

INSERTION SORT

ALUNOS:
ANA JULIA VIEIRA PEREIRA ANDRADE DA COSTA
GABRIEL PEIXOTO MENEZES DA COSTA

PROFESSOR:
HERBERT OLIVEIRA ROCHA



DESCRIÇÃO DO INSERTION SORT

Começa com o elemento $A[0]$ já ordenado.

Para cada i de 1 a $n-1$:

Armazena $key = A[i]$.

Desloca elementos maiores que key na parte ordenada ($A[0..i-1]$) uma posição à direita.

Insere key na posição correta.

ALGORITMO

Algoritmo InsertionSort(A):

Para i de 1 até $\text{tamanho}(A) - 1$ faça:

$\text{chave} \leftarrow A[i]$

$j \leftarrow i - 1$

Enquanto $j \geq 0$ e $A[j] > \text{chave}$ faça:

$A[j + 1] \leftarrow A[j]$

$j \leftarrow j - 1$

FimEnquanto

$A[j + 1] \leftarrow \text{chave}$

FimAlgoritmo

FUNÇÃO DE CUSTO E COMPLEXIDADE

INSERTION SORT FUNÇÃO DE CUSTO

1 Pior Caso

Cada elemento precisa ser comparado e movido por todo o subvetor anterior:

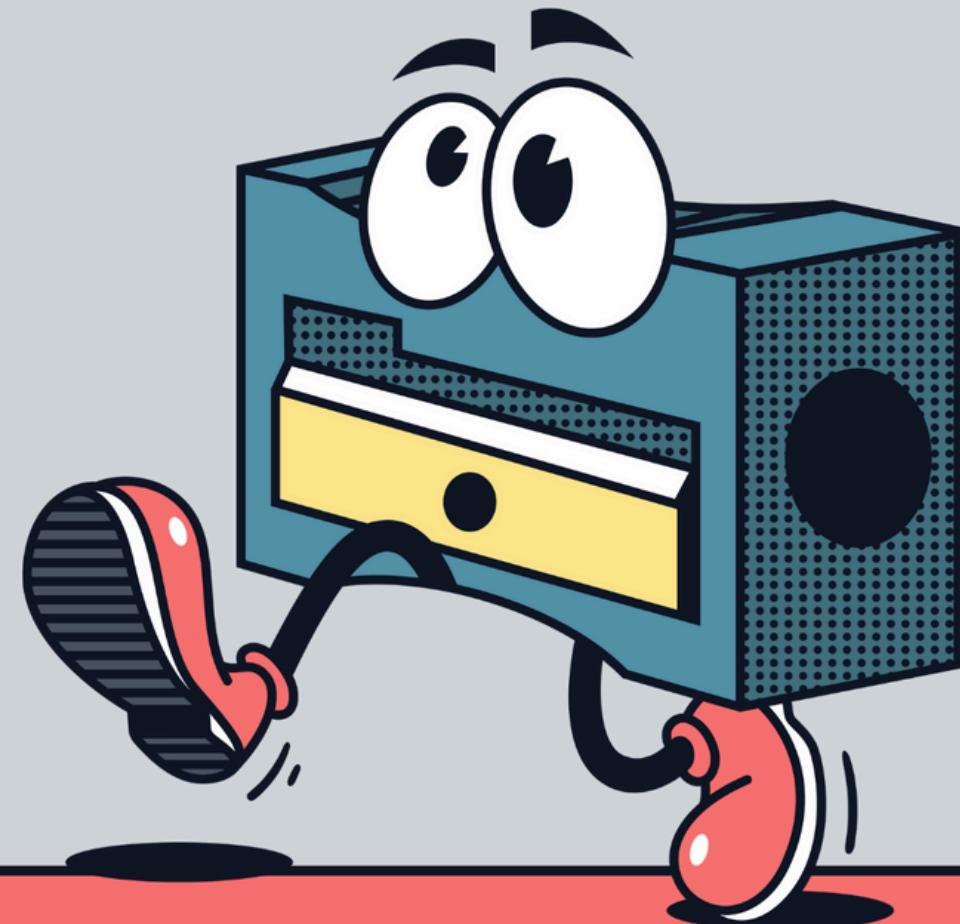
1. Iteração 1: 1 comparação
2. Iteração 2: 2 comparações
- 3...
4. Iteração i: i comparações

$$T(n) = 1 + 2 + 3 + \dots + (n - 1) = \sum_{i=1}^{n-1} i = \frac{n(n - 1)}{2}$$

2 Melhor Caso

O while ($\text{arr}[j] > \text{chave}$) nunca entra, então só há uma comparação por iteração

