ECM306 – TÓPICOS AVANÇADOS EM ESTRUTURA DE DADOS

ENGENHARIA DA COMPUTAÇÃO – 3ª SÉRIE – 2025 – E1, E2

LAB – PROF. CALVETTI

SIMULADO COM SOLUÇÃO

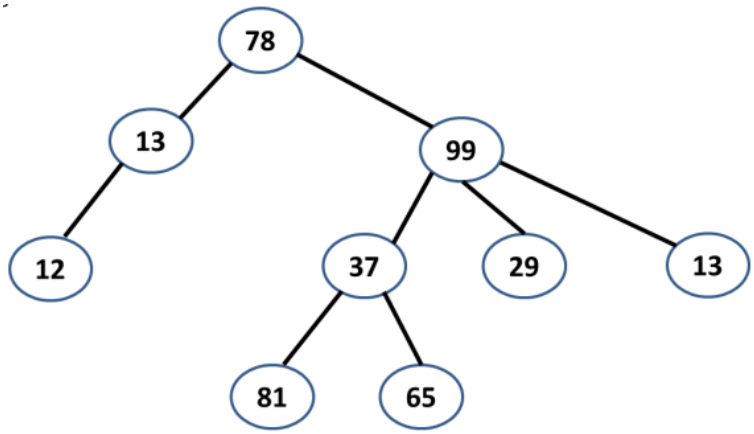
Resolução: Individual;

Prazo: Até o determinado pela atividade;

Entrega: Relatório, em PDF, contendo, obrigatoriamente, as respostas encontradas, bem como uma tabela contendo, em cada linha, o tempo gasto pelo aluno na leitura, entendimento e resolução do item solicitado (exemplo, Exercício 1: 2 min e 48 segundos);

Exercícios: De 1 à 10, a seguir:

1º) ENADE – 2005 (Modificado) - Tendo como base a árvore abaixo, faça o que se pede nos itens a seguir:



A) Descreva a ordem de visita dos nós para uma busca em profundidade, PREORDEM;

RESPOSTA: 78 13 12 99 37 81 65 29 13

B) Descreva a ordem de visita dos nós para uma busca em profundidade, POSORDEM.

RESPOSTA: 12 13 81 65 37 29 13 99 78

2º) Considerando a árvore da questão 1, qual a profundidade do Nó 99?

RESPOSTA: 1

3º) Considerando a árvore da questão 1, qual a altura do Nó 99?

RESPOSTA: 2

4º) ENADE – 2014 (Modificado) - A pilha é uma estrutura de dados que permite a inserção/remoção de itens dinamicamente seguindo a norma de último a entrar, primeiro a sair. Suponha que para uma estrutura de dados, tipo pilha, são definidos os comandos:

- PUSH (p, n): Empilha o número "n" em uma estrutura de dados do tipo pilha "p";
- POP (p): Desempilha o elemento no topo da pilha.

Considere que, em uma estrutura de dados tipo pilha "p", inicialmente vazia, sejam executados os seguintes comandos:

PUSH (p, 10)
PUSH (p, 5)
PUSH (p, 3)
PUSH (p, 40)
POP (p)
PUSH (p, 11)
PUSH (p, 4)
PUSH (p, 7)
POP (p)
POP (p)

Após a execução dos comandos, quais os valores armazenados no topo da pilha "p" e a soma dos elementos armazenados na pilha "p"?

RESPOSTA: 11 e 29

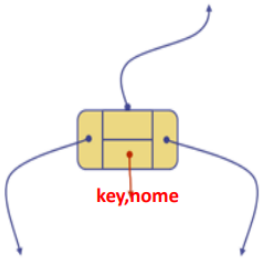
5º) Qual o número máximo de nós armazenados em uma árvore binária com altura 10?

RESPOSTA: 2047

6º) Uma árvore binária completa (*full*) tem 1023 nós. Qual a altura da árvore?

RESPOSTA: 9

7º) Considere um Tipo Abstrato de Dados árvore binária de pesquisa que será utilizado na implementação de uma árvore binária de pesquisa armazenando valores chaves em seus nós, com a seguinte estrutura: referência para o nó pai, referência para o filho à esquerda, referência para o filho à direita e dados armazenados no nó (um valor inteiro, chave e outro valor tipo *String* com nome de uma pessoa. Considere que as funções do TAD estão implementadas e disponíveis para serem utilizadas?



Considere ainda uma árvore binária de pesquisa, implementada a partir dos arrays abaixo declarados, correspondentes aos valores de chave e seus respectivos nomes:

