Universidade Federal de Itajubá
Instituto de Matemática e Computação
Algoritmos e Estruturas de Dados II
CTCO02

Árvores Binárias Balanceadas – AVL e Rubro-Negra

Nome: Breno Yukihiro Hirakawa
Matrícula: 2021001120

Nome: Tiago Antônio Vilela Carvalho
Matrícula: 2022000969

Link para o Git:
https://github.com/Tiago-htm/ComparacaoAvlRubroNegra.git

Introdução

Árvores binárias são estruturas de dados amplamente utilizadas devido à sua versatilidade e eficiência. Entre os tipos mais comuns de árvores binárias balanceadas, destacam-se a AVL e a Rubro-Negra. Ambas se caracterizam por manterem o balanceamento através de rotações e possuem complexidade assintótica de O(log n) tanto para operações de busca, inserção e remoção.

Neste trabalho será apresentado uma análise comparativa do desempenho das árvores binárias AVL e Rubro-Negra. Foram adaptados códigos para medir o tempo de inserção e remoção em ambas as árvores, bem como para contar o número de rotações realizadas durante essas operações. Além disso, foi calculado o tempo necessário para encontrar 1000 elementos aleatórios em cada árvore.

Para garantir a consistência dos testes, foram utilizados 10000 valores ordenados como dados de entrada. Um programa específico foi desenvolvido para gerar um arquivo .txt contendo esses valores, de modo a não interferir no tempo e na performance das operações realizadas pelas árvores.

Desenvolvimento

Avl

Cada nó na árvore AVL possui um fator de balanceamento (FB), que é a diferença entre a altura da subárvore direita e a altura da subárvore esquerda. Os fatores de balanceamento possíveis para qualquer nó são -1, 0 ou 1. Esses valores indicam:

- **FB = 0:** as subárvores esquerda e direita têm a mesma altura.
- **FB = 1:** a subárvore direita é um nível mais alta que a subárvore esquerda.
- **FB = -1:** a subárvore esquerda é um nível mais alta que a subárvore direita.

Durante a inserção de um novo nó na árvore, o fator de balanceamento dos nós ancestrais pode ser alterado. Se um novo nó for inserido na subárvore esquerda, o fator de balanceamento do pai desse nó é decrementado em 1. Se o novo nó for inserido na subárvore direita, o fator de balanceamento do pai é incrementado em 1.

Quando o fator de balanceamento de um nó se torna 2 ou -2, a árvore está desbalanceada e precisa ser ajustada. Para corrigir o desbalanceamento, são realizadas rotações nos nós afetados. Existem quatro tipos principais de rotações usadas para balancear uma árvore AVL:

1. **Rotação simples à direita:** Quando a subárvore da esquerda de um nó está desbalanceada.
2. **Rotação simples à esquerda:** Quando a subárvore direita de um nó está desbalanceada.
3. **Rotação dupla à direita (esquerda-direita):** Quando um nó está desbalanceado para a esquerda e a subárvore direita do nó filho esquerdo está causando o desbalanceamento.
4. **Rotação dupla à esquerda (direita-esquerda):** Quando um nó está desbalanceado para a direita e a subárvore esquerda do nó filho direito está causando o desbalanceamento.

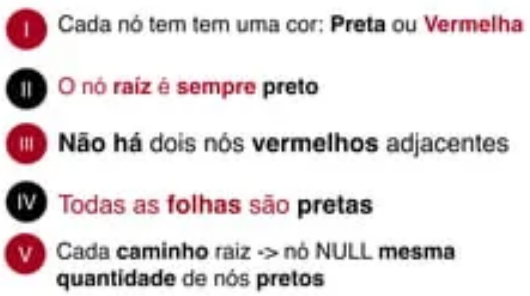
Desta forma, as rotações permitem que a árvore seja balanceada, garantindo que as operações de inserção, remoção e busca continuem funcionando corretamente. Além do mais, com uma árvore balanceada corretamente, até mesmo no pior caso, a árvore possuirá complexidade de tempo O(log n).

Rubro Negra

A rubro negra tem esse nome pois a coloração dos nós da árvore são preto ou vermelho, o algoritmo é bastante recomendado para bancos de dados e sistemas operacionais. Ela garante balanceamento quase uniforme assim como AVL, com esse balanceamento ela garante uma complexidade assintótica de O (log n) para busca, inserção e remoção. Para garantir essa complexidade a rubro Negra mantém a altura da árvore em $2*\log_2(n+1)$, assim até o nó mais alto vai ser em tempo logarítmico.

Estrutura e regras:

As árvores rubro negra possuem nós coloridos de preto ou vermelho, e seguem 5 regras que garantem o balanceamento da árvore:



Na rubro negra tem um conceito chamado de “Altura de Preto”, onde a “altura de preto” de uma árvore Rubro Negra é o número de nós pretos em qualquer caminho de um nó x até uma folha, excluindo o próprio nó x.

Inserção e Remoção:

Na inserção de um novo elemento na árvore na rubro-negra ocorre da mesma forma de uma inserção em uma árvore normal, após inserir o novo nó é colorido de vermelho. Após a coloração é verificado se o balanceamento está correto. Caso o balanceamento esteja incorreto são analisados 3 casos para cada lado do avô (esquerda ou direita), Ambos os lados seguem a mesma verificação mudando somente o lado da rotação.

- **Caso 1**: O “tio” do nó inserido é vermelho.
- **Caso 2**: O “tio” do nó inserido é preto e o nó é um filho direito.
- **Caso 3**: O “tio” do nó inserido é preto e o nó é um filho esquerdo.

