Universidade Federal do ABC
Algoritmos e Estruturas de Dados II 3^o Quadrimestre 2018

Prof. Carlos da Silva dos Santos

Aula prática 05 – Árvore Rubro Negra (continuação).

Nessa aula você vai implementar o procedimento de inserção na árvore rubro-negra. Para isso, vamos reaproveitar o código que você já criou para as rotações.
Vamos nos basear na implementação apresentada na referência *Introduction to Algorithms* (CLRS).
O procedimento de inserção será dividido em duas etapas: (i) inserção propriamente dita e (ii) correção das propriedades da árvore que tenham sido violadas.
Você deve criar uma função `rb_insert` com a seguinte assinatura:

```
rb_insert(Node** T, int key)
```

A função deve se basear no seguinte pseudo-código (conforme apresentado em CLRS):

```
RB-INSERT(T, z)
01    y <- nil[T]
02    x <- root[T]
03    while x != nil[T]
04        y <- x
05        if key[z] < key[y]
06            x <- left[x]
07        else
08            x <- right[x]
09    p[z] <- y
10    if y == nil[T]
11        root[T] <- z
12    else
13        if key[z] < key[y]
14            left[y] <- z
15        else
16            right[y] <- z
17    left[z] <- nil[T]
18    right[z] <- nil[T]
19    color[z] <- RED
20    RB-INSERT-FIXUP(T, z)
```

O procedimento termina com uma chamada para `RB-INSERT-FIXUP`, que corrige eventuais violações das propriedades da árvore rubro-negra. Note que o parâmetro `z` é um nó de árvore para ser inserido, enquanto `rb_insert` recebe uma chave inteira como argumento de entrada. No código acima, `left`, `right` e `p` são operadores que devolvem nós vizinhos esquerdo, direito e o pai de um nó, respectivamente. A constante `nil[T]` representa o valor de sentinela.
Para complementar a implementação, crie uma função `rb_insert_fixup`, baseando-se no pseudo-código a seguir:

```
RB-INSERT-FIXUP(T, z)
01    while color[p[z]] == RED
02        if p[z] == left[p[p[z]]]
03            y <- right[p[p[z]]]
04                if color[y] == RED // O tio (y) é vermelho
05                    color[p[z]] <- BLACK
06                    color[y] <- BLACK
07                    color[p[p[z]]] <- RED
08                else
09                    if z = right[p[z]]
10                        z <- p[z]
11                        LEFT-ROTATE(T, z)
12                    color[p[z]] <- BLACK
13                    color[p[p[z]] <- RED
14                    RIGHT-ROTATE(T, p[p[z]])
15        else
16            (análogo às linhas 03-14 mas invertendo
             esquerda e direita)
```

Depois de implementar as duas funções, teste seu código criando algumas árvores pequenas e verificando o resultado.
Crie agora uma função para calcular a altura de uma árvore rubro negra. Faça testes, variando o número de elementos da árvore e medindo a altura obtida.