

# Prova III - Teoria

**Entrega** 20 jun em 10:30      **Pontos** 10      **Perguntas** 32  
**Disponível** 20 jun em 8:30 - 20 jun em 10:30 2 horas  
**Limite de tempo** 120 Minutos

## Instruções

Nossa 3a prova de AEDs II tem duas partes: teórica e prática. Cada uma vale 10 pontos. A prova teórica será realizada no Canvas e a prática, no Verde.

A prova teórica será realizada no dia 20 de junho, no horário da aula teórica. Essa prova tem 30 questões de verdadeiro ou falso e duas questões abertas de código. As questões de verdadeiro ou falso valem 0,1 e as abertas, 3.5. Abaixo, seguem as regras para a prova.

- 1) O código de acesso à prova será fornecido pelo professor no início da prova.
- 2) Após o envio de uma questão não é permitido que o aluno volte na mesma.
- 3) A prova é individual e é permitida a consulta à cola que contém o nome do aluno.
- 4) A interpretação faz parte da prova.
- 5) Se existir algum erro, após a divulgação do gabarito, peça a anulação da questão.
- 6) Os alunos da manhã 1/manhã farão a prova nos lab 9.
- 7) Os alunos da turma 2/manhã farão a prova nos lab 2.
- 8) Os alunos da tarde farão a prova no lab 10.

Desejamos uma excelente prova para todos.

Este teste foi travado 20 jun em 10:30.

## Histórico de tentativas

	Tentativa	Tempo	Pontuação
MAIS RECENTE	<a href="#">Tentativa 1</a>	83 minutos	1,9 de 10

Pontuação deste teste: **1,9** de 10

Enviado 20 jun em 10:21

Esta tentativa levou 83 minutos.

### Pergunta 1

0 / 0,1 pts

Considere uma tabela *hash* contendo um *array* de árvores binárias de pesquisa. O melhor caso acontece com custo  $\Theta(1)$  e o pior, com o

custo  $\Theta(n)$ .

Você respondeu

☒ Falso

Resposta correta

☐ Verdadeiro

A afirmação é verdadeira. O melhor caso acontece quando o elemento desejado está na raiz da árvore representando sua posição. O pior caso acontece quando todos os elementos inseridos possuem o mesmo valor na função de transformação e eles foram inseridos de tal forma que cada nível da árvore tem um elemento (e.g., inseridos em ordem crescente). Além disso, nesse caso, o elemento desejado encontra-se na folha da árvore ou a pesquisa pelo mesmo passa por essa folha.

## Pergunta 2

0,1 / 0,1 pts

Os métodos *hashing* envolvem um processo de transformação de uma chave em um endereço. Sobre esses métodos, podemos afirmar que o tempo gasto com pesquisas depende do tamanho da tabela e da quantidade de elementos inseridos. Além disso, a função *hash* de transformação deve envolver uma operação simples sobre a chave (TCE-RS'14, adaptada).

Correto!

☒ Verdadeiro

☐ Falso

A afirmação é verdadeira. Primeiro, a dependência do tamanho da tabela e da quantidade de elementos acontece porque tais fatores influenciam nas colisões e o quão maior o número de colisões, maior o custo para encontrar elementos. Segundo, é importante que a operação de transformação seja simples, evitando o *overhead* de processamento.

### Pergunta 3

0 / 0,1 pts

Considere um arquivo sequencial, com 10.000 registros, cujas chaves identificadoras são números inteiros de até 8 dígitos. Para criar um índice tipo *hashing* para esse arquivo, contendo endereços de 0 até 11.999, uma definição adequada para uma função de *hashing*  $f(x)$ , onde  $x$  é uma chave e  $(a \bmod b)$  é o resto da divisão de  $a$  por  $b$ , seria  $f = x \bmod 12000$  (DPE-RJ'14, adaptado).

Resposta correta

☐ Verdadeiro

Você respondeu

☒ Falso

A afirmação é verdadeira. Uma função de transformação tradicional consiste no resto da divisão inteira do elemento pelo tamanho da tabela.

