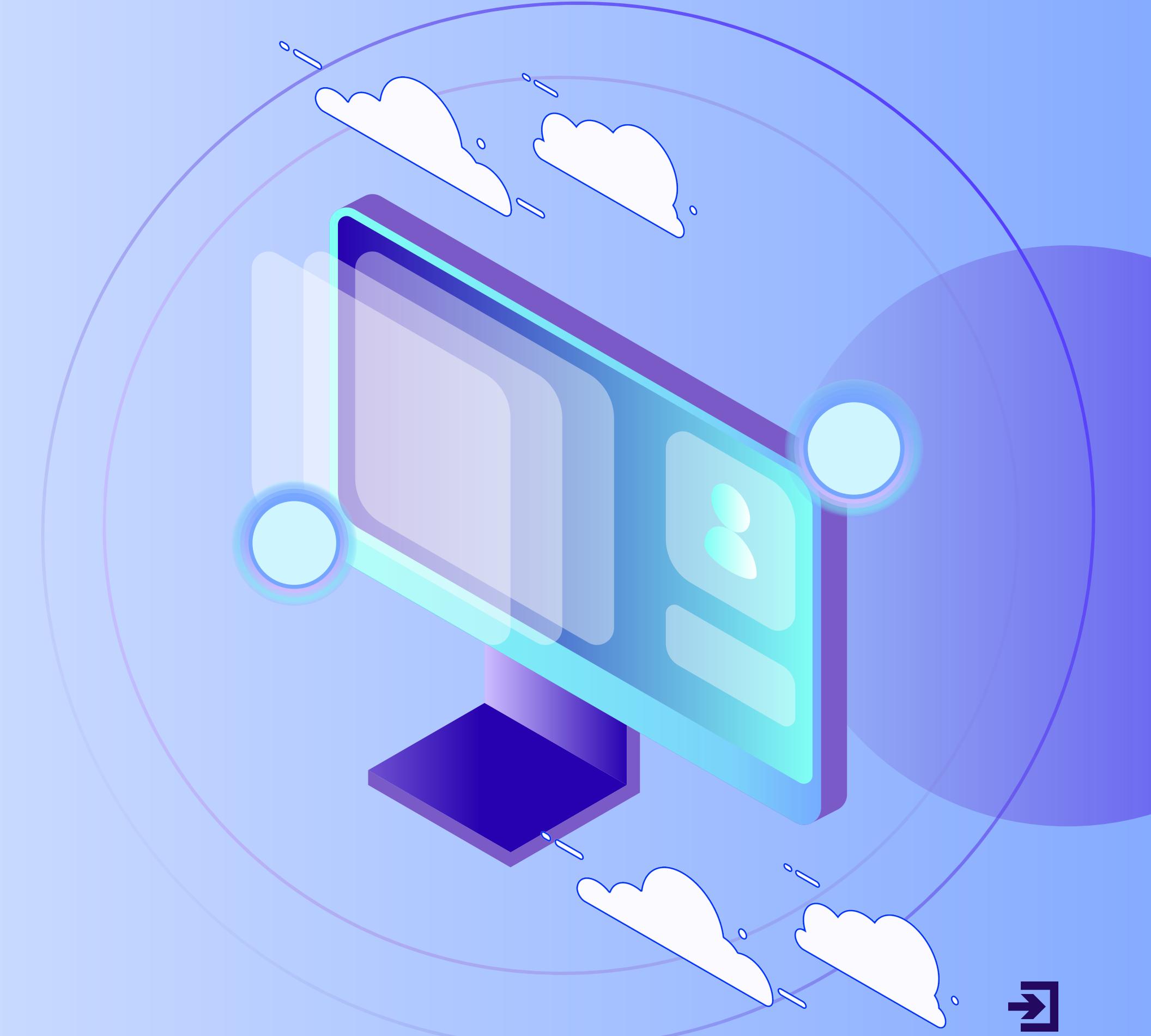


Anna Beatryz Costa - 2025007883

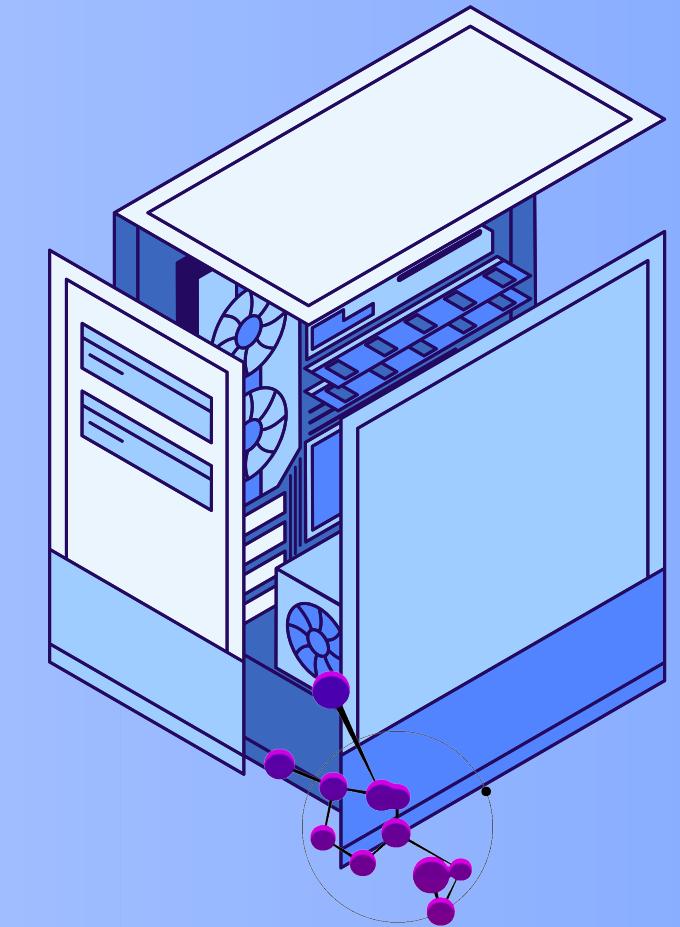
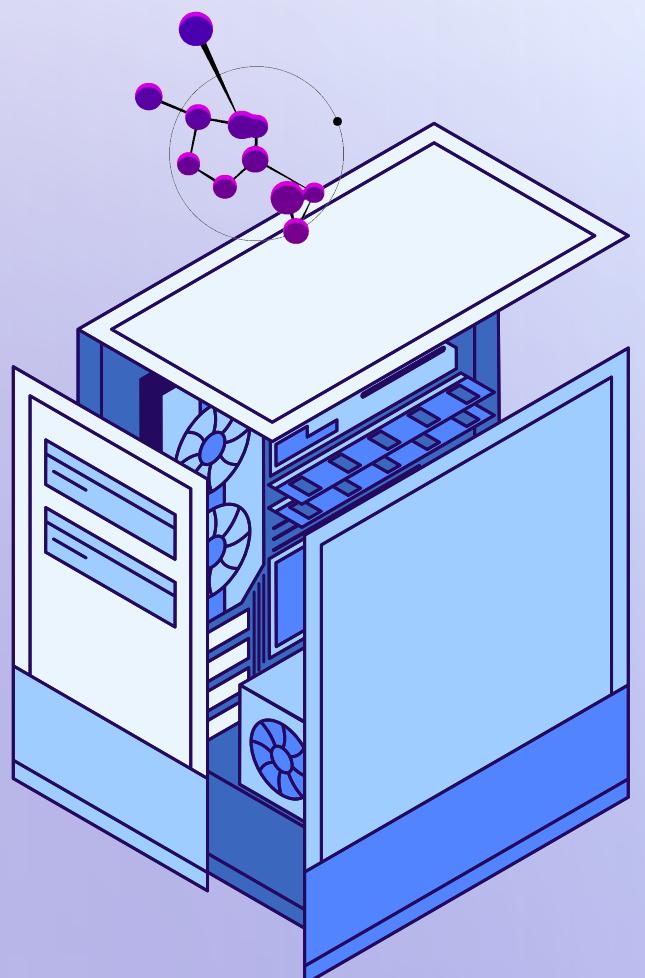
COUNT SORT

Trabalho de Algoritmos e Estrutura de Dados II
Orientadora: Prof.^ª Vanessa Souza



INTRODUÇÃO

Durante as aulas vimos que desempenho dos algoritmos de ordenação pode variar muito dependendo do tamanho e da ordem inicial dos dados. Este trabalho, desenvolvido individualmente para a disciplina de Algoritmos e Estrutura de Dados II, tem como objetivo fazer a análise comparativa do tempo de execução dos algoritmos **Insertion Sort**, **Quick Sort** e **Counting Sort** em linguagem C, avaliando seu comportamento em diferentes cenários de entrada para entendermos melhor suas eficiências na prática.



