# PROVA 3 - NOVO

**Entrega** 10 de jun de 2020 em 12:20 **Pontos** 15,4 **Perguntas** 32
**Disponível** 10 de jun de 2020 em 9:30 - 10 de jun de 2020 em 12:20 aproximadamente 3 horas
**Limite de tempo** 120 Minutos

Este teste não está mais disponível, pois o curso foi concluído.

## Histórico de tentativas

| | Tentativa | Tempo | Pontuação |
|---|---|---|---|
| **MAIS RECENTE** | Tentativa 1 | 97 minutos | 9,78 de 15,4 |

As respostas corretas não estão mais disponíveis.

Pontuação deste teste: **9,78** de 15,4
Enviado 10 de jun de 2020 em 11:10
Esta tentativa levou 97 minutos.

### Pergunta 1

**0,18 / 0,18 pts**

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

- (●) Falso
- ( ) Verdadeiro

A afirmação é falsa. 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, nesse caso, o elemento desejado encontra-se em uma das folhas da árvore ou o mesmo não está contido na árvore.

## Pergunta 2

**0,18 / 0,18 pts**

Uma estrutura de dados eficiente é a tabela *hash*. Um desafio no projeto dessa estrutura é seu dimensionamento. Em uma tabela adequadamente dimensionada, com *n* chaves, o número médio de acessos para localização de uma chave é $\Theta(1)$ comparações (TJ-PI'15, adaptada).

- Falso
- Verdadeiro

Uma tabela *hash* é bem dimensionada quando seu custo é constante, $\Theta(1)$ comparações.

## Pergunta 3

**0,18 / 0,18 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 mod b) é o resto da divisão de a por b, seria f = x mod 12000 (DPE-RJ'14, adaptado).

- Verdadeiro
- Falso

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

## Pergunta 4

**0,18 / 0,18 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 a técnica de endereçamento aberto (*hash* direto) resolve  colisões usando uma matriz esparsa (TRE-PI'16, adaptado).

- Verdadeiro
- Falso

A afirmação é falsa. O *hash* direto consiste em inserir os elementos que sofreram colisões dentro da própria tabela. Por exemplo, podemos utilizar área de reserva ou *overflow* e funções de *rehash*.

## Pergunta 5

**0,18 / 0,18 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).

- Verdadeiro
- Falso

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

## Pergunta 6

**0,18 / 0,18 pts**

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

- (•) Verdadeiro
- ( ) 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

**0,18 / 0,18 pts**

A inserção dos números 2, 12, 5, 8, 1, 11, 9, 7, 4, 13, 10, 6 e 3 em uma 2.3.4 usando as técnicas de fragmentação por ascenção e na descida gera a mesma árvore resultante.

- ( ) Falso
- (•) Verdadeiro

A afirmação é verdadeira. A inserção usando as duas técnicas é igual até a inserção do 10. Elas ficam diferentes após a inserção do 6 e voltam a ficar iguais com a inserção do 3.

## Pergunta 8

**0,18 / 0,18 pts**

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

- (●) Verdadeiro
- ( ) Falso

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 9

**0,18 / 0,18 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 ascenção e na descida gera a mesma árvore resultante.

- (●) 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.

Incorreta

## Pergunta 10

**0 / 0,18 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.

- (●) Verdadeiro
- ( ) 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.

Incorreta

## Pergunta 11

**0 / 0,18 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).

- Falso
- Verdadeiro

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

## Pergunta 12

**0,18 / 0,18 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 o nó raiz é preto.

- Verdadeiro
- Falso

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

Incorreta

## Pergunta 13

**0 / 0,18 pts**

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

- Falso
- Verdadeiro

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

## Pergunta 14

**0,18 / 0,18 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.

- (●) Falso
- ( ) Verdadeiro

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

## Pergunta 15

**0,18 / 0,18 pts**

Em uma árvore alvinegra, quando um nó tem 2 filhos pretos, fazemos a inversão das cores entre os filhos e o pai.

- (●) Verdadeiro
- ( ) Falso

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

Incorreta

## Pergunta 16

**0 / 0,18 pts**

A AVL é uma árvore de busca autobalanceada. Isso significa que as alturas das duas sub-árvores a partir de cada nó são exatamente iguais (SEFAZ-PI'15, adaptado).

