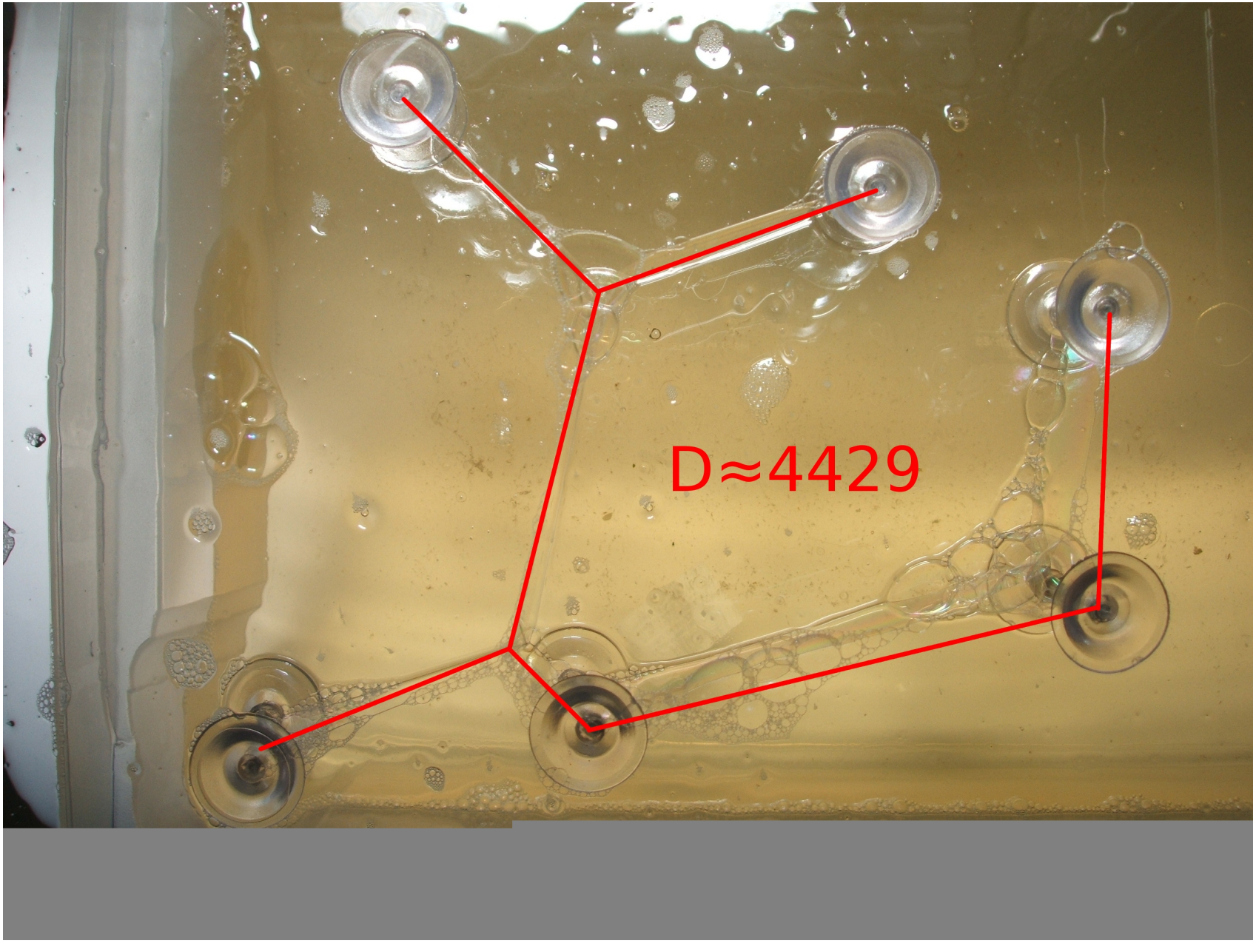
For example, the Steiner Tree problem (STP) is known to be NP-complete. Nonetheless, a simple physical process (termed an **analog computation**) can solve it quickly. [...] The physical process in question runs as follows. Make two parallel glass plates, and insert $n$ pins between the plates to represent the points. The dip the structure into a soap solution, and remove it. The soap film will connect the $n$ pins in the minimum Steiner-tree graph [...] Building the structure and the solution (and the container for the solution) requires steps linear in the size of $n$, and dipping and withdrawing make two steps, [...]

