

# Disciplinas CCM-001 / MCTA003-17

## Análise de Algoritmos (e Estruturas de Dados)

Primeira aula: introdução e exercícios de revisão

---

**Profa. Carla Negri Lintzmayer**

[carla.negri@ufabc.edu.br](mailto:carla.negri@ufabc.edu.br)

[www.professor.ufabc.edu.br/~carla.negri](http://www.professor.ufabc.edu.br/~carla.negri)

Centro de Matemática, Computação e Cognição – Universidade Federal do ABC



Estes slides contêm uma motivação (Seção 1) para o estudo desta disciplina e alguns exercícios de revisão (Seção 2), sobre o conteúdo recomendado.

# Motivação

---

Sequência finita de passos descritos de forma não ambígua que corretamente resolvem um problema.

Sequência finita de passos descritos de forma não ambígua que corretamente resolvem um problema.

Recebe um conjunto de dados como *entrada* e devolve um conjunto de dados como *saída*.

Sequência finita de passos descritos de forma não ambígua que corretamente resolvem um problema.

Recebe um conjunto de dados válidos como *entrada* e devolve um conjunto de dados como *saída*.

Sequência finita de passos descritos de forma não ambígua que **corretamente resolvem** um problema.

Recebe um conjunto de dados válidos como *entrada* e devolve um conjunto de dados como *saída*.

Para toda entrada possível ele produz uma saída que seja solução do problema para aquela entrada.

Nos permite

- verificar corretude,



Nos permite

- verificar corretude,
- prever desempenho,

Nos permite

- verificar corretude,
- prever desempenho,
- comparar soluções.

Nos permite

- verificar corretude,
- prever desempenho,
- comparar soluções.

**Sem que seja necessário implementá-los em um dispositivo específico!**

## Exemplo

A sequência de Fibonacci é a sequência infinita de números: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ....

## Exemplo

A sequência de Fibonacci é a sequência infinita de números: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ....

Por definição, o  $n$ -ésimo número da sequência, escrito como  $F_n$ , é dado por

$$F_n = \begin{cases} 1 & \text{se } n = 1 \\ 1 & \text{se } n = 2 \\ F_{n-1} + F_{n-2} & \text{se } n > 2. \end{cases} \quad (1)$$

## Exemplo

A sequência de Fibonacci é a sequência infinita de números: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ....

Por definição, o  $n$ -ésimo número da sequência, escrito como  $F_n$ , é dado por

$$F_n = \begin{cases} 1 & \text{se } n = 1 \\ 1 & \text{se } n = 2 \\ F_{n-1} + F_{n-2} & \text{se } n > 2. \end{cases} \quad (1)$$

PROBLEMA: NÚMERO DE FIBONACCI

Dado um inteiro  $n \geq 1$ , encontrar  $F_n$ .

- 1: **Função**  $\text{FIB1}(n)$
- 2:     **Se**  $n \leq 2$  **então**
- 3:         **Devolve** 1
- 4:     **Devolve**  $\text{FIB1}(n - 1) + \text{FIB1}(n - 2)$

- 1: **Função**  $\text{FIB1}(n)$
- 2:     **Se**  $n \leq 2$  **então**
- 3:         **Devolve** 1
- 4:     **Devolve**  $\text{FIB1}(n - 1) + \text{FIB1}(n - 2)$

1. Esse algoritmo resolve o problema?
2. Quanto tempo ele leva?



## Exemplo

Seja “Máquina 1” um computador que faz 4 bilhões de instruções por segundo.

Seja “Máquina 2” um computador que faz 40 trilhões de instruções por segundo.

## Exemplo

Seja “Máquina 1” um computador que faz 4 bilhões de instruções por segundo.

Seja “Máquina 2” um computador que faz 40 trilhões de instruções por segundo.

	$n = 10$	$n = 100$	$n = 200$
Máquina 1	$< 1s$	$\approx 6174$ anos	$\approx 5 \times 10^{21}$ milênios

## Exemplo

Seja “Máquina 1” um computador que faz 4 bilhões de instruções por segundo.

Seja “Máquina 2” um computador que faz 40 trilhões de instruções por segundo.

