BOA PROVA — Estruturas de Dados

Instruções

1. Consideraremos apenas as respostas contidas nas folhas em branco.
2. Letras grandes e legíveis.
3. A interpretação faz parte do processo.
4. Cada questão sem atenção será desconsiderada.
5. A prova vale 10 pontos e as notas das últimas avaliações também contam.

1. Implementação

Implemente o método `insere()` na estrutura de dados descrita abaixo.
A estrutura possui três níveis: T1, T2 e T3.

- **T1** é um vetor de tamanho fixo.
- Cada célula de **T1** possui um número inteiro que referencia uma posição na tabela **T2**.
- **T2** é uma tabela hash de tamanho definido por `hashT2(int)`.
- Cada posição de **T2** é uma lista encadeada.

Cada nó de **T2** possui:

- Uma chave
- Um ponteiro para uma lista **T3**
- Um ponteiro para a esquerda
- Um ponteiro para a direita

A função `hashT3` é utilizada da seguinte forma:

- Se T2 não estiver vazia, inserimos na tabela hash T3 somente o elemento acessado.
- Se não existir T3, será inserida uma lista inteira em T3.
- Caso ocorra colisão em T3, deve-se utilizar uma **árvore AVL**.

Utilize a classe `Celula1`. O atributo `CelulaT1[]` é responsável por armazenar T1.
As classes:

- `CelulaT2`

- `CelulaT3`

possuem atributos próprios.

A classe `HashT2` possui como atributo:

`int[] hash`

Considere também:

- `Celula primeiroT3`
- `Celula ultimaT3`

No `hashT1`, a classe principal é `Celula1`. No `hashT2`, a classe principal é `CelulaT2`.
Implemente o método `insere()` para essa estrutura completa.

2. Árvores

Dada a sequência de números:

$$4, 30, 8, 12, 2, 18, 14, 16, 22, 10$$

a) Ilustre o passo a passo da inserção dessa sequência em uma **Árvore AVL**, mostrando os balanceamentos.

b) Após isso, insira os valores em uma **Árvore B**, em ordem crescente, como se estivesse em disco.

3. Trie

Implemente as classes:

- `Trie`
- `StringTrie`

Implemente o método:

`insereTrie(String s)`

Esse método deve inserir palavras na estrutura **Trie**.
Crie também o método:

`insereStringT3`

Esse método deve verificar se a string de uma árvore **não foi previamente inserida** em memória antes de inseri-la em outra árvore.

Foque na implementação correta dos métodos de inserção e verificação de duplicidade.

4. Estrutura do nó da Trie

O nó da árvore Trie deve conter uma estrutura que armazene os caracteres do alfabeto.

Faça:

- A definição da estrutura do nó
- Método para **inserir**
- Método para **remover**
- Método para **buscar**

Em seguida, explique como a estrutura pode ser representada em formato tridimensional (conceito estrutural).

Por fim, implemente o método de inserção baseado no modelo visto em sala:

```
void inserir(String s, No no, int i) {
    if (no.prox[s.charAt(i)] == null) {
        no.prox[s.charAt(i)] = new No(s.charAt(i));
        if (i == s.length() - 1) {
            no.prox[s.charAt(i)].folha = true;
        } else {
            inserir(s, no.prox[s.charAt(i)], i + 1);
        }
    }
    else if (no.prox[s.charAt(i)].folha == false
             && i < s.length() - 1) {
        inserir(s, no.prox[s.charAt(i)], i + 1);
    }
}
```