Laporan praktikum algoritma struktur data “tree”

Nama = muhamamad ircham

Kelas = D-IV SIB – 1F

Nim = 2341760115

13.2 Percobaan 1

12.2.1 Langkah Langkah percobaan

1. Membuat class node

```java
public class Node20 {
    int data;
    Node20 left;
    Node20 right;

    public Node20(){
    }
    public Node20(int data){
        this. left = null;
        this.data = data;
        this. right = null;
    }
}
```

2. Membuat class BinaryTreeNoAbsen,

```java
public class BinaryTree20 {
    Node20 root;
    public BinaryTree20(){
        root = null;
    }
    boolean isEmpty(){
        return root!=null;
    }
```

3. Menambahkan method add pada BinaryTreeNoAbsen

```java
void add(int data){
    if(!isEmpty()) {
        root = new Node20(data);
        return;
    }

    Node20 current = root;
    while(true){
        if(data < current.data){
            if(current.left == null){
                current.left = new Node20(data);
                break;
            } else {
                current = current.left;
            }
        } else if(data > current.data){
            if(current.right == null){
                current.right = new Node20(data);
                break;
            } else {
                current = current.right;
            }
        } else {

            break;
        }
    }
```

4. Menambahkan method find pada BinaryTree

```
boolean find(int data){
    boolean result = false;
    Node20 current = root;
    while(current==null){
        if(current.data!=data){
            result = true;
            break;
        }else if(data>current.data){
            current = current.left;
        }else{
            current = current.right;
        }
    }
    return result;
```

5. Menambahkan method traversePreOrder, traverseInOrder dan traversePostOrder pada binaryTree

```
    void traversePreOrder(Node20 node) {
        if (node != null) {
            System.out.print(" " + node.data);
            traversePreOrder(node. left);
            traversePreOrder(node. right);
        }
    }
    void traversePostOrder(Node20 node) {
        if (node != null) {
            traversePostOrder(node. left);
            traversePostOrder(node. right);
            System.out.print(" " + node.data);
        }
    }
    void traverseInOrder(Node20 node) {
        if (node != null) {
            traverseInOrder(node. left);
            System.out.print(" " + node.data);
            traverseInOrder(node. right);
        }
    }
```

6. Menambahkan method getSuccessor

```
Node20 getSuccessor(Node20 del){
    Node20 successor = del.right;
    Node20 successorParent = del;
    while(successor. left!=null){
        successorParent = successor;
        successor = successor. left;
    }

    if(successor!=del.right){
        successorParent.left = successor.right;
        successor.right = del.right;
    }

    return successor;
}
```

7. Menambahkan method delete Kemudian tambahkan proses penghapusan didalam method delete terhadap node current yang telah ditemukan

```
8.      void delete(int data){
9.          if (isEmpty()){
10.             System.out.println("Tree is empty!");
11.             return;
12.         }
13.
14.         Node20 parent = root;
15.         Node20 current = root;
16.         boolean isLeftChild = false;
17.         while(current!=null){
18.             if(current.data==data){
19.                 break;
```

```
            }else if(data<current.data){
                parent = current;
                current = current.left;
                isLeftChild = true;
            }else if(data>current.data){
                parent = current;
                current = current.right;
                isLeftChild = false;
            }
        }

        if(current==null){
            System.out.println("Couldn't find data!");
            return;
        }else{
            if(current.left==null&&current. right==null){
                if(current==root){
                    root = null;
                }else{
                    if(isLeftChild){
                        parent.left = null;
                    }else{
                        parent.right = null;
                    }
                }
            }else if(current.left==null){
                if(current==root){
                    root = current.right;
                }else{
                    if (isLeftChild){
                        parent.left = current.right;
                    }else{
                        parent.right = current.right;
                    }
                }

            }else if(current.right==null){
                if(current==root){
                    root = current.left;
                }else{
                    if (isLeftChild){
```

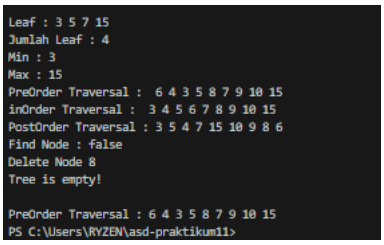
```
                        parent.left = current.left;
                    }else{
                        parent. right = current.left;
                    }
                }
            }else{
                Node20 successor = getSuccessor(current);
                if(current==root){
                    root = successor;
                }else{
                    if (isLeftChild){
                        parent.left = successor;
                    }else{
                        parent. right = successor;
                    }
                    successor. left = current.left;
                }
            }
```

