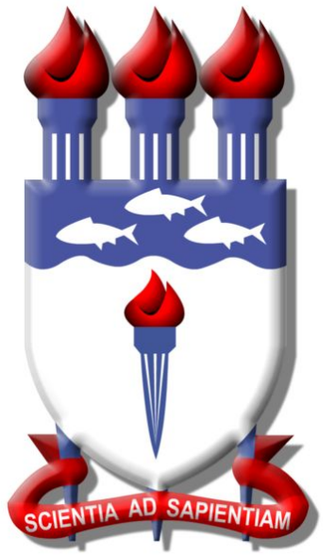


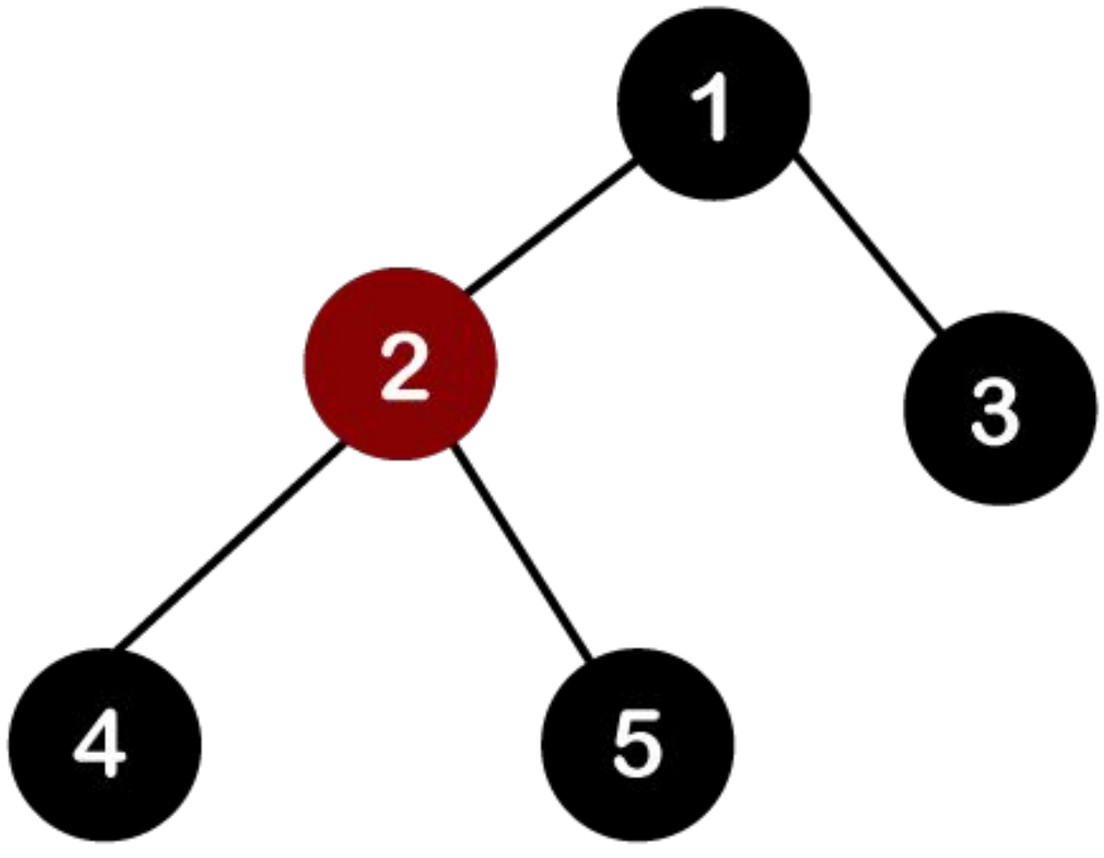
Árvore Rubro-Negra

José Félix - Gabriela Pontes

github.com/GabrielaPMRS/ProjetoED

Motivação

- Bibliotecas, bancos de dados, compiladores
- Inserção e remoção garantidamente em tempo logarítmico O(log n);

