PRAKTIKUM 25
BINARY TREE

A. TUJUAN

Mahasiswa diharapkan mampu :

1. Memahami konsep dari BinaryTree
2. Memahami cara membangun Binary Tree secara manual
3. Memahami konsep dan implementasi dari menghitung kedalaman, mengkopi dan menghapus Binary Tree

B. DASAR TEORI

Struktur pohon (tree) biasanya digunakan untuk menggambarkan hubungan yang bersifat hirarkis antara elemen-elemen yang ada. Contoh penggunaan struktur pohon :

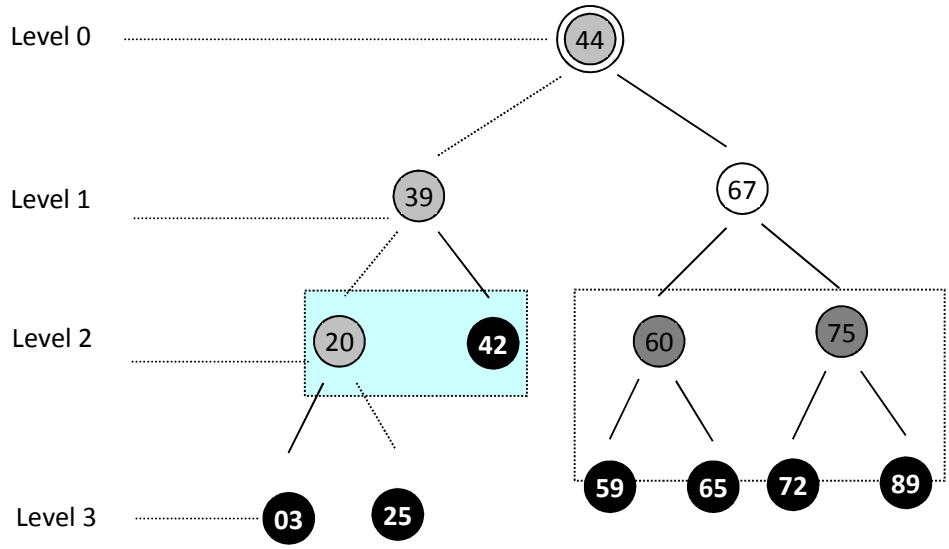
- Silsilah keluarga
- Hasil pertandingan yang berbentuk turnamen
- Struktur organisasi dari sebuah perusahaan

B.1 Pohon Biner (*Binary Tree*)

Sebuah binary tree adalah sebuah pengorganisasian secara hirarki dari beberapa buah simpul, dimana masing-masing simpul tidak mempunyai anak lebih dari 2. Simpul yang berada di bawah sebuah simpul dinamakan anak dari simpul tersebut. Simpul yang berada di atas sebuah simpul dinamakan induk dari simpul tersebut.

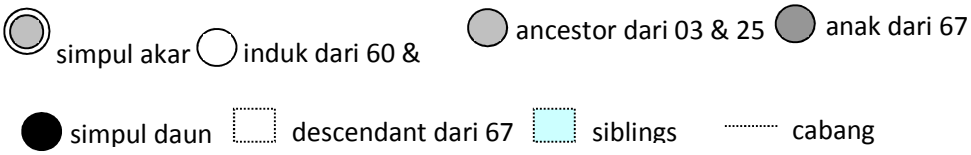
Masing-masing simpul dalam binary tree terdiri dari tiga bagian yaitu sebuah data dan dua buah pointer yang dinamakan pointer kiri dan kanan. Simpul juga mempunyai *sibling*, *descendants*, dan *ancestors*. ***Sibling*** dari sebuah simpul adalah anak lain dari induk simpul tersebut. ***Descendants*** dari sebuah simpul adalah semua simpul-simpul merupakan cabang (berada di bawah) simpul tersebut. ***Anchestors*** dari sebuah simpul adalah semua simpul yang berada di atas antara simpul tersebut dengan *root*.

Level suatu simpul ditentukan dengan menentukan akar sebagai level 0. Jika suatu simpul dinyatakan sebagai tingkat N, maka simpul-simpul yang merupakan anaknya akan berada pada tingkatan N + 1. Tinggi atau kedalaman dari suatu pohon adalah level maksimum dari simpul dalam pohon tersebut.



