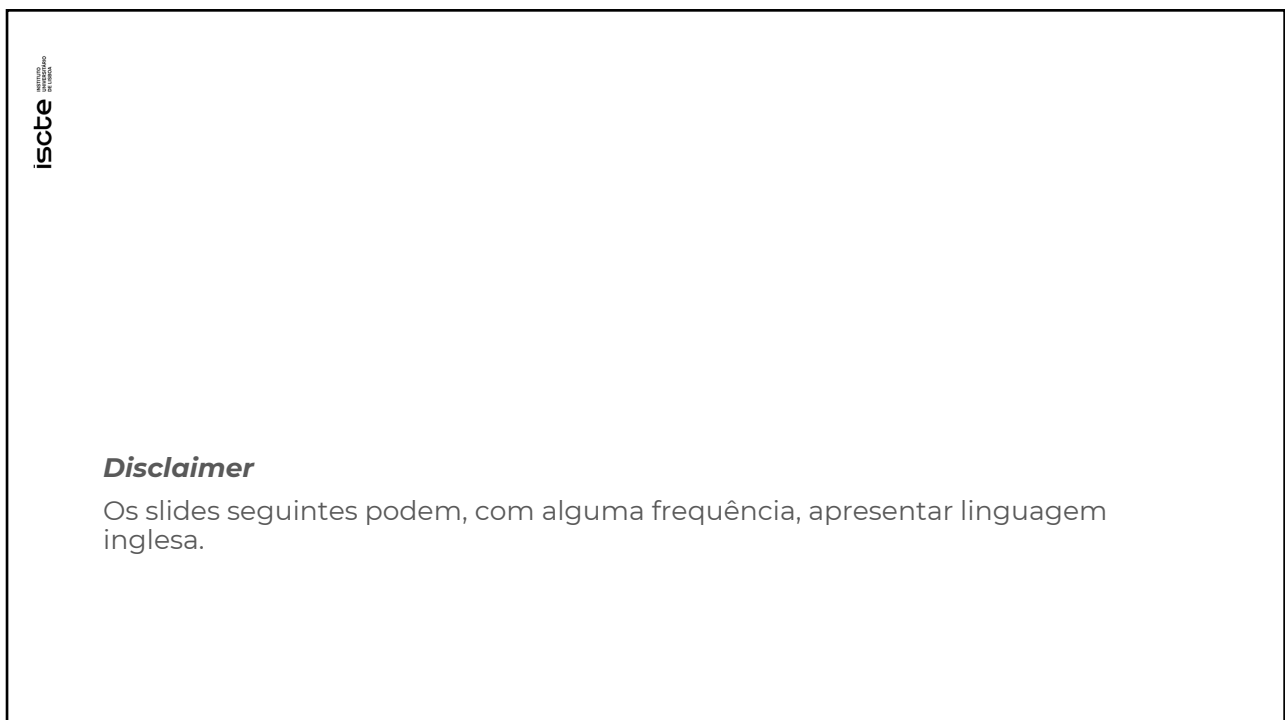




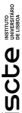
1



2



3



iscte
INSTITUTO UNIVERSITÁRIO DE LISBOA

Plano para esta aula

- Introdução:
 - Conceitos básicos:
 - Algoritmos e Programas (revisitados)
 - Sistemas de numeração
 - Análise de Algoritmos: pontos chave

4

Temas dos slides

Definição: uma definição

- Texto normal

- **Algoritmos (pseudo-código):**

```
um algoritmo
```

- **Notas importantes:**

chamada de atenção

5

"The Feynman Problem-Solving Algorithm:

- (1) write down the problem;*
- (2) think very hard;*
- (3) write down the answer."*

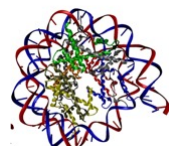
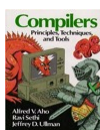
Murray Gell-mann,
Physics Nobel Laureate

Introdução

Algoritmos e Análise de Algoritmos

6

Algoritmos são úteis.



