NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET XI - TREE

PERTEMUAN MINGGU 13 - TREE

Kegiatan Praktikum 1
Implementasi Binary Search Tree menggunakan Linked List (45 Menit)

```java
package Jobsheet_11;

public class Main_07 {
    Run | Debug
    public static void main(String[] args) {
        BinaryTree_07 bt = new BinaryTree_07();
        bt.add(data:6);
        bt.add(data:4);
        bt.add(data:8);
        bt.add(data:3);
        bt.add(data:5);
        bt.add(data:7);
        bt.add(data:9);
        bt.add(data:10);
        bt.add(data:15);

        System.out.print(s:"PreOrder Traversal:");
        bt.traversePreOrder(bt.root);
        System.out.println(x:"");
        System.out.print(s:"InOrder Traversal:");
        bt.traverseInOrder(bt.root);
        System.out.println(x:"");
        System.out.print(s:"PostOrder Traversal:");
        bt.traversePostOrder(bt.root);
        System.out.println(x:"");
        System.out.println("Find Node : " + bt.find(data:5));
        System.out.println(x:"Delete node 8");
        bt.delete(data:8);
        System.out.println(x:"");
        System.out.print(s:"PreOrder Traversal:");
        bt.traversePreOrder(bt.root);
        System.out.println(x:"");
    }
}
```

src > Jobsheet_11 > Node_07.java > {} Jobsheet_11

```java
package Jobsheet_11;

public class Node_07 {
    int data;
    Node_07 left, right;

    public Node_07(int d) {
        data = d;
        left = right = null;
    }
}
```

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET XI - TREE

src > Jobsheet_11 > BinaryTree_07.java > BinaryTree_07 > add(int)

```java
package Jobsheet_11;

public class BinaryTree_07 {
    Node_07 root;
    int size;

    public BinaryTree_07() {
        root = null;
        size = 0;
    }

    boolean isEmpty() {
        return root == null;
    }

    void add(int data) {
        if (isEmpty()) {
            root = new Node_07(data);
        } else {
            Node_07 current = root;
            while (true) {
                if (data < current.data) {
                    if (current.left == null) {
                        current.left = new Node_07(data);
                        break;
                    } else {
                        current = current.left;
                    }
                } else if (data > current.data) {
                    if (current.right == null) {
                        current.right = new Node_07(data);
                        break;
                    } else {
                        current = current.right;
                    }
                } else {
                    break;
                }
            }
        }
    }

    boolean find(int data) {
        Node_07 current = root;
        while (current != null) {
            if (data < current.data) {
```

src > Jobsheet_11 > BinaryTree_07.java > BinaryTree_07 > add(int)

```java
public class BinaryTree_07 {

    boolean find(int data) {
        Node_07 current = root;
        while (current != null) {
            if (data < current.data) {
                current = current.left;
            } else if (data > current.data) {
                current = current.right;
            } else {
                return true;
            }
        }
        return false;
    }

    void traversePreOrder(Node_07 Node_07) {
        if (Node_07 != null) {
            System.out.print(Node_07.data + " ");
            traversePreOrder(Node_07.left);
            traversePreOrder(Node_07.right);
        }
    }

    void traversePostOrder(Node_07 Node_07) {
        if (Node_07 != null) {
            traversePostOrder(Node_07.left);
            traversePostOrder(Node_07.right);
            System.out.print(Node_07.data + " ");
        }
    }

    void traverseInOrder(Node_07 Node_07) {
        if (Node_07 != null) {
            traverseInOrder(Node_07.left);
            System.out.print(Node_07.data + " ");
            traverseInOrder(Node_07.right);
        }
    }

    Node_07 getSuccessor(Node_07 del) {
        Node_07 successor = del.right;
        Node_07 successorParent = del;
        while (successor.left != null) {
            successorParent = successor;
            successor = successor.left;
        }
```

