**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS**

**ESTRUTURA DE DADOS II - C01 – SEGUNDO SEMESTRE DE 2023**

**FREDERICK VITORINO DE LIMA**

**AED – RED BLACK TREES**

**Árvore Red-Black**

O código implementa uma Árvore Red-Black em C++. A árvore Red-Black é uma estrutura de dados que mantém suas propriedades balanceadas.

Uma árvore Red-Black é semelhante a uma árvore binária de busca, mas possui algumas propriedades adicionais que garantem seu balanceamento e desempenho eficiente em operações de inserção, exclusão e busca.

As principais propriedades de uma árvore Red-Black são as seguintes:

1. Cada nó da árvore é atribuído a uma cor: vermelho ou preto.
2. A raiz da árvore é sempre preta.
3. Todas as folhas da árvore (nós nulos ou sentinelas) são consideradas pretas.
4. Se um nó é vermelho, seus filhos devem ser pretos.
5. Todo caminho da raiz até uma folha contém o mesmo número de nós pretos. Esse é o requisito que garante o balanceamento da árvore.

**Descrição Inicial do Código**

O código possui várias classes e arquivos;

* A classe “No” representa um nó na árvore Red-Black e contém informações como valor, cor e ponteiros para o pai, filho esquerdo e filho direito. Representa o arquivo “No.h”.
* A classe “ArvoreRB” representa a árvore Red-Black em si e contém métodos para várias operações na árvore, como inserção, remoção, busca, obtenção do valor mínimo e máximo, e impressão da árvore. Representa o arquivo “ArvoreRB.h”.
* O arquivo “No.cpp” possui a implementação dos métodos da classe “No.h”.
* O arquivo “ArvoreRB.cpp” possui a implementação dos métodos da classe “ArvoreRB.h”.
* O arquivo “RedBlack.cpp” possui a função main e um menu interativo para o usuário realizar operações na árvore Red-Black. O usuário pode escolher inserir um valor na árvore, remover um valor, buscar um valor, obter o menor e o maior valor, ou sair do programa, fechando assim o menu.

Para compilar corretamente esse código, você precisa compilar todos os arquivos (g++ RedBlack.cpp No.Cpp ArvoreRB.cpp No.h ArvoreRB.h). O arquivo executável gerado pela compilação do código se chama “RedBlack.exe” e ele possui 2 versões: Uma para Windows e outra para Linux.

**Explicação dos métodos e funções Complexas**

**imprimirAuxiliar:** É usada para imprimir a estrutura da árvore vermelho-preto no console, mostrando os valores dos nós, suas cores (P para preto e V para vermelho) e a relação com seus filhos esquerdo e direito.

A função começa verificando se o nó passado como parâmetro é diferente de nullptr, garantindo que seja um nó válido.

Dentro do bloco if, a função imprime o espaçamento, que é usado para criar uma indentação visual para representar a profundidade do nó na árvore. Em seguida, imprime a direção do nó em relação ao seu pai (Esq: para filho esquerdo e Dir: para filho direito). Depois disso, imprime o valor do nó e sua cor. Se a cor for preta, é exibido "P"; se for vermelha, é exibido "V".

Após imprimir as informações do nó atual, a função é chamada recursivamente para imprimir os nós filhos, tanto o filho esquerdo quanto o filho direito. O espaçamento é aumentado para criar uma indentação adicional e a direção é atualizada para refletir a relação com o nó atual.

**balancearRemocao:** A função “balancearRemocao” é responsável por realizar as operações necessárias para balancear a Árvore Red-Black após a remoção de um nó.

A função começa com um loop que é executado enquanto o nó atual não é a raiz e sua cor é preta. Os passos para balancear a árvore após a remoção são:

1. Verifica se o nó atual é filho esquerdo do pai.
2. Se o nó for filho esquerdo do pai:

* Obtém o irmão do nó (irmão do pai) armazenando-o no ponteiro irmão.
* Caso 1: O irmão é vermelho.

O irmão é definido como preto, o pai como vermelho, realiza uma rotação esquerda no pai e atualiza o irmão para o novo irmão do nó.

* Caso 2: O irmão e os filhos do irmão são pretos.

O irmão é definido como vermelho e o nó é atualizado para o pai.