$$T(n) = n - 1$$



FUNÇÃO DE CUSTO E COMPLEXIDADE

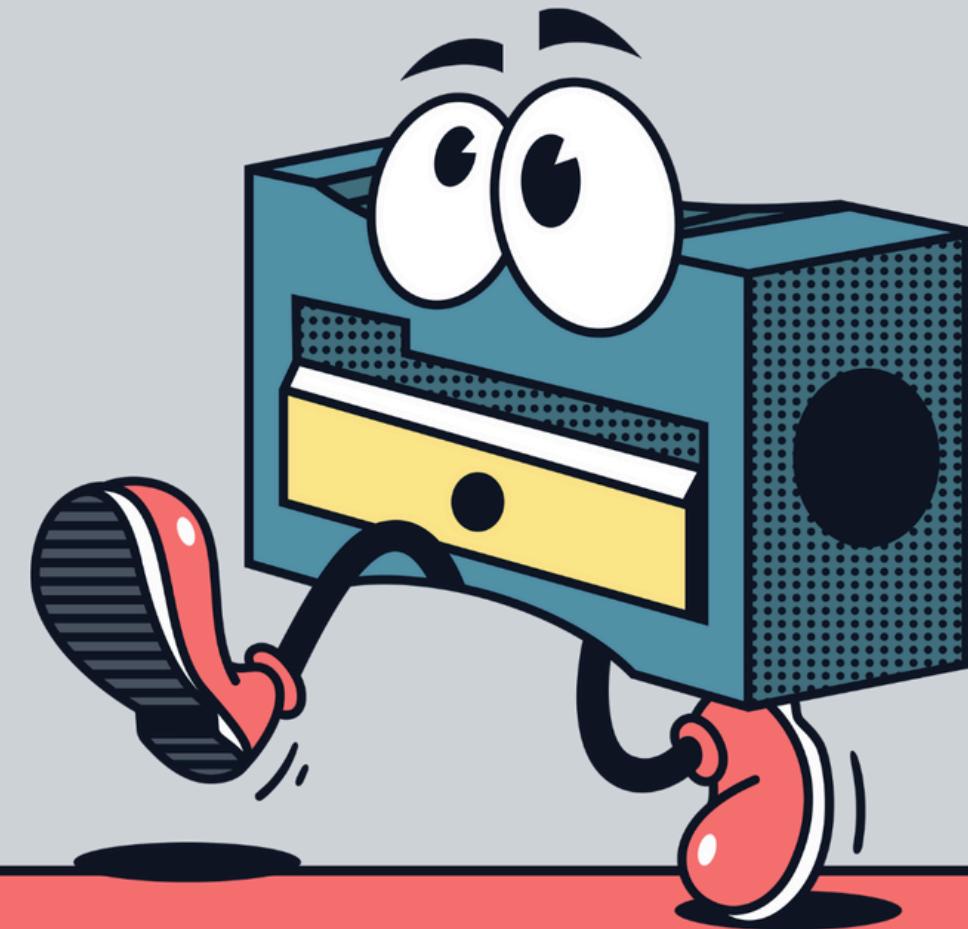
INSERTION SORT COMPLEXIDADE

1 Pior Caso

$O(n^2)$

2 Melhor Caso

$O(n)$



DESCRIÇÃO DO MERGE SORT

O Merge Sort é um algoritmo de ordenação que funciona segundo o princípio “divide e conquista”: ele divide recursivamente ao meio o vetor até que cada subvetor tenha tamanho 1 (que já está ordenado). Em seguida, realiza a etapa de “merge”, em que dois subvetores contíguos ordenados são unidos em um só. A complexidade do Merge Sort é $O(n \log n)$ em todos os casos.

CÓDIGO EM C

**Função MERGE(arr, l, m, r,
comparacoes):**

$n1 \leftarrow m - l + 1$

$n2 \leftarrow r - m$

**Criar vetores L[0...n1-1] e R[0...n2-
1]**

Para i de 0 até n1 - 1:

$L[i] \leftarrow arr[l + i]$

Para j de 0 até n2 - 1:

$R[j] \leftarrow arr[m + 1 + j]$

$i \leftarrow 0, j \leftarrow 0, k \leftarrow l$

Enquanto $i < n1$ e $j < n2$:
 $comparacoes \leftarrow comparacoes + 1$

Se $L[i] \leq R[j]$:

$arr[k] \leftarrow L[i]$

$i \leftarrow i + 1$

Senão:

$arr[k] \leftarrow R[j]$

$j \leftarrow j + 1$

$k \leftarrow k + 1$

Enquanto $i < n1$:

$arr[k] \leftarrow L[i]$

$i \leftarrow i + 1$

$k \leftarrow k + 1$

Enquanto $j < n2$:

$arr[k] \leftarrow R[j]$

$j \leftarrow j + 1$

$k \leftarrow k + 1$

CÓDIGO EM C

Função MERGE_SORT(arr, l, r, comparacoes):

