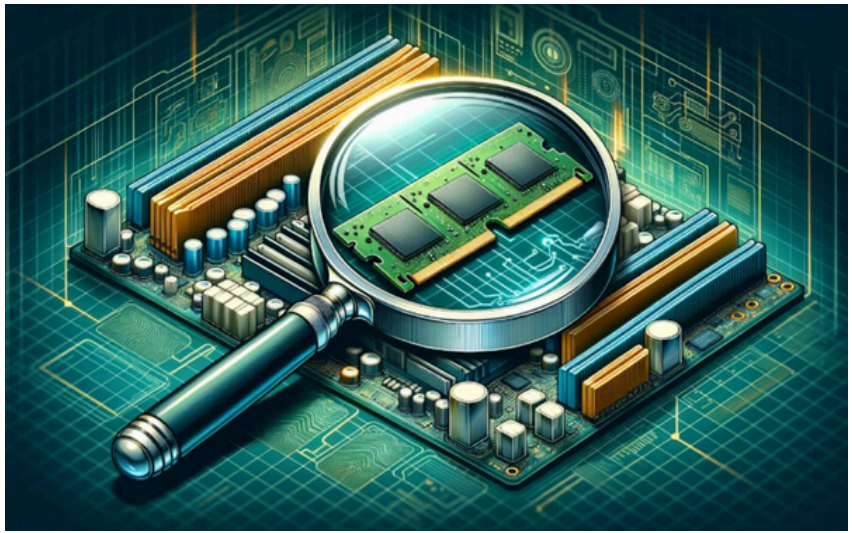
Estrutura de Dados / Programação2
Árvore Rubro-Negra

Luiz Eduardo
Marco Albuquerque
Matheus Henrique
Nícolas de Almeida

https://github.com/Nico050/Trabalho_Estrutura_de_Dados/tree/main

Problemas no gerenciamento de memória



Como podemos organizar?