- (●) Verdadeiro
- ( ) Falso

A afirmação é falsa conforme o conceito de árvore AVL.

## Pergunta 17

**0,18 / 0,18 pts**

Todos os n nomes de uma lista de assinantes de uma companhia telefônica foram inseridos, em ordem alfabética, em três estruturas de dados: uma árvore binária de busca, uma árvore AVL e uma árvore bicolor. As alturas resultantes das três árvores são $\Theta(n)$, $\Theta(lg(n))$ e $\Theta(lg(n))$, respectivamente (PETROBRAS'12, adaptada)

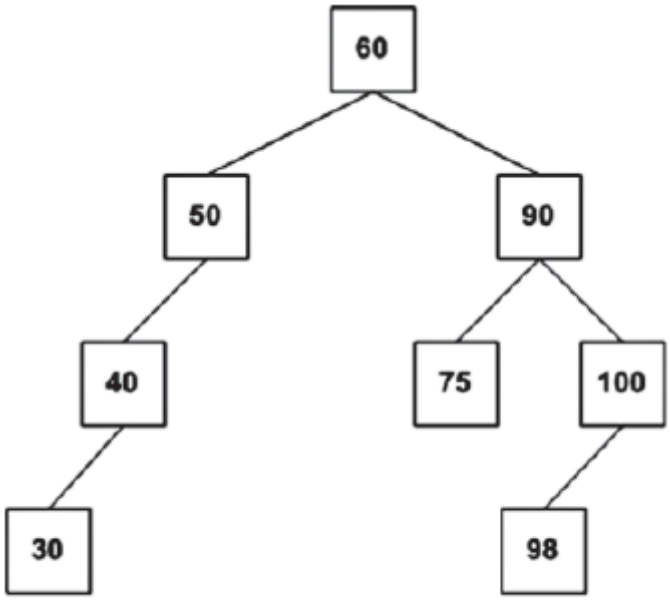
- (●) Verdadeiro
- ( ) Falso

A afirmação é verdadeira. A AB não é balanceada e as demais são.

## Pergunta 18

**0,18 / 0,18 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).



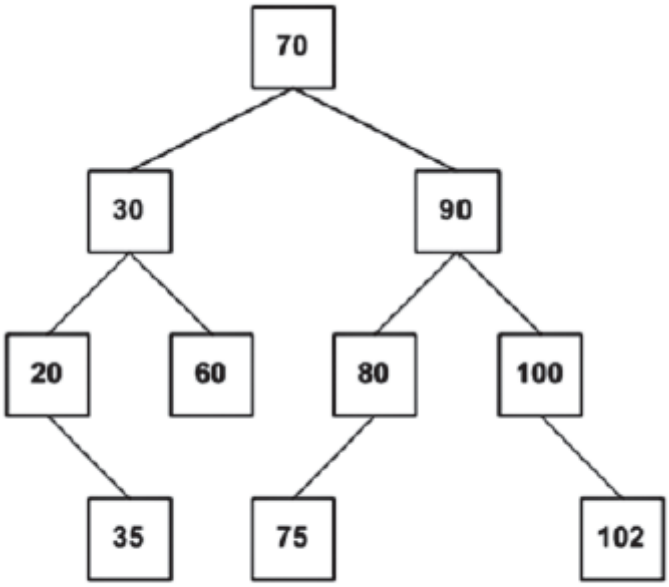

- Verdadeiro
- Falso

A afirmação é falsa conforme o conceito de árvore AVL.

## Pergunta 19

**0,18 / 0,18 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).




- Falso
- Verdadeiro

A afirmação é verdadeira conforme o conceito de árvore AVL.

## Pergunta 20

**0,18 / 0,18 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);
}
```

- Verdadeiro
- Falso

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

## Pergunta 21

**0,18 / 0,18 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.

- Verdadeiro
- Falso

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

## Pergunta 22

**0,18 / 0,18 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;
}
```

- ○ Falso
- ◉ Verdadeiro

A afirmação é verdadeira conforme discutido em sala de aula.