Se $l < r$:

$$m \leftarrow l + (r - l) / 2$$

MERGE_SORT(arr, l, m, comparacoes)

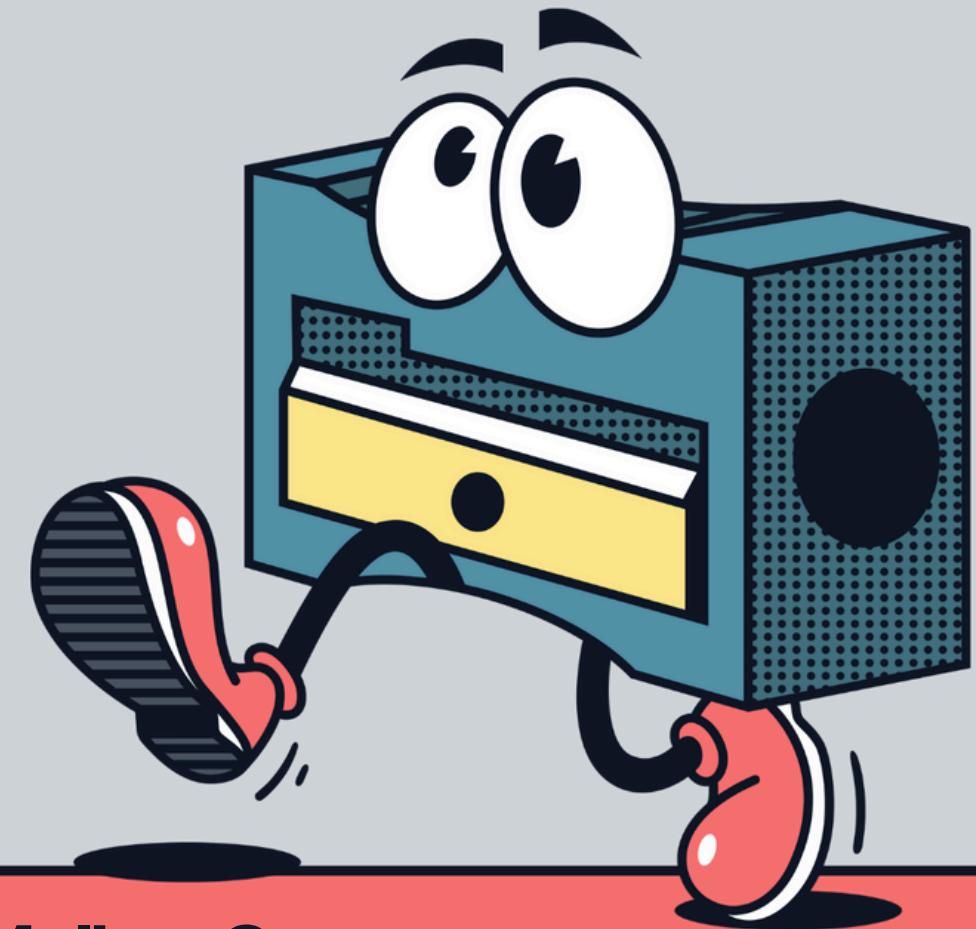
MERGE_SORT(arr, m + 1, r, comparacoes)

MERGE(arr, l, m, r, comparacoes)

FUNÇÃO DE RECORRÊNCIA E COMPLEXIDADE

MERGE SORT

FUNÇÃO DE RECORRÊNCIA



1

Pior Caso e Melhor Caso

$$T(n) = \begin{cases} c & \text{se } n = 1, \\ 2T(n/2) + cn & \text{se } n > 1, \end{cases}$$

FUNÇÃO DE RECORRÊNCIA E COMPLEXIDADE

MERGE SORT

FUNÇÃO DE RECORRÊNCIA

2 Como Funciona

1. 1^a iteração:

$$T(n) = 2T\left(\frac{n}{2}\right) + cn$$

2. 2^a iteração:

$$\begin{aligned} T(n) &= 2\left[2T\left(\frac{n}{4}\right) + c \cdot \frac{n}{2}\right] + cn \\ &= 4T\left(\frac{n}{4}\right) + 2c \cdot \frac{n}{2} + cn \\ &= 4T\left(\frac{n}{4}\right) + 2cn \end{aligned}$$