* Caso 3: O irmão é preto, o filho esquerdo do irmão é preto e o filho direito do irmão é vermelho.

O filho esquerdo do irmão é definido como preto, o irmão é definido como vermelho, realiza uma rotação direita no irmão e atualiza o irmão para o novo irmão do nó.

* Caso 4: O irmão é preto e o filho direito do irmão é vermelho.

O irmão é definido com a cor do pai, o pai é definido como preto, o filho direito do irmão é definido como preto, realiza uma rotação esquerda no pai e atualiza o nó para a raiz.

1. Se o nó não for filho esquerdo do pai:

* Obtém o irmão do nó (irmão do pai) armazenando-o no ponteiro irmão.
* Repete os mesmos casos (Caso 1, Caso 2, Caso 3 e Caso 4), mas com as operações de rotação e coloração aplicadas de forma simétrica para o lado direito.

1. Define a cor do nó como preto.

Esses passos são executados para garantir que a propriedade da Árvore Red-Black seja mantida após a remoção de um nó.

**balancearInsercao:** É responsável por realizar as operações necessárias para balancear a árvore vermelho-preto após a inserção de um novo nó.

A função começa com a declaração dos ponteiros pai e avo para armazenar o pai e o avô do nó atual. Em seguida, um loop é iniciado, que será executado enquanto o nó não é a raiz e a cor do nó e do pai é vermelha.

Dentro do loop, os ponteiros pai e avo são atualizados para o pai e o avô do nó atual. Em seguida, verifica-se se o pai do nó é filho esquerdo do avô.

1. Se o pai for filho esquerdo do avô:

* Obtém o tio (irmão do pai) do nó armazenando-o no ponteiro tio.
* Caso 1: O tio é vermelho.

O avô é definido como vermelho, o pai e o tio são definidos como pretos, e o nó é atualizado para o avô.

* Caso 2: O tio é preto.

Verifica se o nó é filho direito do pai.

Realiza uma rotação esquerda no pai.

Atualiza o nó e o pai para seus novos valores após a rotação.

Realiza uma rotação direita no avô.

Troca as cores do pai e do avô.

Atualiza o nó para o pai.

1. Caso contrário (o pai não é filho esquerdo do avô):

* Obtém o tio (irmão do pai) do nó armazenando-o no ponteiro tio.
* Repete os mesmos casos (Caso 1 e Caso 2), mas com as operações de rotação e troca de cores aplicadas de forma simétrica para o lado direito.

Após sair do loop, define-se a cor da raiz como preto para garantir que a propriedade da Árvore Red-Black seja mantida.

Essa função é responsável por ajustar as cores e realizar as rotações necessárias para manter as propriedades da Árvore Red-Black após a inserção de um novo nó.

**remover:** A função “remover” é usada para remover um nó específico contendo um valor da Árvore Red-Black.

A função começa buscando o nó com o valor fornecido usando a função “buscarNo”. Se o nó não for encontrado, a função simplesmente retorna.

Em seguida, são declarados dois ponteiros de nó, substituto e filho, inicializados como nullptr. Esses ponteiros serão usados para manipular os nós durante a remoção.

O próximo bloco de código verifica se o nó a ser removido tem no máximo um filho. Se sim, o próprio nó é atribuído como substituto. Caso contrário, o sucessor do nó é atribuído como substituto. O sucessor é o nó mínimo na sub-árvore direita do nó a ser removido.

Depois, é determinado qual será o filho do substituto. Se o substituto tiver um filho à esquerda, esse filho é atribuído à variável filho. Caso contrário, o filho à direita é atribuído.

Em seguida, são feitas atualizações nos pais e nas conexões dos nós para refletir a remoção do substituto da árvore. Se o substituto for a raiz, o filho se torna a nova raiz. Caso contrário, o substituto é removido e o filho toma seu lugar como filho do pai do substituto.

Após isso, se o substituto não for o nó original, o valor do substituto é copiado para o nó original. Isso ocorre porque o substituto está sendo removido, mas sua posição na árvore é ocupada pelo nó original.

Em seguida, é realizada a função “balancearRemocao” para garantir que as propriedades da árvore Red-Black sejam mantidas após a remoção do nó.