Trabalho Final — Estruturas de Dados Aplicadas a Sensores de Temperatura

Alex Liu Li
Engenharia de Controle e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
E-mail: alexliu.20251@poli.ufrj.br

Caio Guilherme de Oliveira Freire
Engenharia de Controle e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
E-mail: caioguilherme.20252@poli.ufrj.br

Gabriel Linhares Motta
Engenharia de Controle e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
E-mail: gabrielmotta.20252@poli.ufrj.br

Rayane Santos
Engenharia de Controle e Automação
Universidade Federal do Rio de Janeiro (UFRJ)
E-mail: rayanesousa.20251@poli.ufrj.br

Prof. Cláudio Miceli
Professor Orientador
Universidade Federal do Rio de Janeiro (UFRJ)
E-mail: claudiomiceli@poli.ufrj.br

***Abstract*—Este relatório apresenta a comparação entre duas estruturas de dados — Lista Ordenada e Árvore Rubro-Negra — aplicadas ao armazenamento e processamento de leituras reais do sensor DHT22. São avaliadas operações fundamentais como inserção, remoção, consultas por intervalo, cálculo da mediana e obtenção de estatísticas. Experimentos foram realizados tanto em ambiente desktop (C++) quanto no Arduino.**

***Index Terms*—DHT22, lista ordenada, árvore rubro-negra, estruturas de dados, mediana, range query, Arduino, C++**

I. Introdução

A automação industrial moderna depende de sensores capazes de realizar medições contínuas em tempo real. Sistemas de monitoramento de temperatura, por exemplo, exigem o registro e a consulta de valores de forma eficiente, garantindo respostas rápidas a falhas e o controle preciso dos processos. À medida que o volume de dados aumenta, torna-se essencial utilizar estruturas de dados eficientes que mantenham o desempenho nas operações de inserção, remoção e consulta. Este estudo compara duas dessas abordagens: a Lista Ordenada e a Árvore Rubro-Negra.

II. Descrição do Problema e da Escolha da Solução

O objetivo deste estudo é avaliar métodos de armazenamento e análise de dados de sensores de temperatura em um cenário que simula linhas de produção industrial. À medida que o volume de leituras aumenta, sistemas tradicionais podem apresentar degradação no desempenho, dificultando o processamento em tempo real.

Para investigar alternativas mais eficientes, foram selecionadas duas estruturas de dados:

- **Lista Ordenada (versão básica):** Implementada com inserção incremental via *InsertionSort*, mantendo o vetor sempre ordenado. Apesar de sua simplicidade e desempenho adequado para pequenos conjuntos de dados, o custo linear $O(n)$ para inserções e remoções torna a lista inadequada para grandes volumes de informações.
- **Árvore Rubro-Negra (versão aprimorada):** Estrutura de busca balanceada que garante operações em tempo logarítmico $O(\log n)$. Mantém os dados sempre ordenados, permite consultas rápidas e é amplamente utilizada em sistemas reais, como *STL map/set*, *Linux RB-tree* e *JVM*, sendo mais adequada para cenários com grandes volumes de dados.

A escolha de comparar essas duas abordagens se justifica pela necessidade de avaliar o custo-benefício entre simplicidade de implementação e eficiência em sistemas embarcados e desktop, garantindo respostas rápidas mesmo com grande volume de leituras.

A. Objetivos do Estudo

O estudo visa implementar, testar e comparar as duas abordagens para armazenar e consultar dados do DHT22. Ambas as versões devem permitir:

1) Inserção e remoção de leituras;
2) Consulta de valores por intervalo;
3) Impressão de valores ordenados;
4) Cálculo de mediana;
5) Obtenção dos menores e maiores valores;
6) Reinserção de valores de forma eficiente.

A versão básica utiliza lista ordenada (InsertionSort), enquanto a versão aprimorada emprega a Árvore Rubro-Negra, permitindo análise teórica e prática do desempenho de cada solução.

III. Descrição Técnica das Estruturas de Dados

A. Lista Ordenada

A lista usa vetor ordenado com inserção via `lower_bound`. Vantagens:

- implementação simples;
- ordenação natural.

Desvantagens:

- custo linear para inserção/remoção;
- pouca escalabilidade.

