Relatório sobre Árvores Binárias: AVL vs. Árvore Normal

Alunos: Francisco Ochoa Bonato e Victor Hugo Brunetto

Neste relatório, discutiremos três códigos relacionados a Árvores Binárias: um para uma árvore binária normal e outro para uma Árvore AVL (Adelson-Velsky e Landis), bem como um código auxiliar para elementos. Avaliaremos esses códigos e discutiremos qual tipo de árvore funciona melhor em qual situação.

Código da Árvore Binária Normal:

O primeiro código apresenta uma implementação de uma Árvore Binária Normal. Abaixo estão algumas das principais funcionalidades deste código:

- Arvore é a classe principal que representa a árvore binária.
- adicionar(TIPO valor) insere um elemento na árvore.
- remover(TIPO valor) remove um elemento da árvore.
- emOrdem(Elemento<TIPO> atual), preOrdem(Elemento<TIPO> atual) e posOrdem(Elemento<TIPO> atual) são métodos para percorrer a árvore em diferentes ordens (em ordem, pré-ordem e pós-ordem, respectivamente).
- buscar(TIPO valor) busca um elemento na árvore.

Código da Árvore AVL

O segundo código apresenta uma implementação de uma Árvore AVL. Abaixo estão algumas das principais funcionalidades deste código:

- ArvoreAVL é a classe principal que representa a árvore AVL.
- inserir(int chave) insere um elemento na árvore AVL.
- remover(int chave) remove um elemento da árvore AVL.
- buscar(int chave) busca um elemento na árvore AVL.
- O código inclui funções para calcular alturas e fatores de equilíbrio e realizar rotações para manter a árvore balanceada.
- Código Auxiliar - Elemento
- O código auxiliar apresenta a classe Elemento, que é usada em ambos os tipos de árvores para representar os elementos da árvore. Cada elemento possui um valor e referências para os elementos à esquerda e à direita.

Comparação entre Árvore Binária Normal e AVL

A principal diferença entre uma Árvore Binária Normal e uma Árvore AVL está no balanceamento. A Árvore AVL é uma árvore binária balanceada, o que significa que a diferença entre as alturas das subárvores esquerda e direita em cada nó (fator de equilíbrio) é mantida dentro de limites específicos. Isso garante um desempenho mais consistente em termos de tempo de pesquisa, inserção e remoção em comparação com uma Árvore Binária Normal, que pode se tornar desbalanceada e levar a pior desempenho em casos extremos.

Situações de Uso:

Árvore Binária Normal (Sem Balanceamento):

- A Árvore Binária Normal pode ser usada quando não é necessário manter o balanceamento da árvore e a estrutura é atualizada com menos frequência.
- Pode ser adequada quando os dados inseridos seguem uma distribuição natural e não são inseridos ou removidos com muita frequência.

Árvore AVL (Com Balanceamento):

- A Árvore AVL é ideal quando é crucial manter um bom desempenho, mesmo com inserções e remoções frequentes.
- É especialmente útil em situações onde os dados são inseridos em ordem não natural, tendendo a desbalancear uma Árvore Binária Normal.

Porquê:

- A Árvore AVL garante que a altura da árvore seja logarítmica em relação ao número de elementos, o que resulta em operações de busca, inserção e remoção com complexidade O(log n), onde n é o número de elementos na árvore. Isso é vantajoso para grandes conjuntos de dados e operações frequentes.
- A Árvore Binária Normal pode degenerar em uma estrutura de lista vinculada em casos extremos, levando a operações com complexidade O(n), onde n é o número de elementos. Isso ocorre quando os dados são inseridos em ordem crescente ou decrescente, por exemplo.

Em resumo, a escolha entre Árvore Binária Normal e AVL depende da situação. Se a manutenção do balanceamento é importante e as operações frequentes, a Árvore AVL é uma escolha sólida. No entanto, se o balanceamento não é crítico e as atualizações são raras, a Árvore Binária Normal pode ser mais simples de implementar.