Inserção Árvore Rubro-Negra

ANÁLISE DE ALGORITMOS

TALLES BEZERRA

Regras

- A raiz é preta
- Nós vermelhos só podem ter filhos pretos
- Para cada nó, todos os caminhos do nodo até qualquer folha passa pelo mesmo número de nós pretos(altura negra)

Altura

Lema 1: Numa árvore rubro-negra, a sub-árvore enraizada em um nó x tem no mínimo $2^{an(x)}-1$ nós não nulos.

Lema 2: Uma árvore rubro-negra com n nós não nulos tem no máximo altura 2.lg (n + 1).

Prova por indução.

Custo

A inserção em uma árvore rubro-negra possui complexidade O(log n).

Inserção

Caso 1: tio do nó z é vermelho

- Z->pai->cor = Preto;
- Tio->cor = Preto;
- Z->pai->pai->cor = Vermelho;
- Z= z->pai->pai;

Caso 2: tio do nó z é preto e z é o filho da direita

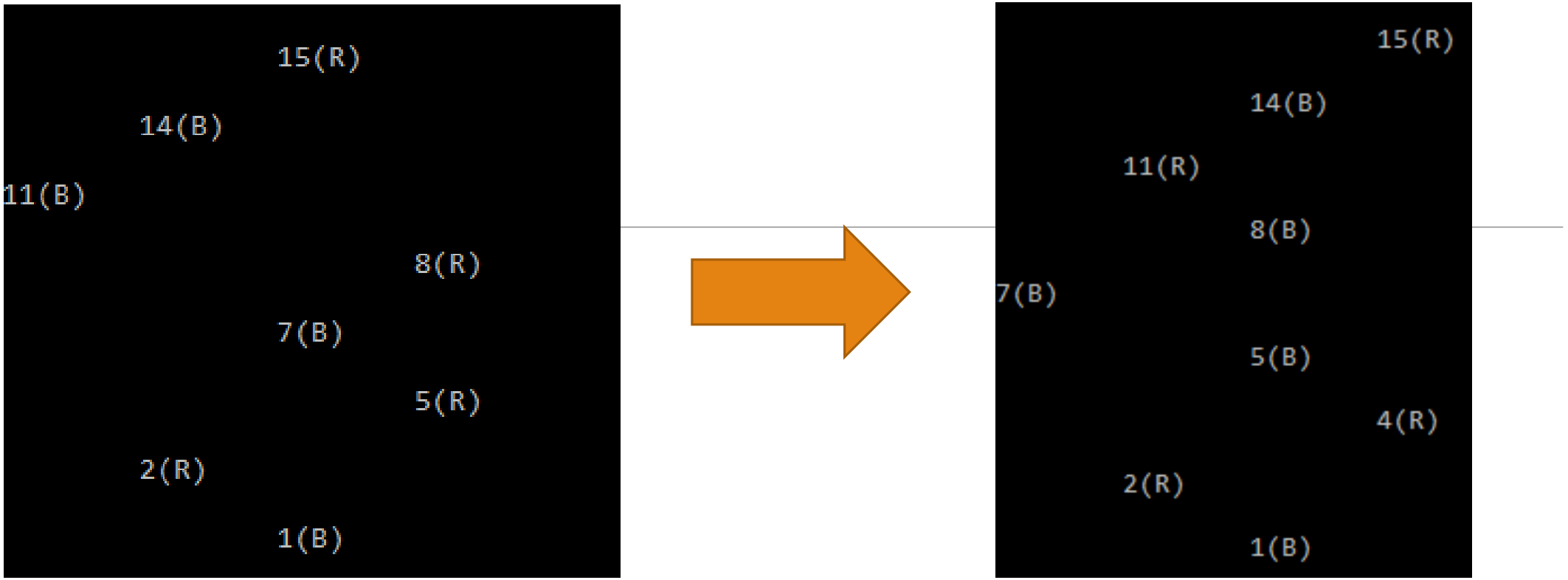
- Z = z->pai;
- Rotacao_esquerda(z);

Caso 3: tio do nó z é preto e z é o filho da esquerda

- Z->pai->cor = Preto;
- Z->pai->pai->cor = Vermelho;
- Rotacao_direita(z->pai->pai);

```
void insere_arvrb(no_arvrb **raiz, int valor){
    no_arvrb *z,*y,*x;
    y = NULL;
    z = (no_arvrb*)malloc(sizeof(no_arvrb));
    z->valor = valor;
    z->cor = RED;
    z->esq = NULL;
    z->dir = NULL;
    x = *raiz;
    while(x!=NULL){
        y = x;
        if (z->valor < x->valor)
            x = x->esq;
        else
            x = x->dir;
    }
    z->pai = y;
    if (y==NULL){
        *raiz = z;
    }
    else{
        if (z->valor < y->valor)
            y->esq = z;
        else
            y->dir = z;
    }
```

```
while(z!=NULL && z->pai != NULL && z->pai->cor == RED){
    if(z->pai == z->pai->pai->esq){
        y = z->pai->pai->dir;
        if(y==NULL){
            rot_dir(&(*raiz),z->pai->pai);
            z->pai->cor = BLACK;
            z->cor = RED;
            z->pai->dir->cor = RED;
            z = z->pai->pai;
        }
        else{
            if (y->cor == RED){
                z->pai->cor = BLACK;
                y->cor = BLACK;
                z->pai->pai->cor = RED;
                z = z->pai->pai;
            }
            else if (z == z->pai->dir){
                z = z->pai;
                rot_esq(&(*raiz),z);
                z->pai->cor = BLACK;
                z->pai->pai->cor = RED;
                rot_dir(&(*raiz),z->pai->pai);
            }
        }
    }
    else{ mesmo código, mas troca todos os esq por dir e virce-versa}
}
(*raiz)->cor = BLACK;
```


15(R)
14(B)
11(B)
8(R)
7(B)
5(R)
2(R)
1(B)
15(R)
14(B)
11(R)
8(B)
7(B)
5(B)
4(R)
2(R)
1(B)