### Pergunta 4

0,1 / 0,1 pts

No método de transformação (*hashing*), os registros armazenados em uma tabela são diretamente endereçados a partir de uma transformação aritmética sobre a chave de pesquisa. Com relação às funções de transformação e colisões, podemos afirmar que uma função de transformação deve mapear as chaves em números

inteiros, dentro de um intervalo da forma  $[0, m - 1]$ , em que  $m$  representa o valor da chave (TRE-PI'16, adaptado).

Correto!

☒ Falso

☐ Verdadeiro

A afirmação é falsa, pois  $m$  representa o tamanho da tabela.

### Pergunta 5

0,1 / 0,1 pts

Considere uma tabela *hash* com *rehash* contendo duas posições e usando as funções de transformação apresentadas abaixo, podemos afirmar que após a inserção dos elementos 1, 4, 5, 2, e 7, a tabela resultante será  $[4 \quad 1]$ .

```
int hash(int key){  
    return key%2;  
}
```

```
int rehash(int key){  
    return ++key%2;  
}
```

Correto!

☒ Verdadeiro

☐ Falso

A afirmação é verdadeira, pois o 5, 2 e 7 não serão inseridos.

**Pergunta 6****0 / 0,1 pts**

Altura de uma árvore 2.3.4 só aumenta quando existe uma fragmentação da raiz da árvore.

Resposta correta

☐ Verdadeiro

Você respondeu

☒ Falso

A afirmação é verdadeira. Nas demais fragmentações, o elemento do meio "sobe" para seu pai. No caso da fragmentação da raiz, como não existe um pai, esse é criado.

**Pergunta 7****0,1 / 0,1 pts**

Dadas duas árvores alvinegras A e B obtidas a partir de uma árvore 2.3.4, A e B possuem os mesmos números de nós e arestas.

☐ Falso**Correto!**☒ Verdadeiro

A afirmação é verdadeira. O número de nós na alvinegra é igual ao número de elementos existentes na 2.3.4. O número de arestas da alvinegra é igual à soma entre o número de arestas da 2.3.4 e o de relacionamento entre gêmeos da 2.3.4.

**Pergunta 8****0,1 / 0,1 pts**

A inserção dos números 4, 24, 10, 16, 2, 22, 18, 14, 8, 26, 20, 12 e 6 em uma 2.3.4 usando as técnicas de fragmentação por ascensão e na descida gera a mesma árvore resultante.

**Correto!**☒ Verdadeiro☐ Falso

A afirmação é verdadeira. A inserção usando as duas técnicas é igual até a inserção do 20. Elas ficam diferentes após a inserção do 12 e voltam a ficar iguais com a inserção do 6.

**Pergunta 9****0 / 0,1 pts**

Na árvore 2.3.4, após uma inserção utilizando fragmentação na descida, podemos garantir a inexistência de nós do tipo 4 no caminho entre a raiz e a folha em que aconteceu a inserção.

**Você respondeu**☒ Verdadeiro**Resposta correta**☐ Falso

A afirmação é falsa. Na inserção com fragmentação na descida, quando chegamos em um nó do tipo 4, esse é fragmentado. Nessa fragmentação, se o pai era do tipo 3, ele ficará sendo do tipo 4. Da mesma forma, se inserirmos em uma folha do tipo 3, essa ficará sendo do tipo 4.

**Pergunta 10****0 / 0,1 pts**

A técnica de inserção com fragmentação na descida na 2.3.4 pode aumentar, reduzir ou manter o número de nós do tipo 4.

Você respondeu

☒ Falso

Resposta correta

☐ Verdadeiro

A afirmação é verdadeira. Na inserção com fragmentação na descida, quando temos um nó do tipo 4 esse será fragmentado o que pode levar a uma redução do número de nós do tipo 4. Por outro lado, quando inserimos o elemento em uma folha do tipo 3, essa vira do tipo 4, aumentando o número de nós desse tipo. Temos ainda que durante a inserção podemos não ter fragmentações, mantendo o número de nós do tipo 4 existentes na árvore. Esse número também é mantido quando para cada Fragmentação, temos um nó do tipo 3 que vira do tipo 4.

