

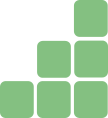
**TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**

**RELATÓRIO PARA A DISCIPLINA DE ALGORITMOS**

**SOBRE ORDENAÇÃO**

# DISCENTE: BERNARDO MEDEIROS

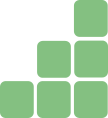
Natal – RN

2025

# RESUMO DO PROJETO

O projeto consiste na implementação de algoritmos de ordenação, que são procedimentos fundamentais na organização de elementos em uma lista ou array em uma ordem específica, normalmente crescente ou decrescente. A ordenação é uma operação básica e amplamente utilizada, sendo essencial em diversas aplicações, como busca binária, sistemas de banco de dados e análise de dados.

No meu projeto, estou aplicando e comparando algoritmos de ordenação em duas linguagens de programação: C e Java.



Natal – RN

2025

.

# Sumário

1. **METODOLOGIA 7**
2. **ALGORITMOS 7** 
   1. **BUBBLE SORT 7** 
      1. **C 7** 
         1. **Implementação 7**
         2. **Gráfico 7**
         3. **Análise 7**
      2. **Java 8** 
         1. **Implementação 8**
         2. **Gráfico 8**

**1.1.2.3. Análise 8**

* 1. **INSERTION SORT 8** 
     1. **C 8** 
        1. **Implementação 8**
        2. **Gráfico 8**
        3. **Análise 8**
     2. **Java 8** 
        1. **Implementação 8**
        2. **Gráfico 8**
        3. **Análise 8**
  2. **SELECTION SORT 8** 
     1. **C 8** 
        1. **Implementação 8**
        2. **Gráfico 8**
        3. **Análise 8**
     2. **Java 8** 
        1. **Implementação 8**

* + - 1. **Gráfico 8**
      2. **Análise 8**
  1. **MERGE SORT 8** 
     1. **C 8** 
        1. **Implementação 8**
        2. **Gráfico 8**
        3. **Análise 8**
     2. **Java 9** 
        1. **Implementação 9**
        2. **Gráfico 9**
        3. **Análise 9**
  2. **QUICK SORT 9** 
     1. **C 9** 
        1. **Implementação 9**
        2. **Gráfico 9**
        3. **Análise 9**
     2. **Java 9** 
        1. **Implementação 9**
        2. **Gráfico 9**
        3. **Análise 9**

1. **CONSIDERAÇÕES FINAIS 10**

**REFERÊNCIAS BIBLIOGRÁFICAS 11**

## 1. METODOLOGIA

A metodologia utilizada consiste na implementação dos algoritmos de ordenação nas linguagens C e Java, contendo variadas quantidades de dados e valores aleatórios desordenados para os testes, a quantidade de dados utilizados variam entre 100, 1000, 10000, 100000 e 1000000. Afim de coletar dados do comportamento do algoritmo com relação ao tamanho de -entrada exigida.