Source: Amelie Byun

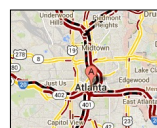
- Com o crescimento da quantidade de dados a que assistimos hoje em dia, construir bons algoritmos tornou-se **cada vez mais importante!**

Large-scale data is Everywhere!

- Devido ao avanços que possibilitam tecnologias de geração (automática) de dados e sua recolha, a quantidade de dados em bases de dados **comerciais** and **científicas** tem aumentado em larga escala.

- New mantra:**

"Gather **whatever** data you can **whenever** and **wherever** possible."



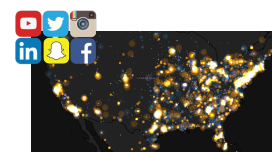
Traffic Patterns



E-Commerce



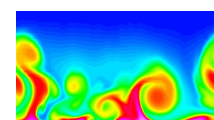
Sensor Networks



Social Networking: Twitter



Corporate data



Computational Simulations

Data deluge (Torrente de Dados, tradução livre)

- **Crescimento explosivo de dados** em (quase) todos os domínios
 - **Smartphones** de sofisticação crescente que nos mantêm conectados 24/7
 - **Internet** e redes sociais facilitam a publicação de dados
 - Experiências científicas e simulações geram volumes (quantidades) extraordinários de dados
- **IoT - Internet of Things**
 - Sensores and redes de sensores capazes de monitorizar tudo 24/7 (desde temperaturas e poluição até sinais vitais)
- **Datificação: qualquer aspecto da nossa vida é transformado em dados**
(o que gostamos/apreciamos transforma-se numa **stream** de "likes")

It's a world of data

The point is not simply that algorithms have many applications. The deeper issue is that the subject of algorithms is a powerful lens through which to view the field of computer science in general. Algorithmic problems form the heart of computer science, but they rarely arrive as cleanly packaged, mathematically precise questions. Rather, they tend to come bundled together with lots of messy, application-specific detail, some of it essential, some of it extraneous.

-J. Kleinberg e É. Tardos

Algoritmos [Algorithms]

“Start at the beginning”

-L. Carroll, “Alice in Wonderland”

11

Algorithms make the world go round!

- Uma das inovações e invenções mais espetaculares da antiga Babilónia é o **sistema de numeração posicional**
- Representamos um número como uma sequência de dígitos, cuja posição determina o valor:



- Com um sistema como o da numeração romana, onde cada dígito tem o seu valor, independentemente da sua posição, torna-se mais complicado, quer representar números, quer operar. Por exemplo, a distância média da terra à lua é de aproximadamente 259 956 milhas romanas escrita em numeração indo-árabe.

12

Algorithms make the world go round!

- Al Khwarizmi (Séc. IX):

procedimentos não ambíguos, mecânicos (mecanizáveis), eficientes e correctos –

Algoritmos



- **Algoritmos + Sistema Decimal** (600 AC) + Tipografia (Gutenberg, 1448):

Renascença, Revolução Científica, Industrial e Económica

- Algoritmos + Computadores = Reinvenção do Mundo
- Algorithms are forever:
 - Um algoritmo é independente da Linguagem de Programação
 - Um algoritmo resolve um problema específico mas (de descrição) geral

16

- **Algoritmo:**
 - sequência ordenada de instruções não ambíguas e corretas
 - realizável numa quantidade finita de tempo

- **Sistema Decimal:**

$$N = n_0 n_1 \dots n_k = \sum_{i=0}^k n_i \times 10^i, \quad k \geq 0$$

$123 = 123_{(10)} = 3 \times 10^0 + 2 \times 10^1 + 1 \times 10^2$

mais à esquerda
maior valor

base 10

- **Sistema Binário:**

$$N = n_0 n_1 \dots n_k = \sum_{i=0}^k n_i \times 2^i, \quad k \geq 0$$

$123 = 01111011_{(2)} = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 1 \times 2^5 + 1 \times 2^6 + 0 \times 2^6$

mais à esquerda
maior valor

base 2

17

Multiplicação Inteira

Como fazemos para multiplicar num computador (qual o algoritmo)?

