

**LAPORAN PRAKTIKUM**  
**MATA KULIAH PRAKTIKUM ALGORITMA STRUKTUR DATA**

Dosen Pengampu : Triana Fatmawati, S.T, M.T

**PERTEMUAN 14 : TREE**



Nama : Yonanda Mayla Rusdiaty

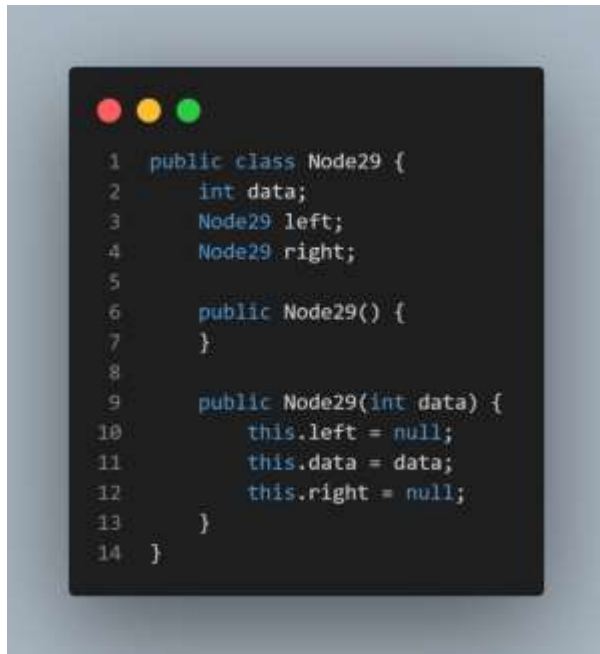
NIM : 2341760184

Prodi : D-IV Sistem Informasi Bisnis

**JURUSAN TEKNOLOGI INFORMASI**  
**POLITEKNIK NEGERI MALANG**  
**2024**

## PRAKTIKUM 1

Kode program class Node29 :



```
1  public class Node29 {  
2      int data;  
3      Node29 left;  
4      Node29 right;  
5  
6      public Node29() {  
7      }  
8  
9      public Node29(int data) {  
10         this.left = null;  
11         this.data = data;  
12         this.right = null;  
13     }  
14 }
```

Kode program class BinaryTree29 :



Kode program class BinaryTreeMain29 :

```
1 public class BinaryTreeMain29 {
2     public static void main(String[] args) {
3         BinaryTree29 bt = new BinaryTree29();
4         bt.add(6);
5         bt.add(4);
6         bt.add(8);
7         bt.add(3);
8         bt.add(5);
9         bt.add(7);
10        bt.add(9);
11        bt.add(10);
12        bt.add(15);
13        System.out.println("Traverse Pre Order : ");
14        bt.traversePreOrder(bt.root);
15        System.out.println("");
16        System.out.println("Traverse In Order : ");
17        bt.traverseInOrder(bt.root);
18        System.out.println("");
19        System.out.println("Traverse Post Order : ");
20        bt.traversePostOrder(bt.root);
21        System.out.println("");
22        System.out.println("Find Node : " + bt.find(5));
23        System.out.println("Delete Node 8");
24        bt.delete(8);
25        System.out.println("");
26        System.out.println("Traverse Pre Order : ");
27        bt.traversePreOrder(bt.root);
28        System.out.println("");
29    }
30 }
31 }
```

Output :

```
Traverse Pre Order :
 6 4 3 5 8 7 9 10 15
Traverse In Order :
 3 4 5 6 7 8 9 10 15
Traverse Post Order :
 3 5 4 7 15 10 9 8 6
Find Node : true
Delete Node 8

Traverse Pre Order :
 6 4 3 5 9 7 10 15
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P14>
```

### Pertanyaan :

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

**Jawab :**

Karena dalam binary search tree (BST), setiap node memiliki properti khusus dimana semua node di subtree kiri memiliki nilai yang lebih kecil daripada nilai node tersebut, dan semua node di subtree kanan memiliki nilai yang lebih besar. Hal ini berarti, bahwa setiap kali kita mencari nilai, kita bisa "mengabaikan" setengah dari tree, yang membuat pencarian menjadi lebih efisien.