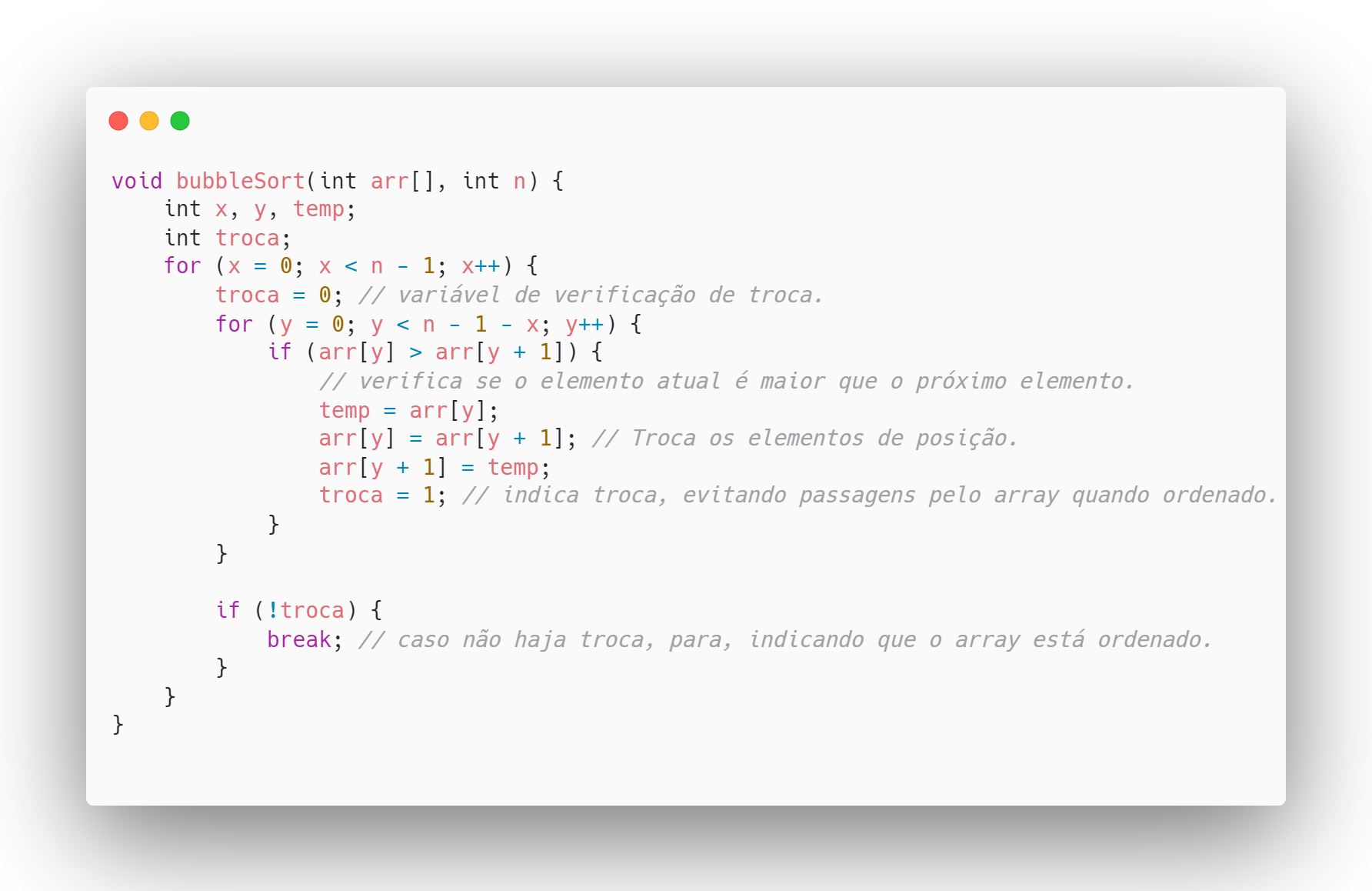
## 2. ALGORITMOS

### 2.1. BUBBLE SORT

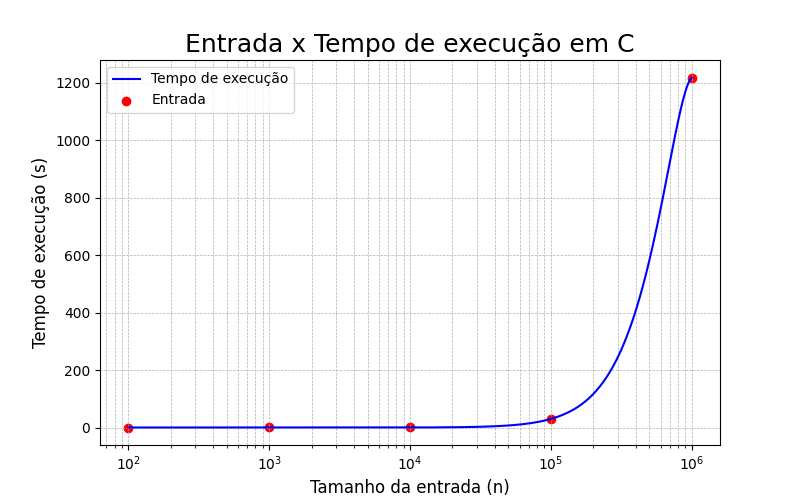
É um dos algoritmos de ordenação mais simples e ineficientes. Funciona comparando repetidamente pares consecutivos de elementos em uma lista e trocando-os caso estiverem na ordem errada. De modo que depois de passar pela lista toda, o maior elemento estará no final da lista (como uma bolha que sobe para o topo). Esse processo continua até que não sejam necessárias mais trocas, quando a lista estará completamente ordenada.