### Pergunta 11

0,1 / 0,1 pts

Em uma árvore alvinegra, se tivermos dois nós pretos seguidos, temos que fazer uma rotação com o avô.

Correto!

☒ Verdadeiro

☐ Falso

A afirmação é verdadeira conforme os conceitos de árvores alvinegras apresentados na sala de aula.

### Pergunta 12

0,1 / 0,1 pts

Uma das estruturas de dados utilizadas na modelagem de sistemas de software denomina-se árvore rubro-negra. Nesse caso, um nó será vermelho quando, na árvore 2-3-4 correspondente, o valor desse nó for gêmeo do nó pai. Caso contrário, o nó é preto. Em uma árvore rubro-negra temos que a quantidade de nós vermelhos é sempre par.

☐ Verdadeiro

Correto!

☒ Falso

A afirmação é falsa conforme os conceitos de árvores alvinegras apresentados na sala de aula.

### Pergunta 13

0,1 / 0,1 pts

Em uma árvore alvinegra, após uma rotação, o NÓ central será branco e, seus, filhos, pretos.

☐ Falso

Correto!

☒ Verdadeiro

A afirmação é verdadeira conforme os conceitos de árvores alvinegras apresentados na sala de aula.

### Pergunta 14

0 / 0,1 pts

Uma árvore rubro-negra possui 18 valores inteiros distintos armazenados em seus 18 nós. A função recursiva boolean busca (int val) visita os nós desse tipo de árvore à procura de um determinado valor (val). O algoritmo utilizado tira partido das características de uma



árvore rubro-negra, com o objetivo de ser o mais eficiente possível. Podemos afirmar que o número máximo de chamadas da função que será necessário para informar se um determinado valor está, ou não, armazenado na árvore é igual a seis (BNDES'13, adaptada).

Você respondeu

☒ Falso

Resposta correta

☐ Verdadeiro

A afirmação é verdadeira e pode ser confirmada executando o código disponibilizado pelo professor.

**Pergunta 15**

0,1 / 0,1 pts

Uma das estruturas de dados utilizadas na modelagem de sistemas de software denomina-se árvore rubro-negra. Nesse caso, um nó será vermelho quando, na árvore 2-3-4 correspondente, o valor desse nó for gêmeo do nó pai. Caso contrário, o nó é preto. Em uma árvore rubro-negra temos que se um nó é vermelho, seus filhos são vermelhos.

Correto!

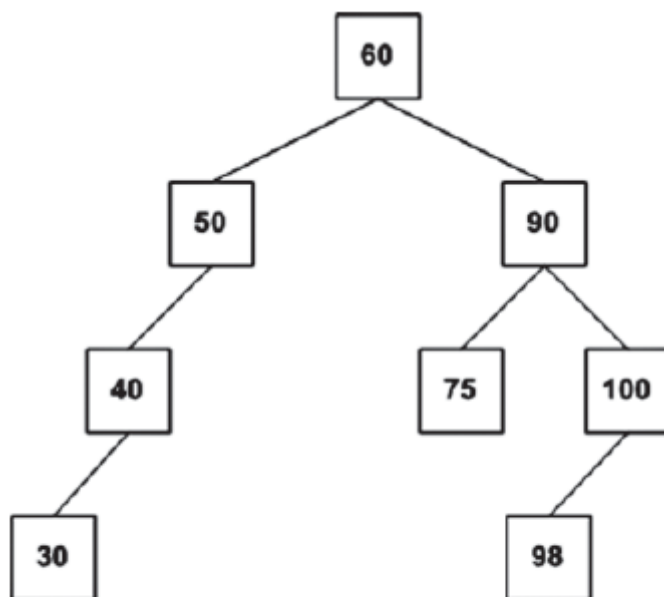
☒ Falso☐ Verdadeiro

A afirmação é falsa conforme os conceitos de árvores alvinegras apresentados na sala de aula.

**Pergunta 16**

0,1 / 0,1 pts

A estrutura de dados AVL é uma árvore binária de busca balanceada criada pelos soviéticos Adelson, Velsky e Landis em 1962. Podemos afirmar que a figura abaixo representa uma árvore AVL (PETROBRAS'12, adaptada).