METODOLOGIA



ETAPAS DESENVOLVIDAS

- Estudo teórico dos algoritmos de ordenação.
- Implementação em C dos algoritmos:
 - Insertion Sort;
 - Quick Sort;
 - Counting Sort.
- Desenvolvimento de sistema de testes automatizados, com geração de dados e medição de desempenho.
- Análise empírica dos resultados, com organização em gráficos e tabelas.



O QUE É O COUNTING SORT

- Algoritmo de ordenação não baseado em comparação
- Ideal para valores inteiros em intervalo conhecido
- Complexidade:
 - Tempo: $O(n + k)$
 - Espaço: $O(n + k)$
- Estável



FUNCIONAMENTO



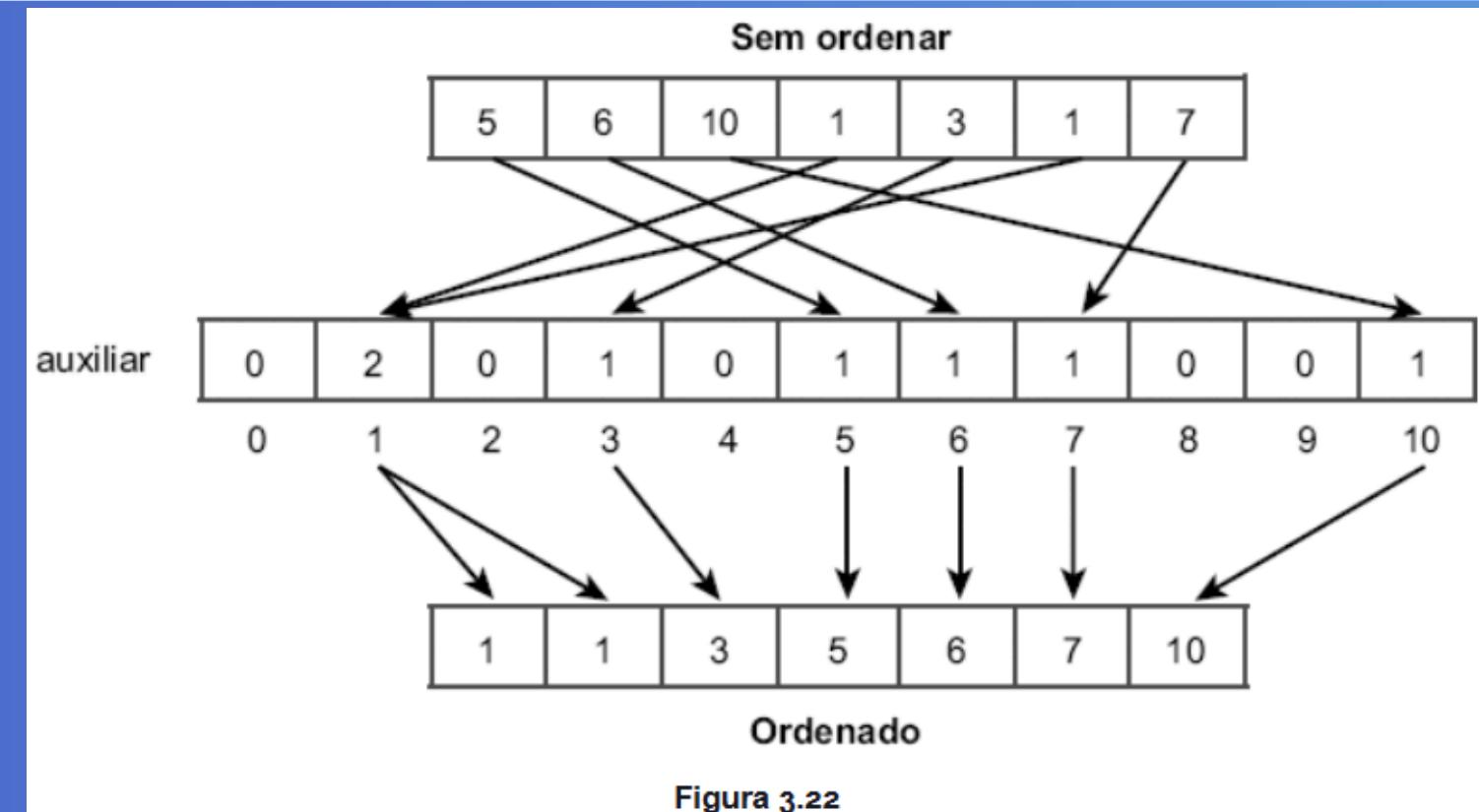
Método *counting sort*

```

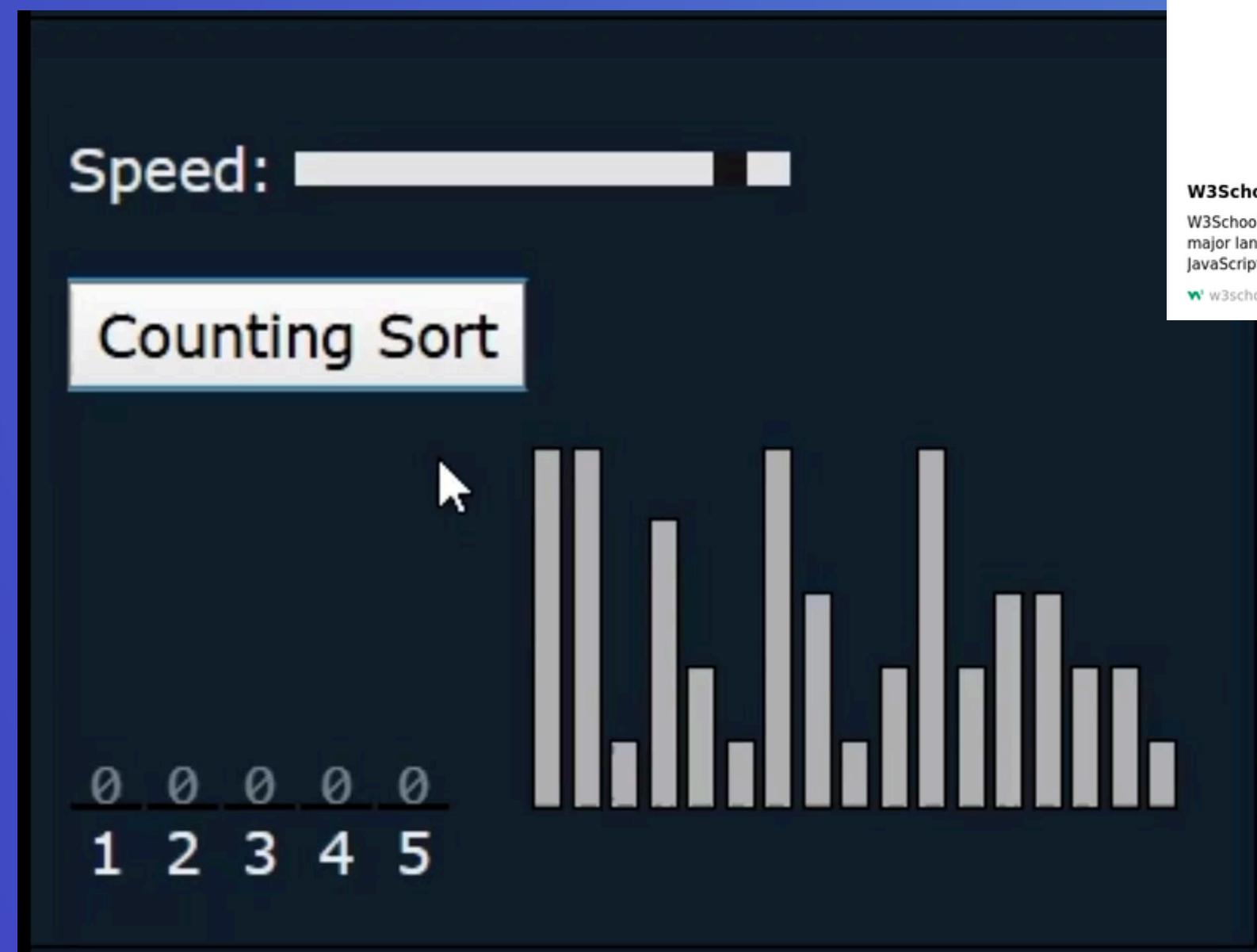
01 #define MAX 100
02 void countingSort(int *v, int N){
03     int i, j, k;
04     int auxiliar[MAX];
05     for(i = 0; i < MAX; i++)
06         auxiliar[i] = 0;
07
08     for(i = 0; i < N; i++)
09         auxiliar[v[i]]++;
10
11    for(i = 0, j = 0; j < MAX; j++)
12        for(k = auxiliar[j]; k > 0; k--)
13            v[i++] = j;
14 }
```