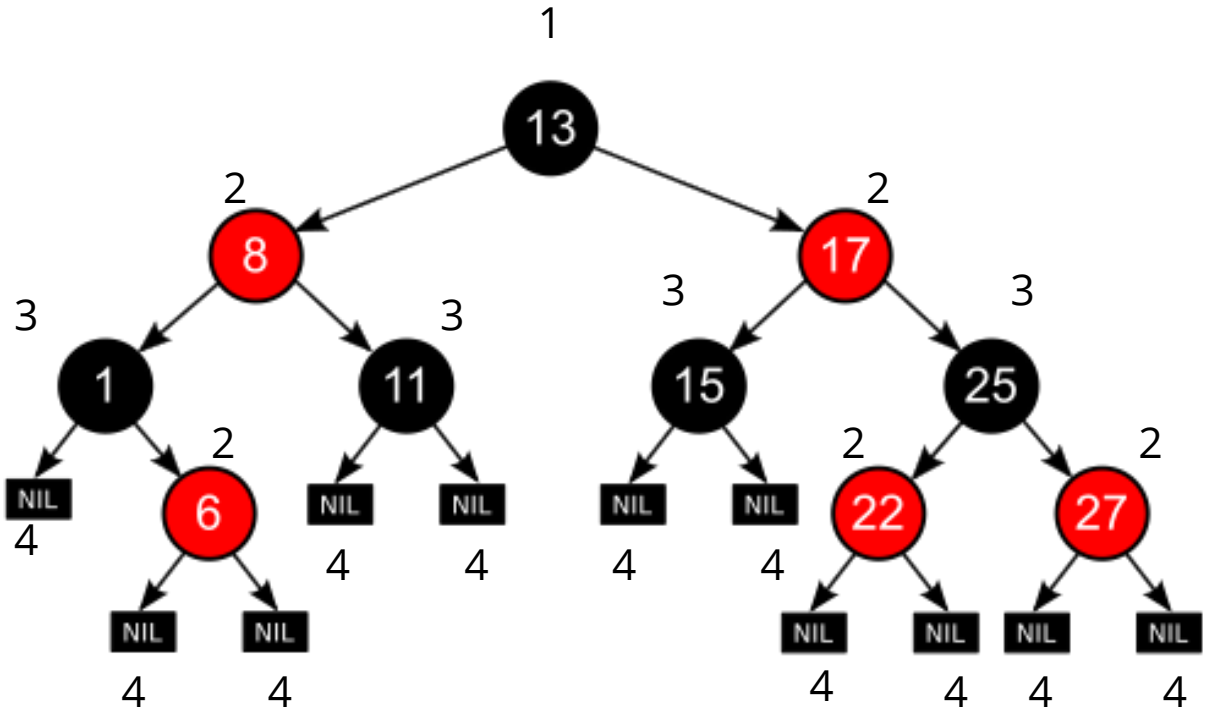
Rubro-Negra

Introdução

- A árvore rubro-negra tem esse nome devido a “coloração” de seus nós
- Uma árvore rubro-negra é uma árvore binária de busca com uma campo adicional que armazena se o nó é rubro ou negro
- O fato de um nó ser rubro ou negro é usado como fator de balanceamento da árvore
- O(logN)
- O caminho mais longo da raiz até qualquer folha nunca será mais do que o dobro do caminho mais curto

Regras da árvore rubro-negra

1. Todo nó da árvore é rubro ou negro
2. A raiz é negra
3. Toda folha (nula) é negra
4. Se um nó for rubro, então seus filhos são negros
5. Qualquer caminho da raiz até uma folha vazia tem um número igual de nós negros



1. Raiz
2. Nó
3. Nó
4. Folhas

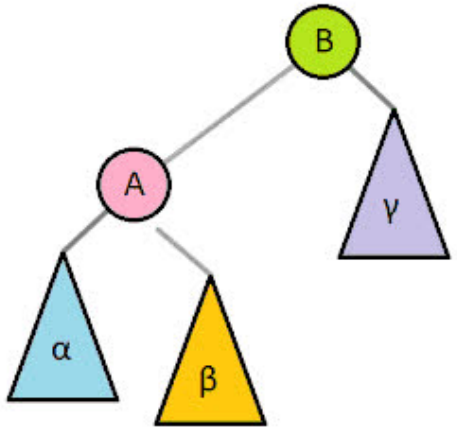
Como Funciona?

Balanceamento

- Inserção e remoção podem causar desequilíbrio

Correções são feitas com:

- Troca de cores
- Rotações (Simples e Duplas)



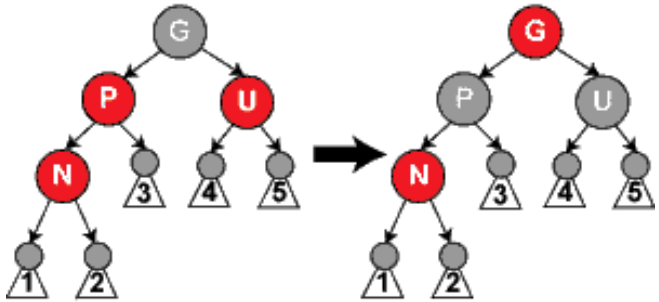
Troca de cores

Tio Rubro: Se o tio do novo nó (o irmão do nó pai) é Rubro

- Troque a cor do pai e do tio para negro
- Troque a cor do avô para rubro
- Se o avô também for a raiz, ele deve permanecer negro

Tio Negro: Se o tio é negro ou nulo:

- Nesse caso, as rotações se tornam necessárias



Inserção

- O nó inserido será sempre rubro
- As cores dos nós podem mudar para atendar às regras
- Rotações também podem ser necessárias para atender às regras

Código

```
#define RED 1
#define BLACK 0

struct node {
  int key;
  int color;
  struct node *left, *right;
};
```

```
parent(struct node *n)

TreeInsert(struct rbtree *T,int n)

struct rbtree_s{
  struct node *raiz;
};
```