Correto!

☒ Falso

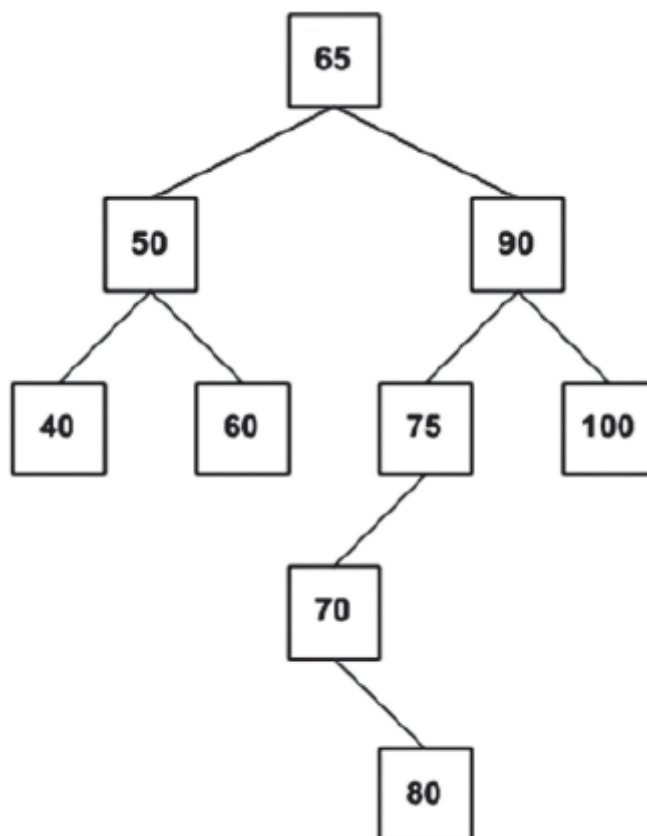
☐ Verdadeiro

A afirmação é falsa conforme o conceito de árvore AVL.

### Pergunta 17

0,1 / 0,1 pts

A estrutura de dados AVL é uma árvore binária de busca balanceada criada pelos soviéticos Adelson, Velsky e Landis em 1962. Podemos afirmar que a figura abaixo representa uma árvore AVL (PETROBRAS'12, adaptada).



☐ Verdadeiro

☒ Falso

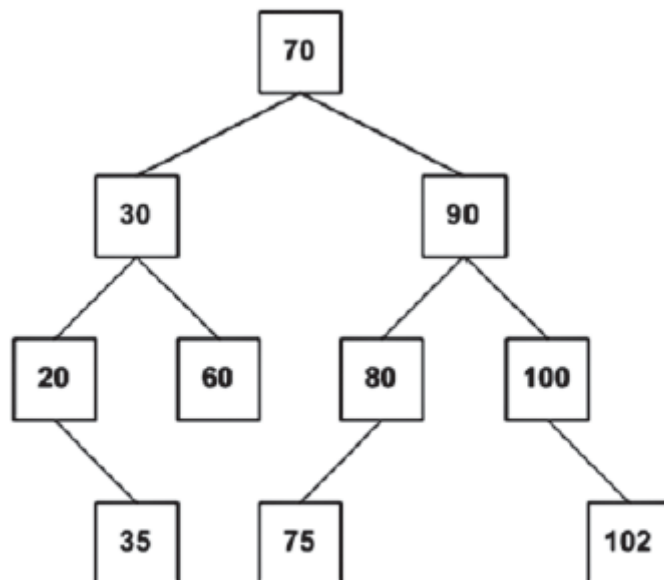
Correto!

A afirmação é falsa conforme o conceito de árvore AVL.

### Pergunta 18

0 / 0,1 pts

A estrutura de dados AVL é uma árvore binária de busca balanceada criada pelos soviéticos Adelson, Velsky e Landis em 1962. Podemos afirmar que a figura abaixo representa uma árvore AVL (PETROBRAS'12, adaptada).



Você respondeu

☒ Verdadeiro

Resposta correta

☐ Falso

A afirmação é falsa dado que a árvore em questão não é de busca (pesquisa). Veja que o 35 está à esquerda do 30.