Etapas:

1. Contagem de frequência
2. Soma acumulada
3. Distribuição dos elementos



FUNCIONAMENTO



W3Schools.com

W3Schools offers free online tutorials, references and exercises in all the major languages of the web. Covering popular subjects like HTML, CSS, JavaScript, Python, SQL, Java, and many, many more.

w3schools.com

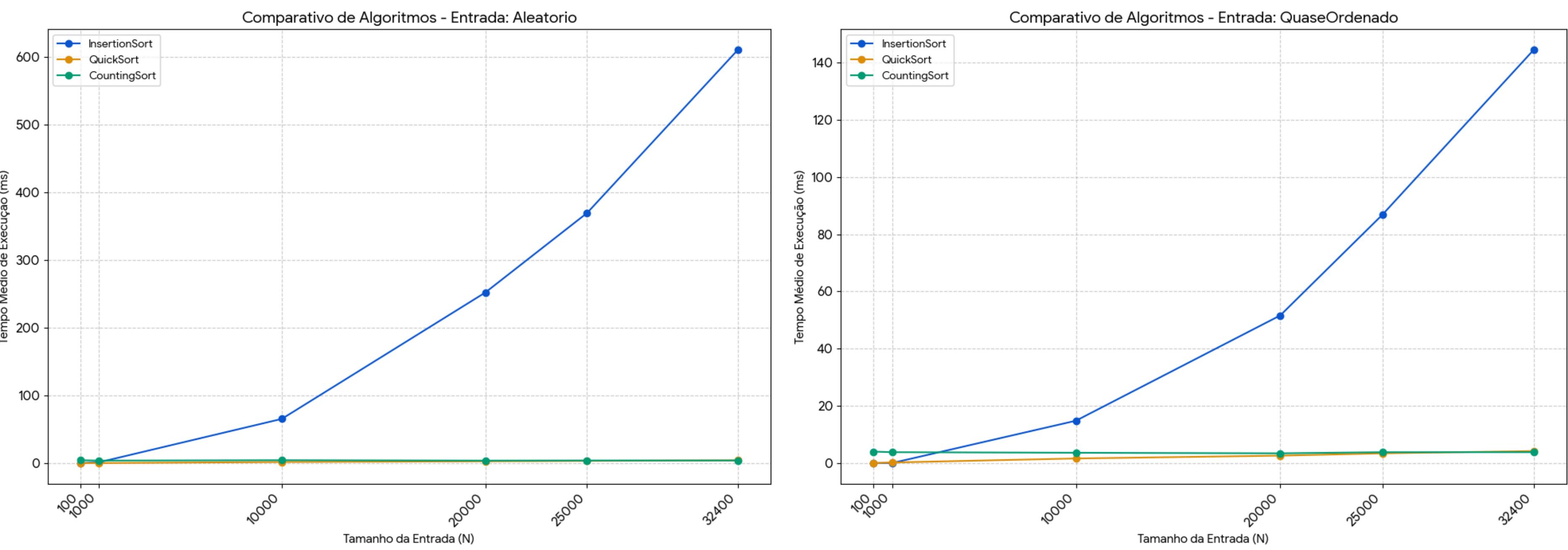


ÁNALISE BENCHMARK

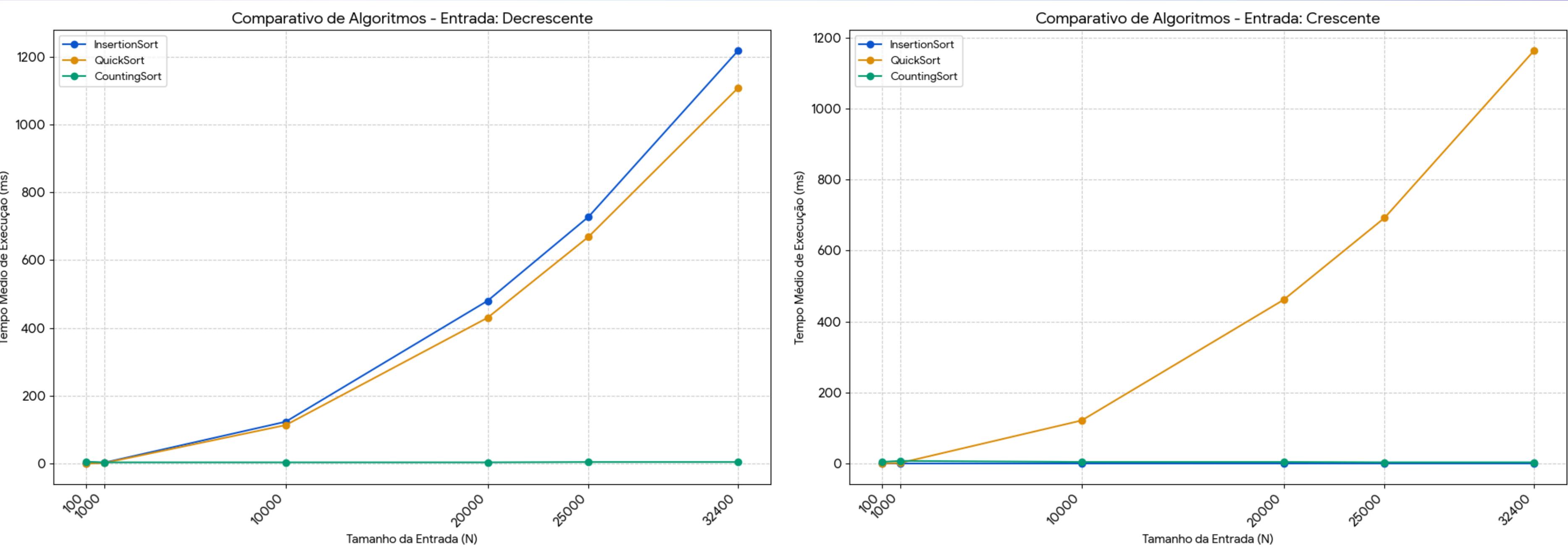
- Implementado em C
- Geração de vetores:
 - Aleatórios, Crescentes, Decrescentes, Quase ordenados (10%)
- Tamanhos testados: de 100 até 32.400 elementos
- Métrica: tempo médio de execução



