INSTITUTO FEDERAL DE SÃO PAULO
CAMPUS GUARULHOS
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DIEGO ROCHA VITALI
ERICK OLIVEIRA DANTAS
CHRISTOPHER WILLIANS SILVA COUTO
GABRIEL VITOR GROSSI LOURENÇO

ANÁLISE DE DESEMPENHO COMPARATIVO: ÁRVORE AVL vs. ÁRVORE RUBRO-NEGRA

GUARULHOS
2025

SUMÁRIO

1. INTRODUÇÃO

No universo da ciência da computação, a eficiência na manipulação de grandes volumes de dados é um desafio constante. Algoritmos e estruturas de dados são as ferramentas essenciais que nos permitem organizar e acessar informações de maneira rápida e otimizada. Quando se trata de operações de busca, inserção e remoção, as árvores de busca binária auto balanceáveis, como as árvores AVL e Rubro-Negra, destacam-se como soluções robustas e eficientes.

Este projeto tem como objetivo realizar uma análise experimental e comparativa do desempenho entre essas duas estruturas de dados clássicas. Através da medição do tempo necessário para popular cada árvore com uma massa de dados significativa, investigaremos qual delas oferece a melhor performance em diferentes cenários de inserção, fornecendo uma base prática para a escolha entre uma e outra em aplicações reais.

2. CONCEITOS FUNDAMENTAIS

2.1 ÁRVORE DE BUSCA BINÁRIA

É uma estrutura de dados em que cada "nó" (que guarda uma informação) tem no máximo dois filhos. Os valores menores que o nó ficam à sua esquerda, e os maiores a sua direita. Isso torna as buscas muito rápidas. Seu grande problema é que dependendo da ordem em que os dados são inseridos, a árvore pode ficar "torta" ou desbalanceada, parecendo uma lista e perdendo muita eficiência e propósito.

2.2 ÁRVORE AVL

Foi uma das primeiras soluções para o problema do desbalanceamento. Seu método é bastante rigoroso: em cada nó, ela calcula o "fator de balanceamento", que é a diferença entre a altura da subárvore da esquerda e a da direita. Se essa diferença for maior que 1 a árvore realiza operações chamadas "rotações" para se reorganizar e manter o equilíbrio. Esse controle rígido garante que a árvore nunca fique desbalanceada, fazendo as buscas extremamente rápidas.

2.3 ÁRVORE RUBRO-NEGRA

É outra abordagem inteligente para o mesmo problema. Em vez de controlar a altura, ela utiliza um sistema de cores. Cada nó é pintado de "vermelho" ou "preto" e deve

seguir um conjunto de regras, como "um nó vermelho não pode ter um filho vermelho" e "todo caminho da raiz até uma folha deve ter o mesmo número de nós pretos". Embora pareça simples, essas regras garantem que o caminho mais longo da raiz até uma folha nunca seja mais que o dobro do caminho mais curto. Ela é menos rígida que a AVL, exigindo menos rotações em inserções e remoções, o que a torna uma excelente opção de uso geral.

3. DESENVOLVIMENTO DO CÓDIGO

3.1 ADAPTAÇÃO DO CÓDIGO FONTE

Os códigos-fonte base das árvores AVL e Rubro-Negra, que originalmente manipulavam tipos de dados simples como números inteiros precisaram ser adaptados. Para que pudessem armazenar os dados completos dos funcionários foi necessário alterar a estrutura de seus nós.

O campo de informação de cada nó, que antes continha um int, foi substituído pela struct Funcionario. Com isso, todas as funções de comparação interna das árvores foram ajustadas para utilizar o campo id da struct como chave de ordenação. Essa modificação permitiu que cada nó da árvore armazenasse o registro completo de um funcionário, enquanto o balanceamento e a organização da estrutura continuam sendo guiados pelo código de identificação.