$$\begin{array}{r} \text{1233925720752752384623764283568364918374523856298} \\ \times \text{42562323582342395285623467235019130750135350013753} \\ \hline \end{array}$$

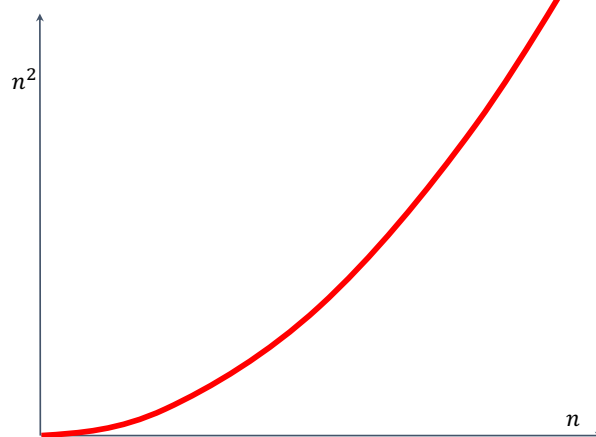
Quantas operações com um dígito são efetuadas?

???

19

Que podemos fazer para melhorar?

Ordem de crescimento do número de operações cresce de acordo a uma função quadrática (em n^2).

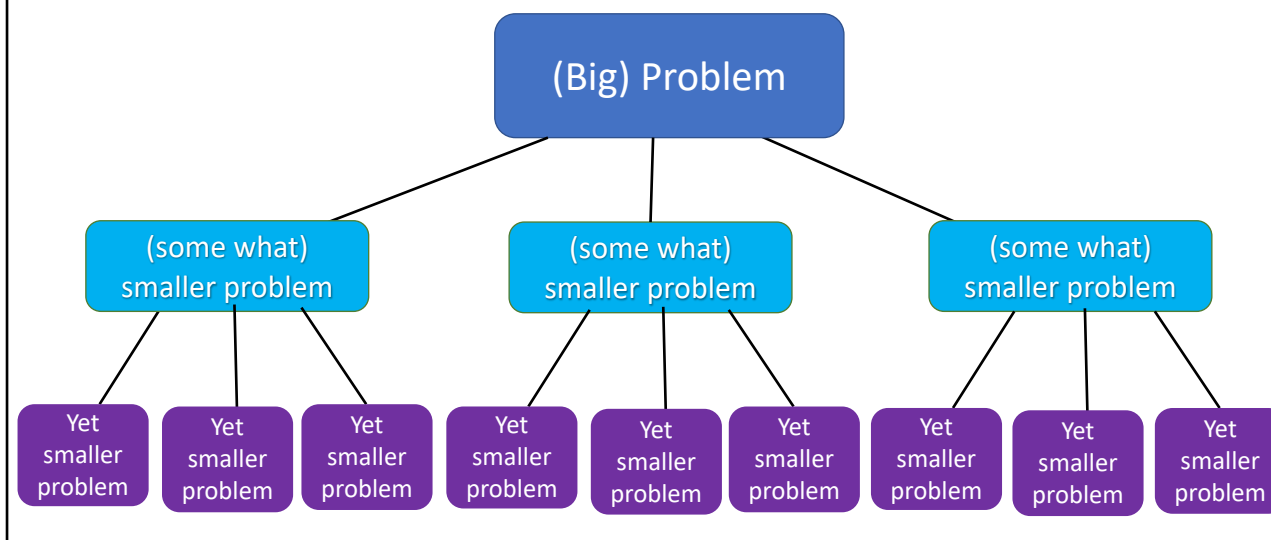


Que podemos fazer para multiplicar números inteiros com n dígitos usando menos operações, ou seja, mais rapidamente que $O(n^2)$?