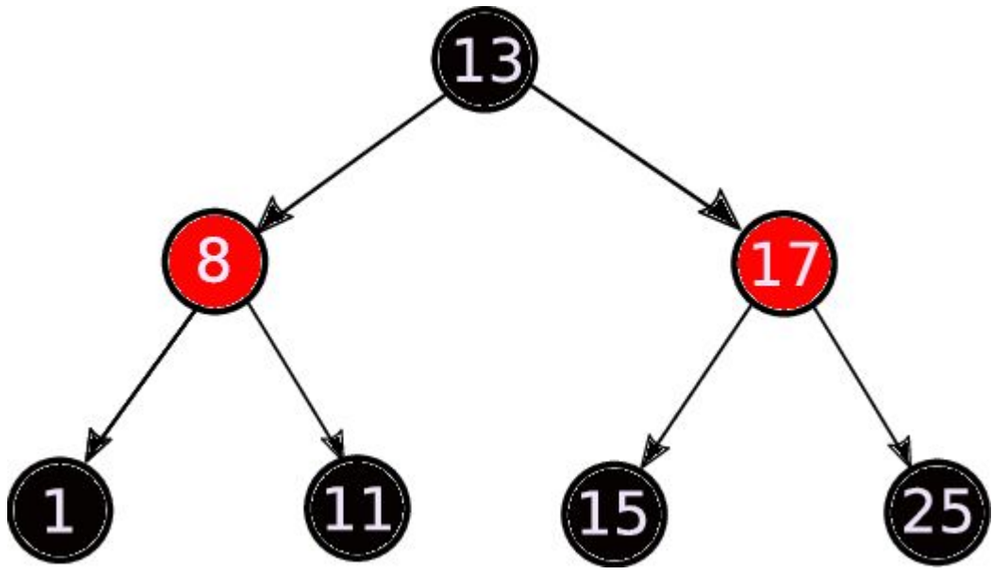


Árvore Rubro-Negra

- É um tipo de árvore binária balanceada que utiliza um esquema de **coloração dos nós** para manter o balanceamento da árvore;
- Originalmente criada em 1972 por Rudolf Bayer chamada de árvores binárias simétricas, e posteriormente em um trabalho de Leonidas J. Guibas e Robert Sedgewick em 1978, adquiriu o nome atual;
- Possui esse nome pois cada em nó dela existe um atributo de cor (além dos ponteiros para os filhos), sendo essas cores o vermelho ou o preto;

Árvore Rubro-Negra

- Possui algumas propriedades que devem ser seguidas:
  - Todo nó é **vermelho** ou **preto**;
  - A raíz é sempre preta;
  - Todo nó folha(NULL) é preto;
  - Não existem nós vermelhos consecutivos;
  - Para cada nó, todos os caminhos desse nó para os nós folhas descendentes contém o mesmo número de nós pretos.



Árvore Rubro-Negra Caída para a Esquerda

- Em 2008, Robert Sedgewick, desenvolveu a **árvore rubro-negra caída para a esquerda,** que é uma variante da árvore rubro negra;
- Garante a mesma complexidade de operações, mas possui uma implementação ainda mais simples na inserção e remoção de nós;
- Possui uma propriedade extra: se um nó é vermelho, então ele é o filho esquerdo do seu pai;

Definições

- Seja “ARB” uma árvore rubro-negra e “a” e “b” nós de ARB;
  - todo nó é uma folha se não possuir nenhum filho;
  - raíz é o nó que está no nível mais acima de ARB;
  - todo nó que não é uma folha é um nó interno;
  - uma aresta é a conexão entre dois nós;
  - um caminho é uma sequência de arestas que conectam dois nós, e o comprimento do caminho é o número de arestas desse caminho
  - a altura de um nó é o comprimento do caminho desse nó até uma folha;
  - a profundidade de um nó é o comprimento do caminho desse nó até a raíz

Tipo Abstrato de Dados (ADT)

```
ArvRB* create_ArvRB();
void free_ArvRB(ArvRB* raiz);
ArvRB* insert_ArvRB(ArvRB* raiz, int valor);
ArvRB* remove_ArvRB(ArvRB* raiz, int valor);
int isEmpty_ArvRB(ArvRB* raiz);
int search_ArvRB(ArvRB* raiz);
void preOrder_ArvRB(ArvRB* raiz);
void inOrder_ArvRB(ArvRB* raiz);
void posOrder_ArvRB(ArvRB* raiz);
```