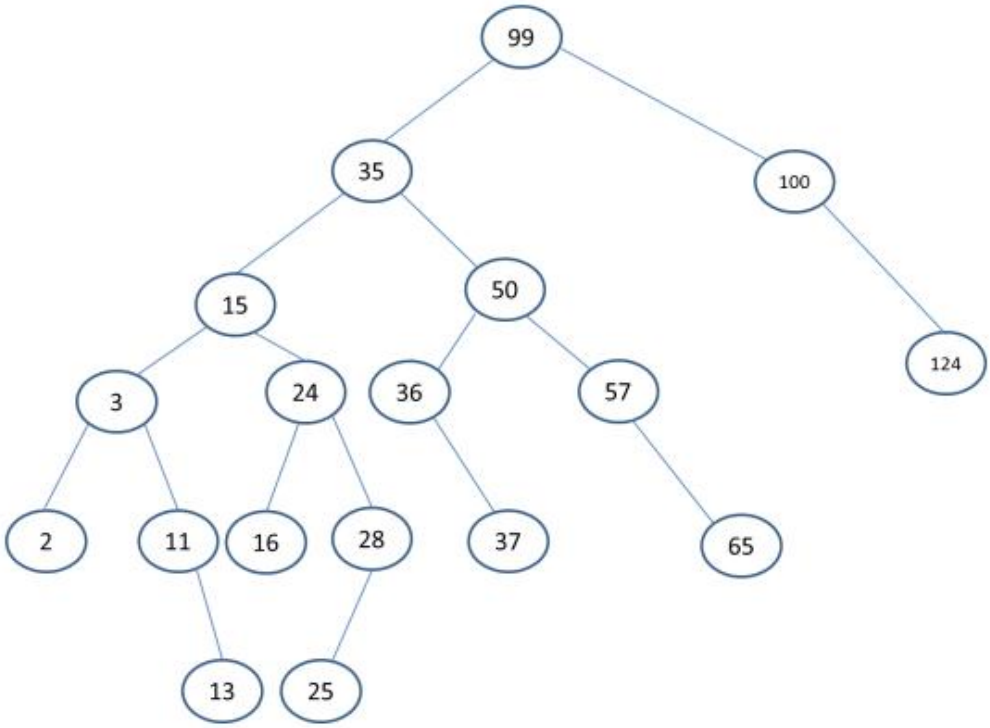
int [ ] valores = {99,35,50,15,24,28,16,3,2,36,11,13,25,57,100,124,65,37};

String [ ] nomes = { "Mauro","Silvio","Leandro","Ana","Paula","Bia","Selma","Carlos",

"Silvia", "Teo", "Ivo", "Selma", "Jorge", "Pedro", "Sueli", "Artur", "Mark", "Camila"};

A) Esboce a árvore binária de pesquisa para os valores acima definidos no array.

RESPOSTA:



B) Quais as comparações de chaves serão processadas para se buscar a chave 25?

RESPOSTA:

1) Compara-se 25 com 99

2) Compara-se 25 com 35

3) Compara-se 25 com 15

4) Compara-se 25 com 24

5) Compara-se 25 com 28

6) Compara-se 25 com 25

Portanto, serão necessárias 6 (seis) comparações

8º) Seja o Heap com sete chaves especificado por: 92 60 78 39 28 66 70. Considere que no Heap foram inseridas as seguintes chaves: 93 e 19 (nesta ordem). Uma possível configuração do Heap após estas inserções é:

A) 92 93 19 78 60 39 28 66 70.

B) 92 93 19 78 60 39 28 66 70.

C) 93 70 92 66 78 60 28 19 39.

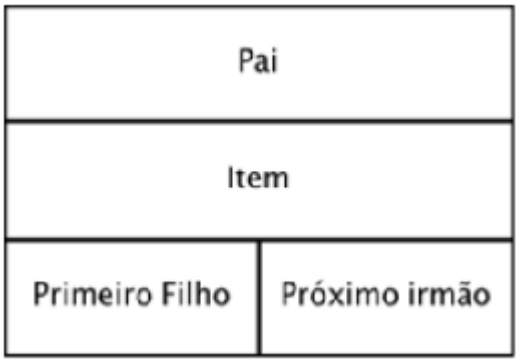
D) 93 92 78 60 28 66 70 39 19.

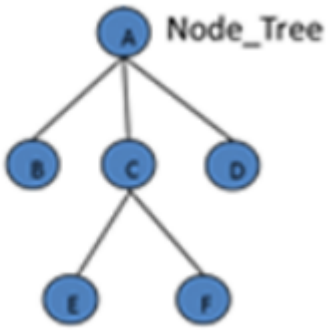
E) Nenhuma das alternativas anteriores.

9º) Verificar se a sequência: 33 32 28 31 26 29 25 30 27 é ou não um Heap. Justifique!

RESPOSTA: Não é *heap*. O nó 28 está no segundo nível e há um nó no terceiro nível (29) que é maior que 28. Logo não é *max-heap*. A árvore também não é *min-heap*, pois o nó raiz (33) é maior que os nós do segundo nível (32 e 28).

10º) Considere um Tipo Abstrato de Dados árvore que armazena valores inteiros em um nó com a seguinte estrutura:





Considere que diversas funções para manipular a árvore estão implementadas e disponíveis para serem utilizadas, numa classe chamada Node_Tree, cuja definição do Node é apresentada no ANEXO.

A) Escrever uma função chamada *ImprimeQtdeFilhos()* que ao ser aplicada em um nó da árvore imprime a quantidade de filhos desse nó. Caso o nó seja folha, a função deverá imprimir a mensagem “Nó sem filhos...”;

RESPOSTA:

```
public void imprimeQteFilhos() {
        int contador = 0;
        if (this.firstChild == null)
                System.out.println("Nó sem filhos....");
                return;
        else {

                Node_Tree trab = this.firstChild;

                while (trab != null ) {
                        contador++;
                        trab = trab.next;
                }
        }
        System.out.println(“Nó tem = “ + contador + “ filhos!);
}
```

B) Escrever uma função chamada *DobraValoresFilhos()* que ao ser aplicada em um nó da árvore, multiplica por 2 os valores de todos os filhos do nó. Caso o nó seja folha, a função deverá imprimir a mensagem “Nó sem filhos...”.

RESPOSTA:

```
public void DobraValoresFilhos() {
      if (this.firstChild == null)
            System.out.println("Nó sem filhos...");
      else {
            Node_Tree trab = this.firstChild;
            while (trab != null ) {
                  trab.item = trab.item * 2;
                  trab = trab.next;
            }
      }
}
```

ANEXO:

```
public class Node_Int {

Integer item;
Node_Int parent;
Node_Int firstChild;
Node_Int next;


public Node_Int(Integer item) {

this.item = item;
this.parent = null;
this.firstChild = null;
this.next = null;

}
```