21

Dividir e Conquistar

Estratégia “dividir e conquistar”: Dividir o problema inicial em k sub-problemas mais simples



23

Dividir e conquistar: multiplicação inteira

- usar a representação decimal inteira:

$$N = n_0 n_1 \dots n_k = \sum_{i=0}^k n_i \times 10^i, \quad k \geq 0$$

Ex: $1234 = 12 \times 100 + 34$

$$\begin{aligned}
 &1234 \times 5678 \\
 &= (12 \times 100 + 34) (56 \times 100 + 78) \\
 &= \underbrace{(12 \times 56)}_{\textcircled{1}} 10000 + \underbrace{(34 \times 56)}_{\textcircled{2}} + \underbrace{(12 \times 78)}_{\textcircled{3}} 100 + \underbrace{(34 \times 78)}_{\textcircled{4}}
 \end{aligned}$$

Uma multiplicação de 4 dígitos

Problema original



4 multiplicações de 2 dígitos

Problema dividido em subproblemas

24

Questão....

- Uma multiplicação de números inteiros de 4 dígitos pode ser partida em 4 multiplicações mais simples de 2 dígitos

$$1234 \times 5678$$

- Se partirmos recorrentemente nos problemas mais simples, no total, quantas multiplicações de um dígito vamos fazer?



Source: oannis kounadeas

Este algoritmo é “melhor” ou “pior?”

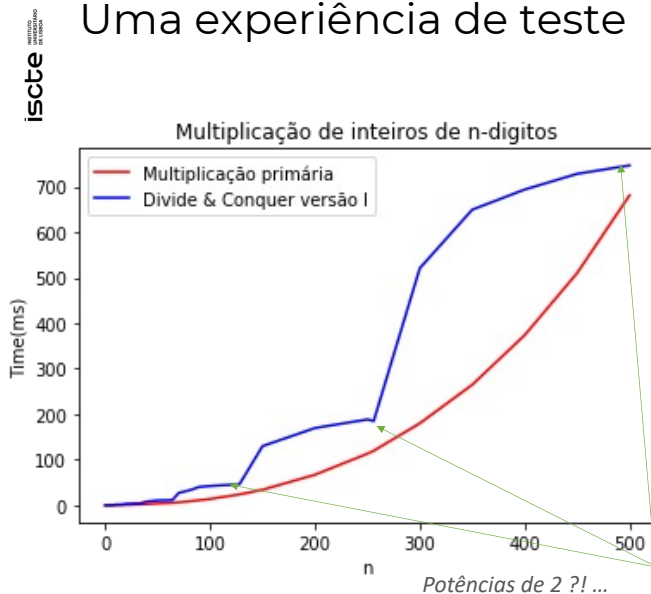
- Como vamos definir “ser melhor”?

1. Tem de estar correto, ou seja, responder corretamente

2. Tem de ser mais rápido:

- Efetuar testes (experiências)
- Tentar compreender analiticamente qual ordem de grandeza que caracteriza o tempo de execução

Uma experiência de teste



- Conjecturas acerca do tempo?

Não parece bom mas...

se aumentarmos o número n de dígitos ($n=10000, 100000$), a linha vermelha irá ultrapassar a azul ou não?

28

Analisar o tempo de execução no(s) teste(s) e...

Hipótese:

- O tempo de execução parece ser, pelo menos, da ordem de n^2 .