# Nada mais fácil que isso!

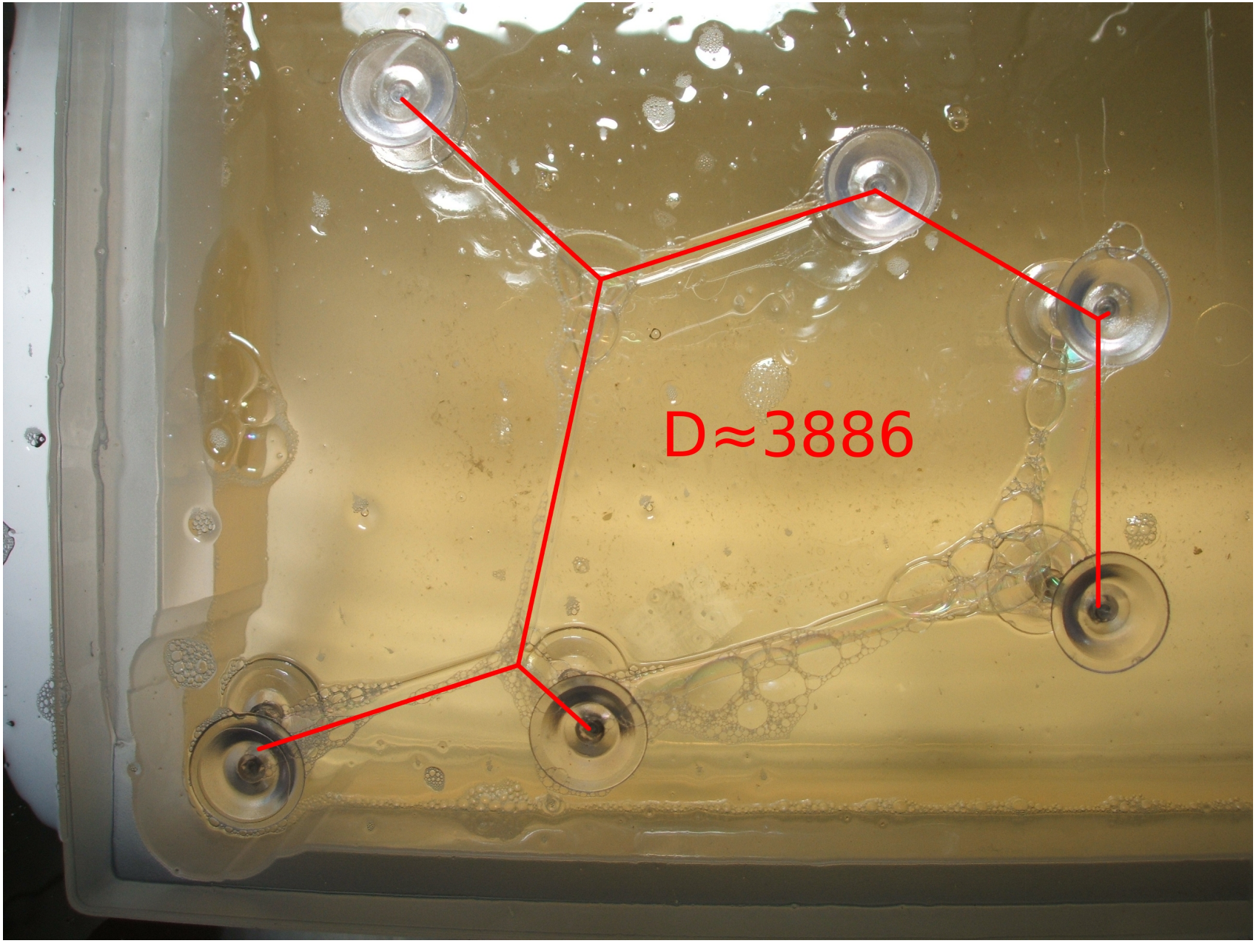




# Gaia...



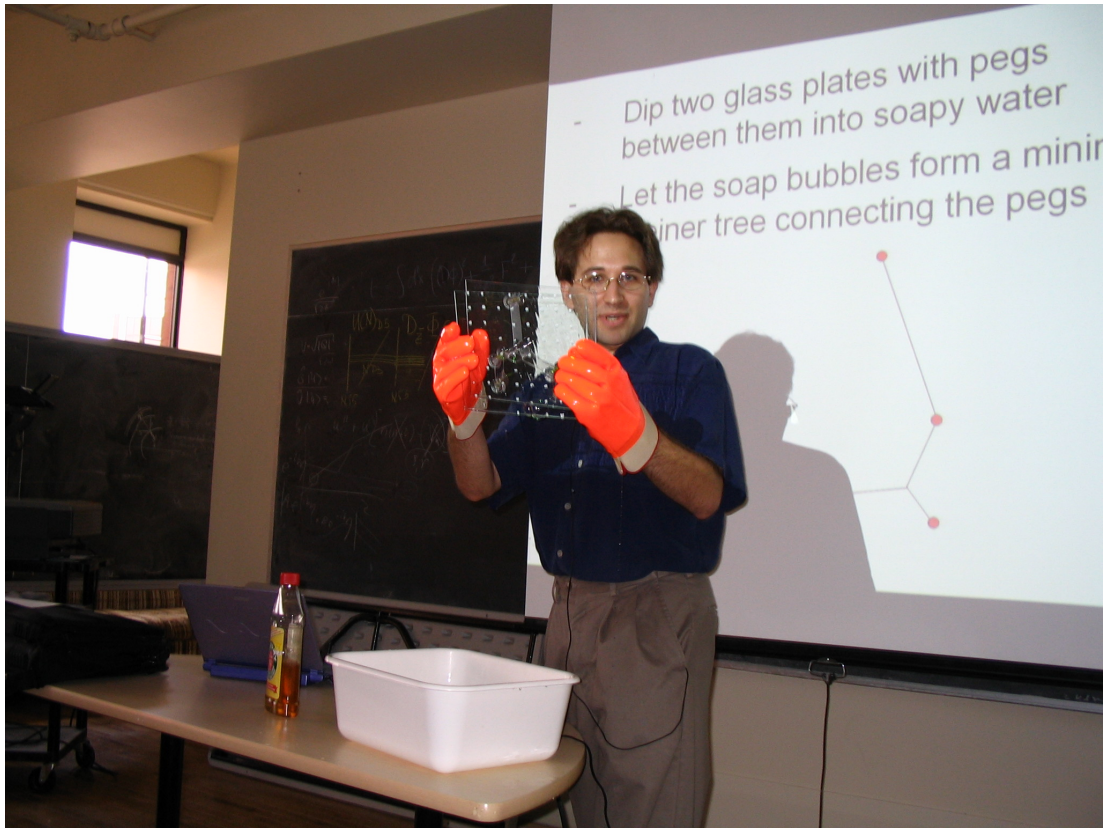
# ... vs. GeoSteiner



Opa! Não deu!

# Não estamos sozinhos...

Long story short, I went to the hardware store, bought some glass plates, liquid soap, etc., and found that, while Nature does often find a minimum Steiner tree with 4 or 5 pegs, it tends to get stuck at local optima with larger numbers of pegs. Indeed, often the soap bubbles settle down to a configuration which is not even a tree (i.e. contains "cycles of soap"), and thus provably can't be optimal.



