Estruturas de Dados II (DEIN0083) 2024.2
Curso de Ciência da Computação
Atividade Avaliativa (70% da 2ª nota)

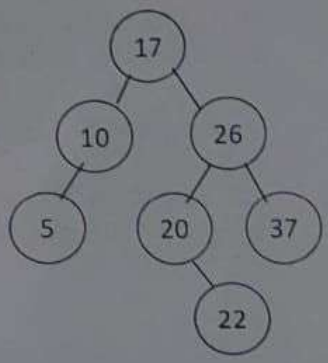
Prof. João Dallyson Sousa de Almeida **Data**: 27/01/2025

Aluno: João Guilherme Lopes Lobato **Matrícula**: 2023041509

Regras durante a prova2:

- É vetada: cópia de respostas dos colegas. A não observância de algum dos itens acima acarretará a anulação da prova. Após a avaliação, você poderá ser selecionado para uma entrevista para verificar a propriedade de suas respostas.

I. (2.0pt) Use a seguinte árvore AVL para completar a tabela. A primeira coluna deve fornecer um item que ainda não esteja na árvore. A segunda coluna deve indicar o número de rotações que seriam feitas ao inserir esse item. A terceira coluna deve identificar o tipo de rotação realizada (em ordem). O valor de cada linha será inserido independentemente das outras linhas. Ou seja, "redefinido" para essa árvore entre as linhas.



Novo Item Inserido	N. de Rotações	Tipo(s) de rotação
	0	Sem rotação
21		
	1	Esquerda
30		
	2	Dupla à Direita
3		

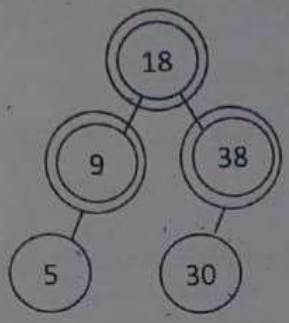
II. (2.0pt) Apresente um algoritmo em pseudocódigo/java para encontrar o primeiro **caractere não repetido** em uma string. Por exemplo, o primeiro caractere não repetido na string "cbp(ss(cb(" é 'p'. Seu algoritmo **deve ter complexidade O(n)**. OBS: Considere a tabela ASCII de 8 bits.

III. (2.0pt) Considere a inserção das chaves 10, 24, 35, 2, 15, 90, 21 em uma tabela de espalhamento de tamanho m = 11 usando endereçamento aberto com a função Hash primária H1(k) = kmod11. Informe se não for possível inserir algum dos elementos. Ilustre o resultado da inserção dessas chaves utilizando:

a) Sondagem linear

b) Hash duplo H2(k) = 1 + (k mod (m-1)).

IV. (2.0pt) Descreva os casos que devem ser cobertos por um algoritmo para realizar a inserção de chaves em uma Árvore Rubro-Negra (RN). Exemplifique-os na árvore RN abaixo:



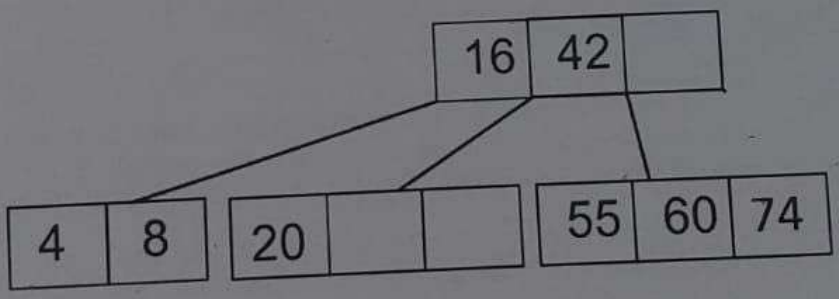
V. (2.0pt) Considerando a Árvore B abaixo, mostre o resultado das seguintes operações na árvore original:

(a) Inserção das chaves 59 e 3, nesta ordem.

(b) Apresente a quantidade total de operações de merge/split, leitura e escrita em disco após a inserção dos itens da letra a.