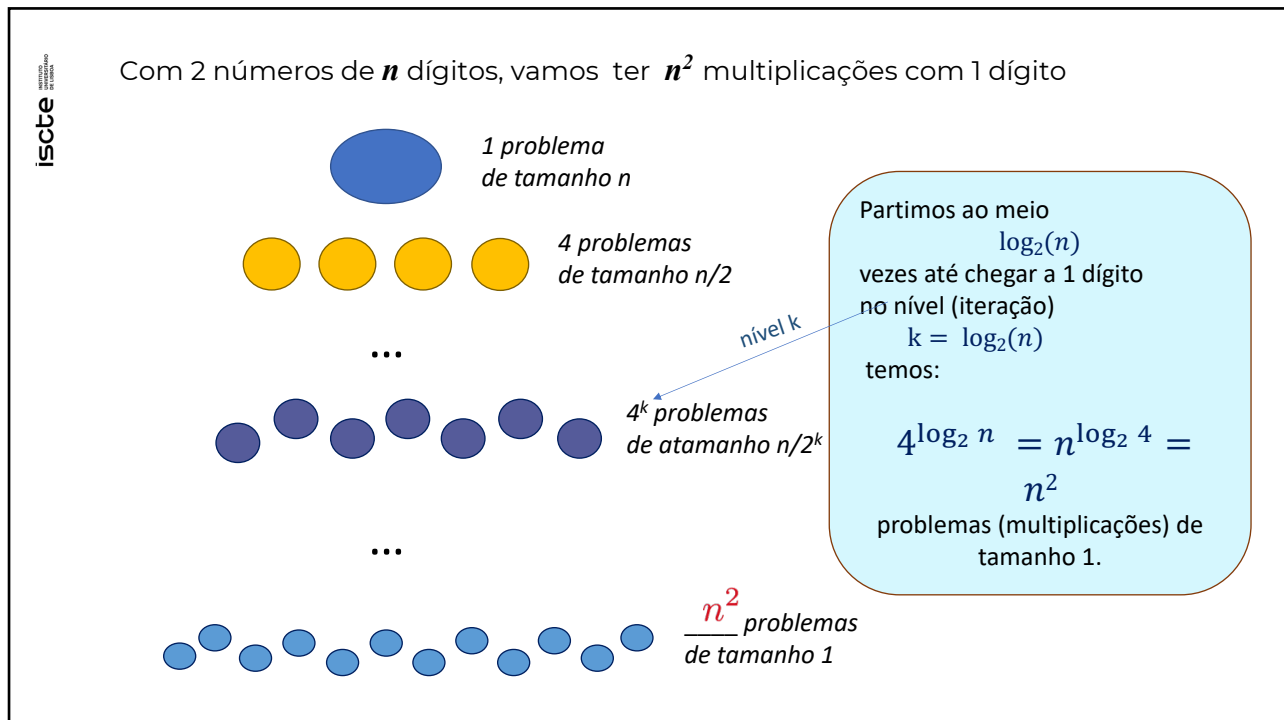
Mas será pior ?

Para responder vamos ter de usar a cabeça: pensar analiticamente.

Necessitamos de ferramentas que nos ajudem a caracterizar/classificar um algoritmo em termos de um **limite máximo para a sua execução**.

Nota: os testes experimentais, ou empíricos, devem ser **repetidos** um **$k (> 1)$** número de vezes antes de olhar para um gráfico de tempos de execução

29



30

Dividir e conquistar **pode mesmo** melhorar!

- Karatsuba usou esta estratégia de modo inteligente (smart!)

$$xy = (a \cdot 10^{n/2} + b)(c \cdot 10^{n/2} + d)$$

$$= ac \cdot 10^n + (ad + bc)10^{n/2} + bd$$

em seguida calculamos estes 3 subproblemas

- Diminuimos uma parcela de recursão: 3 em vez de 4

31

Procedimento preciso e mecanizável

x, y são números com n dígitos

Multiply(x, y):

If $n=1$:

return xy

Write $x = a 10^{\frac{n}{2}} + b$ and $y = c 10^{\frac{n}{2}} + d$ a, b, c, d são números com $n/2$ dígitos

