Estruturas de Dados

Aula 3 – Complexidade Computacional e Notação Big O

Prof. Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

Complexidade Computacional

Notação Big O

Complexidade de Tempo

Complexidade de Espaço

Referências

Complexidade Computacional

Medindo a eficiência de algoritmos

O que é Complexidade Computacional?

Definição

A **complexidade computacional** analisa o **tempo de execução** e o **uso de memória** de um algoritmo em função do **tamanho da entrada**. Permite comparar a eficiência de algoritmos diferentes para o mesmo problema.

Por que é importante?

Entender a complexidade é essencial para desenvolver programas que processem **grandes volumes de dados** sem consumir recursos excessivos ou levar tempo demais.

As duas dimensões

- **Complexidade de Tempo** — quantas operações o algoritmo realiza
- **Complexidade de Espaço** — quanta memória o algoritmo utiliza

Notação Big O

Descrevendo o pior caso

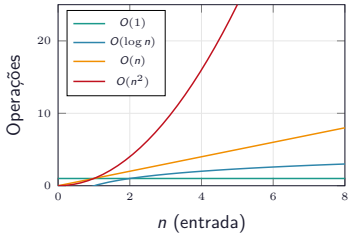
O que é a Notação Big O?

▪ Definição

A **notação Big O** descreve o comportamento de um algoritmo no **pior caso** (*worst case*) à medida que o tamanho da entrada cresce. É uma notação assintótica (família Bachmann–Landau) que permite comparar algoritmos de forma padronizada. [Cormen et al. 2009][D.E. Knuth 1973]

▪ O que ela mede?

A quantidade de **memória** ou **tempo** que o algoritmo requer à medida que o tamanho da entrada n aumenta. É a ferramenta-padrão para análise e comparação de algoritmos.



Principais Complexidades — Visão Geral

$O(1)$ e $O(\log n)$

O(1): tempo constante — acesso direto a um elemento de vetor.

O(log n): logarítmico — Busca Binária, árvores balanceadas.

$O(n)$ e $O(n \log n)$

O(n): linear — busca em vetor não ordenado.

O(n log n): quase-linear — Quicksort, Mergesort.

$O(n^2)$ Quadrático

O tempo é o **quadrado** do tamanho da entrada. Pouco eficiente para grandes volumes.

Exemplos: *Bubble*, *Insertion* e *Selection Sort*.

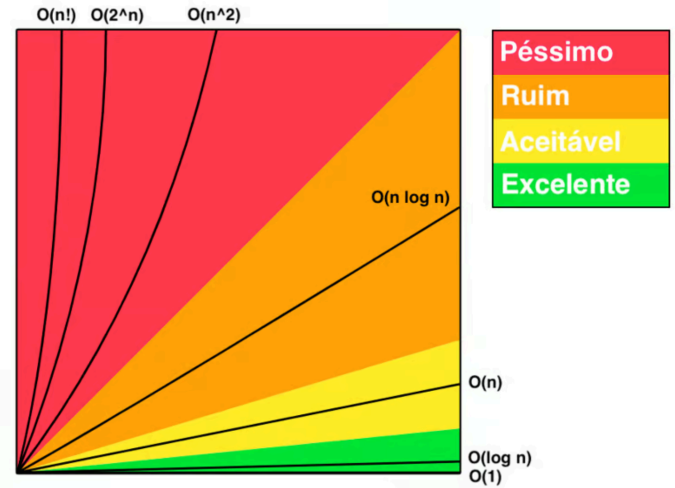
$O(n!)$ Fatorial

Cresce de acordo com o **fatorial** da entrada. Extremamente custoso e pouco escalável.

Exemplo: **Problema do Caixeiro Viajante:** encontrar o menor caminho que passe por todas as cidades exatamente uma vez e retorne à origem.

NOTAÇÃO BIG O	ALGORITMO DE EXEMPLO
O(log n)	Busca binária
O(n)	Busca simples
O(n * log n)	Ordenação Quicksort
O(n2)	Ordenação de seleção
O(n!)	Problema do caixeiro viajante

Gráfico Comparativo das Complexidades



Complexidade de Tempo

Quantas operações o algoritmo realiza?

Complexidade de Tempo — Conceito

Definição

Refere-se ao **tempo máximo** que o algoritmo leva para executar dado um determinado tamanho de entrada — analisado sempre no **pior caso**.

Boas práticas

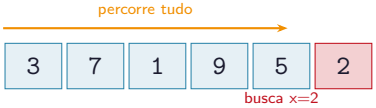
- Reduzir trabalho desnecessário
- Evitar soluções exponenciais
- Escolher algoritmos com menor Big O
- Aproveitar estruturas de dados adequadas

Exemplos

Pesquisa Binária	$O(\log n)$
Busca Linear	$O(n)$
Quicksort	$O(n \log n)$
Selection Sort	$O(n^2)$
Caixeiro Viajante	$O(n!)$

▪ Busca em vetor não ordenado

No pior dos casos, o elemento buscado está na **última posição**. É necessário percorrer todos os n elementos para encontrá-lo.



$\Rightarrow$ $O(n)$

▪ Raciocínio

- Custo no pior caso: n comparações
- $O(n)$ — tempo linear
- O tempo cresce **proporcionalmente** ao tamanho do vetor

Busca Binária em vetor ordenado

Inicia no **meio** do vetor. A cada passo, **descarta metade** da entrada. No pior dos casos, o número de comparações é $\log_2 n$.

$\Rightarrow$ $O(\log n)$

Raciocínio

- A cada passo, descarta **metade** da entrada
- Custo no pior caso: $\log_2 n$ comparações
- $O(\log n)$ — muito mais eficiente que $O(n)$
- **Condição**: o vetor deve estar **ordenado**

Complexidade de Espaço

Quanta memória o algoritmo utiliza?

Complexidade de Espaço

▪ Definição

A **complexidade de espaço** Big O descreve quanta **memória** é necessária para executar um algoritmo no pior caso. É tão importante quanto a complexidade de tempo em sistemas com recursos limitados.

▪ Funcionamento

A análise é análoga à de tempo: avalia-se o quanto de memória extra o algoritmo usa em função do tamanho n da entrada, sempre no **pior caso**.

▪ Exemplo: Selection Sort

O *Selection Sort* tem complexidade de espaço $O(1)$: armazena apenas um **valor mínimo** e seu **índice** para comparação — o espaço usado **não cresce** com o tamanho da entrada.

Selection Sort usa $O(1)$

$O(1)$ | $O(\log n)$ | $O(n)$ | $O(n^2)$

Referências I

CORMEN, T. H. et al. *Introduction to Algorithms*. 2nd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262032937.

D.E. Knuth. *The Art of Computer Programming*. [S.l.]: Addison-Wesley, 1973. v. 1 - 3.