Universidade do Estado do Amazonas
Escola Superior de Tecnologia
Curso de Sistemas de Informação
Projeto e Análise de Algoritmos

Trabalho Prático 2
Avaliação empírica de Árvores Binárias de Busca

Discente:
Victor Emanuel Barros de Lima

Docente:
Marcio Palheta Piedade

Manaus - AM
2024

Sumário

1. Introdução

A avaliação empírica de algoritmos de estrutura de dados é essencial para entender seu desempenho em diferentes cenários. Este trabalho se concentra na análise comparativa de duas estruturas de dados balanceados: a árvore AVL e a árvore Rubro-Negra. Ambas as árvores são amplamente utilizadas para garantir operações eficientes de inserção, remoção e busca, mantendo uma estrutura balanceada.

Neste estudo, as árvores AVL e Rubro-Negra serão preenchidas com números inteiros sem repetições, variando de 1 a n, em ordem aleatória. Os valores de n considerados são: 1.000, 10.000, 100.000, 1.000.000 e 10.000.000. A avaliação será realizada com base nos seguintes critérios:

1. **Tempo Médio de Busca**: Avaliação do tempo médio para a busca de um elemento que existe (por exemplo, o valor 89) e de um elemento que não existe (por exemplo, o valor n + 1).
2. **Tempo Médio de Remoção**: Avaliação do tempo médio para a remoção de um elemento que existe e de um elemento que não existe.
3. **Tempo Médio de Inserção**: Avaliação do tempo médio para a inserção de elementos.

Para cada valor de n, serão realizados 50 experimentos distintos, e os valores médios serão apresentados em gráficos e tabelas para uma análise comparativa detalhada. O eixo das abscissas (eixo x) dos gráficos representará os valores de n utilizados nos experimentos.

Através desta análise, esperamos obter insights valiosos sobre o comportamento e a eficiência das árvores AVL e Rubro-Negra em diferentes condições de carga, contribuindo para uma melhor compreensão das vantagens e desvantagens de cada estrutura em cenários práticos.

2. Manual

Como solicitado, o código foi executado e desenvolvido no subsistema Linux do Windows (Ubuntu) por meio do WSL. Foi desenvolvido na linguagem C.

2.1 Estrutura

- O código foi estruturado com quatro arquivos diferentes que compõem o seu funcionamento. Dentre eles estão o arquivo principal (main.c), o arquivo da árvore AVL (avl_tree.h), o arquivo da árvore Rubro-Negra (rb_tree.h) e o arquivo para as funções auxiliares (aux.h).

- Para fins de facilidade e simplicidade, é necessário compilar somente o arquivo principal (main.c). Os demais arquivos foram definidos como arquivo de cabeçalho (.h).

2.2 Execução

Como solicitado nas especificações do trabalho, a execução do código é realizada por linha de comando. A compilação e execução acontecem da seguinte forma:

- A compilação do código e por meio do compilador gcc com o comando:
    - **gcc main.c -o main**

- A execução se dá pelo comando:
    - **./main**

Durante a execução, o código irá exigir do usuário certas informações para executar o mesmo de acordo com os parâmetros do usuário. Os passos e as informações a serem preenchidas pelo usuário são:

- **O tamanho do vetor**, sendo ele entre 1 e 10.000.000.
  Se o tamanho inserido pelo usuário ultrapassar esse valor, o programa irá retornar ao usuário:
    - "O tamanho inserido é muito grande. Insira um valor entre 0 e 10000000."

  Se o tamanho inserido pelo usuário for menor ou igual a 10.000.000, o programa segue para a sua próxima etapa.

- Em seguida, os valores a serem inseridos para a criação da árvore vão ser gerados de forma que não tenha **repetições** e em **ordem aleatória** para que mais posteriormente seja criada a árvore com esses valores.

- **A árvore desejada,** sendo especificado para o usuário escolher com números: **1** para **Árvore AVL** e **2** para **Árvore Rubro-Negra.** Se algum número diferente for inserido o programa irá retornar para o usuário:
    - "Escolha Inválida."