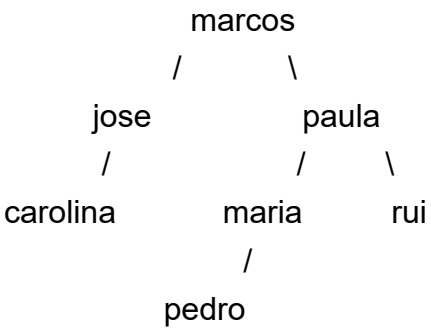
## Pergunta 23

**0,18 / 0,18 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 os nomes Rui e Maria localizados em nós do tipo folha (MANAUSPREV'15, adaptada).

- (●) Falso
- ( ) Verdadeiro

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




Incorreta

## Pergunta 24

**0 / 0,18 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

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.

Incorreta

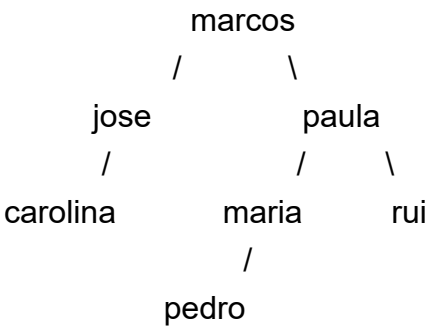
## Pergunta 25

**0 / 0,18 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

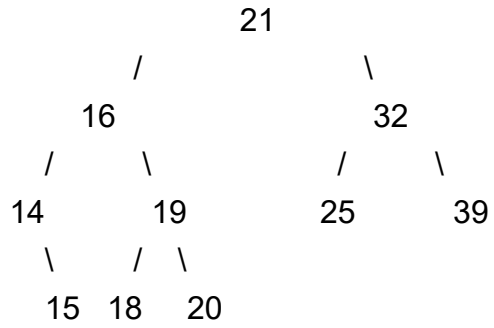
A afirmação é falsa e a árvore resultante é mostrada abaixo.




## Pergunta 26

**0,18 / 0,18 pts**

Dada a árvore binária abaixo e o código do método pesquisar, podemos afirmar que a chamada do método pesquisar(18) imprime na tela duas vezes a letra "A".




```
public boolean pesquisar(int x) {
     return pesquisar(x,raiz);
}

private boolean pesquisar(int x, No i) {
    boolean encontrado;
    if(i == null) {
        encontrado = false;
    } else if(x == i.elemento) {
        encontrado = true;
    } else if(x < i.elemento ) {
        encontrado = pesquisar(x, i.esq);
        System.out.println("A");
    } else {
        encontrado = pesquisar(x, i.dir);
    }
    return encontrado;
}
```

- Falso
- Verdadeiro

A afirmação é verdadeira. A impressão acontece da letra acontece quando o algoritmo caminha para a esquerda a partir dos nós contendo os números 21 e 19.

Incorreta

## Pergunta 27

0 / 0,18 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()”.

- Falso
- 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.

Incorreta

## Pergunta 28

0 / 0,18 pts

Nas linguagem Java e C++, ao instanciarmos um objeto usando new, fazemos com que tal objeto seja acessível a partir de um ponteiro/referência. Entretanto, diferente do Java, a linguagem C++ também permite a criação literal de objetos.

- Verdadeiro
- Falso

Nas linguagem Java e C++, o comando new cria um objeto e retorna seu endereço de memória que deve ser armazenado em um ponteiro/referência. A linguagem Java não permite a criação literal de um objeto o que é possível em C++.

## Pergunta 29

**0,18 / 0,18 pts**

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

- ○ Falso
- ◉ Verdadeiro

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

Incorreta

## Pergunta 30

**0 / 0,18 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()”.

- ◉ Falso
- ○ Verdadeiro

Ambos os comandos criam um ponteiro (ou referência) chamada "no" que armazena um endereço de memória. Além disso, o "new No()" cria um objeto de tipo No e retorna seu endereço de memória.

## Pergunta 31

**2,5 / 5 pts**

Seja a árvore Trie apresentada na sala de aula modifique o método INSERIR supondo que nossa árvore permite a existência de prefixos. Por exemplo, assim, podemos ter as palavras AMO, AMOR, AMORA, AMORES. Apresente a ordem de complexidade do seu método. Sua resposta deve conter somente o método solicitado e sua complexidade.