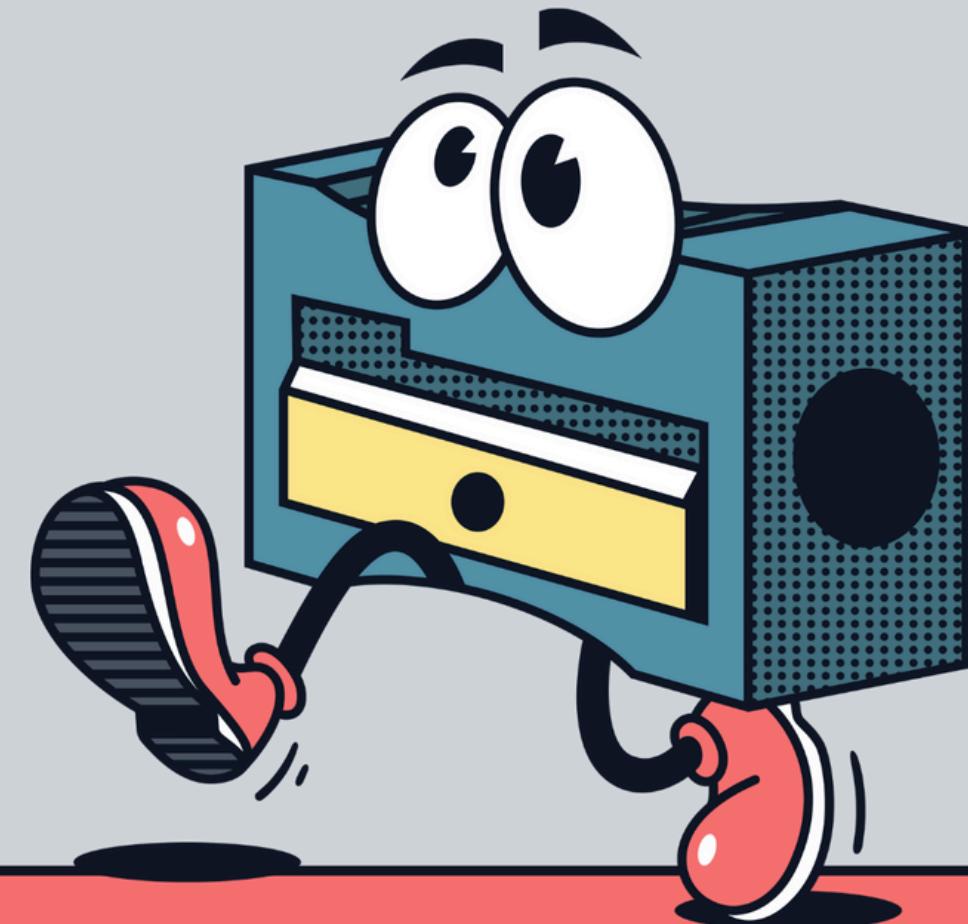
3. k^a iteração:

