Árvores Binárias

Criação, Remoção, Busca e Inserção

1 Estrutura de Dados

Construa uma estrutura de dados para representar uma árvore binária. A estrutura deve conter os seguintes campos:

- Ponteiro para o nó esquerdo
- Ponteiro para o nó direito
- Dado armazenado no nó

2 Criação de uma Árvore Binária

Algorithm 1: Implementação de uma árvore binária

Input: Chave c para o nó a ser inserido (criarNo(c))
Output: Ponteiro T para a raiz da árvore
Alocar espaço para T;
if *T is not NULL* **then**
    T.esquerdo ← NULL;
    T.direito ← NULL;
    T.dado ← c;
    return T;
end

3 Remoção de uma Árvore Binária

Algorithm 2: Remover de uma árvore binária T (Remover(T))

Input: Ponteiro T para a raiz da árvore
Output: Ponteiro T para a raiz da árvore
if *T is not NULL* **then**
    Remover(T.esquerdo);
    Remover(T.direito);
    Liberar(T);
end

4 Busca em uma Árvore Binária

Algorithm 3: Busca em uma árvore binária T - Buscar(c)

Input: Chave c para o nó a ser buscado
Output: Ponteiro T para o nó buscado
if T *is NULL* **then**
  return criarNo(c);
end
if $T.dado = c$ **then**
  return T;
end
if $c < T.dado$ **then**
  return Busca(T.esquerdo, c);
end
if $c > T.dado$ **then**
  return Busca(T.direito, c);
end

5 Inserção em uma Árvore Binária

Algorithm 4: Inserir em uma árvore binária T - Inserir(c)

Input: Chave c para o nó a ser inserido
Output: Ponteiro T para a raiz da árvore
if T *is NULL* **then**
  return criarNo(c);
end
if $c < T.dado$ **then**
  T.esquerdo $\leftarrow$ Inserir(T.esquerdo, c);
end
if $c > T.dado$ **then**
  T.direito $\leftarrow$ Inserir(T.direito, c);
end
if $c = T.dado$ **then**
  return T;
end

6 Exercícios

1. Implemente os algoritmos abaixo:

   a) Implemente a estrutura de dados para a árvore binária de busca. Por exemplo, em C:

   Listing 1: Estrutura de um nó da árvore binária de busca

```c
typedef struct No {
    int dado;
    struct No* esquerdo;
    struct No* direito;
} No;
```

   b) Implemente as funções de criação, remoção, busca e inserção.

   c) Teste as funções com diferentes casos de teste.

   d) Crie um algoritmo para percorrer a árvore em ordem (in-ordem).

   e) Crie um algoritmo para percorrer a árvore em pré-ordem (pre-ordem).

   f) Crie um algoritmo para percorrer a árvore em pós-ordem (pos-ordem).

   g) Crie um algoritmo para calcular a altura da árvore.

   h) Crie um algoritmo para calcular a profundidade de um nó na árvore.

   i) Crie um algoritmo para calcular a soma dos valores armazenados na árvore.

   j) Crie um algoritmo para calcular o nível de um nó na árvore.

   k) Crie um algoritmo para calcular o número de nós na árvore.

   l) Crie um algoritmo para calcular o número de folhas na árvore.

2. Modifique a estrutura para guardar uma dados dos alunos com os seguintes campos:

   - Nome
   - Matrícula
   - Nota

   a) Implemente a estrutura de dados para a árvore binária de busca.

   Listing 2: Estrutura de um nó da árvore binária de busca

```c
typedef struct Aluno {
    char nome[50];
    int matricula;
    float nota;
    struct Aluno* esquerdo;
    struct Aluno* direito;
} Aluno;
```

3. Implemente um algoritmo para buscar um aluno pelo nome.

4. Crie um algoritmo para calcular a média das notas dos alunos.