**Pergunta 19**

0,1 / 0,1 pts

A AVL é uma árvore de busca autobalanceada. Isso significa que pode possuir até duas raízes (SEFAZ-PI'15, adaptado).

Correto!

☒ Falso☐ Verdadeiro

A afirmação é falsa conforme o conceito de árvore AVL.

### Pergunta 20

0 / 0,1 pts

Considere a assinatura das duas funções abaixo na linguagem C++:

```
void troca(double &x, double &y)
```

```
void troca(double* x, double* y)
```

Podemos afirmar que a primeira função utiliza passagem de parâmetros por referência e a segunda, por valor.

Resposta correta

☐ Verdadeiro

Você respondeu

☒ Falso

A afirmação é verdadeira considerando os conceitos sobre passagem de parâmetros na linguagem C++ apresentados na sala de aula.

### Pergunta 21

0,1 / 0,1 pts

Podemos afirmar que o código abaixo em C++ imprime "1 1" como saída

```
void f (int val, int& ref) {  
    val ++; ref++;  
}
```

```
void main () {  
    int i = 1, j = 1;  
    f(i, j);  
}
```

```
printf("%i -- %i", i, j);  
}
```

**Correto!**☒ Falso☐ Verdadeiro

A afirmação é falsa. A variável ref foi passada por referência, assim, seu valor final é dois.

## Pergunta 22

0 / 0,1 pts

Avaliando os códigos abaixo nas linguagens C e C++, é correto afirmar que os dois imprimem os mesmos valores para as variáveis "x" e "y" no final de suas execuções. Isso acontece mesmo com a passagem de parâmetros do primeiro código sendo por valor e a do segundo, sendo por referência.

### Código em C:

```
void troca(double *x, double *y){  
    double aux = *x;  
    *x = *y;  
    *y = aux;  
}  
  
int main(){  
    double x = 7.0;  
    double y = 5.0;  
    troca(&x, &y);  
    printf("X: %lf Y: %lf", x, y);  
}
```

### Código em C++:

```
void troca(double& x, double& y){  
    double aux = x;  
    x = y;  
    y = x;  
}
```

```
int main(){
    double x = 7.0;
    double y = 5.0;
    troca(x, y);
    cout << "X: " << x;
    cout << "Y: " << y;
}
```

Você respondeu

☒ Falso

Resposta correta

☐ Verdadeiro

A afirmação é verdadeira conforme discutido em sala de aula.

**Pergunta 23**

0 / 0,1 pts

A altura mínima de uma árvore binária com  $n$  nós é  $\lg(n)$ .

Resposta correta

☐ Falso

Você respondeu

☒ Verdadeiro

A afirmação é falsa a altura mínima de uma árvore binária com  $n$  nós é  $\lfloor \lg(n) \rfloor + 1$ .

**Pergunta 24**

0,1 / 0,1 pts

As rotações são utilizadas no balanceamento das árvores de tal forma que uma árvore desbalanceada para a esquerda será rotacionada para à direita. Uma desbalanceada para a direita será para à esquerda. Nós

temos quatro tipos de rotações: simples para esquerda, dupla esquerda-direita, simples para à direita e a dupla direita-esquerda. As duas primeiras são efetuadas em árvores desbalanceadas para à esquerda e as outras duas, desbalanceadas para à direita.

☐ Verdadeiro

☒ Falso

Correto!

A afirmação é falsa. As rotações simples para esquerda e a dupla direita-esquerda são efetuadas em árvores desbalanceadas para à esquerda e as outras duas, desbalanceadas para à direita.

### Pergunta 25

0 / 0,1 pts

Considere que a Manausprev armazena os nomes dos beneficiários de aposentadorias em uma Árvore Binária de Busca (ABB). Ao se armazenar, nesta ordem, os nomes Marcos, José, Carolina, Paula, Rui, Pedro e Maria, a ABB resultante tem o nó contendo José possui dois filhos (MANAUSPREV'15, adaptada).