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

**Jawab :** Dalam konteks struktur data binary tree, atribut left dan right dalam class Node digunakan untuk merujuk ke node left child dan right child dari node saat ini.

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

**Jawab :** Atribut root dalam class BinaryTree digunakan untuk merujuk ke node awal atau node paling atas dalam struktur binary tree. Root adalah satu-satunya node yang tidak memiliki node induk/parent.

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

**Jawab :** Ketika objek BinaryTree29 pertama kali dibuat, nilai dari root adalah null, karena root belum diinisialisasi dan default value nya yaitu null. Hal ini juga merupakan alasan mengapa metode isEmpty() mengembalikan true jika root == null. Jika root adalah null, itu berarti pohon masih kosong karena belum ada node yang ditambahkan ke dalamnya.

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

**Jawab :** Ketika tree masih kosong dan sebuah node baru akan ditambahkan, berikut adalah proses yang akan terjadi:

- 1) Method add(int data) dipanggil dengan data yang akan ditambahkan ke pohon.
- 2) Di dalam method add, pertama-tama memeriksa dulu apakah tree kosong dengan memanggil metode isEmpty(). Metode isEmpty() mengembalikan true jika root adalah null, yang berarti tree masih kosong.
- 3) Jika tree kosong (yaitu, isEmpty() mengembalikan true), maka code di dalam blok if dieksekusi. Di sini, sebuah node baru dibuat dengan data yang diberikan dan node ini ditetapkan sebagai root dari pohon.

Dengan kata lain, jika tree awalnya kosong, node baru yang ditambahkan akan menjadi root dari pohon.

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;  
    }  
}
```

**Jawab :** Kode program tersebut digunakan untuk menambahkan node baru ke binary search tree dengan aturan bahwa nilai pada node kiri harus lebih kecil dari nilai pada node induk/parent.

## LATIHAN 2

Kode program class BinaryTreeArray29 :

```
1 public class BinaryTreeArray29 {
2     int[] data;
3     int idxLast;
4
5     public BinaryTreeArray29(int size) {
6         data = new int[10];
7     }
8
9     void populateData(int[] data, int idxLast) {
10        this.data = data;
11        this.idxLast = idxLast;
12    }
13
14    void traverseInorder(int idxStart) {
15        if (idxStart <= idxLast) {
16            traverseInorder(2 * idxStart + 1);
17            System.out.print(data[idxStart] + " ");
18            traverseInorder(2 * idxStart + 2);
19        }
20    }
21 }
22
23
24
```

BinaryTreeMain29 :

```
1 public class BinaryTreeArrayMain29 {
2     public static void main(String[] args) {
3         BinaryTreeArray29 bta = new BinaryTreeArray29(10);
4         int[] data = { 6, 4, 8, 3, 5, 7, 9, 0, 0, 0};
5         int idxLast = 6;
6         bta.populateData(data, idxLast);
7         System.out.println("\ninorder traversal: ");
8         bta.traverseInorder(0);
9         System.out.println("\n");
10    }
11 }
12
```

Output :

