

# Algoritmos e Estruturas de Dados II

## Roteiro de Laboratório – Árvore Bicolor

### Objetivo

Implementar uma Árvore Bicolor a partir de uma Árvore Binária de Pesquisa (ABP).

### Pré-requisitos

- Código funcional de uma Árvore Binária de Pesquisa (classes No e ArvoreBinaria).
- Compreensão dos conceitos de Árvore Bicolor e seus reequilíbrios a partir de fragmentações e rotações simples e duplas

### Atividades

#### 1. Renomear a classe ArvoreBinaria

Renomear a classe ArvoreBinaria para ArvoreBicolor para explicitar a estrutura de dados com a qual está trabalhando.

#### 2. Adicionar atributo cor

Alterar a classe No para incluir um novo atributo chamado “cor” do tipo booleano que indica se o nó é colorido ou não.

#### 3. Criar método para detectar 4-nó

Criar um método “boolean isNoTipo4()” na classe No que verifica e retorna verdadeiro se o nó é do tipo 4, isto é, se é um 4-nó.

#### 4. Criar método para fragmentar um nó

Criar um método “void fragmentar(No i)” na classe ArvoreBicolor que realiza um reequilíbrio do tipo fragmentação no nó, isto é, que seria equivalente à fragmentação na Árvore 2-3-4 correspondente.

#### 5. Criar métodos para os casos de balanceamento por rotação

Criar, na classe ArvoreBicolor, os métodos para os casos de balanceamento:

- Rotação simples à esquerda
- Rotação simples à direita
- Rotação dupla direita-esquerda
- Rotação dupla esquerda-direita

Cada método deve receber um ponteiro para o nó a ser rotacionado, aplicar a rotação e retornar um ponteiro para o nó que é a nova raiz do conjunto após a rotação.

#### 6. Criar método para balancear um nó

Criar um método “void balancear(No bisavo, No avo, No pai, No i)” na classe ArvoreBicolor que recebe os ponteiros para o nó, seu pai, seu avô e seu bisavô e efetua um reequilíbrio do tipo balanceamento por rotação. O balanceamento é feito com o nó, o pai e o avô. Ao final, a nova raiz do conjunto é conectada como filho do bisavô, daí a necessidade de passar o ponteiro para o bisavô como parâmetro.

Lembre-se que 4 tipos de rotação podem ser executados, de acordo com o alinhamento entre avô, pai e o nó:

- a. Se  $\text{pai} > \text{avô} & \text{nó} > \text{pai}$ , então aplica-se uma rotação simples à esquerda
- b. Se  $\text{pai} > \text{avô} & \text{nó} < \text{pai}$ , então aplica-se uma rotação dupla direita-esquerda
- c. Se  $\text{pai} < \text{avô} & \text{nó} < \text{pai}$ , então aplica-se uma rotação simples à direita
- d. Se  $\text{pai} < \text{avô} & \text{nó} > \text{pai}$ , então aplica-se uma rotação dupla esquerda-direita

Após a rotação, o novo nó raiz do conjunto passa a ser filho do nó bisavô em substituição ao avô.

Em qual situação o bisavô não irá existir ( $\text{bisavô} == \text{null}$ )? Faça o tratamento deste caso.

Por fim, lembre-se que, ao final do balanceamento, a nova raiz do conjunto rotacionado deve ser não colorida e, seus filhos, coloridos.

## 7. Implementar método de inserção

Será necessário implementar o processo de inserção com todas as regras de reequilíbrios da Árvore Bicolor. Sugere-se uma implementação completamente nova em relação aos métodos de inserção existentes na antiga classe ArvoreBinaria.

Criar um método “`void inserir(int x)`” na classe ArvoreBicolor, que insere a chave  $x$  na árvore, da seguinte maneira: se a árvore estiver vazia, inserir na raiz; caso contrário, chamar o método de inserção recursivo.

## 8. Implementar método recursivo de inserção

Criar um método recursivo “`void inserir(int x, No bisavo, No avo, No pai, No i)`” na classe ArvoreBicolor. Esse método deve realizar a localização do ponto de inserção da chave  $x$  e orquestrar os reequilíbrios necessários na árvore. Lembre-se da sequência da inserção:

- a. Se chegar à posição de inserção, inserir como uma folha colorida.
- b. Caso contrário
  - i. Se for um 4-nó, fragmentar
  - ii. Chamar recursivamente a inserção na subárvore da esquerda ou da direita

Lembre-se dos momentos em que temos que verificar: “o pai é colorido?”

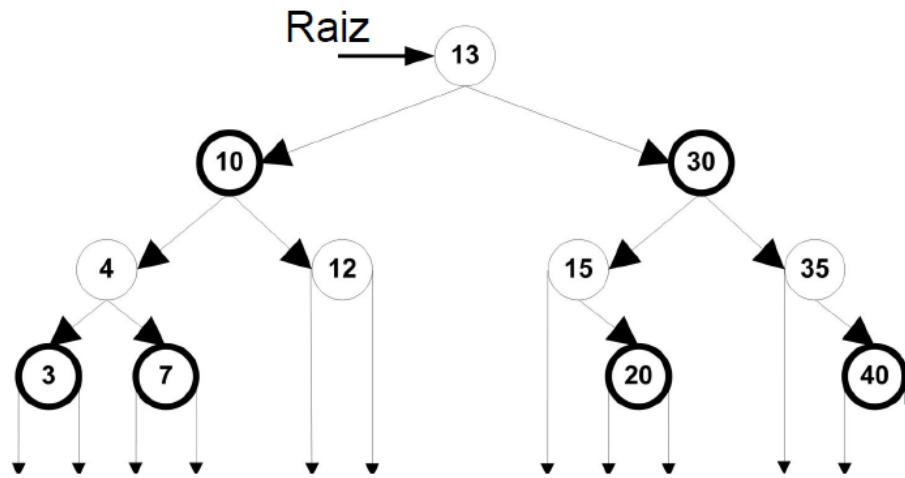
## 9. Validar a Árvore Bicolor

Criar um programa para validar a sua Árvore Bicolor. Seu programa deve inserir elementos na árvore e verificar se os reequilíbrios estão sendo aplicados corretamente. Para isso, seu programa deve:

- a. Criar uma árvore vazia.
- b. Inserir os elementos na árvore conforme exemplo a seguir.
- c. Caminhar na árvore e, em cada visita, imprimir o elemento visitado juntamente com sua cor.

Use o exemplo a seguir para verificação.

Ordem de inserção: 4, 35, 10, 13, 3, 30, 15, 12, 7, 40 e 20



Para essa árvore, um caminhamento central deverá imprimir:

3(cor=1) 4(cor=0) 7(cor=1) 10(cor=1) 12(cor=0) 13(cor=0) 15(cor=1) 20(cor=1) 30(cor=1) 35(cor=0) 40(cor=1)

### Parte Extra

Nosso material de aula apresenta uma simplificação da inserção na árvore bicolor até o 3º elemento. Alterar o método de inserção da classe ArvoreBicolor para que utilize essa abordagem.