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET XI - TREE

```
src > Jobsheet_11 > BinaryTree_07.java > BinaryTree_07 > add(int)
  3   public class BinaryTree_07 {
 81       Node_07 getSuccessor(Node_07 del) {
 88           if (successor != del.right) {
 89               successorParent.left = successor.right;
 90               successor.right = del.right;
 91           }
 92           return successor;
 93       }
 94
 95       void delete(int data) {
 96           Node_07 parent = root;
 97           Node_07 current = root;
 98           boolean isLeftChild = false;
 99           while (current != null) {
100               if (current.data == data) {
101                   break;
102               } else if (data < current.data) {
103                   parent = current;
104                   current = current.left;
105                   isLeftChild = true;
106               } else if (data > current.data) {
107                   parent = current;
108                   current = current.right;
109                   isLeftChild = false;
110               }
111           }
112           if (current == null) {
113               System.out.println(x:"Node_07 not found");
114               return;
115           } else {
116               if (current.left == null && current.right == null) {
117                   if (current == root) {
118                       root = null;
119                   } else {
120                       if (isLeftChild) {
121                           parent.left = null;
122                       } else {
123                           parent.right = null;
124                       }
125                   }
126               } else if (current.left == null) {
127                   if (current == root) {
128                       root = current.right;
129                   } else {
130                       if (isLeftChild) {
131                           parent.left = current.right;
```

```
src > Jobsheet_11 > BinaryTree_07.java > BinaryTree_07 > add(int)
  3   public class BinaryTree_07 {
 95       void delete(int data) {
125                   }
126               } else if (current.left == null) {
127                   if (current == root) {
128                       root = current.right;
129                   } else {
130                       if (isLeftChild) {
131                           parent.left = current.right;
132                       } else {
133                           parent.right = current.right;
134                       }
135                   }
136               } else if (current.right == null) {
137                   if (current == root) {
138                       root = current.left;
139                   } else {
140                       if (isLeftChild) {
141                           parent.left = current.left;
142                       } else {
143                           parent.right = current.left;
144                       }
145                   }
146               } else {
147                   Node_07 successor = getSuccessor(current);
148                   if (current == root) {
149                       root = successor;
150                   } else {
151                       if (isLeftChild) {
152                           parent.left = successor;
153                       } else {
154                           parent.right = successor;
155                       }
156                       successor.left = current.left;
157                   }
158               }
159           }
160       }
161   }
```

Hasil Run Program

```
PROBLEMS 24   OUTPUT   TERMINAL   COMMENTS   DEBUG CONSOLE   ...

PS C:\Users\Admin\OneDrive - ypt.or.id\Documents\Kuliah\Semest
 dan Struktur Data\Praktikum_AlgoritmaStrukturData_07>  & 'C:\
java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\
or.id\Documents\Kuliah\Semester 2\7. Praktikum Algoritma dan S
ritmaStrukturData_07\bin' 'Jobsheet_11.Main_07'
PreOrder Traversal : 6 4 3 5 8 7 9 10 15
InOrder Traversal : 3 4 5 6 7 8 9 10 15
PostOrder Traversal : 3 5 4 7 15 10 9 8 6
Find Node : true
Delete node 8

PreOrder Traversal : 6 4 3 5 9 7 10 15
PS C:\Users\Admin\OneDrive - ypt.or.id\Documents\Kuliah\Semest
 dan Struktur Data\Praktikum_AlgoritmaStrukturData_07>
```

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET XI - TREE

Pertanyaan Percobaan

1. Mengapa dalam binary search tree proses pencarian data bisa lebih efektif dilakukan dibanding binary tree biasa?
Jawab : Dalam binary search tree (BST), data disusun secara terurut sedemikian rupa sehingga semua node di sebelah kiri dari suatu node memiliki nilai yang lebih kecil, sedangkan semua node di sebelah kanan memiliki nilai yang lebih besar. Sifat ini memungkinkan proses pencarian menjadi lebih efisien karena kita dapat membuang setengah dari ruang pencarian pada setiap langkah pencarian.

2. Untuk apakah di class Node, kegunaan dari atribut left dan right?
Jawab : Atribut left dan right digunakan untuk menyimpan referensi (pointer) ke node-node anak di sebelah kiri dan kanan. Setiap node dalam binary tree dapat memiliki maksimal dua anak, yaitu anak di sebelah kiri (left child) dan anak di sebelah kanan (right child). Atribut left dan right digunakan untuk menyimpan referensi ke node-node anak tersebut.

3.
a. Untuk apakah kegunaan dari atribut root di dalam class BinaryTree?
Jawab : Atribut root dalam class BinaryTree digunakan untuk menyimpan referensi (pointer) ke node akar (root node) dari binary tree. Root node merupakan node paling atas dalam hierarki binary tree, dan semua node lainnya merupakan keturunan (descendant) dari root node.
b. Ketika objek tree pertama kali dibuat, apakah nilai dari root?
Jawab : Nilai dari root biasanya diinisialisasi dengan null atau nilai default yang sesuai. Ini karena pada awalnya, tree masih kosong dan belum memiliki node apapun, termasuk node akar.

4. Ketika tree masih kosong, dan akan ditambahkan sebuah node baru, proses apa yang akan terjadi?
Jawab : Ketika tree masih kosong dan akan ditambahkan sebuah node baru, node baru tersebut akan menjadi root node (node akar) dari tree. Dalam hal ini, nilai dari atribut root akan diubah untuk menunjuk ke node baru tersebut.

