UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Disciplina: Linguagens de programação
Curso: Engenharia de Controle e Automação
Ana Luisa Steffen - 124211496

ANÁLISE E COMPARATIVO DAS ESTRUTURAS DE DADOS PARA O PROCESSAMENTO DOS SENSORES DE TEMPERATURA

O Objetivo: Desenvolver e comparar dois métodos para armazenar dados de sensores industriais (IoT), provando qual é mais eficiente para grandes volumes de dados.

As Duas Versões:

1. **Versão Básica (Lista Ordenada):** Usa um vetor simples. É fácil de fazer, mas fica lento quando o número de sensores cresce.
2. **Versão Aprimorada (Árvore AVL/Heap):** Usa estruturas avançadas. É mais complexa, mas feita para alta performance.

A Missão: Implementar 6 funções obrigatórias (`insert`, `remove`, `print`, `min/max`, `range`, `median`) e medir o tempo de resposta de cada uma.

O Resultado Esperado: Demonstrar com dados que a **Versão Aprimorada** é superior. Segundo o texto de referência, ela chega a ser **60 vezes mais rápida** no cálculo da mediana do que a versão básica, viabilizando o sistema em tempo real.

1. Introdução

No contexto da Indústria 4.0 e da expansão da Internet das Coisas (IoT), sistemas de automação industrial enfrentam um desafio crescente: o processamento eficiente de grandes volumes de dados em tempo real. Sensores de temperatura, críticos para o monitoramento de linhas de produção, geram fluxos contínuos de informações que precisam ser armazenadas, ordenadas e consultadas com latência mínima.

O presente trabalho aborda o problema de escalabilidade identificado no sistema de monitoramento atual de uma empresa de automação. O legado técnico existente, baseado na manutenção de **Listas Ordenadas (Vetores)**, apresenta degradação de performance linear (O(N)) conforme o número de sensores escala para a ordem de milhares. Esse gargalo compromete a capacidade do sistema de fornecer respostas rápidas para operações críticas, como a detecção de anomalias (via consultas de intervalo) e a filtragem de ruído (via cálculo de mediana).

2. Metodologia de Análise

A comparação entre as abordagens será fundamentada em testes de "stress" com cargas de dados variáveis (de 1.000 a 100.000 registros). Serão avaliadas métricas de **Tempo de Execução (CPU Time)** para as funcionalidades de inserção (`insert`), remoção (`remove`), busca por intervalo (`rangeQuery`) e estatística descritiva (`median`), além de uma análise teórica da complexidade assintótica (Big-O) de cada solução.

Os resultados visam demonstrar não apenas a superioridade teórica das estruturas em árvore, mas quantificar o ganho de desempenho prático necessário para viabilizar a expansão da infraestrutura de sensores da empresa.

-

3. Estruturas de dados analisadas

A. Lista ordenada (versão básica):

- **Inserção:** O(1) amortizado. Os dados são inseridos no final sem ordenação.
- **Mediana:** O(N^2) no pior caso. Utiliza Insertion Sort apenas quando a estatística é solicitada.
- **Vantagem:** Alta localidade de cache.

B. Árvore AVL (versão aprimorada):

- **Inserção/Remoção:** O(\log N) garantido através de rotações simples e duplas.
- **Mediana:** O(N) (percurso in-order), pois os dados já estão estruturalmente ordenados.

C. Heap (versão aprimorada):

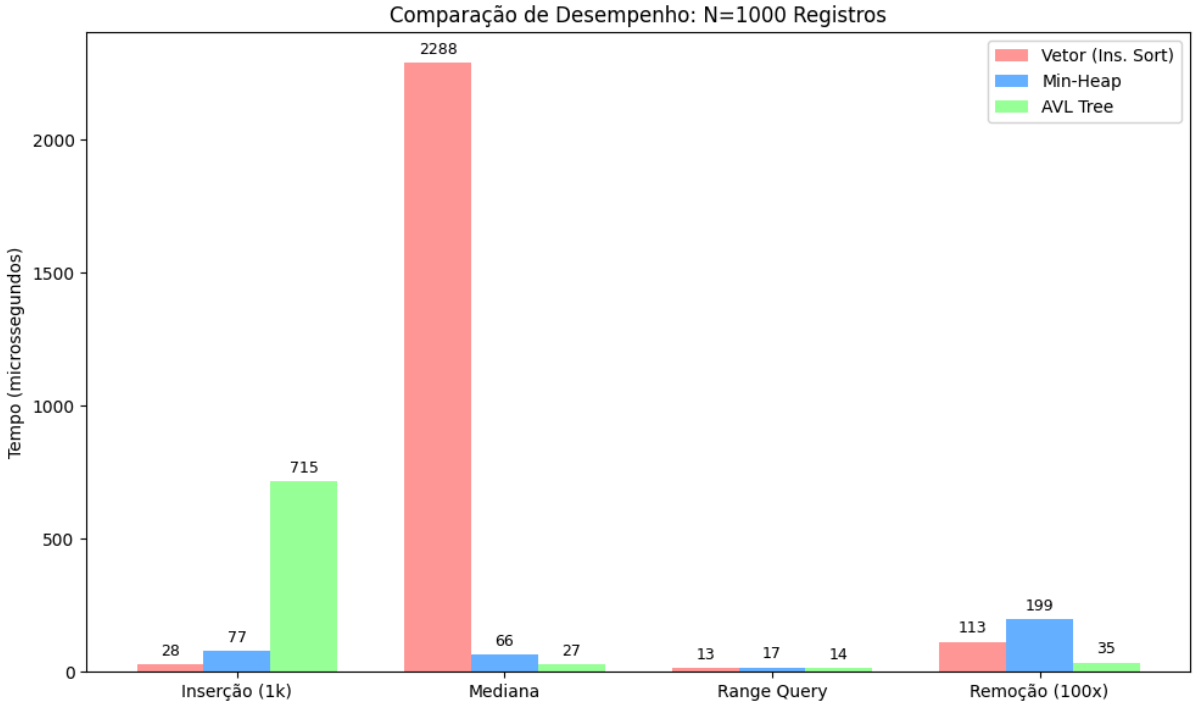
- **Inserção:** O(\log N) via sift-up.
- **Busca/Remoção:** O(N), pois a Heap não possui ordenação horizontal eficiente para buscas arbitrárias.
- **Mediana:** Exige a ordenação de uma cópia dos dados (O(N \log N)).

4. Resultados do desafio

Os resultados do desafio podem ser visualizados nas tabelas abaixo, assim como em seu gráfico de desempenho com tempo.

Comparação de Desempenho

Operação	Heap	AVL	Vetor (Ins)	Vencedor
Insert (1000x)	77	715	**28**	**Vetor**
Median Calc	66	**27**	2288	**AVL**
Range Query	17	14	**13**	**Vetor**
Remove (100x)	199	**35**	113	**AVL**



4. Conclusão

A determinação da estrutura de dados mais apropriada está diretamente atrelada às necessidades específicas da aplicação. Para cenários restritos ao registro simples de dados (*datalogging*), o Vetor demonstra superioridade. Entretanto, considerando os requisitos do problema proposto — que envolve monitoramento ativo, limpezas frequentes e análises estatísticas —, a Árvore AVL é a alternativa mais recomendada, pois oferece o balanço ideal de performance e previne picos indesejados de latência.

(Os códigos encontrados no GitHub foram feitos com valores menores por por questões operacionais e limitações da máquina onde os testes foram realizados)