**Definição ABB**

Uma árvore binária de busca é uma estrutura de dados em forma de árvore onde cada nó tem, no máximo, dois filhos. Em uma árvore binária de busca, o valor de cada nó é maior que o valor de seu filho à esquerda e menor que o valor de seu filho à direita, tornando fácil a realização de buscas. Esta estrutura é amplamente utilizada em aplicações que precisam de eficiência em tempo de busca, inserção e remoção de dados.

A altura da árvore em caso de estar equilibrada (no caso de estar completa) por h=log2 (n+1)

**Complexidade**

A complexidade de uma árvore binária de busca depende do equilíbrio da árvore. Em uma árvore binária de busca equilibrada, a complexidade para operações básicas é geralmente:

* Busca: O(log n), onde n é o número de nós na árvore.
* Inserção: O(log n), onde n é o número de nós na árvore.
* Remoção: O(log n), onde n é o número de nós na árvore.

No pior caso, em uma árvore binária desequilibrada, a complexidade para as operações básicas seria:

* Busca, Inserção e Remoção: O(n), onde n é o número de nós na árvore.

Por isso, é importante garantir que a árvore esteja equilibrada para garantir sua eficiência em tempo de execução.

ABB

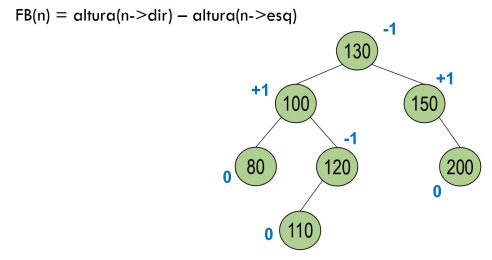
Problemas que a árvore binária de busca pode conter

Desbalanceamento progressivo

Buscas ficam mais custosas

Fator de balanceamento (FB)

Diferença entre a altura da subárvore direita e esquerda



O filho da esquerda sempre é menor do que o da direita, pra qualquer nó na árvore binária de busca.

**Pior caso**

O(n), quando a arvore degenera parecido com uma lista com a altura (n), o tamanho da arvore.

**Utilidade**

Em cenários que necessitam de decisões bidirecionais em cada ponto do processo, é uma boa aplicar a ABB.

**Inserção**

**Balanceamento**

Existe dois métodos aceitados como balanceamento de árvores binárias, o dinâmico que é ocorre a cada inserção de um nó, sendo assim mantendo a árvore balanceada sempre. E o balanceamento global, que é realizado apenas quando necessário permitindo o crescimento da árvore sem limites.

**Árvore RN**

A Árvore Rubro-Negra é uma estrutura de dados que se baseia em uma árvore binária de busca, com algumas propriedades adicionais que a tornam balanceada. A Árvore Rubro-Negra é usada para armazenar e recuperar rapidamente dados em ordem, e é amplamente utilizada em muitos sistemas de banco de dados, sistemas operacionais e outras aplicações de computação.

A Árvore Rubro-Negra é chamada assim porque as arestas entre os nós da árvore são rotuladas como "vermelhas" ou "pretas". Essas cores são usadas para garantir que a árvore esteja balanceada, o que é importante para garantir a eficiência das operações de inserção, deleção e busca na árvore.

As propriedades da Árvore Rubro-Negra incluem:

1. Toda folha (nó null) é preta.
2. Todo nó vermelho tem dois filhos pretos.
3. Todo caminho de raiz a folha contém o mesmo número de nós pretos.
4. A raiz da árvore é sempre preta.

Com base nessas propriedades, a Árvore Rubro-Negra garante que a altura da árvore seja logarítmica em relação ao número de nós na árvore, o que significa que as operações de busca, inserção e deleção são executadas rapidamente.

**Inserção**

1. Adicione o novo nó como uma folha da árvore, como na árvore binária de busca comum.
2. Marque o novo nó como "rubro".
3. Se o pai do novo nó também for "rubro", então você deve reequilibrar a árvore para garantir que as propriedades da árvore rubro-**negra** sejam preservadas. Isso pode ser feito por meio de rotações e colorações.
4. Repita os passos 3 enquanto for necessário, subindo a árvore até que todas as propriedades da árvore rubro-negra sejam restauradas.

Em geral, a inserção na árvore rubro-negra é uma operação complexa, mas o balanceamento da árvore garante que as operações de busca e inserção tenham tempo de execução O(log n), onde n é o número de nós na árvore.

**Árvore-B**

É uma estrutura de dados de árvore que é utilizada para armazenar e recuperar dados de maneira eficiente. É chamada de "Árvore-B" devido a sua propriedade de equilíbrio, que permite que as operações de inserção, exclusão e pesquisa sejam realizadas em tempo logarítmico em média.

Uma Árvore-B é uma árvore binária, o que significa que cada nó pode ter no máximo dois filhos. Além disso, a Árvore-B tem a seguinte propriedade: cada nó deve conter um número fixo de itens (chaves) e todos os itens de um nó devem ser menores do que os itens do nó à sua direita e maiores do que os itens do nó à sua esquerda.

A Árvore-B é amplamente utilizada em sistemas de gerenciamento de banco de dados, pois oferece uma solução de indexação eficiente para grandes quantidades de dados. Além disso, a Árvore-B também é utilizada em aplicações que requerem uma consulta rápida e eficiente de dados, como sistemas de pesquisa, gerenciamento de arquivos e muito mais.