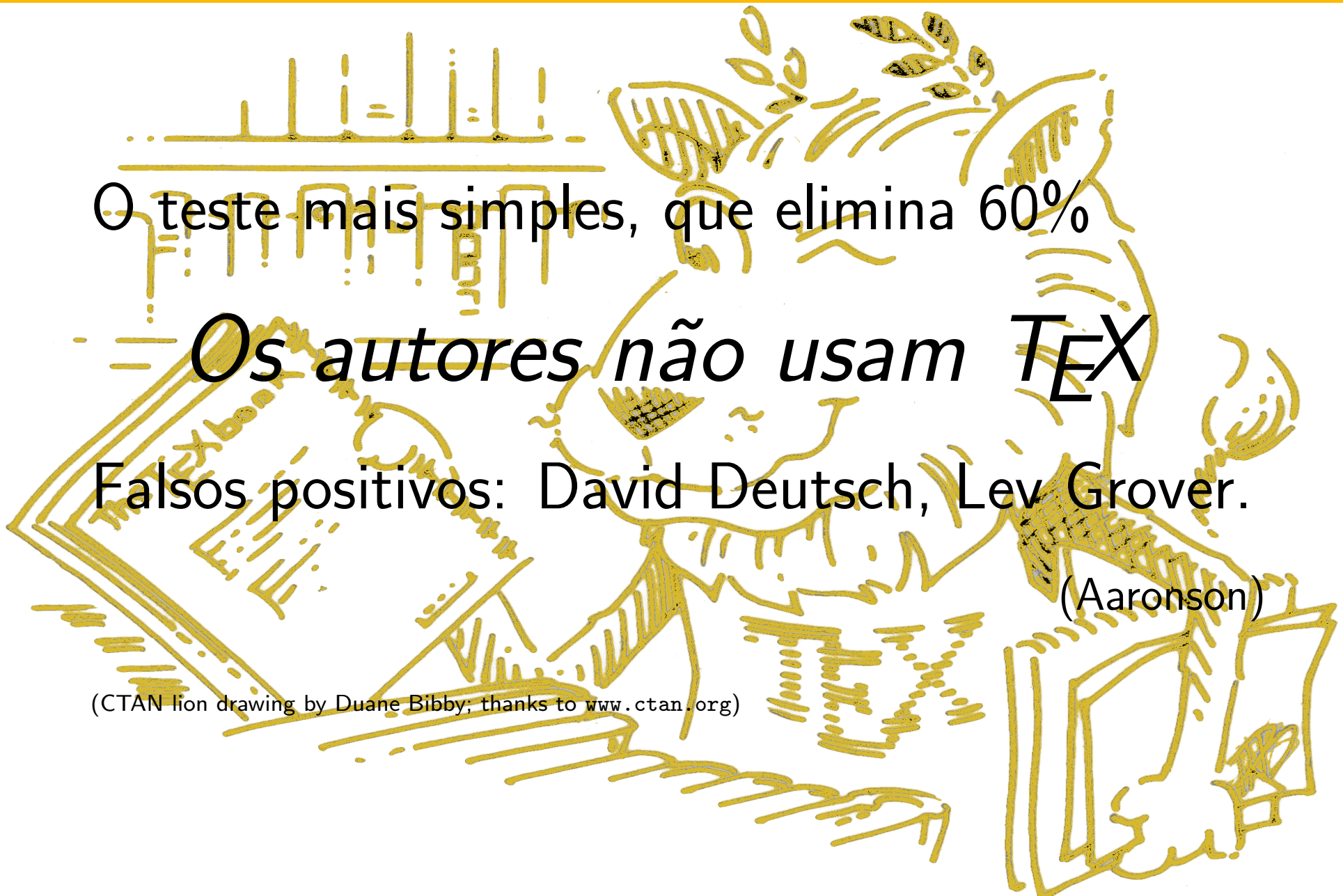

Fonte: `http://www.scottaaronson.com/soapbubble.jpg`

# Alguem precisa mais inspirações?

## The P-versus-NP page

`http://www.win.tue.nl/~gwoegi/P-versus-NP.htm`