Se o nó desbalanceado estiver à esquerda, deve fazer as seguintes considerações para balancear a árvore novamente.

Caso	Cor do Tio	Filho	Ações
1	vermelho	Direita ou Esquerda	Colore o pai de preto Colore o tio de preto Colore o avô de vermelho z = avô – faz nova avaliação
2	preto	direita	z = pai Rotação à esquerda (z) Vira Caso 3
3	preto	esquerda	Colore o pai de preto Colore o avô de vermelho Rotação à direita no avô
1,2,3			Colore a raiz de preto

Caso o nó estiver a direita:

Caso	Cor do Tio	Filho	Ações
1	vermelho	Direita ou Esquerda	Colore o pai de preto Colore o tio de preto Colore o avô de vermelho z = avô – faz nova avaliação
2	preto	esquerda	z = pai Rotação à direita (z) Vira Caso 3
3	preto	direita	Colore o pai de preto Colore o avô de vermelho Rotação à esquerda no avô
1,2,3			Colore a raiz de preto

Na remoção só vai chamar o balanceamento caso o nó removido for preto, pois o vermelho não altera o balanceamento da árvore. Existem três situações para a remoção:

- **Caso 1**: Nó excluído vermelho e sem sucessor.
- **Caso 2**: Nó excluído preto e sucessor vermelho.
- **Caso 3**: Nó excluído preto e sucessor preto.

Para resolver os casos considere a imagem abaixo:

Sit.	z	Sucessor (y)	Irmão (w)	Ação
Substituir direita por esquerda e vice versa quando o **x** estiver à direita do pai			Caso 1 Vermelho	-Colore w de preto - Colore o pai de vermelho - Rotação à **direita** no pai - w = filho da **esquerda** do pai de x - Leva aos casos 2, 3 ou 4
			Caso 2 Preto e ambos os filhos são pretos	- Colore o irmão de vermelho - x = pai - reavalia o x.
3	Preto	Preto	Caso 3 Preto e filho da **direita** é vermelho e o da esquerda é preto	-Colore de preto o filho a direita de w -Colore w de vermelho -Rotação à **esquerda** em w - w = filho da **esquerda** do pai de x - Leva ao caso 4
			Caso 4 Preto e filho da **esquerda** é vermelho	-Colore w com a cor do pai - Colore o pai de preto - Colore filho **esquerdo** de w de preto - Rotação à **direita** no pai de x - x = A->raiz

Resultados

Arquitetura

Para a realização dos testes, foi utilizado um computador equipado com um processador Intel Core i5-9400F de 2.90GHz com 16GB de memória RAM, no sistema operacional Windows 10 x64. Para que os resultados fossem maximizados e sem qualquer interferência, os programas foram executados após a reinicialização da máquina, e os mesmos foram na IDE CLion.

Comparações

Como visto anteriormente, a AVL e a Rubro-Negra possuem algumas diferenças em suas características, como as regras, os modos de balanceamento e as condições de rotações. Porém, a diferença fundamental entre essas duas estruturas está na rigidez do balanceamento:

1. **AVL**: Garante um balanceamento rigoroso, onde a diferença de altura entre subárvores esquerda e direita de qualquer nó é no máximo 1. Isso significa que, após inserções ou remoções, a árvore pode precisar de mais rotações para manter-se balanceada, o que pode custar mais em termos de desempenho.

2. **Rubro-Negra**: É menos rígida quanto ao balanceamento exato das alturas das subárvores. Ela garante um balanceamento aproximado, onde a altura de pretos de

qualquer nó nunca excede duas vezes a altura de pretos de qualquer outro caminho. Isso permite mais flexibilidade durante as inserções e remoções, frequentemente resultando em menos operações de reequilíbrio do que na AVL.

A tabela abaixo exibe os valores encontrados para o número de rotações e inserção e remoção e o tempo resultante destas duas operações para cada árvore.

	AVL	**RUBRO-NEGRA**
ROTAÇÃO-INSERÇÃO	9986	9976
ROTAÇÃO-REMOÇÃO	4988	4989

Portanto, de acordo com a tabela abaixo, os valores de rotações de inserção da AVL e da Rubro-Negra são muito parecidos, tendo uma pequena diferença de apenas 10 rotações. Isso se dá pois a AVL é muito menos “tolerante” com diferenças das alturas do que a rubro negro.

Já nesta tabela, são apresentados os valores do tempo de inserção e remoção, assim como o tempo de busca para 1000 valores aleatórios, todos exibidos em segundos.

	AVL	**RUBRO-NEGRA**
TEMPO-INSERÇÃO	0.0019422 s	0.0022875 s
TEMPO-REMOÇÃO	0.0014459 s	0.0013607 s
TEMPO-BUSCA 1000 VALORES	0.0001275 s	0.0001510 s

Ao analisar a tabela, é nítido que a AVL teve melhor desempenho em velocidade do que a Rubro-Negra. Porém, as árvores AVL são em sua maioria mais lentas que a Rubro-Negra por conta de sua rigidez no quesito balanceamento, ganhando somente no tempo de busca. Tal ocorrido pode ter sido causado por uma diferença de implementação do código, o que causou a discrepância nas velocidades.

Conclusão

Portanto, podemos concluir que na análise teórica, a Rubro-Negra pode superar a AVL em diversos aspectos, perdendo somente no tempo de busca. Porém, na prática, podemos perceber que a AVL teve um desempenho melhor que a Rubro-Negra, além de ambas possuírem praticamente a mesma quantidade de rotações na remoção e na inserção. Muitos destes fatores podem estar relacionados à implementação, onde algum fator permaneceu equivocado