- **O número a ser buscado.** O usuário pode escolher qualquer valor entre 1 e o tamanho informado anteriormente com duas possibilidades:
    - Se o valor estiver na árvore, ele retorna "O valor … foi encontrado na árvore …"
    - Se o valor não estiver na árvore, ele retorna "O número … não foi encontrado na árvore …"

- **O número a ser removido.** O usuário pode escolher qualquer valor entre 1 e o tamanho informado anteriormente com duas possibilidades:
    - Se o valor estiver na árvore, ele retorna “O valor … foi removido da árvore …”
    - Se o valor não estiver na árvore, ele retorna “O número … não foi encontrado na árvore … para remoção”

Após o usuário inserir todos os dados exigidos, o programa finaliza e retorna para o usuário:

- Se a árvore desejada foi criada com sucesso;
- Se o número foi encontrado com sucesso;
- Se o número foi removido com sucesso;
- Tempo de inserção;
- Tempo de busca;
- Tempo de remoção;
- Vetor salvo em arvore.txt.

3. Desempenho

O desempenho das árvores foi determinado por meio das operações de inserção, de busca e de remoção. Para essa análise, o main foi executado 50 vezes para cada tamanho da árvore (1.000, 10.000, 100.000, 1.000.000, 10.000.000). Para uma análise mais profunda, as operações de busca e remoção foram executadas com valores que estão na árvore e com valores que não estão, bem como foi especificado.

3.1 Médias

Os tempos em média obtidos para todos os valores e para os dois cenários (se existem ou não na árvore) seguem abaixo:

- **Valores existem:**

AVL					
	1.000	**10.000**	**100.000**	**1.000.000**	**10.000.000**
Inserção	0,000447 s	0,016022 s	0,143590 s	1,906604 s	34,567922 s
Busca	0,000001 s	0,000001 s	0,000001 s	0,000002 s	0,000004 s
Remoção	0,000001 s	0,000081 s	0,000051 s	0,000061 s	0,000306 s

Rubro-Negra					
	1.000	**10.000**	**100.000**	**1.000.000**	**10.000.000**
Inserção	0,000449 s	0,007598 s	0,070537 s	1,152024 s	23,326334 s
Busca	0,000001 s	0,000001 s	0,000001 s	0,000002 s	0,000003 s
Remoção	0,000001 s	0,000118 s	0,000063 s	0,000063 s	0,000056 s