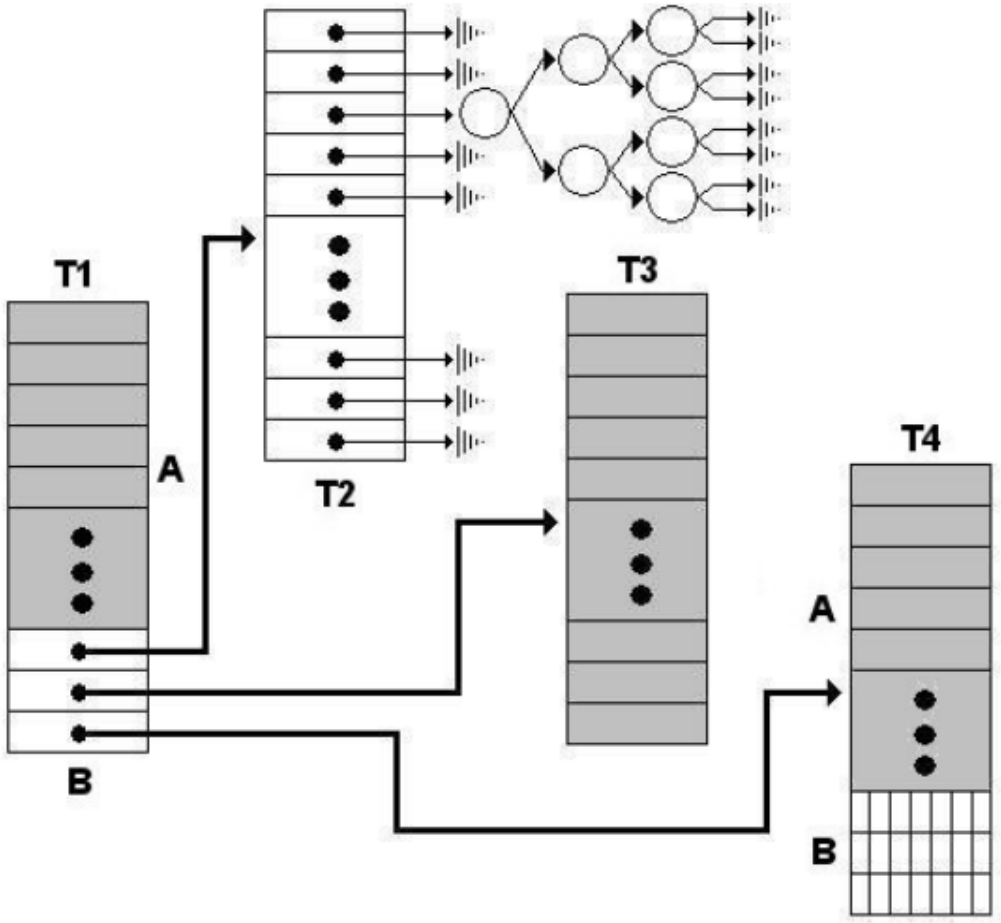
Q31.java
(https://pucminas.instructure.com/files/1333652/download)

A resposta do aluno está incompleta. No caso, no inserir da TRIE, basta remover a condição "no.prox[s.charAt(i)].folha == false" do segundo if.

## Pergunta 32

3,5 / 5 pts

Suponha a estrutura de dados abaixo em que T1 é uma hash com área de reserva (cor branca) "virtual" que nos direciona para T2, T3 e T4. T2 é uma tabela de NoAVL. T3 é uma tabela com duas funções de rehash, int rehashT3A() e int rehashT3B(). T4 é uma tabela com área de reserva. Além disso, o tamanho das quatro tabelas é TAM e que T4 possui mais TAMRESERVA posições para sua área de reserva. Faça um método que retorne o maior elemento de nossa estrutura. Considere que todos os métodos hash estão implementados. Além disso, faça (e justifique) uma análise de complexidade do seu método.




Q32.java
(https://pucminas.instructure.com/files/1334216/download)

O aluno não fez a busca nas árvores de T2.

Pontuação do teste: **9,78** de 15,4