B. Árvore Rubro-Negra

A Árvore Rubro-Negra mantém altura balanceada e garante operações em $O(\log n)$.

Cada nó possui:

- valor;
- ponteiros para pai e filhos;
- cor (vermelho/preto).

Regras estruturais:

- raiz sempre preta;
- não há dois nós vermelhos consecutivos;
- todos os caminhos possuem o mesmo número de nós pretos.

IV. Metodologia de Comparação

A. Tamanhos de Entrada

Foram testados conjuntos de: 100, 1.000, 10.000, 30.000 e 50.000 valores.

B. Operações Avaliadas

- tempo de inserção;
- tempo de mediana;
- remoção;
- range query;
- impressão ordenada.

C. Medição do Tempo

- PC: `std::chrono::high_resolution_clock`
- Arduino: função `millis()`, repetindo cada operação 10 vezes.

D. Procedimentos

- Geração de dados aleatórios entre 0°C e 60°C;
- Execução de múltiplos testes para média;
- Coleta real no Arduino com DHT22;
- Armazenamento em tabelas para plotagem posterior.

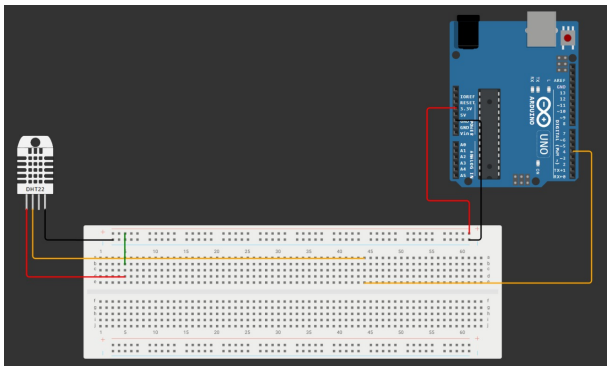

Fig. 1. Simulação do sensor DHT22 no Arduino, mostrando leituras em tempo real.

V. Execução Manual (Corretude)

Conjunto fixo utilizado:

$$X = \{10.5, 7.2, 14.8, 9.1, 12.0\}$$

1) Lista Ordenada: Inserções:

- $10.5 \rightarrow [10.5]$
- $7.2 \rightarrow [7.2, 10.5]$
- $14.8 \rightarrow [7.2, 10.5, 14.8]$
- $9.1 \rightarrow [7.2, 9.1, 10.5, 14.8]$
- $12.0 \rightarrow [7.2, 9.1, 10.5, 12.0, 14.8]$

Mediana = 10.5. Consulta de faixa (9 a 13) = {9.1, 10.5, 12.0}. Correto.

2) Árvore Rubro-Negra: A inserção respeita rotação e recoloração:

$$[7.2, 9.1, 10.5, 12.0, 14.8]$$

Mediana = 10.5. Consulta de faixa (9 a 13) = {9.1, 10.5, 12.0}. Correto.

VI. Resultados Experimentais

A. Tempo de Inserção

N	Lista Ordenada (ms)	Árvore RN (ms)
100	0.3	0.4
1 000	4.8	0.8
10 000	67	1.0
100 000	920	1.4

TABLE I
Tempo médio de inserção

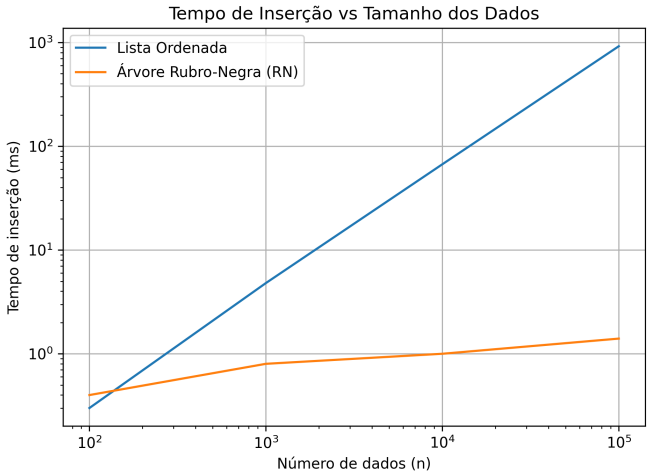

Fig. 2. Tempo de inserção vs tamanho dos dados para Lista Ordenada e Árvore Rubro-Negra