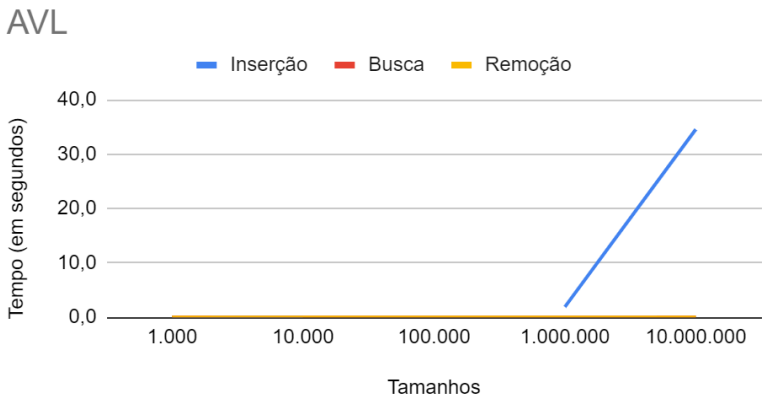
- **Valores não existem:**

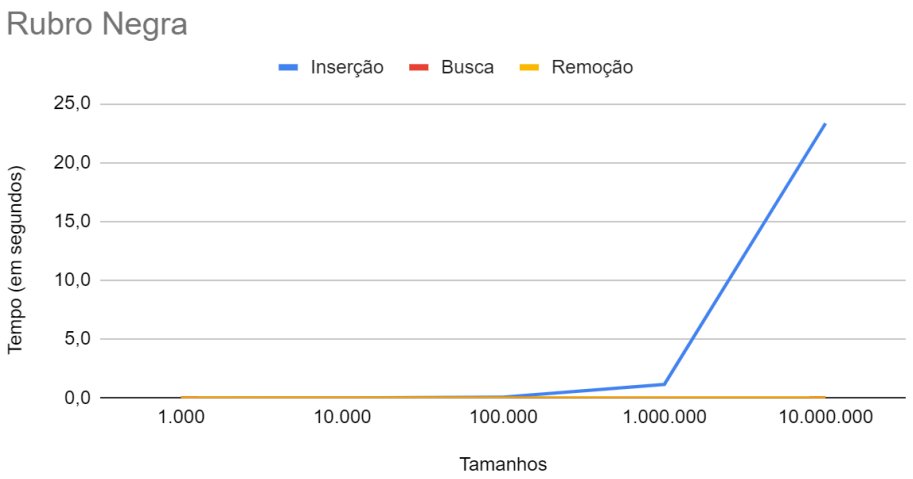
AVL					
	1.000	**10.000**	**100.000**	**1.000.000**	**10.000.000**
Inserção	0,000781 s	0,012713 s	0,084296 s	2,107624 s	33,785596 s
Busca	0,000113 s	0,000077 s	0,000053 s	0,000102 s	0,000292 s
Remoção	0,000011 s	0,000006 s	0,000004 s	0,000006 s	0,000009 s

Rubro-Negra					
	1.000	**10.000**	**100.000**	**1.000.000**	**10.000.000**
Inserção	0,002130 s	0,004576 s	0,068545 s	1,070973 s	22,754154 s
Busca	0,000070 s	0,000082 s	0,000051 s	0,000095 s	0,000305 s
Remoção	0,000007 s	0,000016 s	0,000003 s	0,000005 s	0,000056 s