$$T(n) = 2^k T\left(\frac{n}{2^k}\right) + kc n$$

4. Parar quando $\frac{n}{2^k} = 1 \Rightarrow k = \log_2 n$

5. Substituindo:

$$T(n) = n \cdot T(1) + cn \cdot \log_2 n$$



FUNÇÃO DE RECORRÊNCIA E COMPLEXIDADE

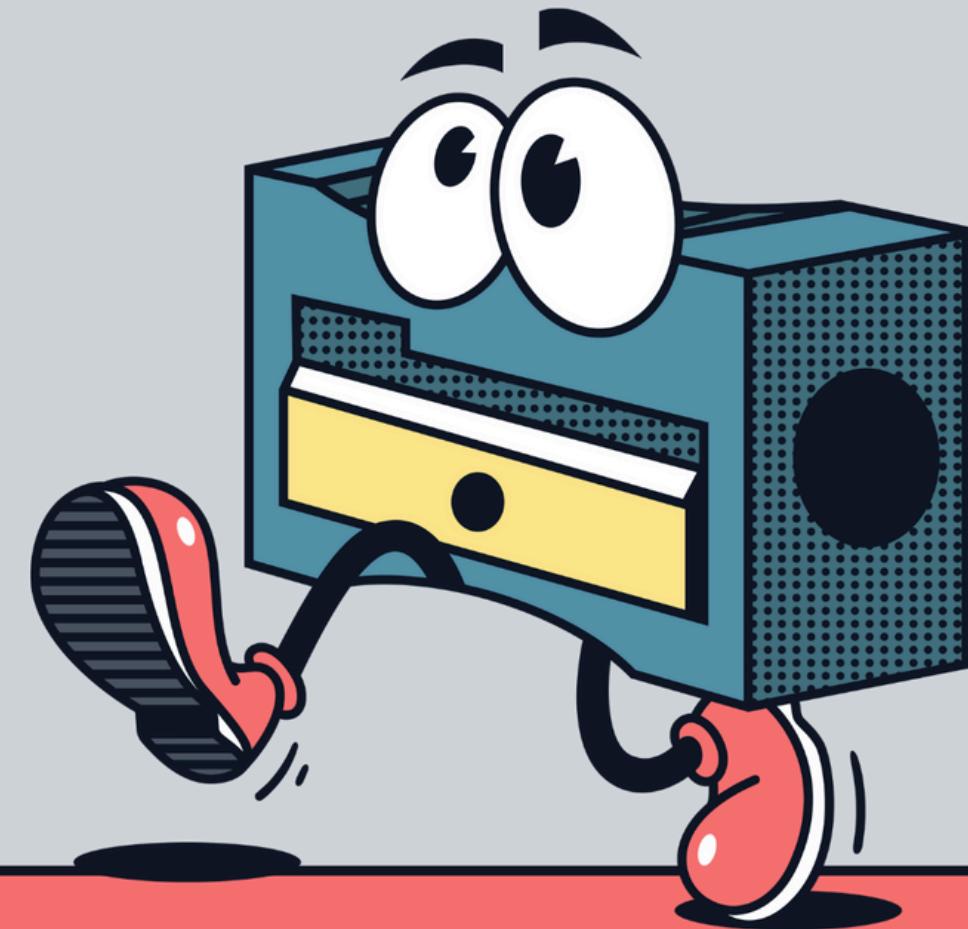
MERGE SORT COMPLEXIDADE

1 Pior Caso

$O(n \log n)$

2 Melhor caso

$O(n \log n)$



AMBIENTE DE TESTES