```
Inorder traversal:
3 4 5 6 7 8 9
```

```
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P14>
```

**Pertanyaan :**

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class BinaryTreeArray?

**Jawab :** Atribut data dan idxLast dalam class BinaryTreeArray29 memiliki fungsi sebagai berikut:

- 1) data: Atribut ini digunakan untuk menyimpan data dari binary tree. Indeks dalam array mewakili posisi node dalam binary tree jika tree tersebut diwakili sebagai array (dengan indeks 0 sebagai root, indeks  $2i+1$  sebagai left child, dan indeks  $2i+2$  sebagai right child, di mana  $i$  adalah indeks node induk).
- 2) idxLast: Atribut ini digunakan untuk menyimpan indeks dari elemen terakhir yang ditambahkan ke dalam array data, yang berguna untuk mengetahui berapa banyak elemen yang saat ini disimpan dalam array dan di mana kita harus menambahkan elemen baru jika kita ingin menambahkan data ke tree. Selain itu, dalam metode traverseInorder, idxLast digunakan untuk memastikan bahwa kita tidak mencoba mengakses elemen di luar batas array.

2. Apakah kegunaan dari method populateData()?

**Jawab :** Metode populateData() digunakan untuk mengisi array data dan menetapkan nilai idxLast dalam objek BinaryTreeArray29.

3. Apakah kegunaan dari method traverseInOrder()?

**Jawab :** Metode traverseInorder() digunakan untuk melakukan penjelajahan inorder pada binary tree yang disimpan dalam array. Penjelajahan inorder adalah method penjelajahan binary tree di mana kita pertama kali mengunjungi left child → root → dan akhirnya right child

4. Jika suatu node binary tree disimpan dalam array indeks 2, maka di indeks berapakah posisi left child dan right child masing-masing?

**Jawab :** Dalam representasi array dari binary tree, left child dan right child dari node pada indeks  $i$  dapat ditemukan pada indeks  $2*i + 1$  dan  $2*i + 2$  masing-masing. Jadi, jika suatu node binary tree disimpan pada indeks 2:

- Left child dari node tersebut akan berada pada indeks  $2*2 + 1 = 5$ .
- Left child dari node tersebut akan berada pada indeks  $2*2 + 2 = 6$ .

5. Apa kegunaan statement `int idxLast = 6` pada praktikum 2 percobaan nomor 4?

**Jawab :** Statement `int idxLast = 6`; digunakan untuk menetapkan nilai awal dari variabel idxLast yang kemudian diteruskan ke method `populateData(data, idxLast)`. Dengan menetapkan `idxLast = 6`, kita memberi tahu method `populateData` bahwa hanya elemen dengan indeks 0 hingga 6 yang harus dianggap sebagai bagian dari data binary tree. Elemen lainnya dalam array data harus diabaikan.



## TUGAS

1. Buat method di dalam class BinaryTree yang akan menambahkan node dengan cara rekursif.

**Jawab :** Berikut adalah hasilnya :

```
void addRecursive(int idxCurrent, int value) { //no 1 tugas
    // Jika indeks saat ini lebih besar dari idxLast, berarti kita telah mencapai akhir array
    if (idxCurrent > idxLast) {
        System.out.println(x:"Array is full, cannot add more nodes");
        return;
    }

    // Jika data pada indeks saat ini adalah 0 (yang berarti node ini kosong),
    // tambahkan nilai ke node ini
    if (data[idxCurrent] == 0) {
        data[idxCurrent] = value;
        // Jika kita menambahkan node baru, perbarui idxLast
        if (idxCurrent > idxLast) {
            idxLast = idxCurrent;
        }
    } else {
        // Jika node ini sudah memiliki nilai, lanjutkan ke node berikutnya
        // Jika nilai yang akan ditambahkan lebih kecil, pergi ke anak kiri (2*idxCurrent + 1)
        // Jika nilai yang akan ditambahkan lebih besar, pergi ke anak kanan (2*idxCurrent + 2)
        if (value < data[idxCurrent]) {
            addRecursive(2 * idxCurrent + 1, value);
        } else if (value > data[idxCurrent]) {
            addRecursive(2 * idxCurrent + 2, value);
        }
    }
}
```

2. Buat method di dalam class BinaryTree untuk menampilkan nilai paling kecil dan yang paling besar yang ada di dalam tree.

**Jawab :**

- Kode program min :

```
int findMin(int idxCurrent) { //no 2 tugas
    // Jika node saat ini kosong atau kita telah mencapai akhir array, akan mengembalikan nilai maksimum int
    if (idxCurrent > idxLast || data[idxCurrent] == 0) {
        return Integer.MAX_VALUE;
    }

    // Jika node saat ini memiliki left child, lanjutkan mencari di subtree kiri
    if (2 * idxCurrent + 1 <= idxLast && data[2 * idxCurrent + 1] != 0) {
        return findMin(2 * idxCurrent + 1);
    }

    // Jika kita mencapai sini, berarti node saat ini adalah node paling kiri
    return data[idxCurrent];
}
```

- Kode program maks :

```

int findMax(int idxCurrent) {
    // Jika node saat ini kosong atau kita telah mencapai akhir array, akan mengembalikan nilai minimum int
    if (idxCurrent > idxLast || data[idxCurrent] == 0) {
        return Integer.MIN_VALUE;
    }