# Como reconhecer provas falsas?



O teste mais simples, que elimina 60%

*Os autores não usam TEX*

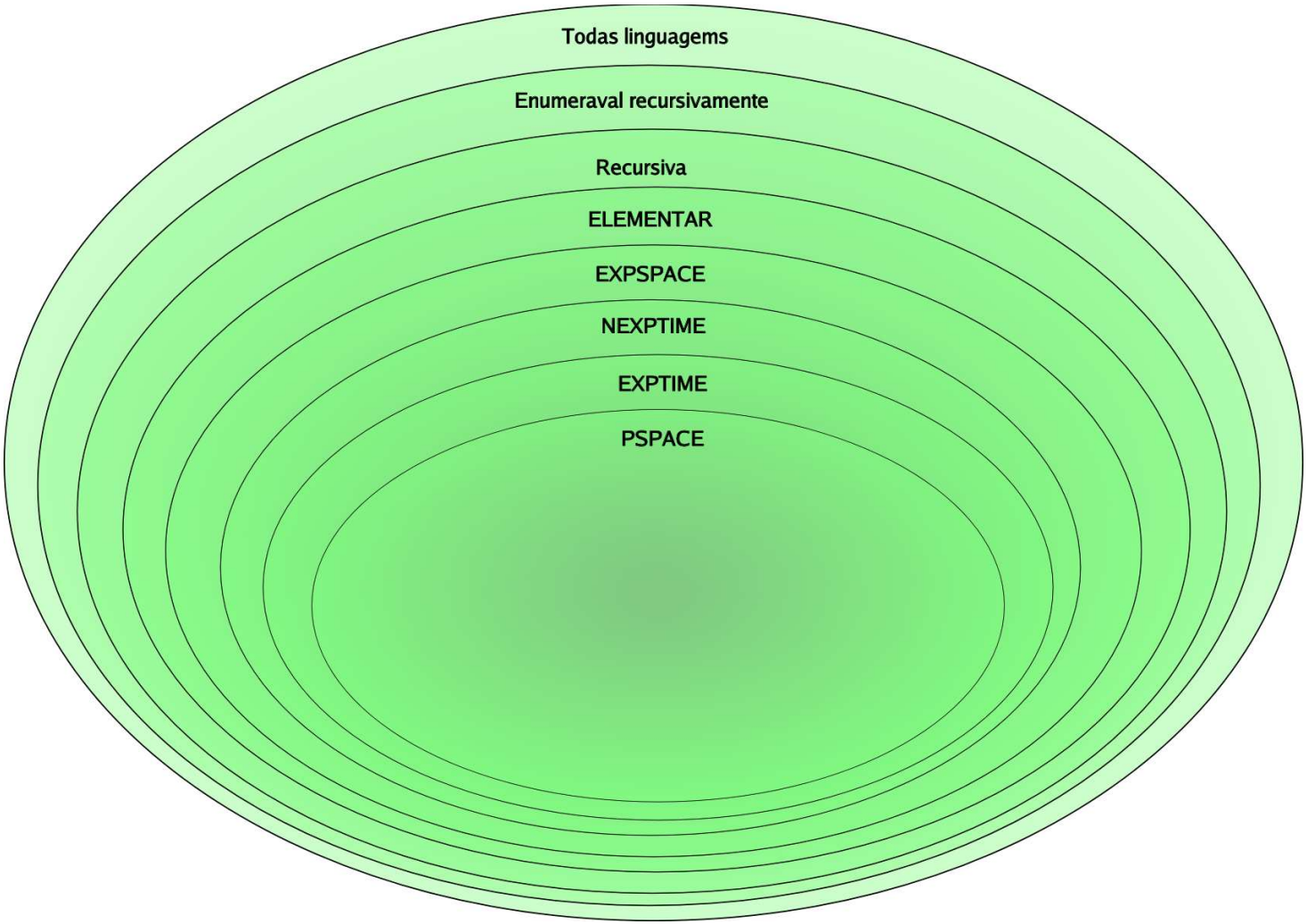
Falsos positivos: David Deutsch, Lev Grover.

