Universidade Federal do Piauí - UFPI

Centro de Ciências da Natureza – CCN

Departamento de Computação

Estrutura de Dados

Docente: Raimundo Santos Moura

Atividade de Participação 07

Participantes:

Kauã Marques do Nascimento

Davi do Nascimento Santos

Lucas Herlon dos Santos Moreira Cunha

- Introdução

A atividade de participação 07, consiste na penúltima atividade da disciplina de Estrutura de Dados no período de 2022.2, sendo que nela apresenta os seguintes de critérios de aplicação:

1. Mostrar informações sobre a árvore: i) tamanho (número de nós); ii) altura; iii) menor elemento; e iv) maior elemento;
2. Escrever um método internalPathLength para calcular e apresentar o comprimento interno de uma BT (árvore binária);
3. Mostrar os percursos (travessias) da árvore, através de métodos para imprimir as chaves de uma BT: i) em-ordem (ou seja, esquerda-raiz-direita); ii) pós-ordem (esquerdadireita-raiz); iii) pré-ordem (raiz-esquerda-direita); e iv) em largura (levelOrder);

Tivemos muita dificuldade com a atividade, a parte da árvore conseguimos resolver de forma rápida e fácil, porém, nossa maior dificuldade foi com a interface gráfica que foi pedida pelo professor, tanto é que não conseguimos terminar de forma plena e apelamos para o uso do ShowOptionPane, fora isso, tudo o que foi pedido pela atividade conseguimos fazer, dessa forma, peço sua compreensão.

- Código

A função Internal_Path é do tipo inteiro, e tem como objetivo calcular o comprimento da Árvore Binária. A função ainda retorna o comprimento da árvore.

```
public int internal_Path(){
    return internal_Path(this.root, -1);
}
private int internal_Path(Node<T> node, int deph){
    if(node == null)
        return 0;
    deph++;
    if(node.right != null && node.left != null)
        return deph + internal_Path(node.left, deph) + internal_Path(node.right, deph);
    else if(node.left == null)
        return deph + internal_Path(node.right, deph);
    else
        return deph + internal_Path(node.left, deph);

}
```

A função printLevelOrder é uma função Void e tem como objetivo printar as chaves da nossa árvore de forma nivelada.

```
public void printLevelOrder() {
    Queue<Node<T>> queue = new LinkedList<>();
    queue.offer(root);
    while (!queue.isEmpty()) {
        Node<T> node = queue.poll();
        System.out.print(node.key + " ");
        if (node.left != null) {
            queue.offer(node.left);
        }
        if (node.right != null) {
            queue.offer(node.right);
        }
    }
}
```

A função printPostOrder é uma função Void novamente e tem como responsabilidade printar as chaves da árvore em Pós-Ordem. Esquerda-Direita-Raiz.

```
public void printPostOrder() {
    printPostOrder(root);
}

private void printPostOrder(Node<T> node) {
    if (node != null) {
        printPostOrder(node.left);
        printPostOrder(node.right);
        System.out.print(node.key + "\n");
    }
}
```

A função printPreOrder é novamente Void e imprime nomes da árvore binária em Pré-Ordem, Raiz-Esquerda-Direita.

```
public void printPreOrder() {
    printPreOrder(root);
}

private void printPreOrder(Node<T> node) {
    if (node != null) {
        System.out.print(node.key + "\n");
        printPreOrder(node.left);
        printPreOrder(node.right);
    }
}
```

A função printInOrder é uma função Void e também imprime os nomes da árvore de forma Em-Ordem.

Esquerda-Raiz-Direita.

```
public void printInOrder() {
    printInOrder(root);
}

private void printInOrder(Node<T> node) {
    if (node != null) {
        printInOrder(node.left);
        System.out.print(node.key + "\n");
        printInOrder(node.right);
    }
}
```

- Conclusão

Pedimos desculpas por não ter feito o trabalho como deveria, adjunto todas as dificuldades, entregamos uma boa atividade e de forma concisa. O código funcionou muito bem para todos os casos as quais testamos, e tivemos êxito em tudo que foi pedido.

Link do Vídeo:

https://youtu.be/r3FBlioFYOI