RESULTADOS GERAIS

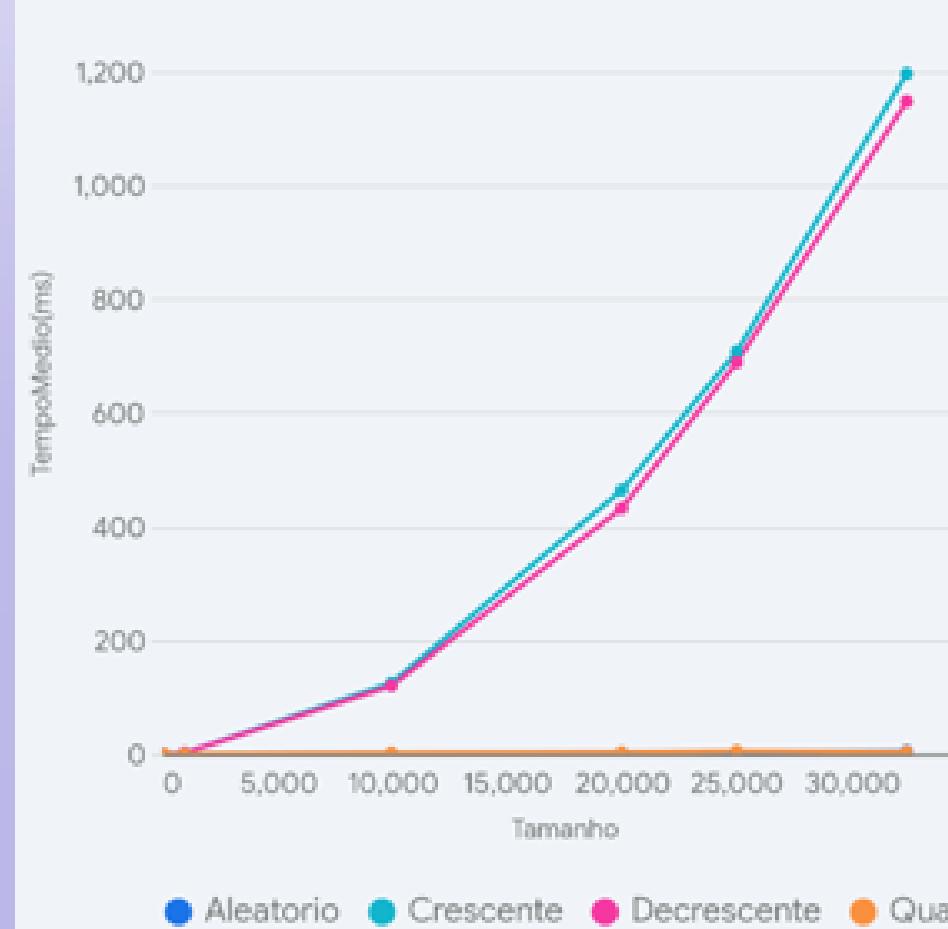


RESULTADOS GERAIS

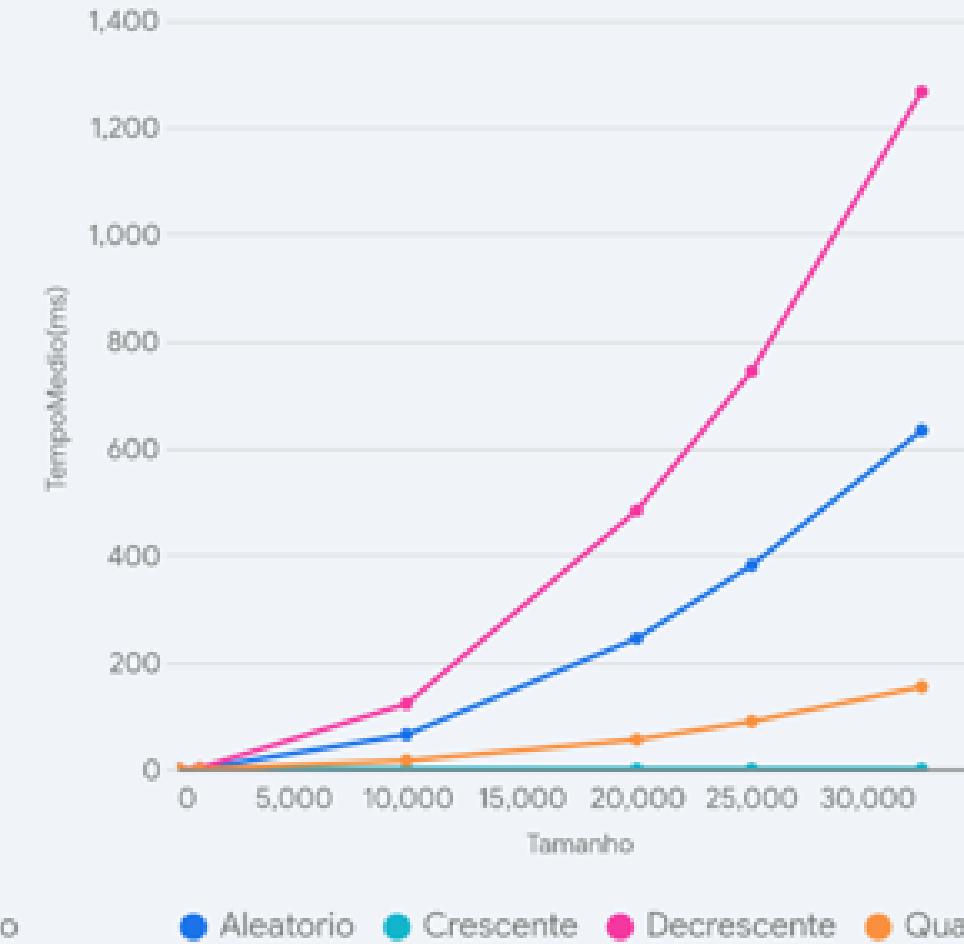


RESULTADOS GERAIS

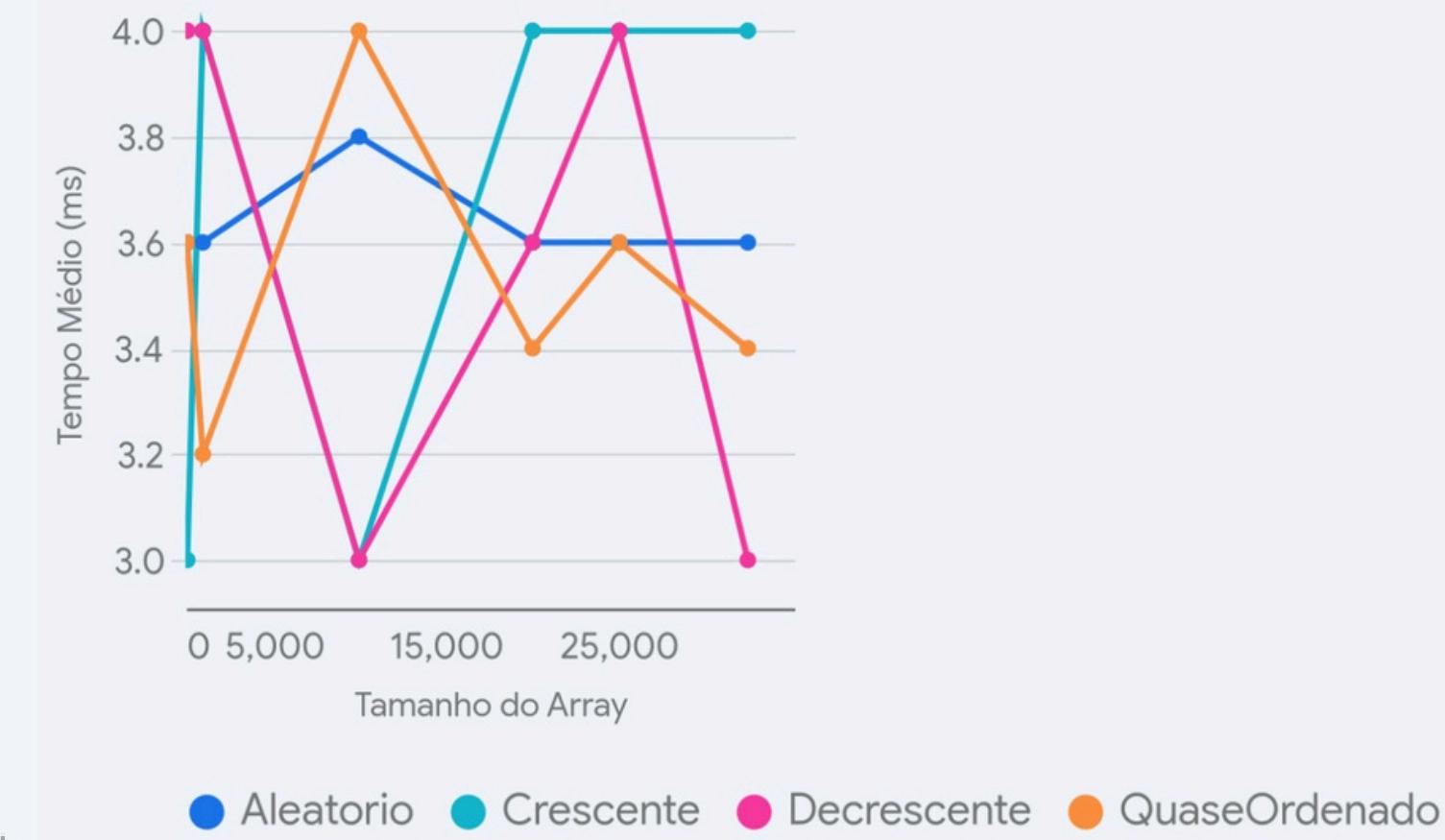
Quick Sort - Tempo Médio por Tamanho e Tipo de Entrada



Insertion Sort - Tempo Médio por Tamanho e Tipo de Entrada



Tempo Médio do Counting Sort por Tamanho e Tipo de Entrada





ÁNALISE RESULTADOS

- Aleatório: Counting Sort e Quick Sort rápidos; Insertion muito lento
- Crescente: Insertion Sort muito eficiente; Quick Sort vai mal (pior caso).
- Decrescente: Insertion e Quick Sort degradam; Counting Sort constante.
- Quase ordenado: Insertion melhora, mas ainda perde para os outros.



CONSIDERAÇÕES FINAIS



- Complexidade teórica foi confirmada na prática.
- Counting Sort se destaca quando as condições ideais são atendidas (valores pequenos, inteiros).
- Quick Sort: bom na média, mas sensível à escolha do pivô.
- Insertion Sort: útil apenas em casos muito específicos



OBRIGADA!

DÚVIDAS?



GitHub: https://github.com/Cheeshiiree/Trabalho1_AEDII_Ordenacao_2025007883