Sistema Operacional: Windows 11 Pro 64 bits

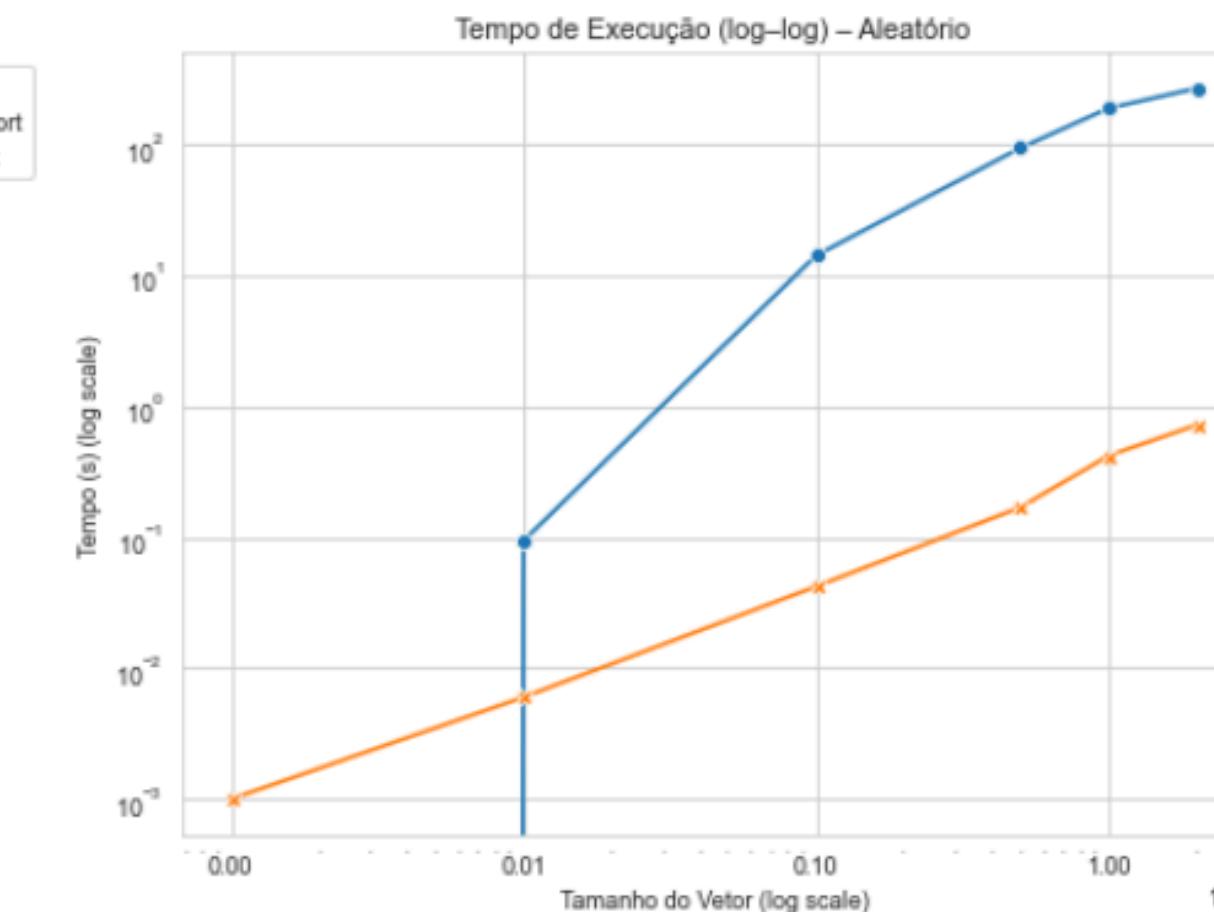
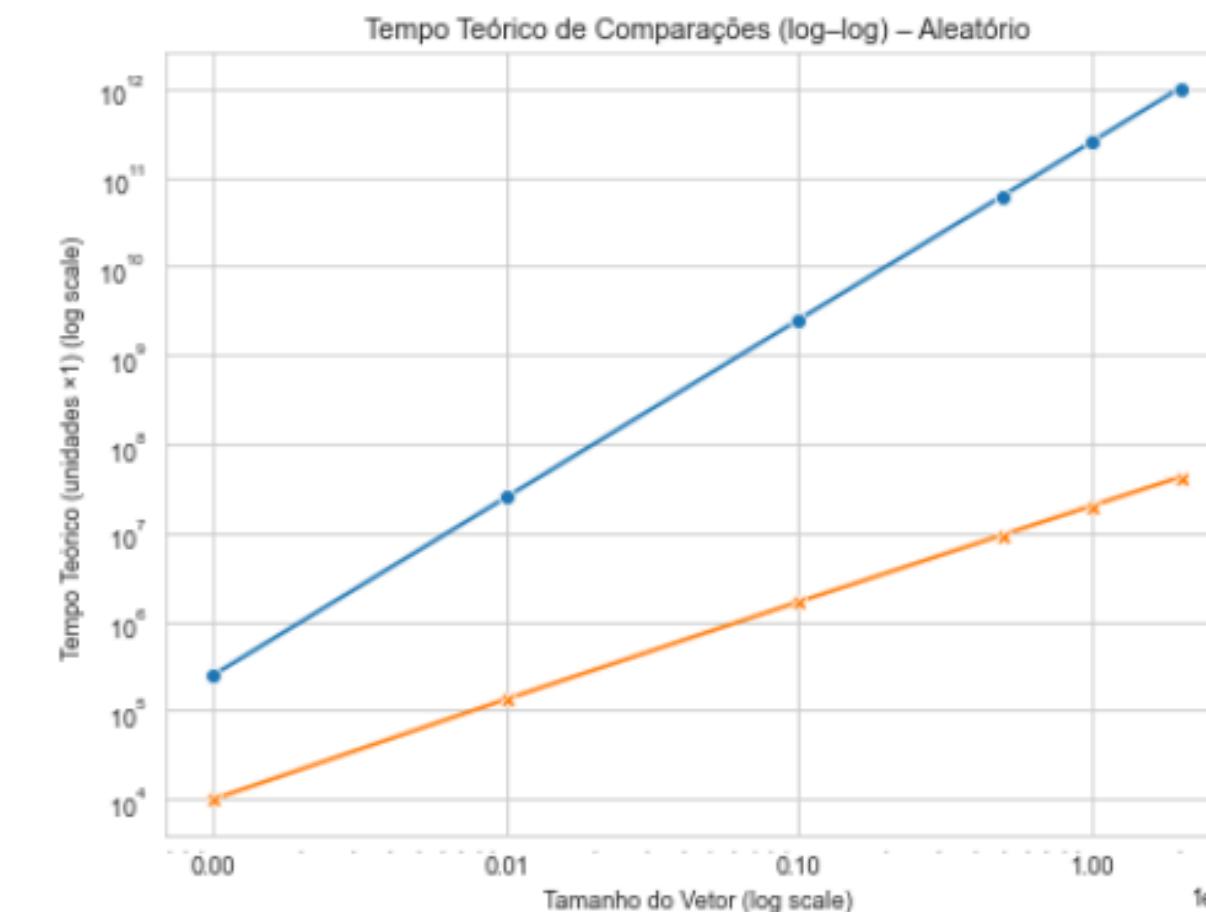
**Plataforma de Testes: Notebook Lenovo Gamer 3i
82CG0005BR**

Especificações:

- **Processador: Intel Core i7-10750H (2.60GHz até 5.00GHz)**
- **Placa de Vídeo: NVidia Geforce GTX1650 (4GB GDDR6)**
- **Memória RAM: 8GB DDR4**
- **SSD: 512GB**