Struct

```
typedef struct node
{
  int data;
  struct node *left;
  struct node *right;
  int color;
} ArvRB;
```

Código - Inserção

```
int insert_ArvRB(ArvRB **root, int value)
{
  int check; //variável de controle
  *root = insertNode(*root, value, &check);

  if (*root != NULL)
  {
    (*root)->color = BLACK;
  }

  return check; // se der errado, retorna 0. Senão, 1
}
```

Código - Inserção

```
ArvLLRB *insertNode(ArvRB *root, int value, int *check)
{
  if (root == NULL)
  {
    ArvLLRB *newNode;
    newNode = (ArvRB *)malloc(sizeof(ArvRB));

    if (newNode == NULL) //Se a alocação der errado
    {
      *check = 0;
      return NULL;
    }

  newNode->data = value;
  newNode->color = RED;
  newNode->right = NULL;
  newNode->left = NULL;
  *check = 1;
  return newNode;
}
```

Código - Inserção

```
if (value == root->data)
 {
   *check = 0; // O valor está duplicado na árvore
 }

 else
 {
   if (value < root->data)//O valor é menor
   {
     root->left = insertNode(root->left, value, check);
   }
   else //O valor é maior
   {
     root->right = insertNode(root->right, value, check);
   }
}
```

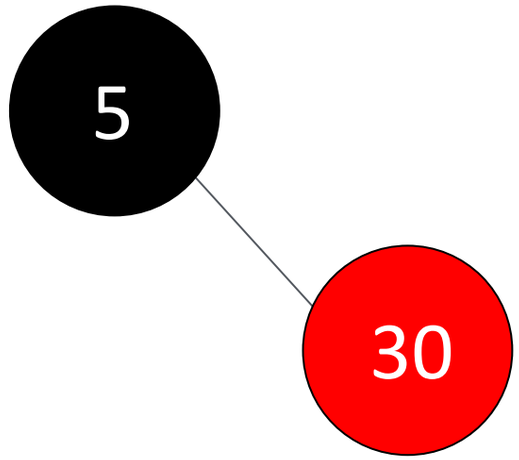
Código - Inserção

```
// Verifica se o filho da esquerda é preto o filho da direita é
vermelho
if (color(root->right) == RED && color(root->left) == BLACK)
  {
    root = rotateLeft(root); //Rotaciona para esquerda
  }

// Verifica se o filho da esquerda e o neto da esquerda são vermelhos

if (root->left != NULL && color(root->left) == RED
    && color(root->left->left) == RED)
  {
    root = rotateRight(root); //Rotaciona para direita
  }

 // Verifica se os dois filhos são vermelhos
  if (color(root->left) == RED && color(root->right) == RED)
  {
       changeColor(root); //Muda a cor dos nós da raíz atual e dos
seus filhos
  }

  return root;
}
```

