Universidade do Estado da Bahia (UNEB)
Departamento de Ciências Exatas e da Terra (DCET-I)
Disciplina: Engenharia de Programas
Docente: Diego G. F. Suarez
Discentes: Davi M. B. Barbosa, Jéssica T. N. de Jesus e Yan S. Barreiro

Trabalho Prático: Análise do algoritmo Bucket Sort

Introdução:

1. Aprender como o algoritmo funciona.
2. Funcionamento do algoritmo de forma visual no Data Structure Visualization
3. Link: https://www.cs.usfca.edu/~galles/visualization/BucketSort.html

1. Relatório de estabilidade do setup experimental (descrição da configuração e dos testes em diferentes horários e dias, com histograma e critério de estabilidade garantido): (2.0)

Davi's laptop = 00
Processador: Intel i7-8550U @ 1.80Ghz
RAM: 8 GB
GPU: AMD Radeon 520
HDD 1 TB

Jessica's laptop = 01
Processador: Intel i7-11700 @ 2.50Ghz
RAM: 32 GB @ 3200 Mhz
GPU: NVidia RTX 2060
SSD PCIe 256GB

Yan's laptop = 02
Processador: Intel i5-10gen
RAM: 8GB
GPU: Integrated Intel
SSD PCIe 256GB

Docker
Versão: 24.0.2
Imagem: jupyter/scipy-notebook
Versão da imagem: lab-4.0.1

Demonstração no Notebook: “01 - Relatório.ipynb”:
Relatórios realizados em dias e horários diferentes encontram-se na pasta /Relatórios.

- ✓ Realizado no laptop 00 no dia 04/06/2023 às 18h51
  R. PDF: ”00 - Relatório Notebook i7 - 04-06 - 18:51.pdf”

- ✓ Realizado no laptop 01 no dia 14/06/2023 às 15h41
  R. PDF: “01.1 - Relatório Notebook i7 - 14-06 - 15:41.pdf”

- ✓ Realizado no laptop 01 no dia 13/06/2023 às 23h30??
  R. PDF: “01.2 - Relatório Notebook i7 - 13-06 - 23:30.pdf”

- ✓ Realizado no laptop 02 no dia 12/06/2023 às 17h41
  R. PDF: “02.1 - Relatório Notebook i5 - 12-06 - 17:41.pdf”

- ✓ Realizado no laptop 02 no dia 14/06/2023 às 16h33
  R. PDF: “02.2 - Relatório - Notebook I5 - 14-06 - 16:33.pdf”

- ✓ Realizado no laptop 02 no dia 17/06/2023 às 18h15
  R. PDF: “02.3 - Relatório Notebook i5 - 17-06 - 18:15.pdf”

2. Calibração do hardware usado para teste “cálculo dos MFLOPS”: (1.0)
R. Notebooks: “01 - Relatório.ipynb” e “02 - Bucket Sort Final.ipynb”.

3. Estudo experimental (tempo médio vs n): (3.0)
R. No pior caso o experimento apresenta no gráfico o comportamento de uma curva exponencial $O(n^2)$
Uso da função “generate_pascal_row(row_index)” para simular o pior caso;
Notebook: “02 - Bucket Sort Final.ipynb”