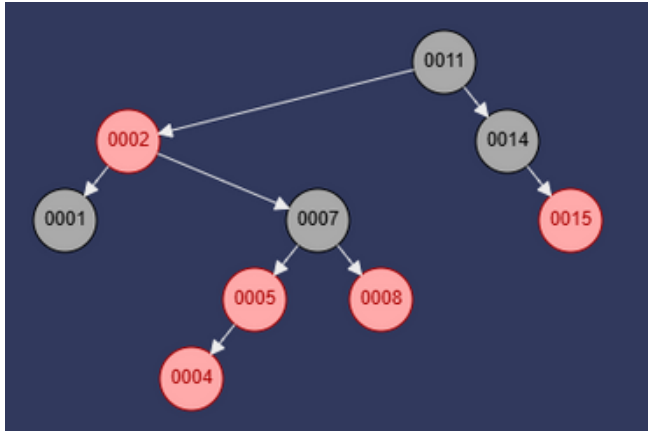
```
void RBinsertfixup(struct rbtree_s *T, int z){
  struct node *n,*pai, *avo;

  n = TreeInsert(T,z);
  n->color = RED;

  pai = parent(n);
  avo = parent(pai);

  while(avo && pai->color == RED){

    struct node *tio;

    if(pai = avo->left){
      tio = avo->right;

      if(tio && tio->color == RED){
        pai->color=BLACK;
        tio->color=BLACK;
        avo->color=RED;
        n=avo;
      }

      else{
        if (n == pai->right){

          n=pai;
          LeftRotate(T,n);
          pai = parent(n);
          avo = parent(pai);

        }

        pai->color=BLACK;
        avo->color=RED;
        RightRotate(T,avo);
      }
    }
```

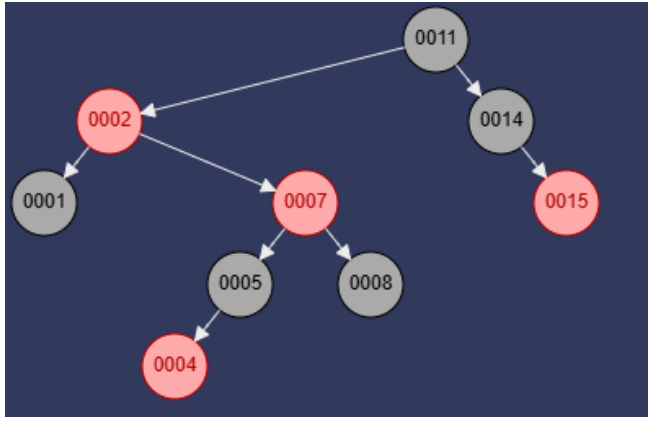
```
    else{
      tio=avo->left;
      if(tio && tio->color == RED){
        pai->color=BLACK;
        n->color=BLACK;
        avo->color=RED;
        n=avo;
      }

      else{
        if(n == pai->left){
          n=pai;
          RightRotate(T,n);
          pai = parent(n);
          avo = parent(pai);
        }

        pai->color=BLACK;
        avo->color=RED;
        LeftRotate(T,avo);
      }
    }
    pai = parent(n);
    avo = parent(pai);
  }

  T->raiz->color=BLACK;
}
```

```
void LeftRotate(struct rbtree_s *T,struct node *x)
{
    struct node *y = x->right;
    struct node *paix;
    struct node *avox;
    paix = parent(x);
    avox = parent(paix);

    x->right = y->left;


    if (paix == NULL) T->raiz = y;

    else if (x == paix->left)paix->left = y;

    else    paix->right = y;

    y->left = x;

}
```

```
void RightRotate(struct rbtree_s *T,struct node *x)
{
    struct node *y = x->left;
    struct node *paix;
    struct node *avox;
    paix = parent(x);
    avox = parent(paix);

    x->left = y->right;


    if (paix == NULL) T->raiz = y;

    else if (x == paix->right)paix->right = y;

    else    paix->left = y;

    y->right = x;

}
```