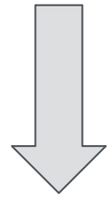
Código - Inserção

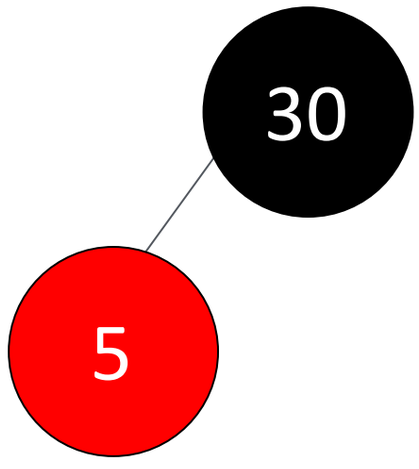
Insere valor 5

Insere valor 30



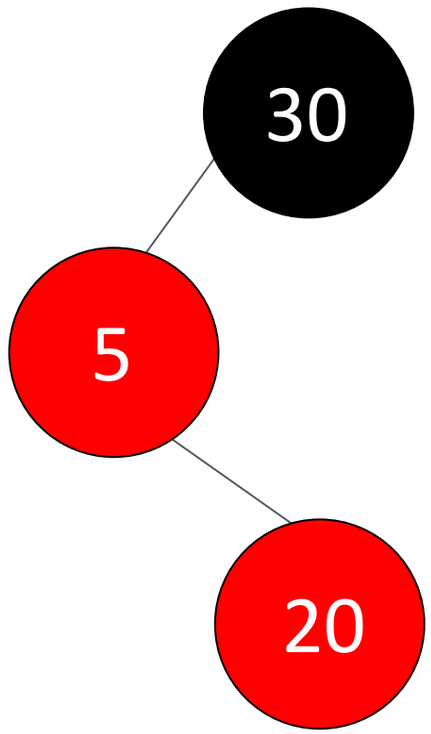
Rotaciona à esquerda em “5”



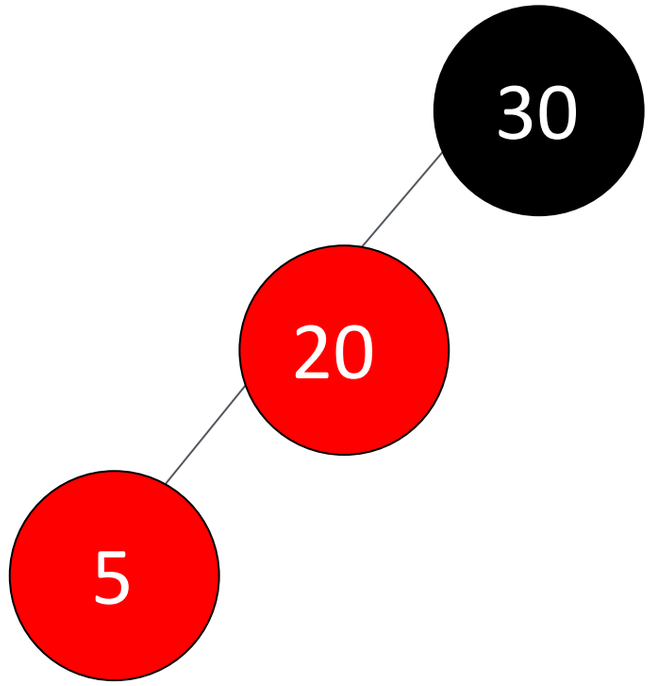


Código - Inserção

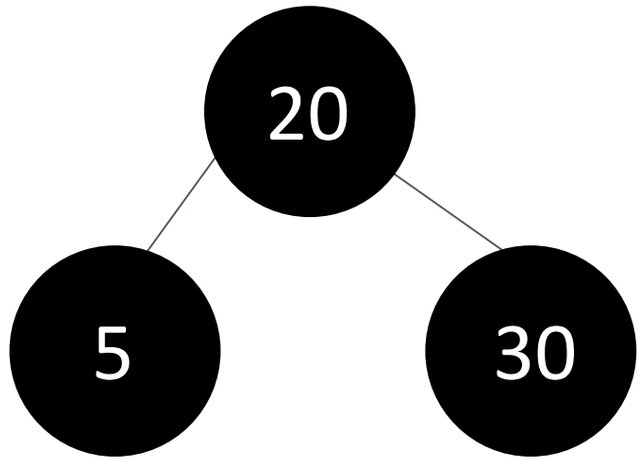
Insere valor 20



Rotaciona à esquerda em “5”