3.2 FLUXO GERAL DO PROGRAMA

O fluxo de execução do programa foi projetado para automatizar os testes e apresentar os resultados de forma clara. A lógica principal pode ser descrita nos seguintes passos:

Preparação do Ambiente: Ao iniciar, o programa executa uma rotina de preparação. Ele verifica se o arquivo com os dados já ordenados (funcionarios_ordenados.csv) existe. Caso não exista, a função alimenta_arvore é chamada com um parâmetro especial (CRIANDO_ARVORE). Essa chamada inicial lê os 15.000 registros do arquivo massaDados.csv, armazena-os em um vetor auxiliar, ordena este vetor pelo id do funcionário utilizando o algoritmo QuickSort e, por fim, grava os dados ordenados no novo arquivo funcionarios_ordenados.csv. Esta etapa ocorre apenas uma vez.

Menu Interativo: Após garantir que ambos os arquivos de dados (ordenado e desordenado) estão disponíveis, o programa exibe um menu principal ao usuário. Utilizando uma interface gráfica no console, o usuário pode escolher qual teste de desempenho realizar:

Inserir na Arvore AVL (Dados Desordenados): Inicia o teste de inserção na AVL usando o arquivo massaDados.csv.

Inserir na Arvore AVL (Dados Ordenados): Inicia o teste na AVL usando o arquivo funcionarios_ordenados.csv.

Inserir na Arvore Rubro-Negra(Dados Desordenados): Inicia o teste de inserção na Rubro-Negra usando o arquivo massaDados.csv.

Inserir na Arvore Rubro-Negra(Dados Ordenados): Inicia o teste na Rubro-Negra usando o arquivo funcionarios_ordenados.csv.

Exibir Tempos: Apresenta os resultados do último teste executado.

Sair: Encerra a aplicação.

Execução dos Testes: Ao selecionar uma das opções de inserção o programa chama a função alimenta_arvore passando qual a árvore e o nome do arquivo correspondente.

Coleta e Armazenamento dos Tempos: A função alimenta_arvore, que centraliza a lógica de teste utiliza a biblioteca <sys/time.h> para medir o tempo de execução. O cronômetro é acionado no início da leitura do arquivo e parado logo após a inserção do último elemento na árvore. Os tempos obtidos para cada árvore e tipo de dado são armazenados em vetores globais para serem posteriormente exibidos pela função exibeTempos.

4. METODOLOGIA DO EXPERIMENTO

4.1 MASSA DE DADOS

O ponto de partida é um arquivo no formato .CSV contendo 15.000 registros de funcionários. Cada linha do arquivo representa um funcionário, com campos como código, nome, idade, empresa, departamento e salário separados por ponto e vírgula. Para armazenar esses dados de forma organizada na memória, foi criada uma struct Funcionario.

4.2 GERAÇÃO DE UM CENÁRIO ORDENADO

Uma das grandes questões sobre árvores auto balanceáveis é como elas se comportam quando os dados já chegam ordenados, o pior caso para uma árvore de busca binária comum. Para testar isso o programa primeiro lê todos os 15.000 registros do arquivo original para um vetor auxiliar na memória. Em seguida, utilizando o algoritmo

de ordenação QuickSort, esse vetor é ordenado pelo código do funcionário. Finalmente, os dados agora ordenados são gravados em um novo arquivo .CSV, chamado funcionarios_ordenados.csv. Esse processo é feito uma única vez no início da execução do programa.

4.3 TESTES DE DESEMPENHO

Com os dois arquivos em mãos (o original desordenado e o gerado ordenado), realizamos os dois testes principais para cada tipo de árvore (AVL e Rubro-Negra):

Teste 1 (Dados Desordenados): Medição do tempo para inserir todos os 15.000 registros do arquivo massaDados.csv em cada uma das árvores.

Teste 2 (Dados Ordenados): Medição do tempo para inserir todos os 15.000 registros do arquivo funcionarios_ordenados.csv em cada uma das árvores.