☐ Falso

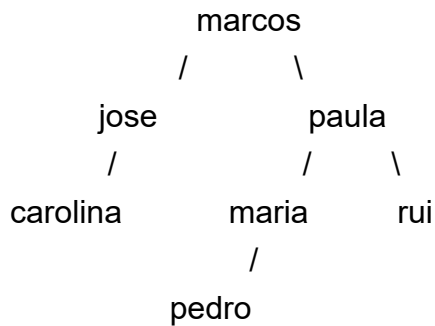
☒ Verdadeiro

Resposta correta

Você respondeu



A afirmação é falsa e a árvore resultante é mostrada abaixo.



### Pergunta 26

0 / 0,1 pts

Considere que a Manausprev armazena os nomes dos beneficiários de aposentadorias em uma Árvore Binária de Busca (ABB). Ao se armazenar, nesta ordem, os nomes Marcos, José, Carolina, Paula, Rui, Pedro e Maria, a ABB resultante tem altura 3, que corresponde à altura mínima para armazenar os 7 nomes (MANAUSPREV'15, adaptada).

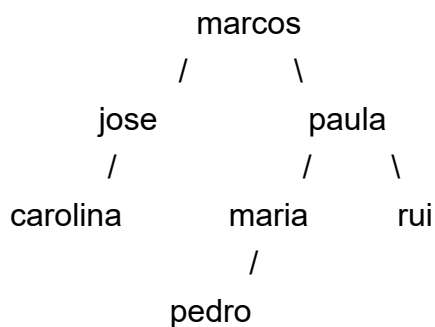
Você respondeu

☒ Verdadeiro

Resposta correta

☐ Falso

A afirmação é falsa e a árvore resultante é mostrada abaixo.



### Pergunta 27

0 / 0,1 pts

O comando Java “No no = new No()” cria um objeto do tipo No e armazena seu endereço na variável no. A mesma coisa acontece em C++ no código “No no = new No()”.

Resposta correta

☐ Falso

Você respondeu

☒ Verdadeiro

O comando C++ apresentado não compila, pois "No no" cria um objeto do tipo No e, não, um ponteiro. Logo, a variável no não armazena um endereço de memória como no comando Java apresentado.

**Pergunta 28**

0,1 / 0,1 pts

Nas linguagens C/C++, acessamos um atributo de um registro/objeto apontado por um ponteiro fazendo ponteiro->atributo.

☐ Falso

Correto!

☒ Verdadeiro

O símbolo "->" permite acessarmos os atributos/funções de um registro/objeto apontados por um ponteiro.

**Pergunta 29**

0,1 / 0,1 pts

A linguagem C++ permite criar funções cuja implementação do código acontece “fora” da classe. Nesse caso, “dentro” da mesma temos a

assinatura da função e, "fora", implementamos a função usando o nome da classe e o operador "::". O exemplo abaixo mostra uma classe Cubo com sua função getVolume implementada de forma externa.

```
class Cubo {  
    public:  
        int lado;  
        int getVolume();  
}  
  
int Cubo :: getVolume() {  
    return lado*lado*lado;  
}
```

☐ Falso

☒ Verdadeiro

Correto!

A afirmação é verdadeira e a implementação externa acontece conforme o exemplo apresentado.

### Pergunta 30

0,1 / 0,1 pts

O código C++ abaixo tem um objeto chamado "quadrado\_teste" do tipo Quadrado que pode ser usado globalmente.

```
class Quadrado {  
    private:  
        int lado;  
    public:  
        void set_values (int);  
  
    //...  
};
```

```
int main() {  
    Quadrado quadrado_teste;  
  
    //...  
    return 0;  
}
```