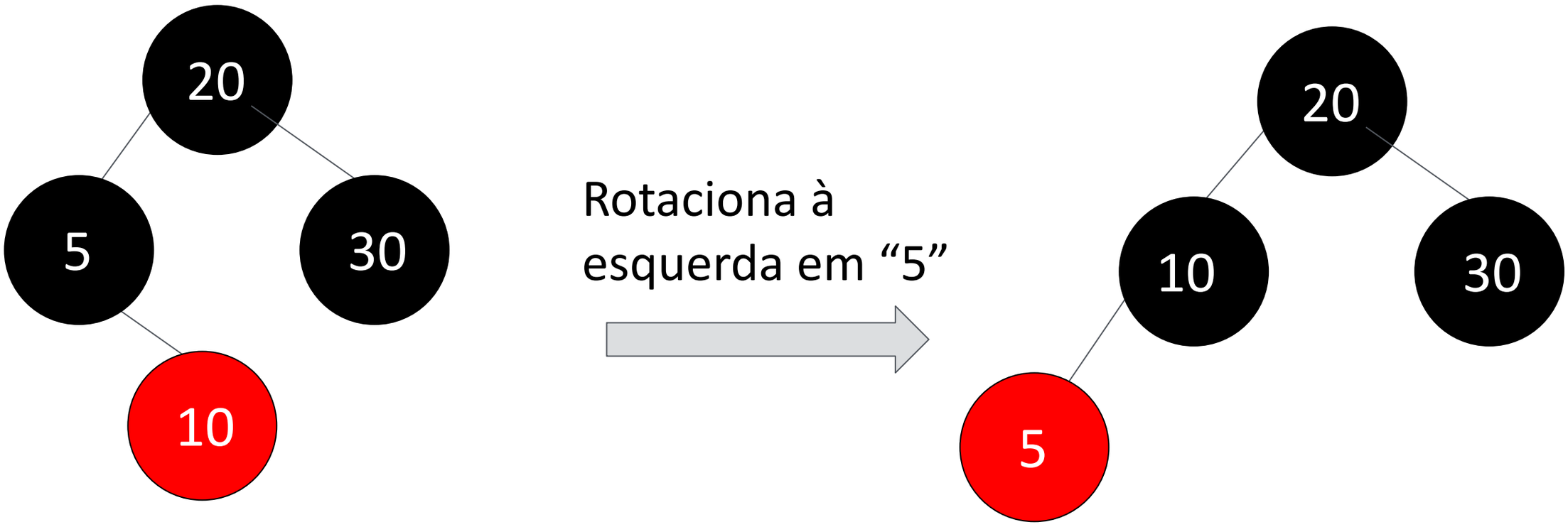


Rotaciona à direita em “20” e muda as cores dos nós

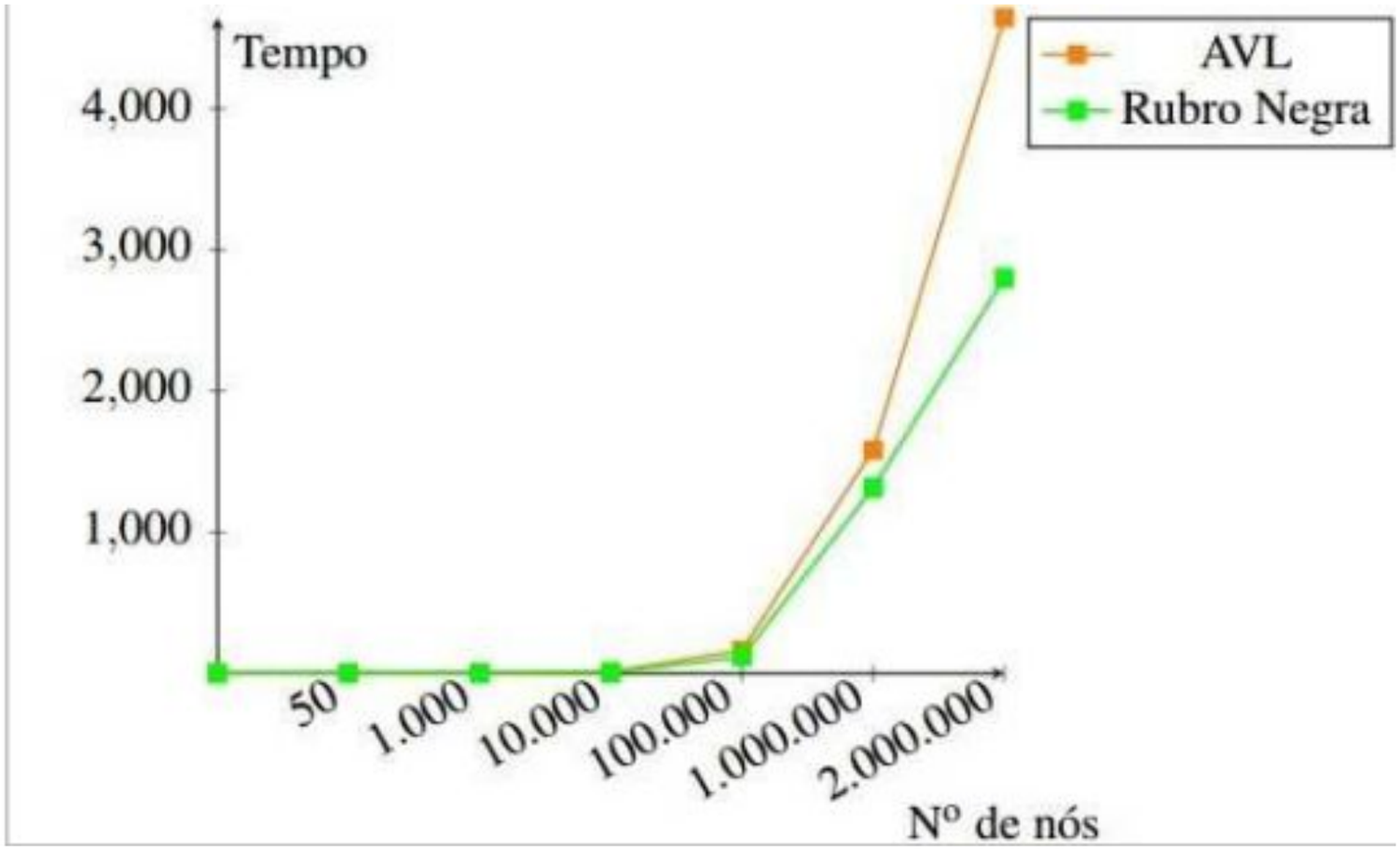


Código - Inserção

Insere valor 10

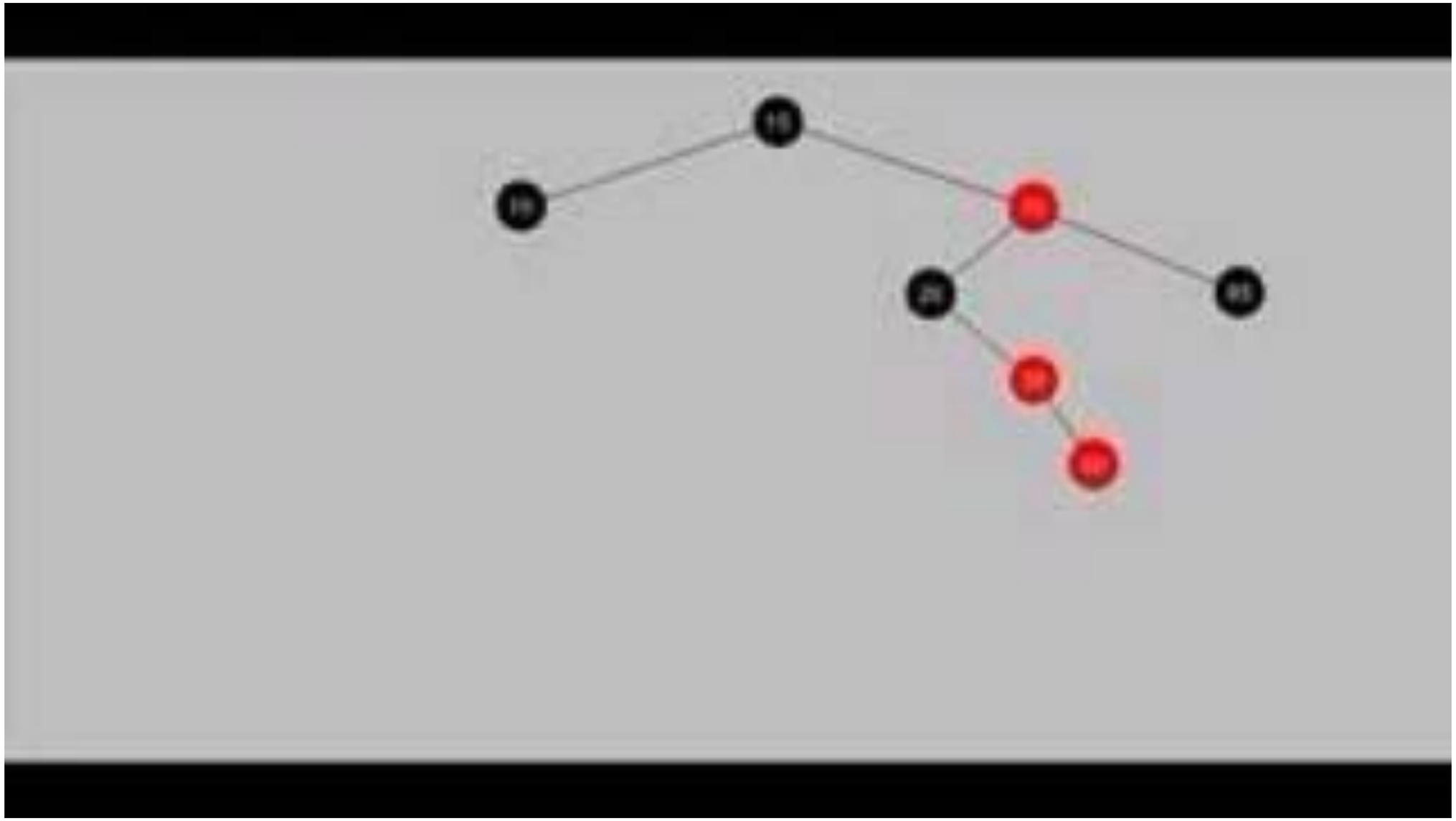