	$n = 10$	$n = 100$	$n = 200$
Máquina 1	$< 1s$	$\approx 6174$ anos	$\approx 5 \times 10^{21}$ milênios
Máquina 2	$< 1s$	$\approx 226$ dias	$\approx 5 \times 10^{17}$ milênios

3. Dá para fazer melhor?

### 3. Dá para fazer melhor?

1: **Função** FIB2( $n$ )

2:     **Se**  $n \leq 2$  **então**

3:         **Devolve** 1

4:     Seja  $F[1..n]$  um vetor de tamanho  $n$

5:      $F[1] = 1$

6:      $F[2] = 1$

7:     **Para**  $i = 3$  até  $n$  **faça**

8:          $F[i] = F[i - 1] + F[i - 2]$

9:     **Devolve**  $F[n]$

### 3. Dá para fazer melhor?

1: **Função** FIB2( $n$ )

2:     **Se**  $n \leq 2$  **então**

3:         **Devolve** 1

4:     Seja  $F[1..n]$  um vetor de tamanho  $n$

5:      $F[1] = 1$

6:      $F[2] = 1$

7:     **Para**  $i = 3$  até  $n$  **faça**

8:          $F[i] = F[i - 1] + F[i - 2]$

9:     **Devolve**  $F[n]$

1. Esse algoritmo resolve o problema?

2. Quanto tempo ele leva?

Seja “Máquina 1” um computador que faz 4 bilhões de instruções por segundo.

Seja “Máquina 2” um computador que faz 40 trilhões de instruções por segundo.

## Exemplo

Seja “Máquina 1” um computador que faz 4 bilhões de instruções por segundo.

Seja “Máquina 2” um computador que faz 40 trilhões de instruções por segundo.

	$n = 10$	$n = 100$	$n = 200$
Máquina 1	$< 1s$	$< 1s$	$< 1s$
Máquina 2	$< 1s$	$< 1s$	$< 1s$



## Exemplo

Seja “Máquina 1” um computador que faz 4 bilhões de instruções por segundo.

Seja “Máquina 2” um computador que faz 40 trilhões de instruções por segundo.

	$n = 10$	$n = 100$	$n = 200$	$n = 1mi$
Máquina 1	$< 1s$	$< 1s$	$< 1s$	$< 1s$
Máquina 2	$< 1s$	$< 1s$	$< 1s$	$< 1s$

## Exemplo

Seja “Máquina 1” um computador que faz 4 bilhões de instruções por segundo.

Seja “Máquina 2” um computador que faz 40 trilhões de instruções por segundo.

	$n = 10$	$n = 100$	$n = 200$	$n = 1mi$	$n = 10bi$
Máquina 1	$< 1s$	$< 1s$	$< 1s$	$< 1s$	$2.5s$
Máquina 2	$< 1s$	$< 1s$	$< 1s$	$< 1s$	$< 1s$

## Exemplo

Seja “Máquina 1” um computador que faz 4 bilhões de instruções por segundo.

Seja “Máquina 2” um computador que faz 40 trilhões de instruções por segundo.

	$n = 10$	$n = 100$	$n = 200$	$n = 1mi$	$n = 10bi$
Máquina 1	$< 1s$	$< 1s$	$< 1s$	$< 1s$	$2.5s$
Máquina 2	$< 1s$	$< 1s$	$< 1s$	$< 1s$	$< 1s$

3. Dá para fazer melhor?

## O que veremos no curso?

0. Uma vez compreendido um problema, como resolvê-lo?
  1. O algoritmo resolve o problema?
  2. Quanto tempo ele leva?
  3. Dá para fazer melhor?

## O que veremos no curso?

0. Uma vez compreendido um problema, como resolvê-lo?
  - Técnicas de solução de problemas (divisão e conquista, gulosos, programação dinâmica).
  - Outros problemas (redução).
  - Complexidade (problemas P, NP e NP-completos).
1. O algoritmo resolve o problema?
2. Quanto tempo ele leva?
3. Dá para fazer melhor?

## O que veremos no curso?

0. Uma vez compreendido um problema, como resolvê-lo?
  - Técnicas de solução de problemas (divisão e conquista, gulosos, programação dinâmica).
  - Outros problemas (redução).
  - Complexidade (problemas P, NP e NP-completos).
1. O algoritmo resolve o problema?
  - Como demonstrar corretude de algoritmos (prova por indução).
2. Quanto tempo ele leva?
3. Dá para fazer melhor?