Gambar 16.1 Pohon Biner (*Binary Tree*) dengan kedalaman 3

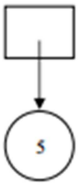
Keterangan:



Binary Tree adalah struktur data yang hampir mirip juga dengan Linked List untuk menyimpan koleksi dari data. Linked List dapat dianalogikan sebagai rantai linier sedangkan Binary Tree bisa digambarkan sebagai rantai tidak linier. Binary Tree dikelompokkan menjadi unordered Binary Tree (tree yang tidak berurut) dan ordered Binary Tree (tree yang terurut). Binary Tree dapat digambarkan berdasarkan kondisinya, sebagai berikut

Binary Tree kosong

Gambar 1. Binary Tree dalam kondisi kosong

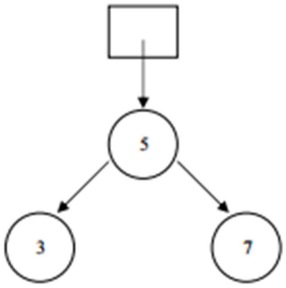


Pointer ke akar (root) dari tree

Gambar 2. Binary Tree berisi satu node yaitu root

Binary Tree sebagai root sekaligus sebagai daun (leaf)

Gambaran dari Binary Tree yang terdiri dari 3 (tiga) node:



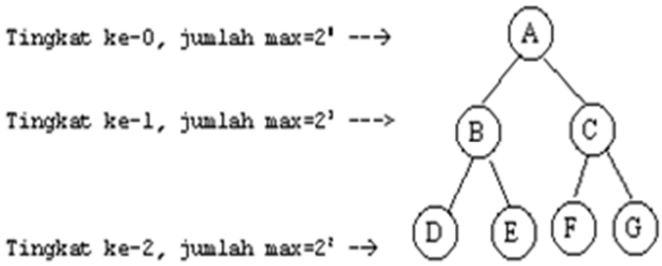
Pointer ke root dari tree

Leaf dari tree

Gambar 3. Binary Tree

Node pada binary tree

- Jumlah maksimum node pada setiap tingkat adalah 2^n



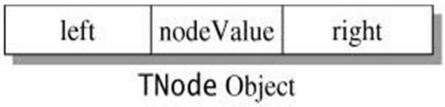
Gambar 4. Jumlah Node di Binary Tree

- Node pada binary tree maksimum berjumlah 2^n-1

Implementasi Binary Tree

Pada Binary Tree sebuah node direpresentasikan oleh class TNode. Sebuah Node terdiri dari 3 bagian yaitu

- Data value disebut nodeValue
- Variabel reference left sebagai anak kiri
- Variabel reference right sebagai anak kanan



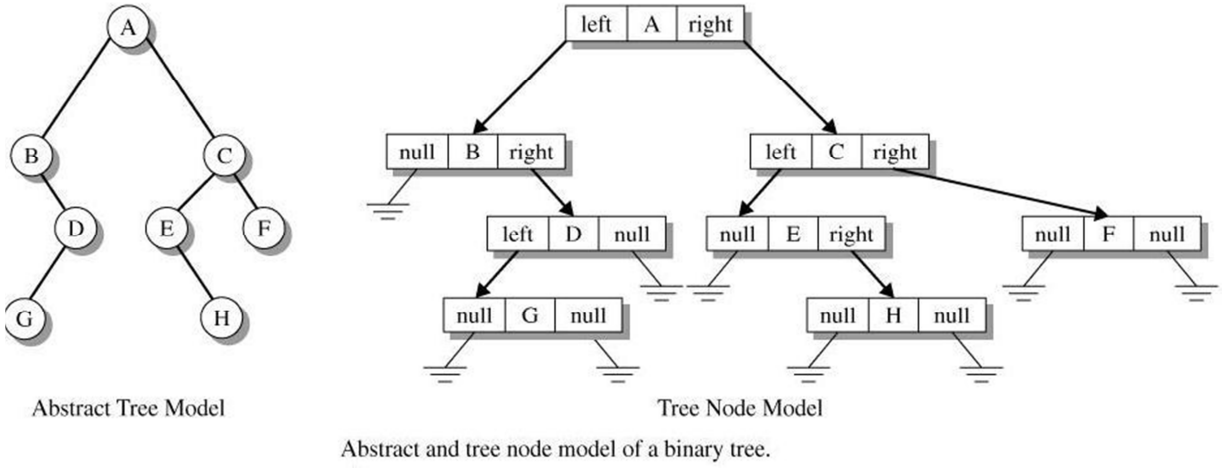

```
public class TNode<T>
{
   // data dari sebuah node
   public T nodeValue;

   // menyimpan alamat dari anak kiri dan anak kanan
   public TNode<T> left, right;

   // membuat obyek dengan sebuah data item
   // dan anak kiri dan anak kanan diset null
   public TNode(T item)
   {
      nodeValue = item;
      left = right = null;
   }

   // membuat obyek dengan sebuah data item
   // dan menentukan alamat dari anak kiri dan anak kanan
   public TNode (T item, TNode<T> left, TNode<T> right)
   {
      nodeValue = item;
      this.left = left;
      this.right = right;
   }
```