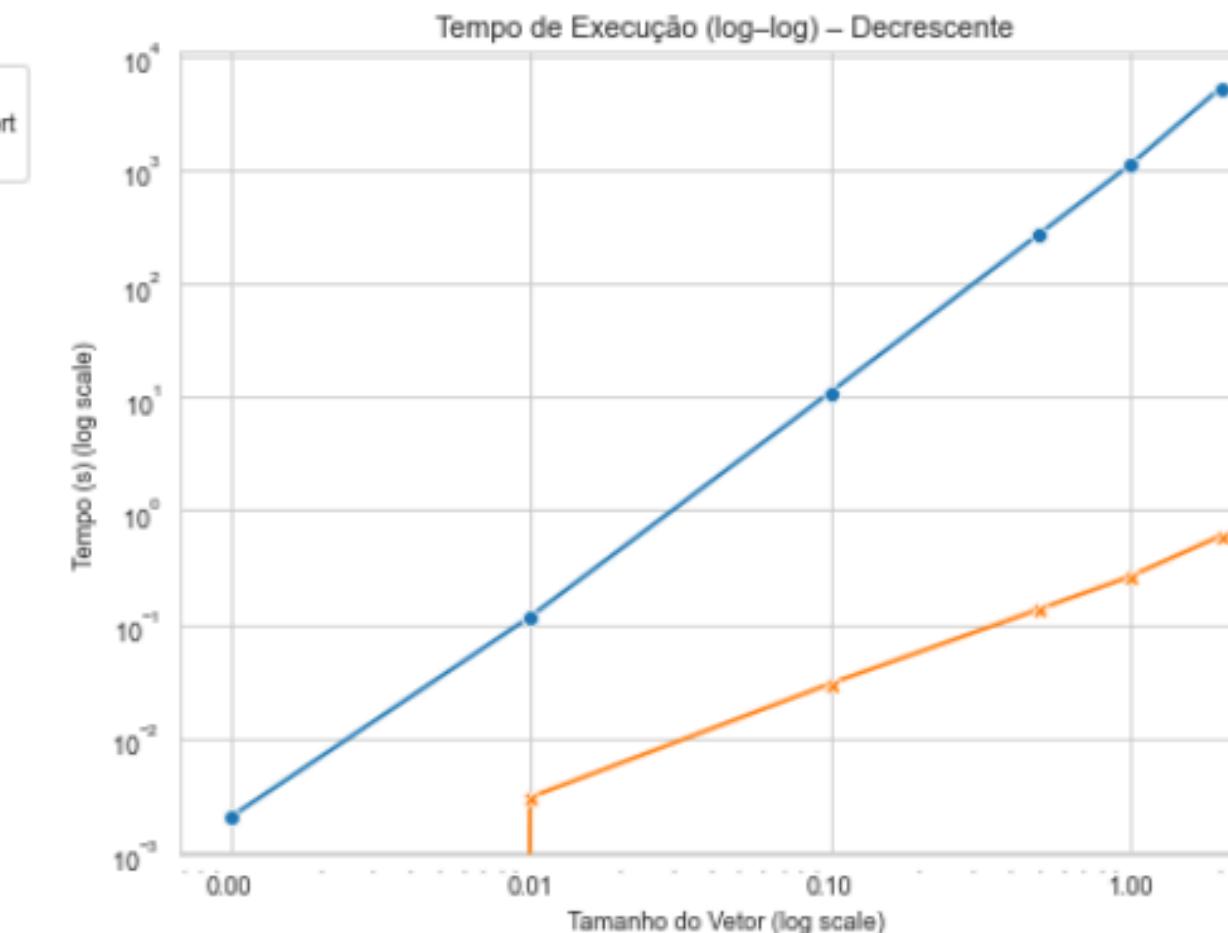
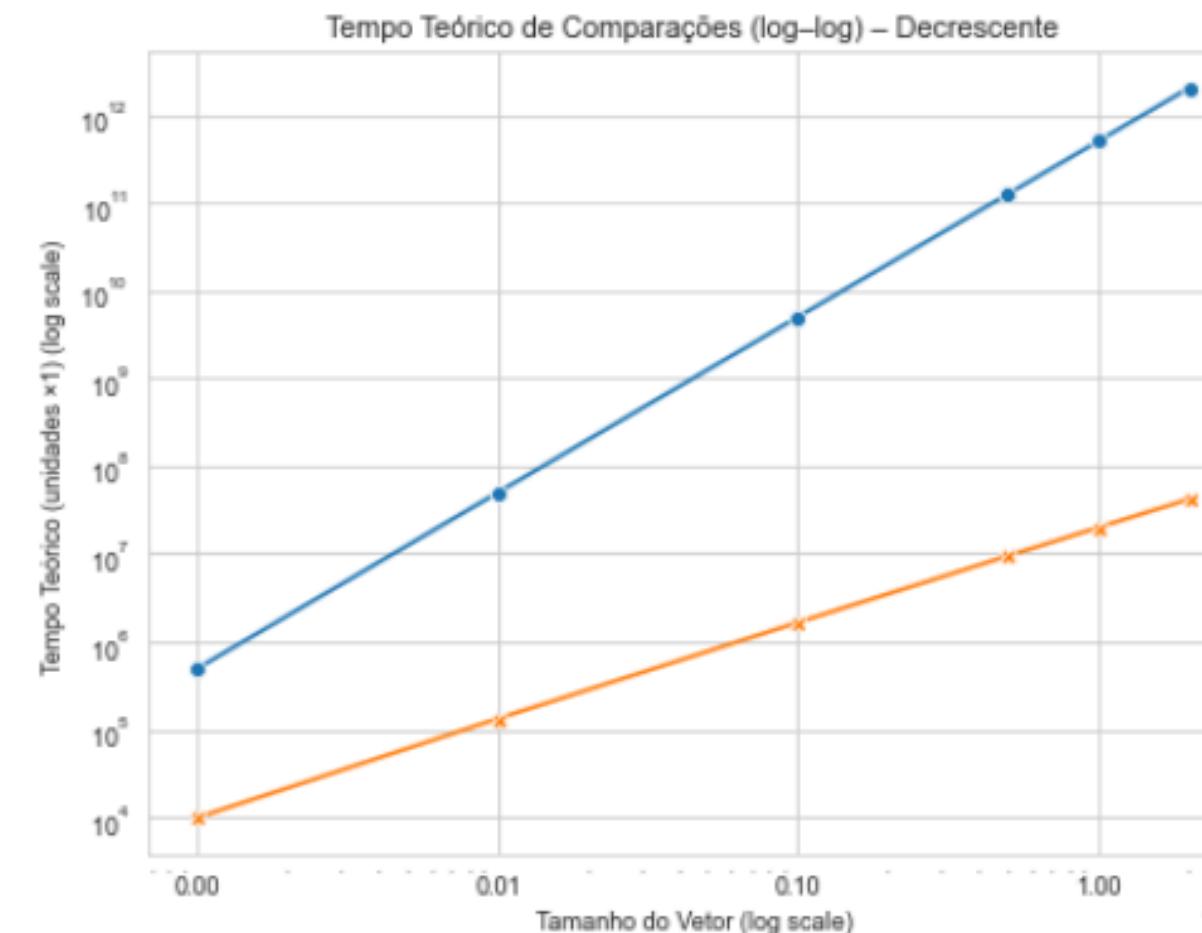
ANALISE DE GRAFICOS