(Aaronson)

(CTAN lion drawing by Duane Bibby; thanks to www.ctan.org)

# Agenda

# O mundo até ELEMENTAR

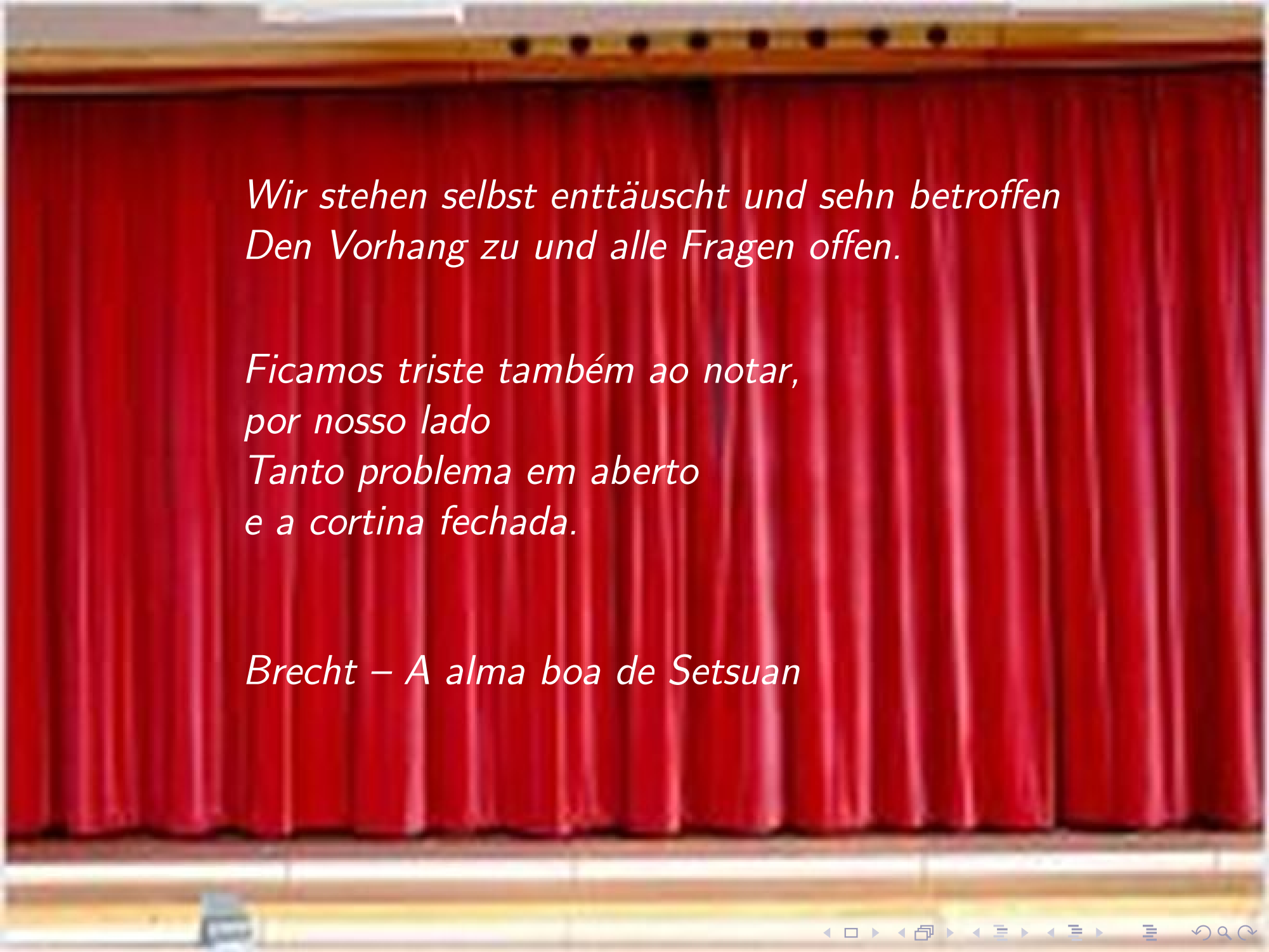


# Exemplo: Expressões regulares

- Uma expressão regular é
  - 0 ou 1 (denota o conjunto $L(0) = \{0\}$ e $L(1) = \{1\}$).
  - $e \circ f$, se $\circ$ é um operador, e $e$, $f$ são expressões regulares.
- Operadores possíveis: $\cup$, $\cdot$, $^2$, $^*$, $\neg$.
- Decisão: Dadas as expressões regulares $e$, $f$, $L(e) \neq L(f)$?