4.4 MEDIÇÃO PRECISA DO TEMPO

Para garantir que a comparação seja focada apenas no desempenho das árvores a medição do tempo é iniciada exatamente no momento em que o programa começa a ler um arquivo de dados e termina assim que o último registro é inserido na árvore. O tempo gasto para criar o arquivo ordenado não entra nessa medição, porque é uma etapa de preparação. Para garantir a confiabilidade dos resultados cada um dos testes foi executado 10 vezes e a média dos tempos foi calculada para a análise final.

5. ANÁLISE DOS RESULTADOS

Após a execução dos testes, foram coletados os tempos de inserção para cada árvore em ambos os cenários (dados ordenados e desordenados). Para uma análise estatística mais robusta, calculamos a média e o desvio padrão dos 10 tempos registrados para cada teste. Os resultados estão consolidados na Tabela 1.

Tipo de Árvore	Entrada Desordenada (Média)	Entrada Ordenada (Média)
Árvore AVL	0.02578s	0.02308s
Árvore Rubro-Negra	0.02458s	0.02288s

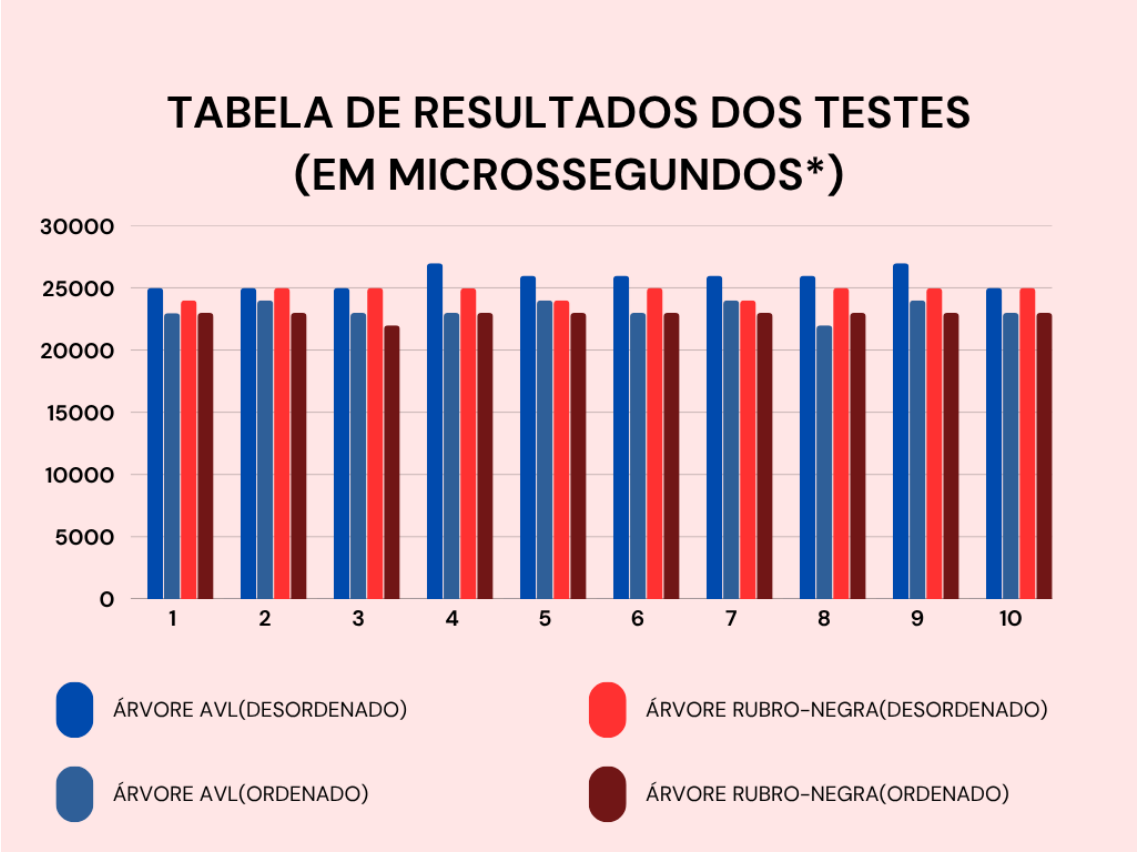
Observando os resultados, podemos fazer algumas análises importantes:

Desempenho Geral: Em ambos os cenários, a Árvore Rubro-Negra apresentou um tempo de inserção ligeiramente inferior ao da Árvore AVL, indicando uma performance

superior para operações de preenchimento massivo de dados. A diferença, embora pequena, na casa dos milissegundos, foi consistente em todas as execuções.