$ac = \mathbf{Multiply}(a, c)$

$bd = \mathbf{Multiply}(b, d)$

$z = \mathbf{Multiply}(a+b, c+d)$

$xy = ac 10^n + (z - ac - bd) 10^{n/2} + bd$

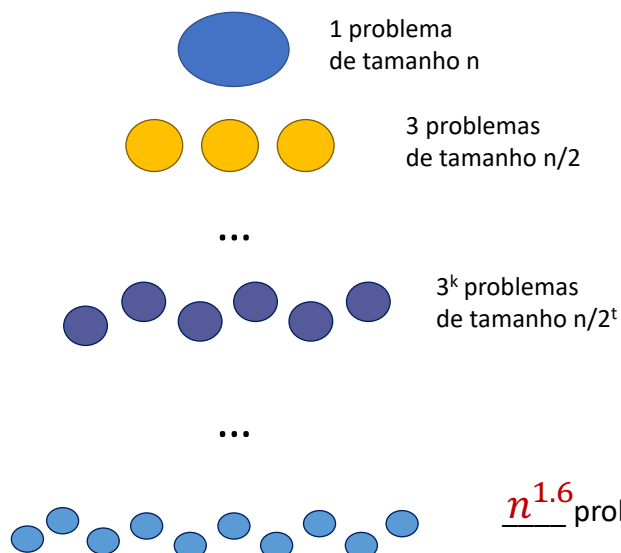
return xy

(Linguagem de *alto nível* = alguma subjetividade,; ver Python notebook para codificação.)

(Assumimos que n é uma potência de 2, sem perda de generalidade.)

33

Tempo de execução?



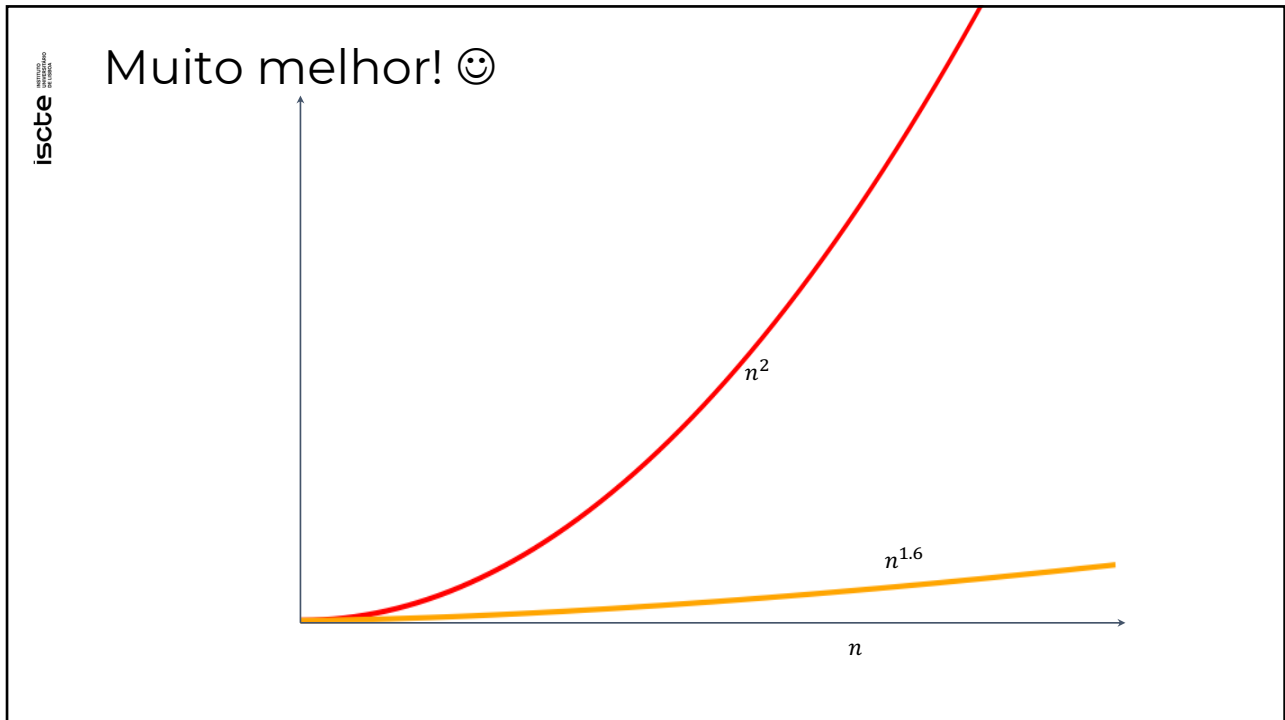
- dividindo n em metade $\log_2(n)$ times, chegamos a 1