| Expressões regulares com | Completo para |
|---|---|
| $\cup, \cdot$ | NP |
| $\cup, \cdot, ^*$ | PSPACE |
| $\cup, \cdot, ^2$ | NEXP |
| $\cup, \cdot, ^2, ^*$ | EXPSPACE |
| $\cup, \cdot, \neg$ | Fora do ELEMENTAR! |

- O tempo do último problema cresce ao menos como uma torre 2 de altura $\lg n$.


*Wir stehen selbst enttäuscht und sehn betroffen*
*Den Vorhang zu und alle Fragen offen.*

*Ficamos triste também ao notar,*
*por nosso lado*
*Tanto problema em aberto*
*e a cortina fechada.*

*Brecht – A alma boa de Setsuan*

# Fontes das imagens I

13: piqs.de, `fotos.piqs.de/0/f/3/a/2/68a6760a3b6969fc3c693e65957a9e4b.jpg`
27: Source links on the pages.
36: `astronomy-pictures.net/beyond-the-stars.jpg`
43, 44: Knuth

# Referências I


Wilkinson microwave anisotropy probe. Online. `http://map.gsfc.nasa.gov`.

Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. **Annals of Mathematics**, 160:781–793, 2004.

J. D. Bekenstein. A universal upper bound on the entropy to energy ratio for bounded systems. **Phys. Rev. D**, 23(2): 287–298, 1981.

C.-J. de la Vallée Poussin. Recherches analytiques la thÃ©orie des nombres premiers. **Ann. Soc. scient. Bruxelles**, 20: 183–256, 1896.

# Referências II


Martin Fürer. Faster integer multiplication. In **STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing**, pages 57–66, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-631-8. doi: http://doi.acm.org/10.1145/1250790.1250800.

Ronald Lewis Graham, Donald Ervin Knuth, and Oren Patashnik. **Concrete Mathematics: a foundation for computer science**. Addison-Wesley, 1988.

J. Hadamard. Sur la distribution des zéros de la fonction zeta(s) et ses conséquences arithmétiques. **Bull. Soc. math. France**, 24:199–220, 1896.

Anatolii Alekseevich Karatsuba and Yu Ofman. Multiplication of many-digital numbers by automatic computers. **Doklady Akad. Nauk SSSR**, 145(2):293–294, 1962. Translation in Soviet Physics-Doklady 7 (1963), pp. 595–596.

# Referências III


Jon Kleinberg and Éva Tardos. **Algorithm design**. Addison-Wesley, 2005.

D. E. Knuth. Diamond signs. http://www-cs-faculty.stanford.edu/ uno/diamondsigns/diam.html.

Seth Lloyd. Computational capacity of the universe. **Physical Review Letters**, 88(23), 2002. `http://focus.aps.org/story/v9/st27`.

John C. Mitchell. **Foundations for programming languages**. MIT Press, 1996.

B. Monien and E. Speckenmeyer. Solving satisfiability in less than $2^n$ steps. **Discrete Appl. Math**, 10:287–295, 1985.

James Stirling. **Methodus differentialis, sive tractatus de summation et interpolation serierum infinitarium**. London, 1730.

# Referências IV


Volker Strassen. Guassian elimination is not optimal. **Numer. Math**, 13:354–356, 1969.

Alan Mathison Turing. On computable numbers with an application to the Entscheidungsproblem. **Proc. London MathŚoc.**, 2(42):230–265, 1936.

Jan van Leeuwen, editor. **Handbook of theoretical computer science**, volume A: Algorithms and complexity. MIT Press, 1990. URL `http://www.amazon.com/Handbook-Theoretical-Computer-Science-Vol/dp/0262720140`.