Tempo Estimado – Vetor Aleatório



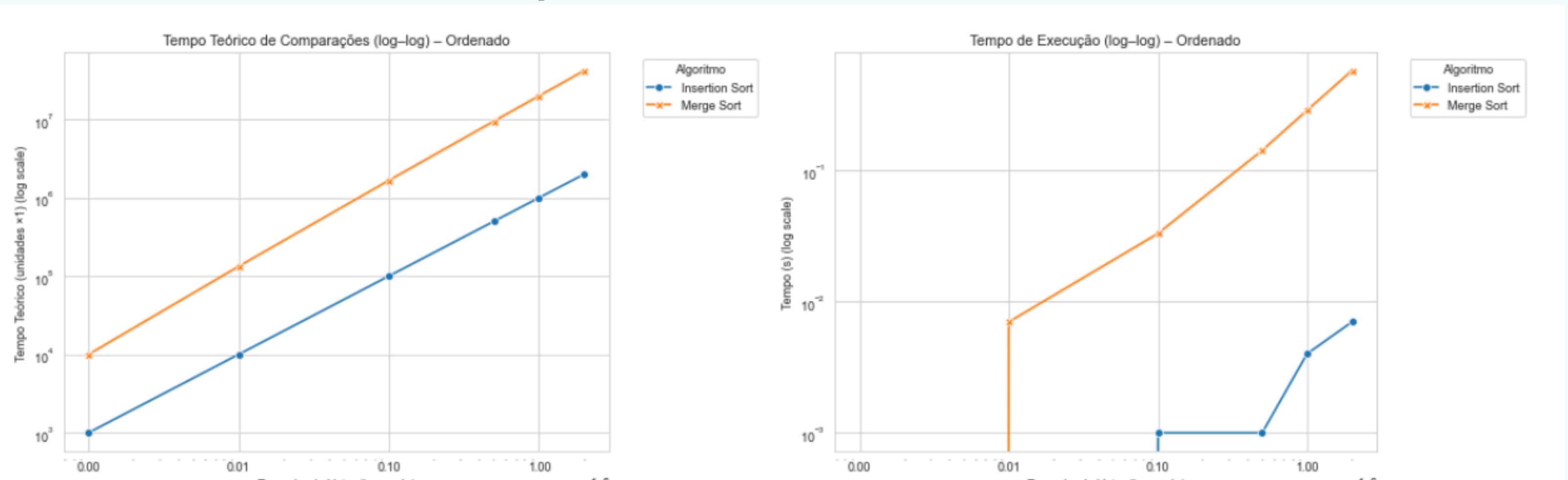
ANALISE DE GRAFICOS

Tempo Estimado – Vetor em Ordem Decrescente



ANALISE DE GRAFICOS

Tempo Estimado – Vetor Ordenado



**Muito Obrigado
Pela Atenção**