##### 2.1.1. LINGUAGEM - C

##### 2.1.1.1. Implementação

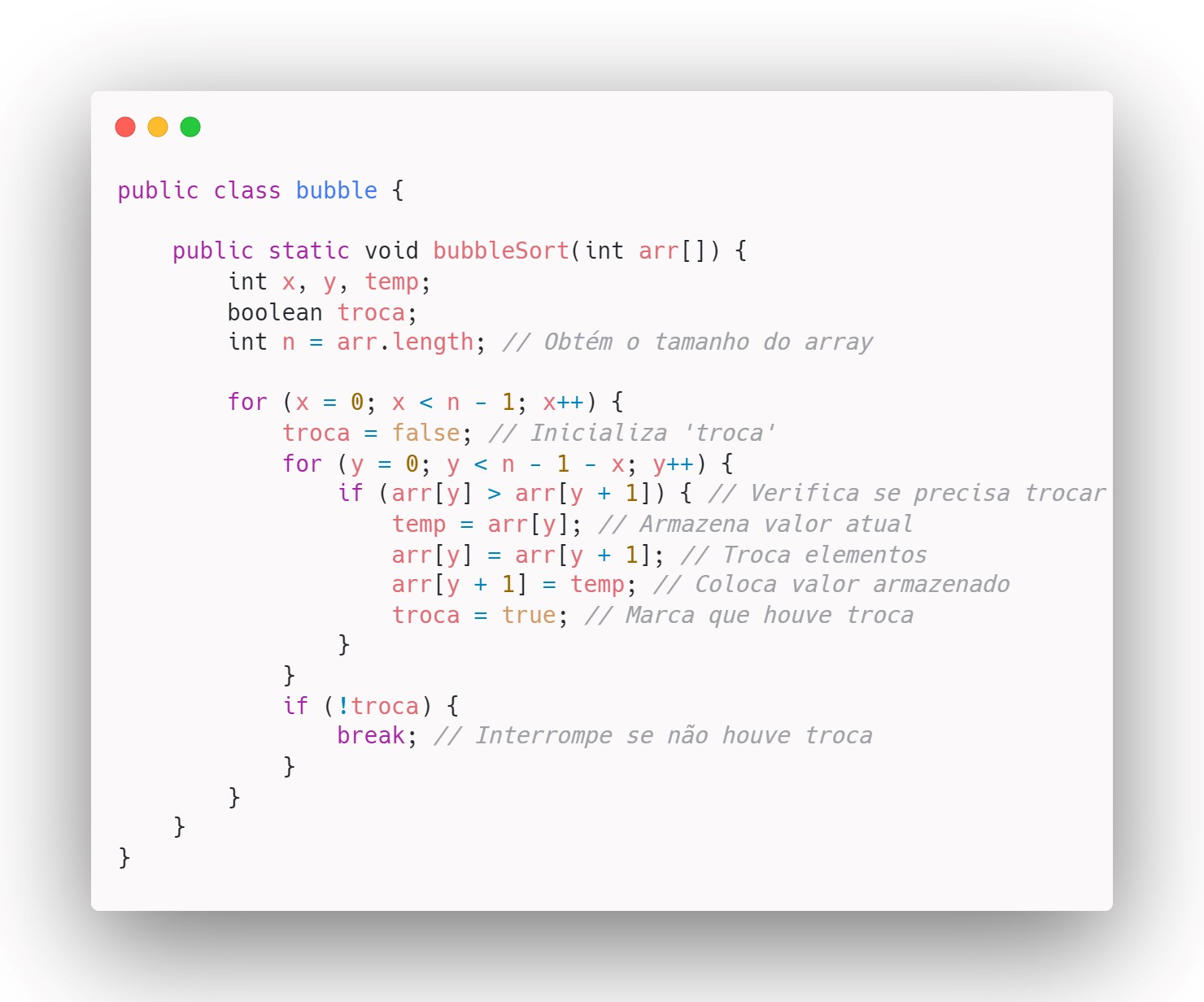


##### 2.1.1.2. Gráfico

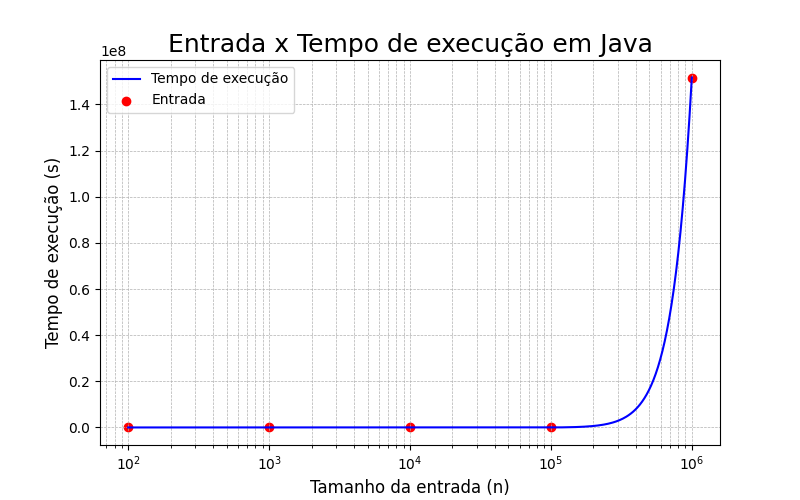
****

#### 2.1.2. LINGUAGEM - JAVA

##### 2.1.2.1. Implementação



**2.1.2.1. Gráfico**



##### 2.1.1.1. Análise Crítica

O algoritmo Bubble Sort, é objetivamente ineficiente e limitado, tendo a complexidade de tempo O(n²), sendo parcialmente adequado apenas para listas pequenas ou já quase ordenadas, mas há opções melhores de sistemas de ordenação. Tem um desempenho aprimorado na linguagem C quando comparado à implementação em Java, porém, o algoritmo em geral tem o desempenho defasado e extremamente lento, mesmo quando a lista já está ordenada ou parcialmente ordenada, o algoritmo ainda pode continuar fazendo comparações desnecessárias e trocas, o que aumenta o tempo de execução, então, mesmo em seu melhor caso de complexidade O(n), para a maioria das situações, ele não é tão eficiente quanto outros algoritmos.

### 2.2. INSERTION SORT

É um algoritmo de ordenação que constrói a lista ordenada, ordenando um elemento de cada vez. Iniciando no segundo elemento, considerando o primeiro já é ordenado por padrão, e em seguida, insere elemento por elemento na posição adequada, comparando com os elementos da parte já ordenada (à esquerda). Repetindo esse processo até que, todos os elementos estejam corretamente posicionados, onde ele não é maior do que o anterior nem menor do que o próximo.

#### 2.1.1. LINGUAGEM - C

##### 2.1.1.1. Implementação



##### 2.1.1.1. Gráfico



##### 2.1.1.1. Análise Crítica

O algoritmo Insertion Sort é eficiente para listas pequenas ou quase ordenadas. Sua complexidade no melhor caso é O(n), o que é bastante eficiente, porém, para listas grandes, é ineficiente devido à sua complexidade de tempo O(n²) no pior caso, sendo um problema quando lidamos com grandes volumes de dados, em listas de dados completamente aleatórios, tende a ser lento, pois requer muitas comparações e movimentações de elementos. Em conclusão, o alto nível de esforço comparativo impacta o desempenho.