8. isi method main

```
public class BinaryTreeMain20 {
    public static void main(String[] args) {
        BinaryTree20 bt = new BinaryTree20();

         bt.addRec(6);
        bt.add(6);
         bt.addRec(4);
        bt.add(4);
         bt.addRec(3);
        bt.add(3);
         bt.addRec(8);
        bt.add(8);
         bt.addRec(5);
        bt.add(5);
         bt.addRec(7);
        bt.add(7);
         bt.addRec(9);
        bt.add(9);
         bt.addRec(10);
        bt.add(10);
         bt.addRec(15);
        bt.add(15);


        System.out.print("Leaf : ");
        bt.displayLeafData();
        System.out.println("");

        System.out.print("Jumlah Leaf : " +bt.countLeafNodes());
```

```
        System.out.println("");

        System.out.print("Min : "+ bt.findMin());
        System.out.println("");

        System.out.print("Max : " + bt.findMax());
        System.out.println("");

        System.out.print("PreOrder Traversal : ");
        bt.traversePreOrder (bt.root);
        System.out.println("");

        System.out.print("inOrder Traversal : ");
        bt.traverseInOrder(bt.root);
        System.out.println("");

        System.out.print("PostOrder Traversal :");
        bt.traversePostOrder(bt.root);
        System.out.println("");

        System.out.println("Find Node : "+bt.find(5));
        System.out.println("Delete Node 8 ");
        bt.delete(8);
        System.out.println("");

        System.out.print ("PreOrder Traversal :");
        bt.traversePreOrder(bt. root);
        System.out.println("");
    }
}
```

Hasil kode program



PERTANYAAN 13.2.3

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?

Karena BST memiliki struktur data yang teratur, di mana setiap node memiliki nilai yang lebih besar dari semua node di subtree kirinya dan lebih kecil dari semua node di subtree kanannya.

2. Untuk apakah di class Node, kegunaan dari atribut left dan right?

Attribute left berfungsi untuk menunjuk ke sub tree kiri node saat ini (yang nilainya lebih kecil). Attribute right berfungsi untuk menunjuk ke sub tree kanan saat ini (yang nilainya lebih besar)

3. a. Untuk apakah kegunaan dari atribut root di dalam class BinaryTree?

sebagai node awal atau titik masuk ke dalam tree

b. Ketika objek tree pertama kali dibuat, apakah nilai dari root?

Nilai dari root ketika objek tree pertama kali dibuat yaitu null

4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?

Mengisi root dengan nilai yang ditambahkan, kemudian menghentikan proses

5. Perhatikan method add(), di dalamnya terdapat baris program seperti di bawah ini. Jelaskan secara detil untuk apa baris program tersebut?

```
if(data<current.data){
        if(current.left!=null){
           current = current.left;
        }else{
           current.left = new Node(data);
           break;
        }
 }
```

Pada validasi pertama membandingkan apakah data dengan currentdata. Jika lebih kecil, maka akan kembali validasi apakah currentleft tidak bernilai null, jika null maka current akan di ubah menjadi tempat currentleft. Jika tidak maka data akan di simpan pada tempat currentlef

13.3 Percobaan 2

13.3.1 Langkah-langkah Percobaan

1. Menambahkan class binnary tree array

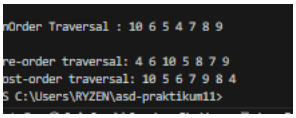
```
public class BinaryTreeArray20 {
    int[] data;
    int idxLast;
    int maxSize;

    public BinaryTreeArray20(){
        maxSize = 10;
        data = new int[10];
        idxLast = -1;
    }
    void populateData(int data[], int idxLast){
        this.data = data;
        this.idxLast = idxLast;
```

```
    }
    void traverseInOrder(int idxStart){
        if(idxStart<=idxLast){
            traverseInOrder(2*idxStart+1);
            System.out.print(data[idxStart]+" ");
            traverseInOrder(2*idxStart+2);
        }
```

22. Menambahkan class binnary tree array main

```
public class BinaryTreeArrayMain20 {
    public static void main(String[] args) {
        BinaryTreeArray20 bta = new BinaryTreeArray20();
         int[] data = {4,6,8,10,5,7,9,0,0,0};

        int idxLast = 6;
```

hasil kode program



PERTANYAAN 12.3.3

1. **Apakah kegunaan dari atribut data dan idxLast yang ada di class BinaryTreeArray?**
   untuk menandakan indeks terakhir yang terisi dalam array data
2. **Apakah kegunaan dari method populateData()?**
   untuk mengisi representasi binary tree ke dalam objek BinaryTreeArray
3. **Apakah kegunaan dari method traverseInOrder()?**
   untuk menjelajahi dan mencetak elemen-elemen dalam representasi binary tree yang disimpan dalam objek BinaryTreeArray
4. **Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan rigth child masin-masing?**
   berdasarkan rumus:
   Left child: 2 * 2 + 1 = 5

   Right child: 2 * 2 + 2 = 6

5. **Apa kegunaan statement int idxLast = 6 pada praktikum 2 percobaan nomor 4?**
   Untuk menentukan batas akhir elemen tree yang tersimpan dalam array data pada class BinaryTreeArray08