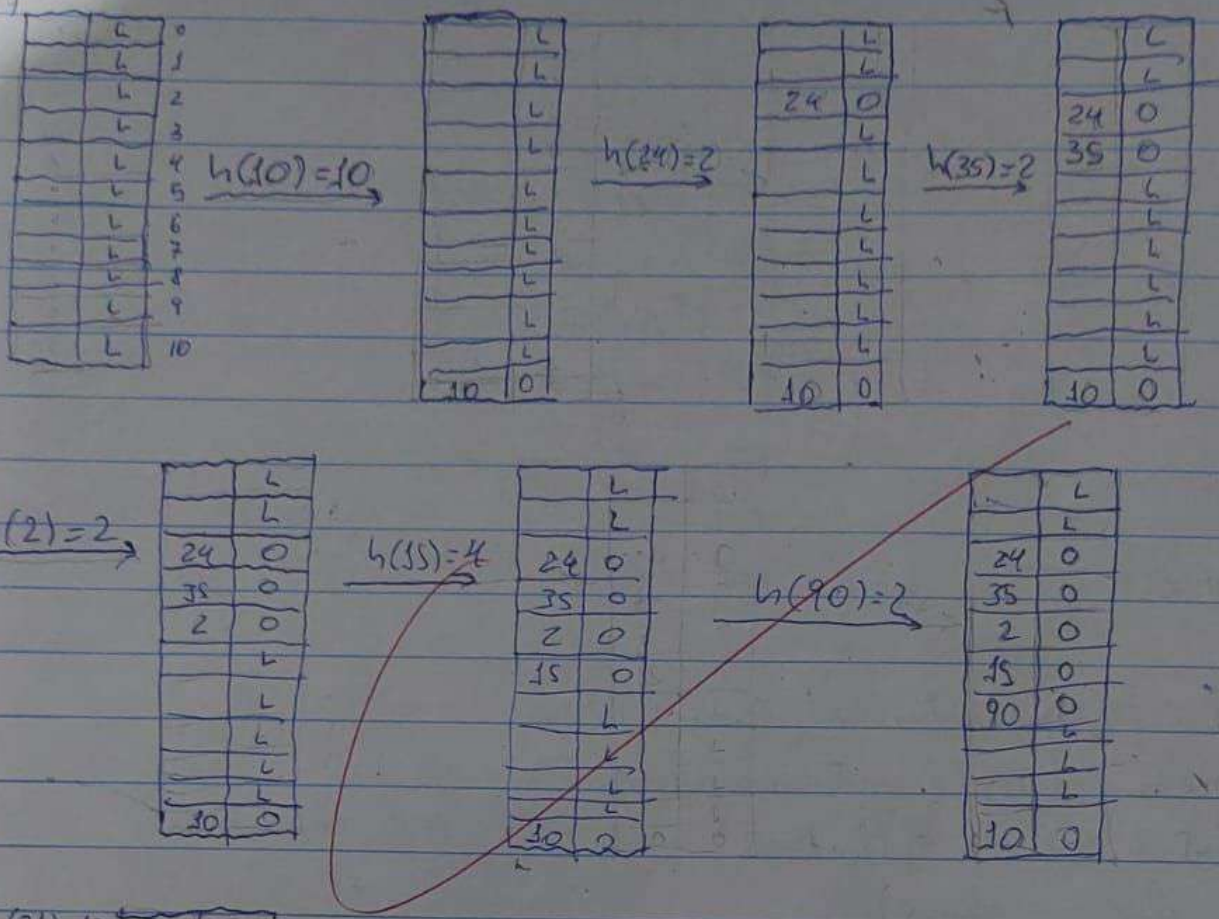
(c) Remoção das chaves 20, 55 e 60, nesta ordem, na árvore resultante da letra a.

(d) Apresente a quantidade total de operações de merge/split, leitura e escrita em disco após a remoção dos itens da letra c.