    // Jika node saat ini memiliki right child, lanjutkan mencari di subtree kanan
    if (2 * idxCurrent + 2 <= idxLast && data[2 * idxCurrent + 2] != 0) {
        return findMax(2 * idxCurrent + 2);
    }

    // Jika kita mencapai sini, berarti node saat ini adalah node paling kanan
    return data[idxCurrent];
}

```

- Main :

```

public class BinaryTreeArrayMain29 {
    Run | Debug
    public static void main(String[] args) {
        BinaryTreeArray29 bta = new BinaryTreeArray29(size:10);
        int[] data = { 6, 4, 8, 3, 5, 7, 9, 0, 0, 0};
        int idxLast = 6;
        bta.populateData(data, idxLast);
        System.out.println(x:"\nInorder traversal: ");
        bta.traverseInorder(idxStart:0);
        int min = bta.findMin(idxCurrent:0);
        int max = bta.findMax(idxCurrent:0);
        System.out.println("\nMinimum value in the tree: " + min);
        System.out.println("Maximum value in the tree: " + max);
        System.out.println(x:"\n");
    }
}

```

You, 1 minute ago • Uncommitted changes

- Output :

```

Inorder traversal:
3 4 5 6 7 8 9
Minimum value in the tree: 3
Maximum value in the tree: 9

```

3. Buat method di dalam class BinaryTree untuk menampilkan data yang ada di leaf.

**Jawab :**

- Kode program :

```

void printLeaves(int idxCurrent) { //no 3 tugas
    // Jika node saat ini kosong atau kita telah mencapai akhir array
    if (idxCurrent > idxLast || data[idxCurrent] == 0) {
        return;
    }

    // Jika node saat ini tidak memiliki child, print nilainya
    if ((2 * idxCurrent + 1 > idxLast || data[2 * idxCurrent + 1] == 0) &&
        (2 * idxCurrent + 2 > idxLast || data[2 * idxCurrent + 2] == 0)) {
        System.out.print(data[idxCurrent] + " ");
    } else {
        // Jika node saat ini memiliki child, lanjutkan penjelajahan ke seluruh childnya
        printLeaves(2 * idxCurrent + 1);
        printLeaves(2 * idxCurrent + 2);
    }
}
}

```

- Main :

```

public class BinaryTreeArrayMain29 {
    Run | Debug
    public static void main(String[] args) {
        BinaryTreeArray29 bta = new BinaryTreeArray29(size:10);
        int[] data = { 6, 4, 8, 3, 5, 7, 9, 0, 0, 0};
        int idxLast = 6;
        bta.populateData(data, idxLast);
        System.out.println(x:"\nInorder traversal: ");
        bta.traverseInorder(idxStart:0);
        int min = bta.findMin(idxCurrent:0);
        int max = bta.findMax(idxCurrent:0);
        System.out.println("\nMinimum value in the tree: " + min);
        System.out.println("Maximum value in the tree: " + max);
        System.out.println(x:"\nLeaf nodes: ");
        bta.printLeaves(idxCurrent:0);
        System.out.println(x:"\n");
    }
}
You, 1 second ago • Uncommitted changes

```

- Output :

Inorder traversal:

3 4 5 6 7 8 9

Minimum value in the tree: 3

Maximum value in the tree: 9

Leaf nodes:

3 5 7 9

PS D:\KULIAH\college\smt 2\29\_yonanda\_asd\P14>

4. Buat method di dalam class BinaryTree untuk menampilkan berapa jumlah leaf yang ada di dalam tree.

**Jawab :**

- Kode program :

```
int countLeaves(int idxCurrent) { // no 4 tugas
    // Jika node saat ini kosong atau kita telah mencapai akhir array, return 0
    if (idxCurrent > idxLast || data[idxCurrent] == 0) {
        return 0;
    }

    // Jika node saat ini tidak memiliki child, return 1
    if ((2 * idxCurrent + 1 > idxLast || data[2 * idxCurrent + 1] == 0) &&
        (2 * idxCurrent + 2 > idxLast || data[2 * idxCurrent + 2] == 0)) {
        return 1;
    } else {
        // Jika node saat ini memiliki child, lanjutkan penjelajahan ke seluruh childnya
        return countLeaves(2 * idxCurrent + 1) + countLeaves(2 * idxCurrent + 2);
    }
}
```

You, 4 seconds ago • Uncommitted changes



- Main :