- Devido aos critérios mais flexíveis (no quesito de rotações) a inserção e a remoção são mais velozes do que utilizando uma AVL



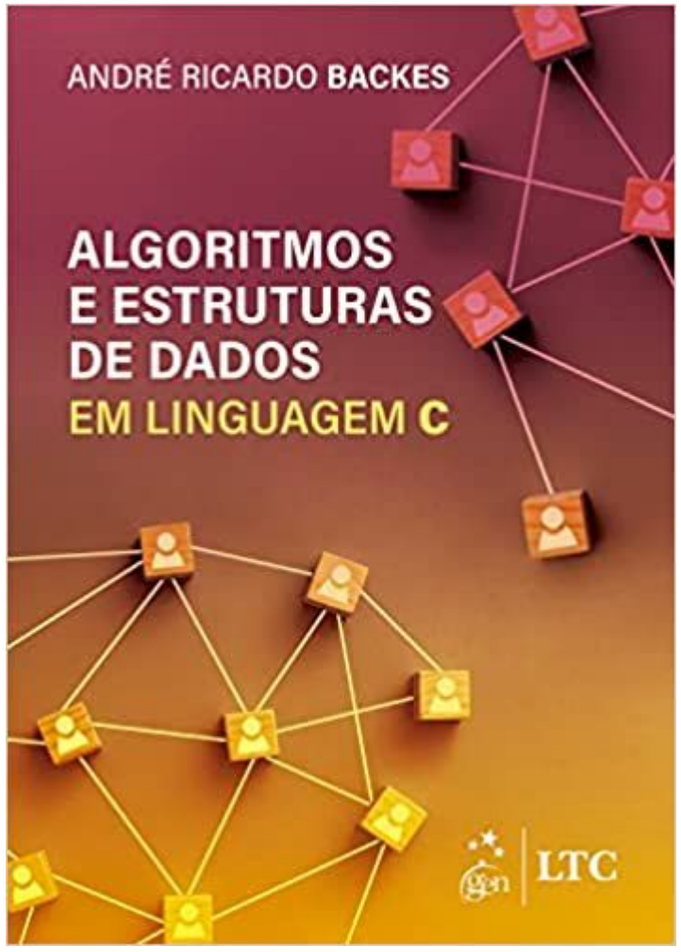
Fonte: Revista Sítio Novo - IFTO. Uma comparação entre árvores AVL e Rubro-Negras

Animação - Inserção de nós em uma Árvore Rubro-Negra



Fonte: Youtube, disponível em: https://www.youtube.com/watch?v=vDHFF4wjWYU

Referência



Capítulo 12