Questão 03 (10, 24, 35, 2, 15, 90, 21), m = 11

a)



03-b) · h(10,0) = 10 (LIVRE) · h(24,0) = 2 (LIVRE) · h(35,0) = 2 (OCUPADO),
h(35,1) = (2 + 1*(1+35 mod 10)) mod 11 = (2+6) mod 11 = 8 (LIVRE) · h(2,0) = 2 (OCUPADO),
h(2,1) = (2 + 1*(1+2 mod 10)) mod 11 = (2+3) mod 11 = 5 (LIVRE) · h(15,0) = 4 (LIVRE)
· h(90,0) = 2 (OCUPADO), h(90,1) = (2 + 1*(1+90 mod 10)) mod 11 = (2+1) mod 11 = 3 (LIVRE)
· h(21,0) = 10 (OCUPADO), h(21,1) = (10 + 1*(1+21 mod 10)) mod 11 = (10+2) mod 11 = 1 (LIVRE)

Portanto a tabela final fica:

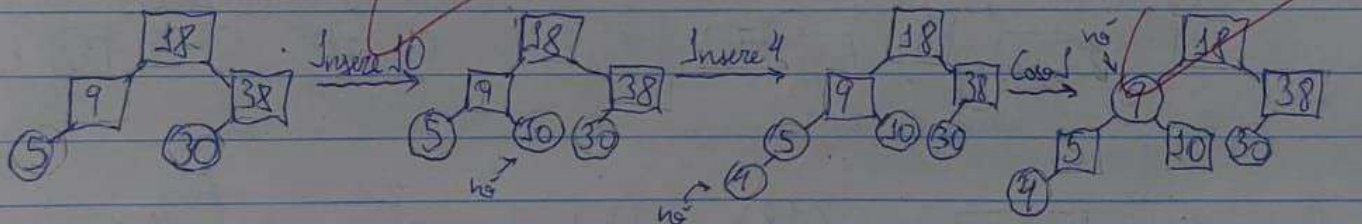
0	1	2	3	4	5	6	7	8	9	10
L	O	O	O	O	O	L	L	O	L	O
	21	24	90	15	2			35		10

2,0

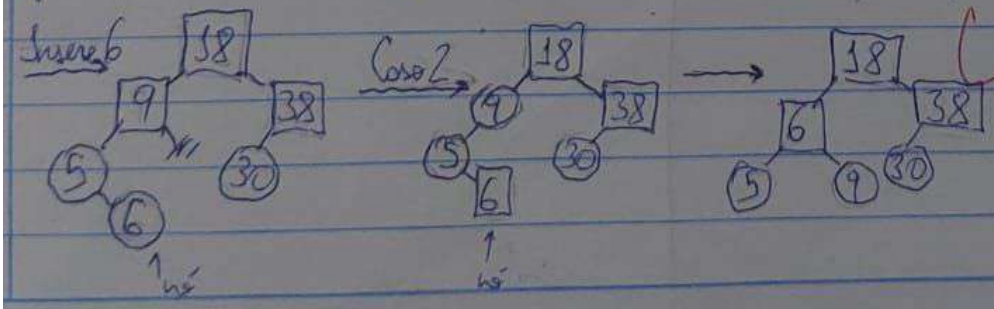
Questão 04

Para a inserção, devemos cobrir 3 casos, sendo que em cada caso é pré-requisito que o pai do nó inserido seja vermelho, e então se este pré-requisito não for atendido em algum momento, não é necessário mais fazer correção.

1° Caso (Tio do nó é vermelho): Para solucionar este caso, basta recolorir o pai e tio para preto e o avô para vermelho, atualizar o nó para que aponte para o avô e então checar os casos novamente. Exemplo: (Seja □ nó preto e O nó vermelho)

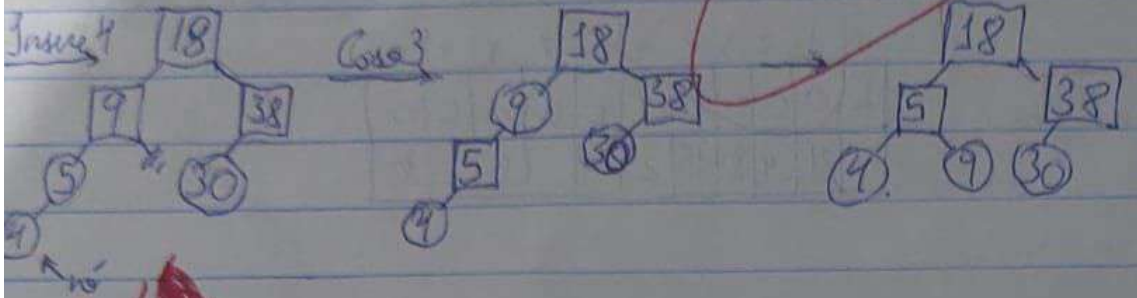


2° Caso (Tio é preto e pai é filho do avô na direção oposta em que nó é filho do pai): Para solucionar este caso, recolorimos o nó para preto e seu avô para vermelho, e então aplicamos uma rotação dupla no avô na direção em que o pai é filho do avô. Exemplo:

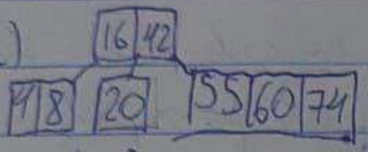


Continuação 04

3° Caso (Tio é preto e pai é filho do avô no mesmo sentido em que o nó é filho do pai): Recolorimos o pai para preto, o avô para vermelho e então aplicamos uma rotação simples no avô na direção em que o nó é filho do pai. Exemplo:

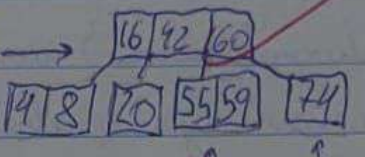


Questão 05 - a)



t = 2

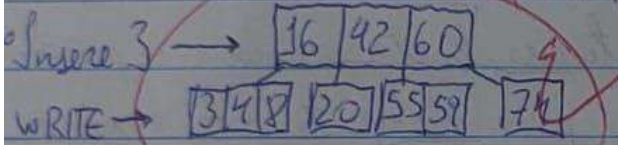
• Insere 59 →



(1 leitura, 2 escritas, 1 split, 0 merges)

Faltou o split raiz

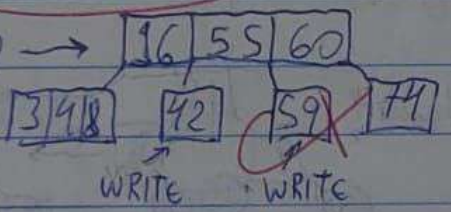
• Insere 3 →



(1 leitura, 1 escrita, 0 splits, 0 merges)

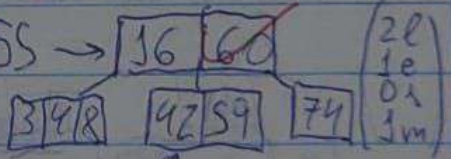
b) Leituras: 2, Escritas: 3
Splits: 1, Merges: 0

c) • Remove 20 →



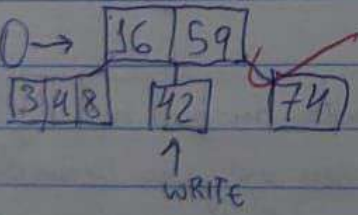
(2 l, 2 e, 0 s, 0 m)

• Remove 55 →



(2 l, 1 e, 0 s, 1 m)

• Remove 60 →



(1 l, 1 e, 0 s, 0 m)

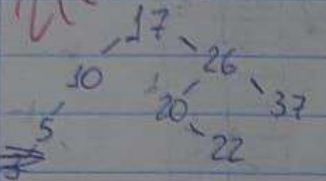
d) Leituras: 5, Escritas: 4
Splits: 0, Merges: 1

9,5 1 1 A

Aluno: João Guilherme Ropero Lobato

Questão 01

2,0



15	0	Sem Rotação
21	2	Dupla à Esquerda
23	1	Esquerda
30	0	Sem Rotação
6	2	Dupla à Direita
3	1	Direita

2,0

Questão 02

```
public char firstNonRepeatingChar(String s) {
    Map<Char, Integer> count = new HashMap<>();          // 1+
    char[] chars = s.toCharArray();                      // 1+
    for (char c : chars)                                 // n*(
        count.put(c, count.getOrDefault(c,0)+1);         // 1)+

    for (char c : chars)                                 // n*(
        if (count.get(c)==1)                             // 1)+
            return c;

    return '\0'; //Não encontrado
}
```

Quantidade de operações: $1+1+n+n = 2n+2 = O(n)$

Basicamente contamos a quantidade de cada caractére c na string, e depois loopamos novamente na string para retornar o primeiro que só aparece 1 vez.