RESUMO ED2

ÁRVORE RUBRO-NEGRA

Rubro-Negra

Uma árvore rubro-negra é uma árvore de busca binária autobalanceável que possui um bit extra de armazenamento, sua cor. Essas árvores asseguram que o comprimento de nenhum desses caminhos seja maior que duas vezes o de qualquer outro, de modo que a árvore seja aproximadamente balanceada. No caso dessa árvore trataremos o nó que aponta para o pai, caso seja nulo, como ponteiros para as folhas, nós externos, dela e os nós normais que possuem chaves como nós internos da árvore.

A árvore rubro-negra deve suprir as seguintes características:

1. Todo nó deverá ser vermelho ou preto
2. Se um nó é vermelho, seus filhos são pretos
3. Toda folha é preta
4. A raiz é preta
5. Todos os caminhos simples de um nó até as folhas contém o mesmo número de nós pretos

Propriedades Gerais

Cor do nó: cada nó é vermelho ou preto

Folhas: Todas as folhas são representadas pelo ponteiro NIL na árvore e são pretas.

Raiz: O nó raiz é preto

Altura das folhas pretas: Todos os caminhos simples de um nó até as folhas contém o mesmo número de nós pretos

Balanceamento: A altura da árvore será no máximo 2log (n + 1), onde n é o número de nós internos da árvore.

Tempo de busca: O tempo de busca na árvore rubro-negra é de O(log n)

Inserir e Deletar: O tempo de inserção e deletar da árvore é O(log n), porém diferente da árvore de busca binária a árvore-rubro-negra é autobalanceavel garantindo que a altura dela será no máximo 2log n – 1, e por conta do autobalanceamento a complexidade fica em torno de O(log n). O máximo leva em consideração o pior caso onde metade dos nós da árvore estão coloridos em vermelho, o que pode dobrar a altura dela.

Rotações

As operações de árvore de busca na árvore rubro-negra possuem complexidade de O(log n) e podem violar as propriedades da árvore rubro-negra. Por isso, utilizamos uma técnica para mudar a estrutura de ponteiros chamada de rotação. Temos dois tipos de rotação, para a esquerda e para a direita. A rotação para a esquerda ocorre quando a subárvore da direita é mais alta que a subárvore da esquerda. Essa operação consiste em rotacionar o nó pai e seu filho da direita para a esquerda fazendo com que o nó pai e o nó da esquerda se torne filho do nó da direita. Essa operação resulta em uma subárvore esquerda mais alta. Já a rotação para a direita envolve rotacionar o nó e seu filho da esquerda para a direita, fazendo com que o filho da direita se torne pai do nó original e de seu filho da direita, essa operação ocorre quando a subárvore esquerda é maior que a subárvore direita.

As rotações são utilizadas para balancear a árvore e manter a altura dela como O(log n).

OPERAÇÕES:

Pesquisa: A operação de pesquisa em uma árvore rubro-negra funciona de maneira semelhante a uma árvore de pesquisa binária, iniciando no nó raiz e comparando a chave de pesquisa com as chaves no nó. Se a chave de pesquisa for igual a uma das chaves do nó, a pesquisa é bem-sucedida e o valor correspondente é retornado. Se a chave de pesquisa for menor que a chave no nó, a pesquisa continua na subárvore esquerda. Se a chave de pesquisa for maior que a chave no nó, a pesquisa continua na subárvore direita. Esse processo continua até que a chave seja encontrada ou fique claro que a chave não está na árvore. A complexidade de tempo desta operação é O(log n), onde n é o número de nós na árvore.

Inserção: A operação de inserção em uma árvore rubro-negra começa encontrando o local apropriado na árvore para inserir o novo par chave-valor, de forma semelhante a uma árvore de busca binária. Uma vez encontrado o local apropriado, o novo par chave-valor é inserido e as restrições da árvore Rubro-Negra são mantidas ajustando as cores dos nós e potencialmente realizando rotações. O novo nó é colorido de vermelho e a árvore é então percorrida em direção à raiz, ajustando as cores e potencialmente realizando rotações conforme necessário para manter as restrições da árvore Rubro-Negra. A complexidade de tempo desta operação é O(log n), onde n é o número de nós na árvore.

Exclusão: A operação de exclusão em uma árvore rubro-negra começa por encontrar a chave a ser excluída, semelhante a uma árvore binária de pesquisa. Uma vez encontrada a chave, ela é removida da árvore e as restrições da árvore rubro-negra são mantidas ajustando as cores dos nós e potencialmente realizando rotações. Se o nó a ser excluído tiver dois filhos, seu sucessor (o nó com a próxima maior chave) é encontrado e sua chave e valor são trocados pela chave e valor do nó a ser excluído. O nó com a próxima maior

chave é excluído. Após a exclusão, a árvore é percorrida em direção à raiz, ajustando as cores e possivelmente realizando rotações conforme necessário para manter as restrições da árvore rubro-negra. A complexidade de tempo desta operação é O(log n), onde n é o número de nós na árvore.

CONCLUSÃO:

Uma árvore Rubro-Negra é um tipo de árvore binária de busca auto-balanceada que impõe um conjunto de restrições para garantir complexidade de tempo logarítmica para suas operações principais.

As principais operações de uma árvore rubro-negra são:

Pesquisar: Encontrar uma chave na árvore rubro-negra e retornar seu valor, caso exista.

Inserção: Inserção de um novo par chave-valor na árvore Rubro-Negra.

Deleção: Removendo um par chave-valor da árvore Rubro-Negro.

Aqui está a complexidade de tempo para essas operações:

Pesquisa: A complexidade de tempo da operação de pesquisa em uma árvore rubro-negra é O(log n), onde n é o número de nós na árvore. Isso ocorre porque uma operação de pesquisa envolve iniciar no nó raiz e seguir o ponteiro filho apropriado com base na chave de pesquisa, até que a chave seja encontrada ou fique claro que a chave não está na árvore.

Inserção: A complexidade de tempo da operação de inserção em uma árvore rubro-negra é O(log n), onde n é o número de nós na árvore. Isso ocorre porque uma operação de inserção envolve encontrar o local apropriado na árvore para inserir o novo par chave-valor e, em seguida, ajustar as cores e rotações dos nós para manter as restrições da árvore Rubro-Negra.

Exclusão: A complexidade de tempo da operação de exclusão em uma árvore rubro-negra é O(log n), onde n é o número de nós na árvore. Isso ocorre porque uma operação de exclusão envolve encontrar a chave a ser excluída e, em seguida, ajustar as cores e rotações dos nós para manter as restrições da árvore rubro-negra.

Em geral, as árvores Rubro-Negra possuem complexidade de tempo logarítmica para suas operações principais, tornando-as adequadas para uso em diversas estruturas de dados que requerem operações eficientes de busca, inserção e exclusão.