A Árvore-B tem as seguintes propriedades importantes:

1. Balanço: A Árvore-B mantém seus nós equilibrados, o que garante que as operações de inserção, exclusão e pesquisa sejam realizadas em tempo logarítmico em média.
2. Ordem Mínima: Cada nó de uma Árvore-B deve conter um número mínimo de itens (chaves) e um número mínimo de filhos. Se um nó é inserido com mais itens do que o mínimo permitido, ele é dividido em dois nós.
3. Ordem Máxima: Cada nó de uma Árvore-B deve conter um número máximo de itens (chaves) e um número máximo de filhos. Se um nó é inserido com mais itens do que o máximo permitido, ele é dividido em dois nós.
4. Propriedade de Busca: A Árvore-B mantém os itens em ordem, o que permite uma busca rápida e eficiente. A busca é realizada comparando a chave de pesquisa com as chaves de cada nó, o que permite determinar o local exato do item na árvore.
5. Propriedade de Inserção: A Árvore-B permite a inserção de novos itens de maneira eficiente, mantendo a estrutura equilibrada da árvore.
6. Propriedade de Exclusão: A Árvore-B permite a exclusão de itens de maneira eficiente, mantendo a estrutura equilibrada da árvore.
7. Flexibilidade: A Árvore-B é uma estrutura de dados flexível, o que significa que ela pode ser facilmente adaptada a diferentes aplicações e requisitos de armazenamento de dados.

A **Árvore AVL** (Adelson-Velsky and Landis) é uma estrutura de dados de árvore binária de busca balanceada.

As propriedades de uma árvore AVL incluem:

1. Balanceamento: Uma árvore AVL mantém um equilíbrio entre as alturas de seus nós esquerdo e direito, de modo que a diferença de altura entre eles não seja maior do que 1.
2. Busca Binária: As propriedades de busca de uma árvore binária de busca são preservadas na árvore AVL. Isso significa que as operações de inserção, remoção e busca podem ser realizadas em tempo logarítmico.
3. Auto-balanceamento: A árvore AVL se ajusta automaticamente após cada operação de inserção ou remoção, mantendo o equilíbrio entre as alturas de seus nós.
4. Altura Otimizada: Devido ao seu balanceamento, a altura de uma árvore AVL é otimizada e geralmente é menor do que a de uma árvore binária de busca não balanceada. Isso torna as operações de busca, inserção e remoção mais eficientes.
5. Rotacionamentos: O balanceamento na árvore AVL é mantido através de rotacionamentos, que são operações simples que reordenam os nós da árvore.

**Diferenças**

AVL X ABB

A principal diferença entre uma árvore AVL e uma árvore de busca binária é o equilíbrio. Uma árvore AVL é uma árvore de busca binária balanceada, enquanto uma árvore de busca binária não é necessariamente balanceada.

AVL X RN

1. Método de balanceamento: A Árvore AVL balanceia-se através do rotacionamento, que são operações simples que reordenam os nós da árvore. Já a Árvore Rubro-Negra balanceia-se através da utilização de duas cores, vermelho e preto, que representam a altura relativa de um nó em relação aos seus filhos.
2. Altura da Árvore: A Árvore AVL tem uma altura média otimizada, mas pode ter uma altura máxima maior do que a Árvore Rubro-Negra. Isso significa que a Árvore Rubro-Negra é mais eficiente em termos de espaço, enquanto a Árvore AVL é mais eficiente em termos de tempo.
3. Complexidade: A Árvore AVL requer mais verificações para manter seu equilíbrio do que a Árvore Rubro-Negra, resultando em uma complexidade maior para inserções e remoções. No entanto, as operações de busca são realizadas de forma mais rápida na Árvore AVL do que na Árvore Rubro-Negra.

Em resumo, a escolha entre uma Árvore AVL e uma Árvore Rubro-Negra depende do equilíbrio entre a complexidade das operações de inserção e remoção e a eficiência das operações de busca. Em geral, a Árvore Rubro-Negra é preferida em aplicações onde a inserção e a remoção são frequentes, enquanto a Árvore AVL é preferida em aplicações onde a busca é mais frequente.

Árvore B x RN

1. Armazenamento de dados: A Árvore B é uma árvore indexada que permite armazenar dados em disco, enquanto a Árvore Rubro-Negra é uma estrutura de dados em memória principal, o que a torna mais rápida para operações de busca e inserção.
2. Balanceamento: A Árvore B balanceia-se através da divisão de nós em várias páginas, enquanto a Árvore Rubro-Negra balanceia-se através da utilização de duas cores, vermelho e preto.
3. Operações: A Árvore B tem uma complexidade maior para as operações de busca e inserção devido à necessidade de lidar com as páginas, mas é mais eficiente para operações de atualização e remoção de dados. Já a Árvore Rubro-Negra é mais rápida para operações de busca e inserção, mas pode ter uma complexidade maior para operações de remoção.
4. Eficiência: A Árvore B é mais eficiente para grandes quantidades de dados que precisam ser armazenados em disco, enquanto a Árvore Rubro-Negra é mais eficiente para pequenas quantidades de dados em memória principal.

**Bibliografia**

<https://www.youtube.com/watch?v=b-0h8gqTrlw>

<https://www.youtube.com/watch?v=DaWNuijRRFY>

*Revista Sítio novo; Uma comparação do cálculo da mediana de inteiros contidos em árvores AVL e Rubro-negras*