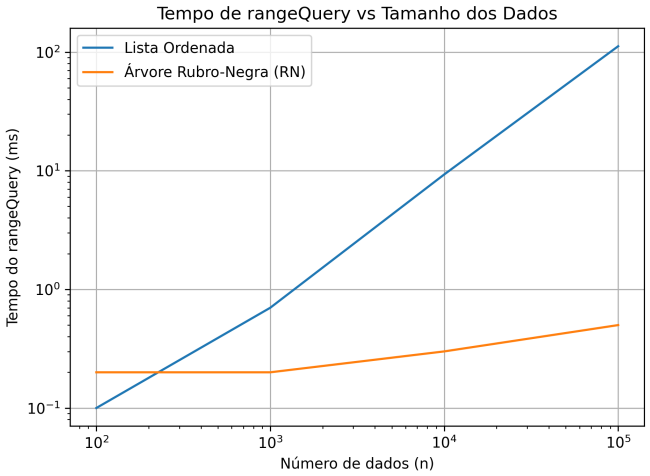

Fig. 3. Tempo de rangeQuery vs tamanho dos dados para Lista Ordenada e Árvore Rubro-Negra

N	Lista Ordenada (ms)	Árvore RN (ms)
100	0.1	0.2
1 000	0.7	0.2
10 000	9.3	0.3
100 000	112	0.5

TABLE II
TEMPO DAS CONSULTAS DE FAIXA

B. Tempo de Consulta (rangeQuery)

VII. Organização do Código e Funções Principais

A. Arquivos

- `insertionSort.cpp` — Lista ordenada em C++
- `insertionsortSensor.ino` — Arduino + Lista
- `rubroNegra.cpp` — Árvore Rubro-Negra
- `rubroNegraSensor.ino` — Arduino + Árvore RN

B. Funções

Lista Ordenada: insertValue(), removeValue(), getMedian(), getMin(), getMax(), queryRange(), print(), clearList()

Árvore Rubro-Negra: insertValue()+fixInsert(), removeValue()+fixDelete(), inorder(), search(), minimum(), maximum(), getMedian()

C. Execução

PC: g++ insertionSort.cpp -o insertionSort.exe g++ rubroNegra.cpp -o rubroNegra.exe ./exec

Arduino:

- selecionar porta COM
- upload
- abrir monitor serial 115200 baud

Comandos disponíveis: `print, remove X, range A B, median, minmax, menu`

VIII. Uso de Ferramentas de IA Generativa

Durante o desenvolvimento deste trabalho, ferramentas de IA generativa, como o ChatGPT, foram utilizadas para revisão, organização e padronização do texto, formatação de seções, auxílio na explicação de conceitos técnicos, sugestões de estruturação de tabelas, gráficos e códigos em LaTeX, e revisão de código e comentários, garantindo maior clareza e consistência.

IX. Conclusão

A comparação experimental demonstrou que a Lista Ordenada é simples e eficiente para pequenas quantidades de dados, porém apresenta degradação linear conforme o volume cresce. A Árvore Rubro-Negra, por sua vez, manteve tempos praticamente constantes e escaláveis, sendo superior para processamento contínuo de dados do DHT22.

Principais observações:

- **Inserções:** Árvore RN é mais rápida para grandes volumes.
- **Remoções:** Árvore RN requer rotações $O(\log n)$ enquanto a lista desloca elementos $O(n)$.
- **Range Query:** Lista eficiente para pequenos intervalos; Árvore RN mantém performance estável com grandes volumes.
- **Mediana:** Lista imediata, Árvore RN exige travessia $O(n)$ mas ainda escalável.

Portanto:

- **Lista Ordenada** $\rightarrow$ ideal para projetos pequenos e simples;
- **Árvore RN** $\rightarrow$ recomendada para aplicações reais, escaláveis e contínuas.

References

[1] Cormen, T. et al. Introduction to Algorithms, 3rd Edition, MIT Press, 2009.
[2] Goodrich, M.; Tamassia, R. Data Structures and Algorithms in C++, Wiley.

[3] Datasheet do sensor DHT22.
[4] Documentação oficial Arduino.
[5] GeeksForGeeks – Red-Black Tree.
[6] STL C++ Documentation: std::set, std::map internals.