4. Estudo teórico (3.0):

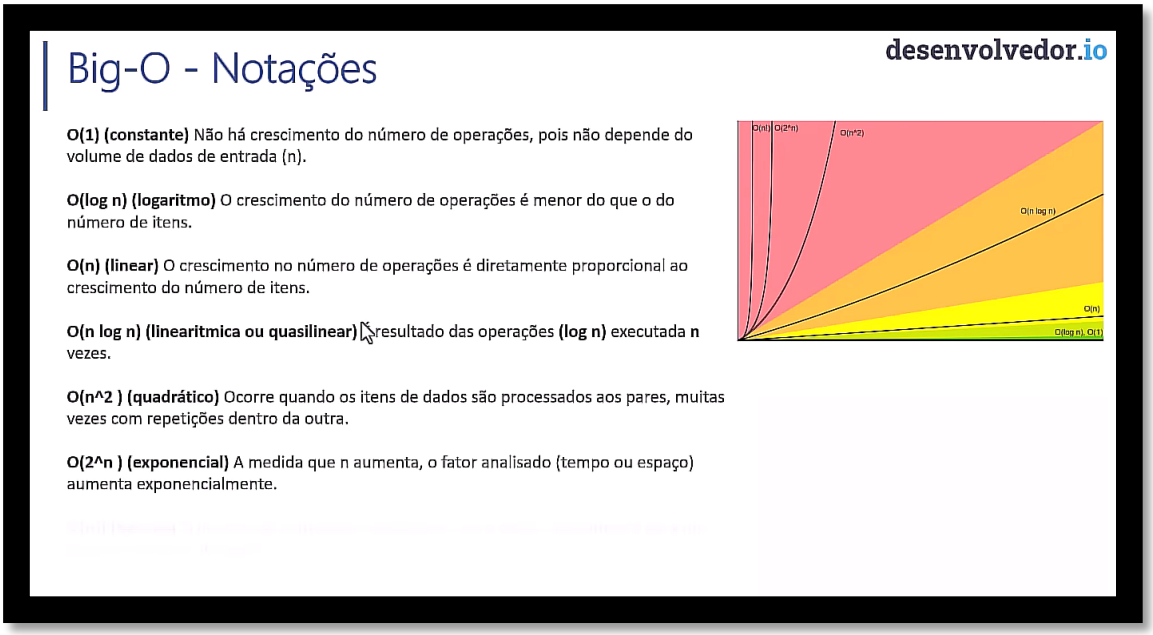


Imagem 01

4.1. Contagem de número de operações:
R. O (A,O,C) foi obtido através da contagem das atribuições explícitas e implícitas, contagem do laço “for” e as operações presentes no algoritmo.

