Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Medianeira
Bacharelado em Ciência da Computação

RELATÓRIO DE ÁRVORES E GRAFOS

Benchmark de árvores binárias

ABB, AVL e Rubro Negra

Gabrieli Machado Bianchin
Amabilly Barbosa Russo

Medianeira
Abril/2024

SUMÁRIO

1. INTRODUÇÃO

As árvores são estruturas não lineares que podem ser implementadas em qualquer linguagem de programação e são amplamente utilizadas no mundo da computação para, principalmente, armazenamento de dados. São organizadas de forma hierárquica, tendo como base um nó-raiz que dá origem às folhas, também chamadas simplesmente de nós. Por tamanha usabilidade existem diversos tipos de árvores e, por isso, este trabalho busca realizar a comparação (Benchmark) entre três tipos de árvores: Binária de Busca (ABB), Binária de Busca Balanceada (AVL) e Rubro-Negras (RB) para avaliar seu desempenho em diferentes situações.

2. MATERIAL E MÉTODOS

2.1. MATERIAL

O material utilizado para a realização dessa atividade foi o CodeBlocks, um ambiente de desenvolvimento integrado para a linguagem C.

2.2. METODOLOGIA

Para a programação das funções das diferentes árvores foram usados diversos sites, listados nas referências, além dos materiais apresentados em aula pelo professor Evando Carlos Pessini.

Para a parte de arquivos foi usado o material feito no semestre anterior na matéria de Fundamentos de Estruturas de Dados, também ministrada pelo professor Evando Carlos Pessini.

Para marcar o tempo de execução das árvores foi utilizada a biblioteca <time.h> com a função clock. As comparações entre as três árvores foram registradas neste documento em forma de tabelas, para mais fácil entendimento.

Com o objetivo de que a comparação entre o desempenho de cada tipo de árvore seja justo, foram usados conjuntos de números randômicos iguais nas árvores dispostas. Para tal, gravamos essas sequências aleatórias em arquivos de texto (CasoUm, CasoDois,

CasoTres, CasoQuatro). Feito isso foi testado o tempo de execução de cada árvore separadamente.

3. ÁRVORES ABB

Uma árvore binária de busca (ABB) é uma estrutura de dados binária onde cada nó tem no máximo dois filhos, um à esquerda e um à direita. Para qualquer nó da árvore, todos os valores na subárvore à esquerda são menores que o valor do nó-raiz, isto é, o nó que origina toda a árvore. Consequentemente, todos os valores da subárvore à direita são maiores. Isso garante que a árvore esteja organizada de maneira que as operações de buscas, inserções e remoções possam ser realizadas de maneira eficiente dentro do código. No entanto, uma árvore binária de busca pode se tornar desbalanceada com o tempo, levando a uma baixa de eficiência nas operações.

3.1 INSERÇÃO

3.1.1. Tabela Árvore ABB

Dados	Tempo
1k	0.00000 µs
10k	0.00800 µs
100k	0.05000 µs
1m	2.08400 µs

3.2. EXCLUSÃO

3.2.1. Tabela Árvore ABB

Dados	Tempo
1k	0.00100 µs
10k	0.00000 µs
100k	0.05200 µs
1m	1.55200 µs

3.3. BUSCA

3.3.1. Tabela Árvore ABB

Dados	Tempo
1k	0.00000 µs
10k	0.00200 µs
100k	0.04800 µs
1m	0.53400 µs

4. ÁRVORES AVL

Uma árvore AVL é um tipo de variação da ABB. A AVL mantém os nós da árvore balanceados automaticamente ao verificar a diferença de altura entre as sub árvores à esquerda e à direita para garantir que as operações permaneçam eficientes ao decorrer do tempo. Qualquer nó com altura x<-1 ou x>1 é considerado desbalanceado e portanto, são realizadas uma ou mais rotações a fim de corrigir tal problema.

Existem quatro tipos principais de rotações: rotação simples à esquerda, rotação simples à direita, rotação dupla à esquerda e rotação dupla à direita.

4.1 INSERÇÃO

4.1.1. Tabela Árvore AVL

Dados	Tempo
1k	12.333 µs
10k	30.560 µs
100k	41.010 µs
1m	57.701 µs

4.2. EXCLUSÃO

4.2.1. Tabela Árvore AVL

Dados	Tempo
1k	14.510 µs
10k	30.907 µs
100k	43.008 µs
1m	61.552 µs

4.3. BUSCA

4.3.1. Tabela Árvore AVL

Dados	Tempo
1k	1.930 µs
10k	2.347 µs
100k	2.680 µs
1m	2.992 µs

5. ÁRVORES RUBRO-NEGRAS (RB)

Uma árvore Rubro-Negra é um tipo de variação da ABB. A Rubro-Negra mantém os nós da árvore automaticamente balanceados por meio das seguintes regras:

1. Cada nó ou é vermelho ou é preto.
2. A raiz da árvore é sempre preta.
3. Todas as folhas nulas (nós sem filhos) são pretas.
4. Se um nó é vermelho então ambos os seus filhos são pretos (não é possível haver dois nós vermelhos consecutivos).