☐ Verdadeiro☒ Falso**Correto!**

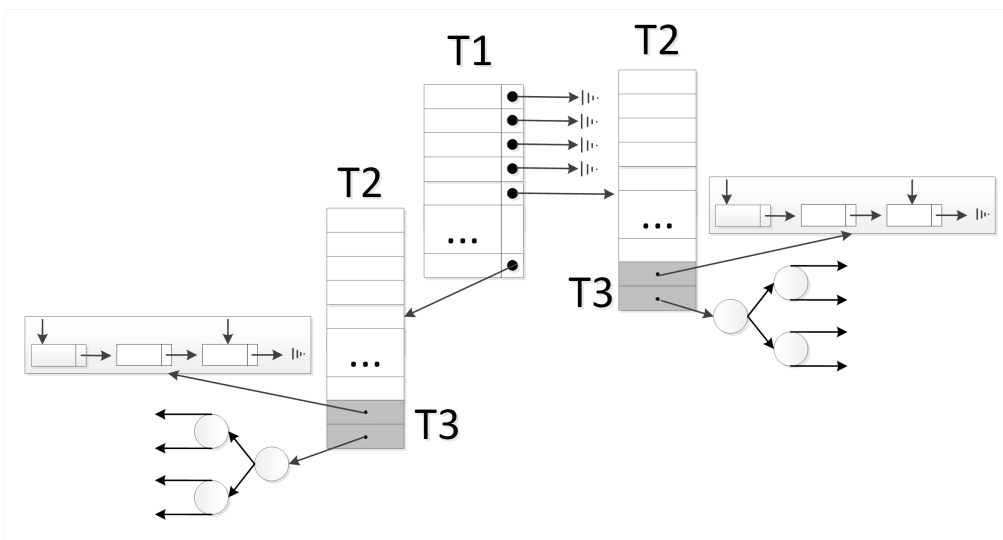
A declaração da classe e a criação do objeto foram feitos de maneira adequada. Entretanto, como o objeto foi declarado dentro da função main, o escopo de vida do mesmo é aquela função a partir do seu ponto de declaração.

**Pergunta 31****0,1 / 3,5 pts**

Seja a árvore Trie apresentada na sala de aula implemente um método para MOSTRAR todas as palavras contidas na árvore. Considere que a classe Nó da árvore Trie contém uma ÁRVORE BINÁRIA SIMPLES para armazenar o endereço dos nós filhos. Apresente a ordem de complexidade do seu método. Sua resposta deve conter somente o método solicitado e sua complexidade.

[!\[\]\(dd161862f9164df98f62b726e9846241\_img.jpg\) mostrar.java](#)[\(https://pucminas.instructure.com/files/6327161/download\)](https://pucminas.instructure.com/files/6327161/download)**Pergunta 32****0,1 / 3,5 pts**

Considere a estrutura de dados Doidona ilustrada abaixo. O primeiro nível dessa estrutura contém uma tabela hash T1 sendo que cada célula dessa tabela possui um número inteiro elemento e um ponteiro T2 para outra tabela hash, T2. Cada célula de T1 aponta para uma T2. O tamanho de T1 é tam1 e de cada T2, tam2. Distribuímos os elemento em T1 com uma função  $\text{int hashT1}(\text{int } x)$  que retorna  $(x \% \text{tamT1})$ . A T2 de cada célula distribui seus elementos com uma função  $\text{int hashT2}(\text{int } x)$  que retorna  $(x \% \text{tamT2})$  e trata as colisões com uma função  $\text{int rehashT2}(\text{int } x)$  que retorna  $(++x \% \text{tam2})$ . Quando a função  $\text{rehashT2}$  não consegue tratar as colisões, a T2 de cada nó trata essas colisões com a tabela hash T3 (logicamente implementada). A função  $\text{int hashT3}(\text{int } x)$  retorna  $(x \% 2)$ . Quando esse retorno é zero, inserimos o valor em uma lista listaT3. Quando um, em uma árvore binária arvT3. Implemente o algoritmo para pesquisar a existência de um elemento cadastrado nessa estrutura. Você N.Ã.O deve usar as estruturas de Árvore Binária nem de Lista, use apenas o Nó da Árvore e a Célula da Lista. A classe Doidona tem o atributo CelulaT1 t1[]. A classe CelulaT1 tem os atributos int elemento e HashT2 t2. A classe HashT2 tem os atributos int[] tab; Celula primeiroT3, ultimoT3; e No raizT3. As classes Celula e nó são iguais as vistas na sala de aula. Faça a análise de complexidade do seu método.



↓ [Pesquisar.java](#)

(<https://pucminas.instructure.com/files/6327410/download>)

Pontuação do teste: **1,9** de 10