# O que veremos no curso?

0. Uma vez compreendido um problema, como resolvê-lo?
  - Técnicas de solução de problemas (divisão e conquista, gulosos, programação dinâmica).
  - Outros problemas (redução).
  - Complexidade (problemas P, NP e NP-completos).
1. O algoritmo resolve o problema?
  - Como demonstrar corretude de algoritmos (prova por indução).
2. Quanto tempo ele leva?
  - Vocabulário e técnicas de análise de algoritmos (notação assintótica e recorrências).
3. Dá para fazer melhor?

# O que veremos no curso?

0. Uma vez compreendido um problema, como resolvê-lo?
  - Técnicas de solução de problemas (divisão e conquista, gulosos, programação dinâmica).
  - Outros problemas (redução).
  - Complexidade (problemas P, NP e NP-completos).
1. O algoritmo resolve o problema?
  - Como demonstrar corretude de algoritmos (prova por indução).
2. Quanto tempo ele leva?
  - Vocabulário e técnicas de análise de algoritmos (notação assintótica e recorrências).
3. Dá para fazer melhor?
  - Técnicas de solução de problemas (divisão e conquista, gulosos, programação dinâmica).
  - Estruturas de dados (não básicas).



- Conteúdo dado no quadro, com alguns slides de apoio eventualmente
  - Notas de aula/livro estão disponíveis no site
  - Recursos extras estão disponíveis no site (incluindo videoaulas)

- Conteúdo dado no quadro, com alguns slides de apoio eventualmente
  - Notas de aula/livro estão disponíveis no site
  - Recursos extras estão disponíveis no site (incluindo videoaulas)
  - **Videoaulas não substituem as aulas presenciais!**

- Conteúdo dado no quadro, com alguns slides de apoio eventualmente
  - Notas de aula/livro estão disponíveis no site
  - Recursos extras estão disponíveis no site (incluindo videoaulas)
  - **Videoaulas não substituem as aulas presenciais!**
- Atendimentos: responda o questionário que está no site!
  - Qualquer pergunta e feedback são **sempre** bem-vindos
  - Não deixe dúvidas acumularem

- Conteúdo dado no quadro, com alguns slides de apoio eventualmente
  - Notas de aula/livro estão disponíveis no site
  - Recursos extras estão disponíveis no site (incluindo videoaulas)
  - **Videoaulas não substituem as aulas presenciais!**
- Atendimentos: responda o questionário que está no site!
  - Qualquer pergunta e feedback são **sempre** bem-vindos
  - Não deixe dúvidas acumularem
- Espero que você
  - Participe dos atendimentos
  - Seja autor das suas atividades
  - Se divirta :)

[professor.ufabc.edu.br/~carla.negri/cursos/2023Q2-AA/](http://professor.ufabc.edu.br/~carla.negri/cursos/2023Q2-AA/)

- Estude bem o conteúdo do site.
- Verifique-o com frequência!

## **Exercícios de revisão**

---

- Os exercícios daqui foram retirados da [lista 0](#).
- Existem [materiais de revisão](#) sobre os conteúdos necessários, principalmente sobre [indução](#).

## Exercício 1

Seja  $A$  um conjunto qualquer de números inteiros. Prove que todo subconjunto  $B \subseteq A$  de tamanho finito e não-vazio tem um elemento mínimo.



## Exercício 2

Qual o erro?



## Exercício 3

Denote por  $F_n$  o  $n$ -ésimo número da sequência de Fibonacci.

Definimos  $F_1 = 1$ ,  $F_2 = 1$  e  $F_n = F_{n-1} + F_{n-2}$  para  $n \geq 3$ . Prove que  $F_n \geq 2^{0.5n}$  para todo  $n \geq 6$ .

## Exercício 4

Escreva um algoritmo recursivo que devolva o elemento de maior valor contido em um vetor de  $n$  elementos.

## Exercício 5

Avalie  $\sum_{k=0}^{40} (\log_2 4^k + 1)$ .