- e no último nível
 $k = \log_2(n)$
temos

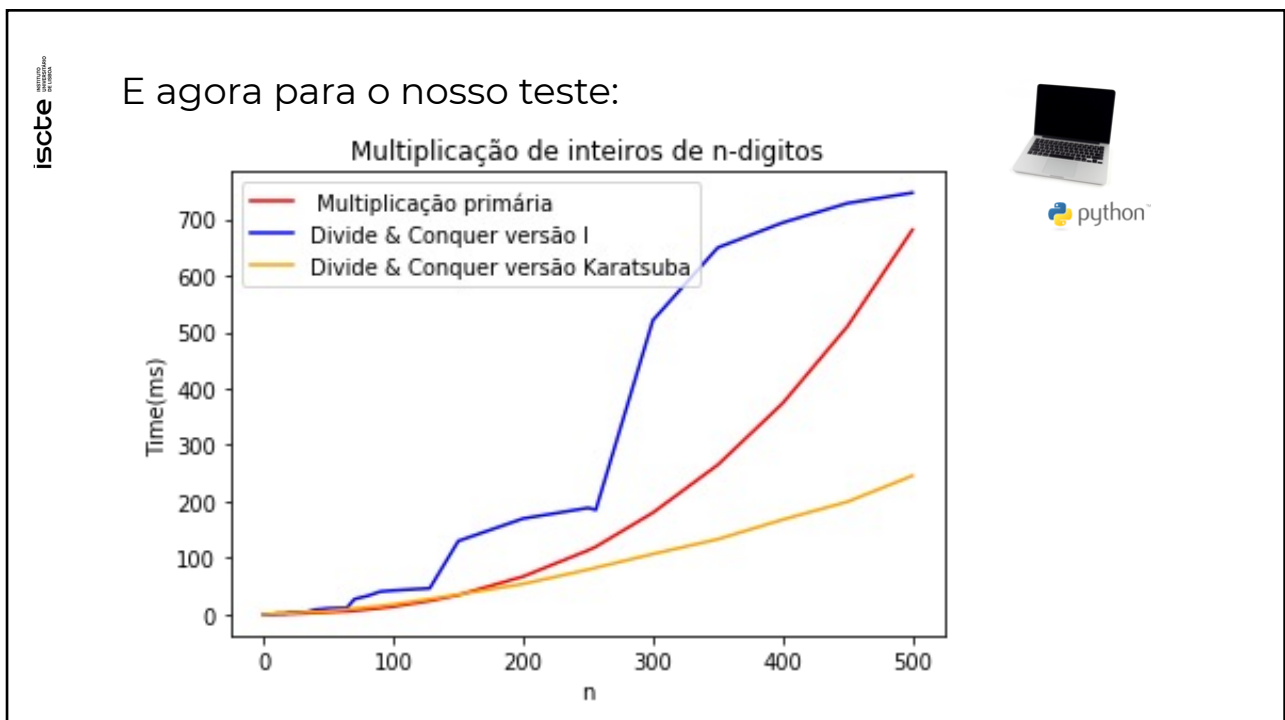
$$3^{\log_2 n} = n^{\log_2 3} \approx n^{1.6}$$

problemas de tamanho 1.

34



35



36

A reter desta aula

- Introdução:
 - Conceitos básicos:
 - Algoritmos e Programas (revisitados)
 - Sistemas de numeração
 - Algoritmos de multiplicação para ilustrar a necessidade de **melhorar**: o uso da estratégia dividir para conquistar como ferramenta de melhoria.