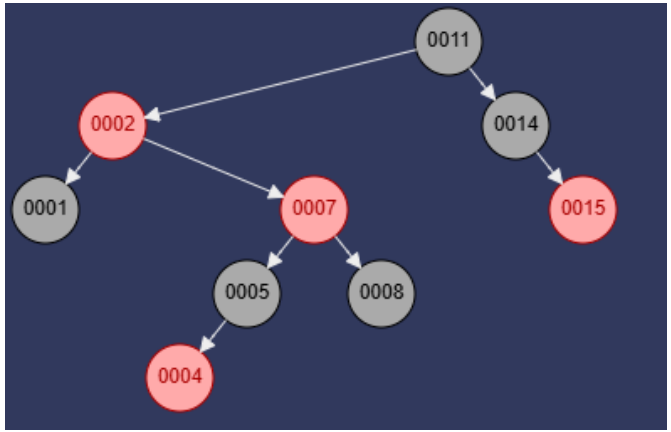
Exemplo

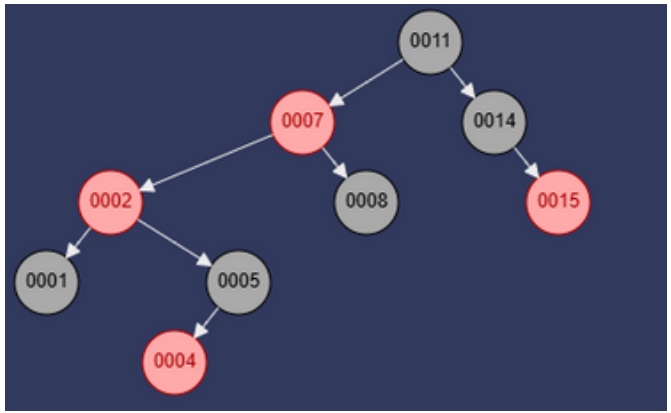




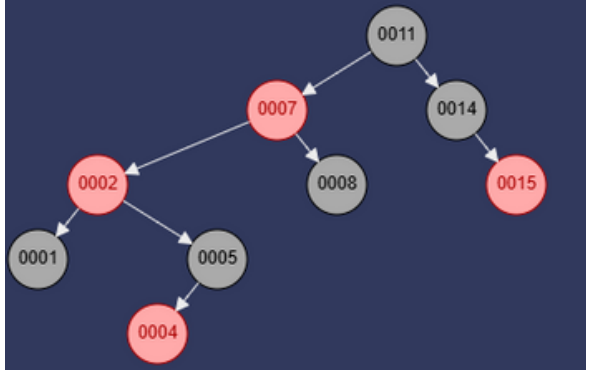

```
void RBinsertfixup(struct rbtree_s *T, int z){
  struct node *n,*pai, *avo;

  n = TreeInsert(T,z);
  n->color = RED;

  pai = parent(n);
  avo = parent(pai);

  while(avo && pai->color == RED){

    struct node *tio;

    if(pai = avo->left){
      tio = avo->right;

      if(tio && tio->color == RED){
        pai->color=BLACK;
        tio->color=BLACK;
        avo->color=RED;
        n=avo;
      }


     pai = parent(n);
     avo = parent(pai);
```

