Universidade Federal da Bahia - UFBA
Instituto de Matemática - IM
Departamento de Ciência da Computação - DCC
Curso de Bacharelado em Engenharia de Automação e Controle

MATA40 - Estrutura de Dados | Período: 2015.2 | Data: 28/04/2016.
Prof. Antonio L. Apolinário Junior
Estagiário Docente: Alan Santos

Roteiro do Laboratório 6 – Árvore Binária de Busca

Objetivos:

- Compreender de forma prática o conceito de **Árvore Binária de Busca** (**ABB**).
- Implementar, em linguagem C uma Árvore binária de acordo com os conceitos apresentados em sala de aula.

Conceitos básicos:

Uma Árvore Binária de Busca é uma árvore binaria ordenada, onde para toda raiz de uma árvore/sub-árvore de valor $\mathbf{k_r}$, todos os nós da sub-árvore esquerda são menores que $\mathbf{k_r}$ e todos os nós da sub-árvore direita são maiores que $\mathbf{k_r}$.

A representação de um TAD para uma Árvore binária pode ser dada da seguinte forma:

```
typedef struct tNode {
      int data;//Valor inserido em cada nó na árvore
      struct tNode *left;//Nó que aponta para a sub-árvore esquerda
      struct tNode *right;//Nó que aponta para a sub-árvore direita
} TreeNode;
```

Roteiro:

1. Baixe do repositório da disciplina os códigos fonte base para esse Laboratório.

2. Analise a estrutura de arquivos que compõe esse Laboratório. Abra os arquivos e entenda onde esta o programa principal e quais são os módulos utilizados.

3. Compile os programas executando na linha do console o comando **`make`**. Após compilar e verificar que nenhum erro foi gerado, execute o programa através do console usando o comando **`./ABB`** . Verifique o que foi impresso no console e siga para o próximo passo.

4. Codifique a função **`clearTree`** para limpar todos os nós da árvore. Em seguida execute o passo 3 novamente.

5. Codifique a função **`createNode`** para criar um novo nó para a árvore. Execute o passo 3 novamente e verifique se nenhum erro ocorreu.

6. Codifique a função **`insertNode`** para inserir um novo valor no nó correto. A inserção consiste em criar um novo nó e inserir o valor neste. Execute o passo 3 novamente para verificar se os nós foram criados com sucesso.

7. Codifique a função **`findNode`** para buscar um valor na árvore. Execute o passo 3 novamente para verificar se o nó foi encontrado corretamente. Faça com que o algoritmo sorteie e busque 5 valores do vetor V na árvore binária. Agora faça a mesma coisa, só que gerando valores aleatórios.

8. Codifique os 3 tipos de percursos em árvores vistos em sala de aula, **`preOrderVisit`**, **`inOrderVisit`** e **`posOrderVisit`**. Para o controle da função, imprima o valor de cada nó visitado no console. Execute o passo 3 e verifique se os nós foram impressos corretamente.

9. Codifique a função **`removeNode`** e verifique se os nós estão sendo removidos e reorganizados corretamente. Execute o passo 3 para verificar se a função está correta.

Desafio para o final de semana

Codifique as seguintes funções para a árvore binária:

- Implementar funções que retornem o menor e o maior valor armazenado na árvore.

- Implementar uma função que remova todos os nós que possuem valor maior que **n**, sendo **n** um parâmetro da função.

- Codificar uma função que remova todos os nós que possuem o seu campo dado no intervalo entre **a** e **b**. Ex: dado o conjunto de nós {1, 2, 3, 4, 5, 6} e o intervalo a = 2 e b = 4, só devem permanecer na árvore os nós {1, 5, 6}.