Universidade Federal Rural de Pernambuco
Dept. de Estatística e Informática - Bacharelado em Sistemas da Informação

Árvore Rubro Negra
Algoritmo e Estrutura de Dados

Aluna: Paula Priscila da Cruz Araujo
Professor: Tiago A. E. Ferreira

RECIFE , 2019.2

INTRODUÇÃO

A árvore Rubro Negra ou *Red Black Tree* é um tipo de estrutura de dados baseado na árvore binária com o acréscimo de atribuir cores ao nodos. As cores podem ser vermelha ou preta. Restringindo as cores dos nodos, as árvores vermelho-preto asseguram que o comprimento de nenhum desses caminhos seja maior que o dobro de qualquer outra, assim fazendo com que ela seja aproximadamente balanceada.

Existem algumas regras sobre a árvore Rubro Negra:

1. Todo nó ou é vermelho,ou é preto;
2. A raiz é sempre preta;
3. Toda folha é preta;
4. Quando se tem um nó vermelho seus filhos são obrigatoriamente pretos;
5. Para todos os caminhos o número de nodos pretos são os mesmos

É importante também salientar que cada nodos tem os atributos cor, chave, esquerda, direita e pai, e se um dos nodos não tiver filhos ou pai, os respectivos “apontam” para nulo.

DESENVOLVIMENTO

Durante o desenvolvimento deste programa foi usado a linguagem Python em sua 3° versão, com o auxílio da orientação a objeto.

A princípio é criado um classe **No** que recebe o dado digitado pelo usuário e a chave do respectivo dado, com os atributos pai, filho esquerdo, filho direito, e a cor do nó. Pai, filho esquerdo e filho direito a princípio apontam para None ou vazio.

Após isso é criado outra classe chamada ***ArvoreRB***, esta será a classe que iria criar a árvore com ajuda de um “sentinela”, um novo objeto da classe nó, que irá receber valores ‘None’ , seguindo as regras 2 e 3 , além da ideia especificada no último parágrafo da introdução, este novo nó recebe a cor preta, a partir daí os outros nós recebem as mesmas características.

A seguir foram criados alguns métodos auxiliares, são eles: ***busca_MinimoArvore***() ,***rotacionaraesquerda***() e ***rotacionaradireita***(). Esses auxiliam na manutenção do balanceamento da árvore após uma nova inserção ou deleção de um nodo.

Os seguintes métodos são ***inserirRBTree***(),***corrigir_InserirRBTree***(), ***transplantar_ArvoreRB***(),***excluir_RBTree***(),***corrigir_ExcluirRBTree***(),***buscar_dados***(),***percorrer_ArvoreRB***() e ***imprimir_arvoreRB***().

O método *busca_MinimoArvore*() como o próprio nome diz busca o valor mínimo, menor da árvore. Ele irá auxiliar o método de *excluir_RBTree*().

Os métodos *rotacionaraesquerda*() e *rotacionaradireita*() são usados para verificar se após uma inserção ou exclusão de algum valor, as propriedades da árvore rubro negra não foram violadas.

O método *inserirRBTree*() faz inserção como qualquer árvore binária, porém é ligeiramente modificada, porque depois de inserido é colorido com a cor vermelha, porém é preciso garantir se as propriedades estão preservadas, por isso é usado o *corrigir_InserirRBTree*().

O *corrigir_InserirRBTree*() corrige problemas que podem ser causados pelas propriedades 2 e 4 da árvore.

O transplatar_ArvoreRB() é um outro método auxiliar a exclusão do *excluir_RBTree*().

Já o método *excluir_RBTree*() rastreiam um nó y que poderia causar violações das propriedades vermelho-preto. Quando queremos eliminar o nó z e ele tem menos do que dois filhos, z é removido da árvore e queremos que y seja z. Quando z tem dois filhos, y deve ser o sucessor de z , e y passa para a posição de z na árvore. Embora ele contenha quase duas vezes o número de linhas de código de *excluir_RBTree*() da árvore binária, os dois procedimentos têm a mesma estrutura básica.

O método corrigir *excluir_RBTree*() assim como o *corrigir_InserirRBTree*() corrige problemas que podem ser causados pelas propriedades 2 e 4 da árvore.

O *buscar_dados*() também auxilia a deleção, verifica antes se o valor dado pelo o usuário está na árvore ou não.

E por fim ,*percorrer_ArvoreRB*() e imprimir_arvoreRB(), varrem a árvore em “Inorder” e imprimi o resultado ao usuário.

CONCLUSÃO

Como a maioria das operações da árvore rubro negra estão em tempo (O(lgn)), e seu balanceamento diferenciado fazem com que ela seja uma das melhores estruturas de dados considerando o custo computacional, comparado a outras estruturas como lista, filas ou pilhas e até árvores AVL.

As árvores rubro negra são muitos úteis em aplicações com muitas inserções e exclusões.