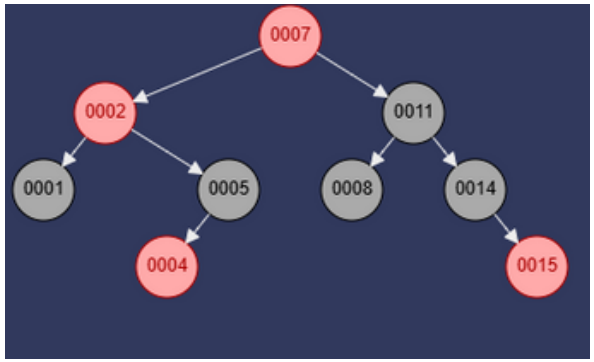
Exemplo

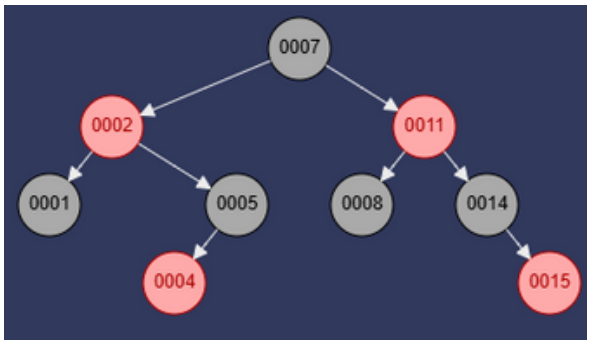





```
while(avo && pai->color == RED){

  struct node *tio;

  if(pai = avo->left){
    tio = avo->right;

     else{
       if (n == pai->right){

         n=pai;
         LeftRotate(T,n);
         pai = parent(n);
         avo = parent(pai);

       }
```

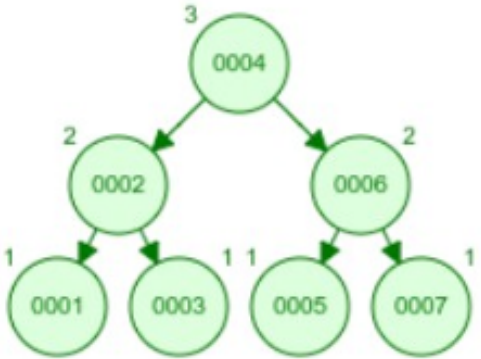


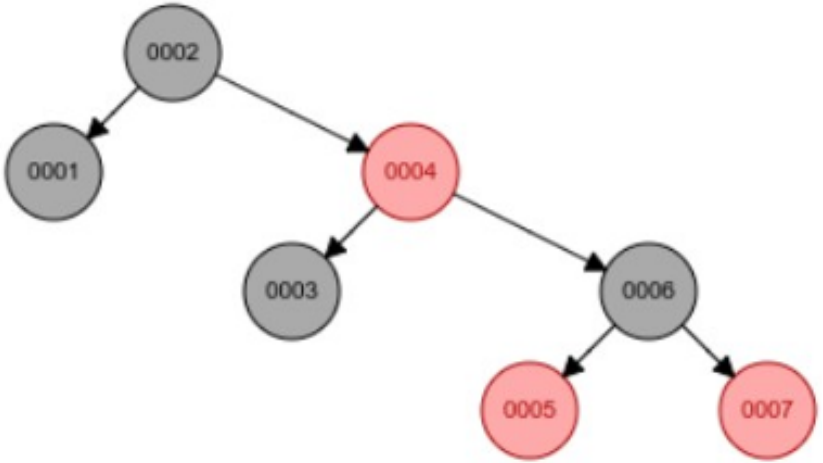





```
while(avo && pai->color == RED){

  struct node *tio;

  if(pai = avo->left){
    tio = avo->right;

      else{
        if (n == pai->right){

          n=pai;
          LeftRotate(T,n);
          pai = parent(n);
          avo = parent(pai);

        }

        pai->color=BLACK;
        avo->color=RED;
        RightRotate(T,avo);
      }
```

AVL x RB


3
0004
2
0002
2
0006
1
0001
1
0003
1
0005
1
0007


0002
0001
0004
0003
0006
0005
0007

Animações:

https://www.inf.ufsc.br/~aldo.vw/estruturas/simulador/RB.html

Referências

1. https://www.ime.usp.br/~song/mac5710/slides/08rb.pdf
2. https://www.slideshare.net/skosta/rvores-rubro-negra#40
3. http://desenvolvendosoftware.com.br/estruturas-de-dados/arvores-vermelha-preta.html#opera%C3%A7%C3%A3o-inserir