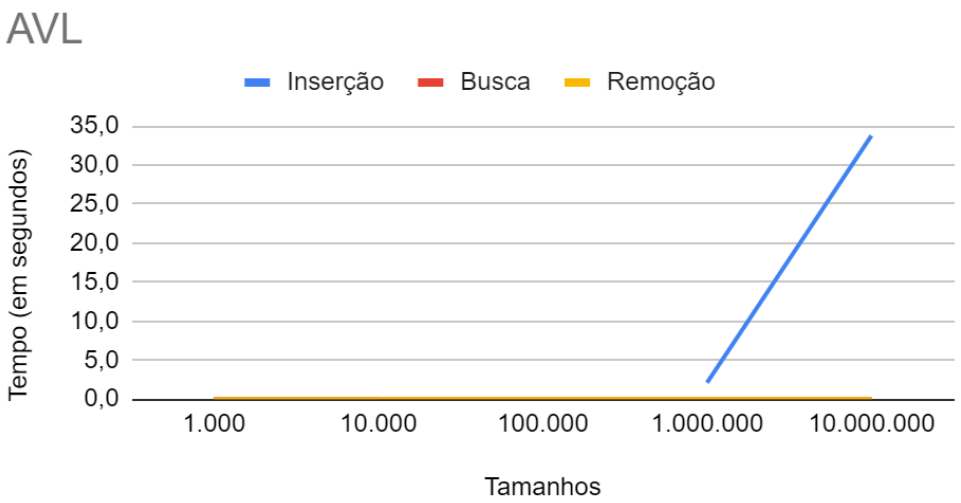
3.2 Gráficos

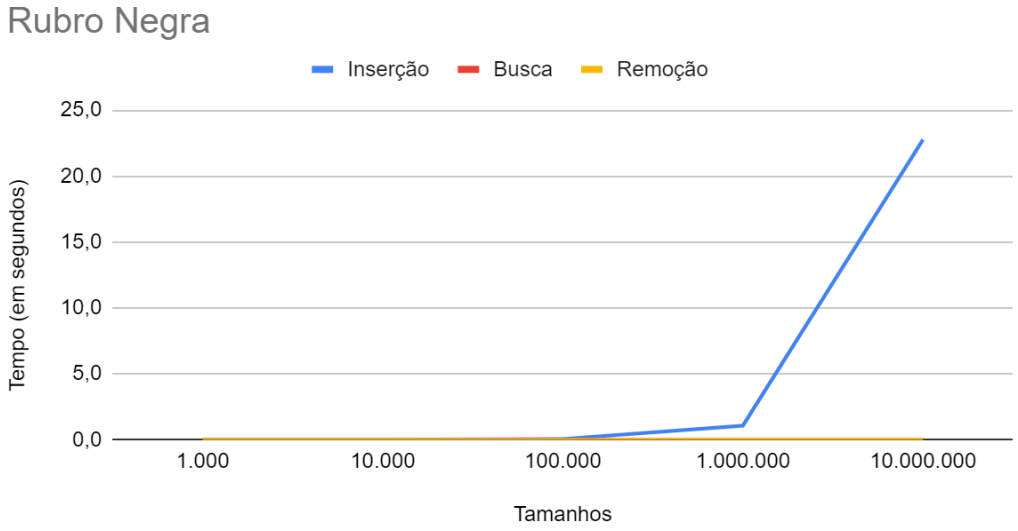
- **Valor existem:**





- **Valores não existem:**





4. Conclusão

Neste trabalho, realizamos uma avaliação empírica das árvores AVL e Rubro-Negra, analisando o desempenho dessas estruturas de dados em diferentes operações: inserção, busca e remoção. Utilizamos números inteiros únicos e aleatórios entre 1 e n, com valores de n variando entre 1.000 e 10.000.000. As árvores foram implementadas em C e executadas no subsistema Linux do Windows (Ubuntu) por meio do WSL, com uma estrutura modular dividida em quatro arquivos distintos.

Resultados Principais

Os experimentos demonstraram que ambas as árvores mantêm tempos de busca e remoção extremamente rápidos, mesmo para grandes valores de n. No entanto, observamos algumas diferenças significativas:

1. **Inserção**:
    - A árvore AVL apresentou tempos de inserção ligeiramente mais altos comparados à árvore Rubro-Negra para grandes valores de n. Isso pode ser atribuído ao maior número de rotações necessárias para manter o balanceamento estrito da AVL.
    - A árvore Rubro-Negra, com seu balanceamento mais relaxado, teve um desempenho de inserção mais eficiente em cenários com grande quantidade de dados.
2. **Busca**:
    - Ambas as árvores demonstraram tempos de busca muito rápidos e consistentes, com diferenças quase imperceptíveis entre os dois tipos de árvore.
    - Em cenários onde o elemento não estava presente, os tempos de busca permaneceram baixos, evidenciando a eficiência de ambas as estruturas na operação de busca.
3. **Remoção**:
    - A remoção de elementos também foi eficiente em ambas as árvores, com tempos médios muito próximos.
    - A árvore Rubro-Negra apresentou ligeira vantagem em alguns casos de remoção, possivelmente devido à menor quantidade de rotações necessárias comparada à AVL.

Considerações Finais

Com base nos resultados obtidos, concluímos que ambas as árvores AVL e Rubro-Negra são altamente eficientes para operações de busca e remoção, mesmo em grandes volumes de dados. A árvore Rubro-Negra mostrou-se ligeiramente mais eficiente em operações de inserção para grandes valores de n, enquanto a árvore AVL manteve-se competitiva e eficiente em todos os cenários testados.

Esta análise empírica fornece uma base sólida para a escolha da estrutura de dados apropriada conforme as necessidades específicas de desempenho em aplicações práticas. A modularidade do código e a simplicidade na execução das operações permitem fácil adaptação e implementação dessas árvores em diversos contextos.