Impacto dos Dados Ordenados: Um dos pontos mais interessantes da análise é que ambas as árvores foram mais rápidas ao inserir dados previamente ordenados. Isso contraria a intuição inicial de que a inserção ordenada seria um "pior caso", como ocorre em árvores de busca binária não balanceadas. O motivo para essa eficiência é que, com dados ordenados, as rotações necessárias para o balanceamento se tornam mais previsíveis e, possivelmente, menos complexas computacionalmente do que as rotações exigidas por inserções em ordem aleatória, que tendem a modificar a estrutura da árvore de forma mais drástica.

AVL vs. Rubro-Negra: A vantagem da Árvore Rubro-Negra, mesmo que sutil, pode ser atribuída à sua natureza menos rígida. A AVL exige que a diferença de altura entre subárvores seja no máximo 1, o que pode forçar mais operações de rotação para manter esse critério estrito a cada inserção. A Árvore Rubro-Negra, por outro lado, permite um "desbalanceamento" um pouco maior, contanto que suas regras de coloração sejam mantidas. Isso resulta em menos reestruturações durante o processo de inserção, explicando sua maior velocidade neste cenário específico.



6. CONCLUSÃO

Com base nos dados experimentais obtidos, concluímos que para a tarefa específica de inserção massiva de 15.000 elementos a Árvore Rubro-Negra demonstrou um desempenho superior ao da Árvore AVL, tanto com dados desordenados quanto com dados previamente ordenados.

Apesar de a Árvore AVL ser teoricamente mais balanceada, o custo computacional para manter seu balanceamento rigoroso a cada inserção se mostrou ligeiramente maior do que o custo das operações de coloração e rotações menos frequentes da Árvore Rubro-Negra. Este resultado confirma a reputação da Rubro-Negra como uma estrutura de dados de uso mais geral e eficiente para aplicações que envolvem um grande número de operações de escrita (inserções e remoções).

Portanto, para o problema proposto a escolha da forma de preenchimento mais eficiente seria com os dados previamente ordenados utilizando a estrutura da Árvore Rubro-Negra.

7. REFERÊNCIAS

IME-USP. Árvores balanceadas. Disponível em: https://www.ime.usp.br/~song/mac5710/slides/08rb.pdf. Acesso em: 26 jun. 2025.

PROGRAMAÇÃO DESCOMPLICADA | Linguagem C. Estrutura de Dados em C | Aula 105 - Árvore Rubro Negra - Definição. YouTube, 6 out. 2015. Disponível em: https://www.youtube.com/watch?v=DaWNuijRRFY. Acesso em: 26 jun. 2025.

OLIVEIRA, Ulysses. Estruturas de Dados: uma abordagem didática. 2. ed. Disponível em: https://www.ulysseso.com/livros/ed2/ApF.pdf. Acesso em: 26 jun. 2025.

UFABC_HAL. 16.6 Estruturas de Dados Puramente Funcionais - Árvores Rubro-Negras: Operações Básicas. YouTube, 30 jul. 2021. Disponível em: https://www.youtube.com/watch?v=W0WFVELrUm0. Acesso em: 26 jun. 2025.

CARVALHO, A. L. Árvores AVL. Disponível em: https://dcm.ffclrp.usp.br/~augusto/teaching/aedi/AED-I-Arvores-AVL.pdf. Acesso em: 26 jun. 2025.

SONG, Siang Wun. **Árvore Rubro-Negra**. [S. l.]: IME-USP, [s.d.]. Disponível em: https://www.ime.usp.br/~song/mac5710/slides/08rb.pdf. Acesso em: 23 jun. 2025.