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

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

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET XI - TREE

```
            current.left = new Node(data);
            break;
        }
}
```

Jawab :

- if(data<current.data) mengecek apakah nilai data yang akan ditambahkan lebih kecil dari nilai data pada node saat ini (current.data). Jika ya, maka node baru harus ditempatkan di subtree sebelah kiri.
- if(current.left!=null) mengecek apakah node saat ini (current) memiliki anak di sebelah kiri. Jika ya, maka pencarian dilanjutkan ke subtree sebelah kiri dengan mengupdate nilai current menjadi current.left.
- current = current.left mengupdate nilai current dengan node anak di sebelah kiri, sehingga pencarian dilanjutkan ke subtree sebelah kiri.
- else bagian ini dieksekusi jika current.left bernilai null, yang berarti tidak ada node anak di sebelah kiri. Dalam kasus ini, node baru dapat ditambahkan sebagai anak kiri dari node saat ini (current).
- current.left = new Node(data) membuat node baru dengan nilai data yang diberikan, dan menetapkannya sebagai anak kiri dari node saat ini (current).
- break digunakan untuk keluar dari loop atau rekursi, karena node baru telah berhasil ditambahkan.

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET XI - TREE

Kegiatan Praktikum 2
Implementasi binary tree dengan array (45 Menit)

src > Jobsheet_11 > BinaryTreeArray_07.java > BinaryTreeArray_07 > BinaryTreeArray_07()

```java
package Jobsheet_11;

public class BinaryTreeArray_07 {
    int[] data;
    int idxLast;

    public BinaryTreeArray_07() {
        data = new int[10];
    }

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

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

src > Jobsheet_11 > ArrayMain_07.java > ArrayMain_07

```java
package Jobsheet_11;

public class ArrayMain_07 {
    Run | Debug
    public static void main(String[] args) {
        BinaryTreeArray_07 bta = new BinaryTreeArray_07();
        int[] data = {6, 4, 8, 3, 5, 7, 9, 0, 0, 0};

        int idxLast = 6;
        bta.populateData(data, idxLast);
        System.out.print(s:"\nInOrder Traversal : ");
        bta.traverseInOrder(idx:0);
        System.out.println(x:"\n");
    }
}
```

Hasil Run Program

```
PS C:\Users\Admin\OneDrive - ypt.or.id\Documents\Kuliah\S
a dan Struktur Data\Praktikum_AlgoritmaStrukturData_07>
n\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-c
pt.or.id\Documents\Kuliah\Semester 2\7. Praktikum Algorit
AlgoritmaStrukturData_07\bin' 'Jobsheet_11.ArrayMain_07'

InOrder Traversal : 3 4 5 6 7 8 9

PS C:\Users\Admin\OneDrive - ypt.or.id\Documents\Kuliah\S
a dan Struktur Data\Praktikum_AlgoritmaStrukturData_07>
```

NAMA : AYLEEN RUHUL QISTHY
NIM : 2341720012
KELAS : 1G
MATERI : JOBSHEET XI - TREE

Pertanyaan Percobaan

1. Apakah kegunaan dari atribut data dan idxLast yang ada di class BinaryTreeArray?
Jawab :
- Atribut data adalah array yang menyimpan elemen-elemen dari binary tree dalam bentuk array. Setiap elemen array mewakili node dalam binary tree dan posisinya di dalam array menentukan hubungannya (parent-child relationship) dengan node lain.
- Atribut idxLast menunjukkan indeks terakhir yang diisi dengan nilai node di dalam array data. Atribut ini digunakan untuk menentukan batas atas saat melakukan operasi pada tree, seperti traversal, sehingga metode dapat berhenti saat mencapai elemen terakhir yang valid.

2. Apakah kegunaan dari method populateData()?
Jawab : Method ini digunakan untuk mengisi data dalam array dengan data yang diberikan sebagai parameter. Method ini juga digunakan untuk menunjukkan indeks terakhir dari array yang digunakan untuk menyimpan data.

3. Apakah kegunaan dari method traverseInOrder()?
Jawab : Method ini digunakan untuk melakukan traversal in-order pada binary tree yang diimplementasikan dalam array. Traversal in-order mengunjungi node dalam urutan left, root, right. Metode ini akan mencetak elemen-elemen tree dalam urutan tersebut mulai dari indeks yang diberikan sebagai parameter.

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 binary tree menggunakan array, jika suatu node berada pada indeks i, maka:
- Left child dari node tersebut akan berada pada indeks 2*i + 1
- Right child dari node tersebut akan berada pada indeks 2*i + 2

Jadi, jika node disimpan dalam array indeks 2, maka:
- Left child akan berada pada indeks 2*2 + 1 = 5
- Right child 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 menginisialisasi nilai dari atribut idxLast pada objek BinaryTreeArray. Nilai 6 menunjukkan bahwa indeks terakhir yang diisi dengan nilai node di dalam array data adalah indeks ke-6. Dengan kata lain, hanya ada 7 node yang diinisialisasi dalam binary tree tersebut (indeks 0 hingga 6).