Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática (ICEI)
Curso de Ciência da Computação
Disciplina: Algoritmos e Estruturas de Dados II

Prova III :

Aluno: _____

1. Considere as classes abaixo que representam uma Árvore Binária. Você deve implementar o método **boolean isMax(double valor)** na classe Arvore, que verifica se a árvore tem altura de, no máximo, o valor passado como parâmetro multiplicado pelo Log2 da quantidade de nós. Por exemplo, se o usuário passar como parâmetro o valor 1.4, o método retorna TRUE se a altura da árvore for, no máximo, 1.4*Log2(quantidadeNós). NÃO estão implementados métodos para obter altura ou quantidade de nós. Faça a análise de complexidade da sua solução.

```
class No {
  public int elemento;
  public No esq, dir;
}
```

```
public class Arvore {
    private No raiz;
    public boolean isMax(double valor) {
        // Implemente o método aqui
    }
}
```

2. Provar ou refutar cada uma das afirmações a seguir, apresentando uma explicação detalhada do motivo pelo qual a afirmação é verdadeira ou falsa. Mesmo que você acredite que a afirmação esteja correta, é necessário justificar com clareza seu raciocínio.

   (a) Gabriel inseriu os seguintes números, nessa ordem, em uma árvore AVL: 3, 13, 17, 23, 7, 9, 21, 25, 2. O quinto elemento da árvore a ser visitado, quando é realizada uma busca em pré-ordem, é o número 9.

   (b) Uma rotação dupla direita-esquerda é usada em subárvores onde o pai está desbalanceado para a esquerda e o filho para a direita.

   (c) A inserção dos números 2, 10, 4, 6, 1, 9, 7, 5, 3, 11, 8 em uma árvore 2.3.4 com fragmentação por ascensão cria uma árvore idêntica à da fragmentação na descida.

   (d) A inserção dos números 2, 10, 4, 6, 1, 9, 7, 5, 3, 11, 8 em uma árvore alvinegra causa duas rotações.

3. Considere uma estrutura de dados *Doidona* ilustrada abaixo. O primeiro nível dessa estrutura contém uma árvore binária composta por exatamente 26 nós cujos valores são os caracteres do alfabeto, que irá indexar as palavras pelo primeiro caracter. Para o desenho foram apresentados apenas alguns caracteres, mas considere que está toda preenchida. Cada nó dessa árvore tem uma referência para uma tabela *hash* $T1$, ou seja, temos 26 tabelas $T1$. Todas possuem tamanho $tam1$. Na figura, por questões de espaço, representamos apenas a $T1$ dos nós rotulados com os caracterse H, U e Y. A $T1$ de cada nó distribui seus elementos com uma função que recebe o último caracter da palavra com assinatura *int hashT1(char x)* que retorna *(x % tam1)* e trata as colisões com uma função *int rehashT1(char x)* que retorna *(++x % tam1)*. Quando a função *rehash* não consegue tratar as colisões, a $T1$ de cada nó trata essas colisões com a tabela *hash* $T2$ (logicamente implementada) como área de reserva e tamanho $tam2$. A tabela *hash* $T2$ possui em cada posição uma lista de Strings. A função *int hashT2(int x)*, que recebe a quantidade de caracteres da palavra, retorna *(x % tam2)* que é o índice da T2 com a lista onde deve ser inserida a palavra. O método de inserção mantém as palavras ordenadas nas Listas de T2. Você deve implementar o método boolean pesquisar(String nome). Sabendo que existe um ponteiro No raiz para a raiz da árvore, considere os atributos das classes:

```
public class Doidona {
    private No raiz;
}

public class No {
    public char caracter;
    public T1 t1;
    public No esq, dir;
}
```

```
public class T1 {
    public String palavras[];
    public T2 t2;
}

public class T2 {
    public CelulaT2 celulast2[];
}
```

```
public class CelulaT2 {
    public Celula inicio;
    public Celula fim;
}

public class Celula {
    public String palavra;
    public Celula prox;
}
```