4.2. Obtenção da fórmula geral:
R. $11n^2+47n+17$

4.3. Estimativa das complexidade:
R. $O(n^2)$

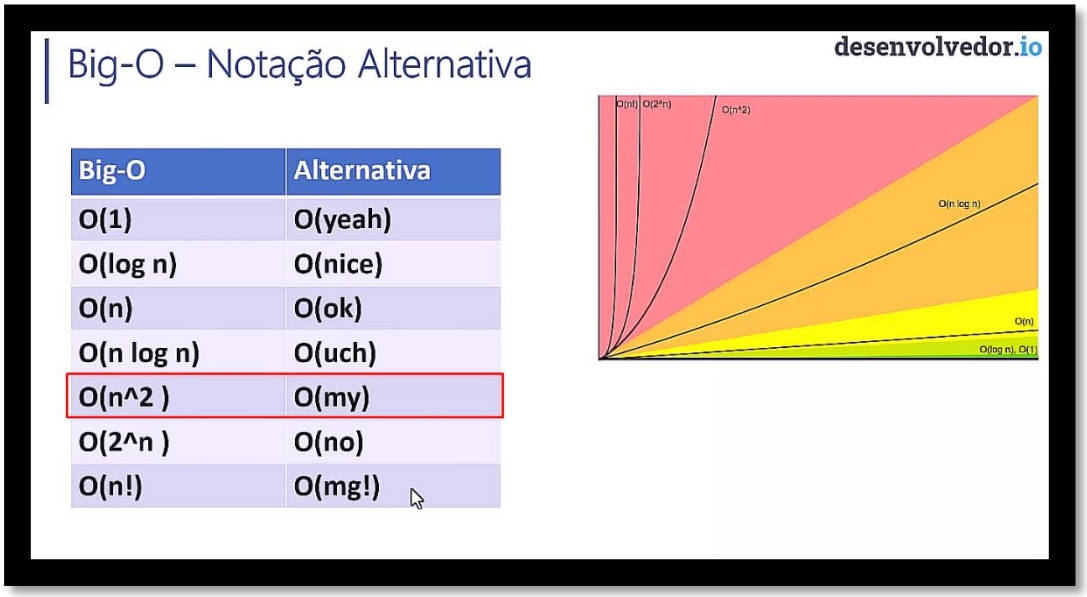
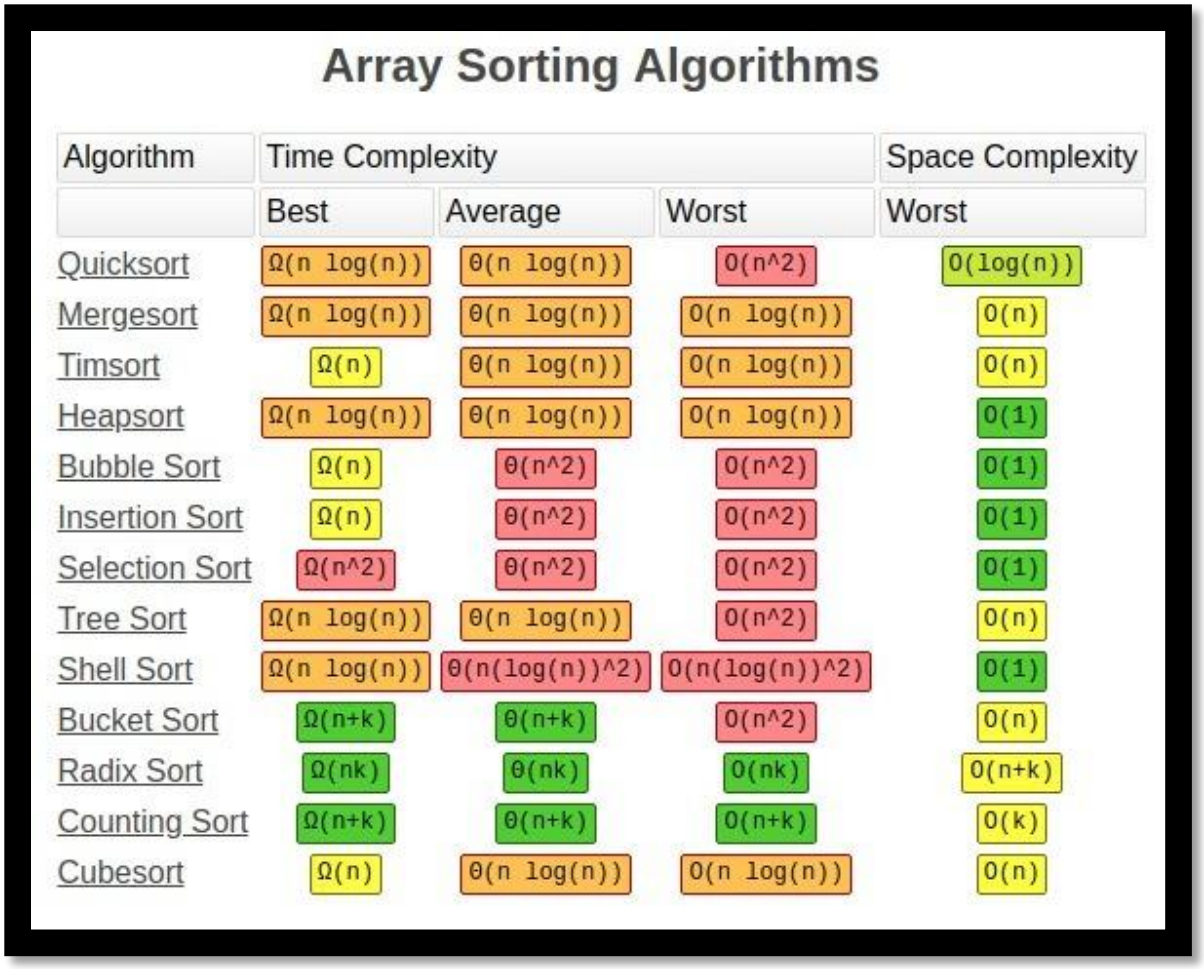


Imagem 02

4.4. Big O Notation - Análise da complexidade no tempo de algoritmos:
R. Resultados obtidos -> inserir aqui imagens do vídeo
Notebook: “02 - Bucket Sort Final.ipynb”
Fonte das imagens 01 e 02:” NOTAÇÃO BIG-O COMO CLASSIFICAR A COMPLEXIDADE DE UM CÓDIGO? VOCÊ PRECISA SABER!”, Link: https://www.youtube.com/watch?v=KjJwx-AB4KI&t=2s
Fonte: “Know Thy Complexities”, Link: https://www.bigocheatsheet.com

5. Comparação teoria-prática (gráfica e discussão de resultados): (1.0)
R. Apresentação da curva exponencial do resultado do experimento prático (blue);
R. Apresentação da curva exponencial do resultado experimento teórico (red);
R. Resultados iniciais apresentaram um gráfico similar a o(N), após encontrar a fórmula para simular o pior caso foi possível identificar uma curva exponencial O (N^2), com sistema não calibrado e resultados favoráveis com sistema calibrado e após encontrar a solução para simulação do pior caso.
Notebook: “02 - Bucket Sort Final.ipynb”



Array Sorting Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	θ(n log(n))	O(n log(n))	O(n)

Imagem 03

Fonte da imagem 03 :" Sorting Algorithms":
Link: https://lamfo-unb.github.io/2019/04/21/Sorting-algorithms/