```
public class BinaryTreeArrayMain29 {  
    Run | Debug  
    public static void main(String[] args) {  
        BinaryTreeArray29 bta = new BinaryTreeArray29(size:10);  
        int[] data = { 6, 4, 8, 3, 5, 7, 9, 0, 0, 0};  
        int idxLast = 6;  
        bta.populateData(data, idxLast);  
        System.out.println(x:"\nInorder traversal: ");  
        bta.traverseInorder(idxStart:0);  
        int min = bta.findMin(idxCurrent:0);  
        int max = bta.findMax(idxCurrent:0);  
        System.out.println("\nMinimum value in the tree: " + min);  
        System.out.println("Maximum value in the tree: " + max);  
        System.out.println(x:"\nLeaf nodes: ");  
        bta.printLeaves(idxCurrent:0);  
        int leafCount = bta.countLeaves(idxCurrent:0);  
        System.out.println("\nNumber of leaf nodes: " + leafCount);  
        System.out.println(x:"\n");  
    }  
}
```

You, 1 second ago • Uncommitted changes

- Output :

```
Inorder traversal:  
3 4 5 6 7 8 9  
Minimum value in the tree: 3  
Maximum value in the tree: 9  
  
Leaf nodes:  
3 5 7 9  
Number of leaf nodes: 4  
  
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P14>
```

5. Modifikasi class BinaryTreeArray, dan tambahkan :

- method add(int data) untuk memasukan data ke dalam tree

**Jawab :**

```
// no 5 tugas modif
public BinaryTreeArray29(int size) {
    data = new int[size];
    for (int i = 0; i < size; i++) {
        data[i] = 0;
    }
}

void populateData(int[] data, int idxLast) {
    this.data = data;
    this.idxLast = idxLast;
}

void add(int newData) {
    if (data[0] == 0) {
        data[0] = newData;
    } else {
        for (int i = 0; i < data.length; i++) {
            if (data[i] == 0) {
                data[i] = newData;
                break;
            }
        }
    }
    idxLast++;
}
}
```

- method traversePreOrder() dan traversePostOrder()

**Jawab :**

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

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

**Main :**

```
public class BinaryTreeArrayMain29 {  
    Run | Debug  
    public static void main(String[] args) {  
        BinaryTreeArray29 bta = new BinaryTreeArray29(size:10);  
        int[] data = { 6, 4, 8, 3, 5, 7, 9, 0, 0, 0};  
        int idxLast = 6;  
        bta.populateData(data, idxLast);  
        System.out.println(x:"\nInorder traversal: ");  
        bta.traverseInorder(idxStart:0);  
        int min = bta.findMin(idxCurrent:0);  
        int max = bta.findMax(idxCurrent:0);  
        System.out.println("\nMinimum value in the tree: " + min);  
        System.out.println("Maximum value in the tree: " + max);  
        System.out.println(x:"\nLeaf nodes: ");  
        bta.printLeaves(idxCurrent:0);  
        int leafCount = bta.countLeaves(idxCurrent:0);  
        System.out.println("\nNumber of leaf nodes: " + leafCount);  
        System.out.println(x:"\nAdding new data to the tree: ");  
        bta.add(newData:10);  
        System.out.println(x:"\nPreOrder traversal after adding new data: ");  
        bta.traversePreOrder(idxCurrent:0);  
        System.out.println(x:"\nPostOrder traversal after adding new data: ");  
        bta.traversePostOrder(idxCurrent:0);  
        System.out.println(x:"\n");  
    }  
}
```

You, 40 seconds ago • Uncommitted changes

**Output a) dan b) :**

```
Inorder traversal:  
3 4 5 6 7 8 9  
Minimum value in the tree: 3  
Maximum value in the tree: 9  
  
Leaf nodes:  
3 5 7 9  
Number of leaf nodes: 4  
  
Adding new data to the tree:  
  
PreOrder traversal after adding new data:  
6 4 3 10 5 8 7 9  
PostOrder traversal after adding new data:  
10 3 5 4 7 9 8 6  
  
PS D:\KULIAH\college\smt 2\29_yonanda_asd\P14>
```