5. Todo caminho simples da raiz para uma folha nula tem o mesmo número de nós pretos.

Ao inserir um novo nó, ele é inicialmente colorido de vermelho. Depois, ao realizar a remoção de um nó é possível causar violações nas regras da árvore, fazendo com que ela precise de ajustes (recoloração(ões) e/ou rotação(ões)) para restaurar o balanceamento e manter suas propriedades. Assim, a Rubro-Negra garante um balanceamento mais simples se comparado à árvore AVL.

5.1. INSERÇÃO

5.1.1. Tabela Árvore Rubro-Negra

Dados	Tempo
1k	11.250 µs
10k	22.450 µs
100k	31.747 µs
1m	43.370 µs

5.2. EXCLUSÃO

5.2.1. Tabela Árvore Rubro-Negra

Dados	Tempo
1k	13.100 µs
10k	24.320 µs
100k	34.485 µs
1m	46.650 µs

5.3. BUSCA

5.3.1. Tabela Árvore Rubro-Negra

Dados	Tempo
1k	4.230 µs
10k	5.870 µs
100k	6.400 µs
1m	6.980 µs

6. RESULTADOS E DISCUSSÃO

Ao comparar as árvores binárias de busca ABB, AVL e Rubro-Negra e seus desempenhos em microssegundos(µs) nas operações de inserção, busca e exclusão de dados, é possível notar que as ABBs são adequadas para casos simples onde o balanceamento não é crítico pelo motivo de tornarem-se mais lentas para realizar operações à medida em que seus nós são modificados; as árvores AVL são ideais para para cenários com poucas modificações, vide aplicações que exigem busca eficiente e operações relativamente estáticas porque o número de rotações para manter o balanceamento destas é relativamente baixo e o custo de manutenção não compromete a eficiência; as árvores Rubro-Negra equilibram bem o custo de manutenção e o desempenho, sendo apropriadas para cenários com frequentes operações de inserção e remoção porque não utilizam-se tanto de rotações complicadas que impactam em todos os seus nós. Portanto, explicita-se que o tempo levado para inserção de dados ABB < RB < AVL, para busca AVL < RB < ABB e para exclusão, ABB < RB < AVL.

Abaixo, todos os dados coletados:

ABB	Inserção	Busca	Exclusão
1k	0.00000 µs	0.00000 µs	0.00000 µs
10k	0.00800 µs	0.00200 µs	0.00000 µs
100k	0.05000 µs	0.04800 µs	0.052 µs

1m	2.08400 µs	0.53400 µs	1.552 µs

AVL	Inserção	Busca	Exclusão
1k	12.333 µs	1.930 µs	14.510 µs
10k	30.560 µs	2.347 µs	30.907 µs
100k	41.010 µs	2.680 µs	43.008 µs
1m	57.701 µs	2.992 µs	61.025 µs

Rubro-Negra	Inserção	Busca	Exclusão
1k	11.250 µs	4.230 µs	13.100 µs
10k	22.450 µs	5.870 µs	24.320 µs
100k	31.747 µs	6.400 µs	34.485 µs
1m	43.370 µs	6.980 µs	46.650 µs

7. CONSIDERAÇÕES FINAIS

Através do estudo realizado sobre a teoria de funcionamento dos três tipos de árvores propostos e observando os exemplos práticos expostos nas tabelas ao decorrer do trabalho, entende-se que a aplicabilidade das árvores binárias é vasta mas pode variar em diferentes contextos computacionais, pois estas dependem das necessidades específicas da aplicação na qual trabalharão.

Ademais, os resultados obtidos neste Benchmark aparentemente estão condizentes em relação à teoria. Estimava-se que todos os valores resultantes estivessem num padrão linear com complexidade O(log n) e essa expectativa foi alcançada apesar de existirem pequenos desacertos no código, como por exemplo a busca da árvore AVL que pode apresentar problemas. Entretanto, como não ocorreu nenhum erro ele não foi modificado. Outro ponto importante é a velocidade da árvore ABB, isso acontece provavelmente porque as árvores não estão balanceadas seguindo um mesmo padrão, reflexo dos conjuntos de números randômicos.

Conclui-se que o Benchmark foi um sucesso, mesmo que a entrega seja tardia.

8. REFERÊNCIAS

1. Deletion in Red-Black Tree. Disponível em: <https://www.geeksforgeeks.org/deletion-in-red-black-tree/?ref=lbp>.
2. Deletion from Red-Black Trees R U O. [s.l: s.n.]. Disponível em: <https://www.cs.purdue.edu/homes/ayg/CS251/slides/chap13c.pdf>. Acesso em: 7 ago. 2024.
3. Red-black tree (C) - LiteratePrograms. Disponível em: <https://web.archive.org/web/20140328232325/http://en.literateprograms.org/Red-black_tree_(C)>. Acesso em: 7 ago. 2024.
4. Materiais do professor Evando Carlos Pessini e projetos de estrutura de dados desenvolvidos ao decorrer dos dois semestres.