Sehingga sebuah Tree direpresentasikan pada gambar5.



Gambar 5. Binary Tree dan representasi dalam TNode

C. TUGAS PENDAHULUAN

Jelaskan istilah-istilah di bawah ini !

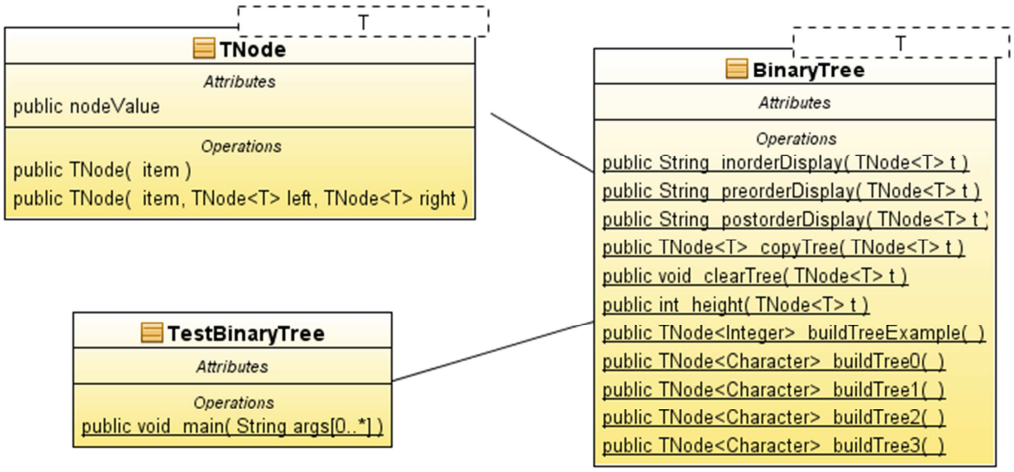
1. Binary Tree
2. Root
3. Parent
4. Child
5. Sibling
6. Leaf Node
7. Level Node
8. Kedalaman Binary Tree

D. PERCOBAAN

Pada percobaan dan latihan ini seperti ditunjukkan pada gambar 6, terdapat Class TNode, Class Binary Tree dan Class Test BinaryTree. Class TNode merepresentasikan sebuah Node pada Binary Tree. Pada Class Binary Tree terdapat method :

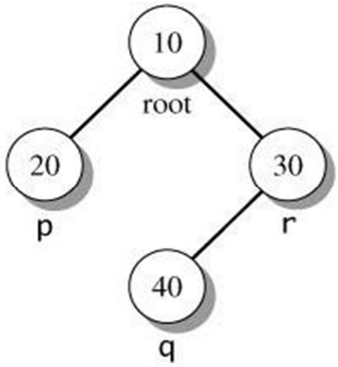
- preorderDisplay(TNode<T> t) : untuk traversal Tree dengan cara Preorder
- inorderDisplay(TNode<T> t) : untuk traversal Tree dengan cara Inorder
- postorderDisplay(TNode<T> t) : untuk traversal Tree dengan cara Postorder
- buildTreeExample(), buildTree0(),buildTree1(),buildTree2(),buildTree3() : method untuk membangun tree berdasarkan studi kasus.
- copyTree(TNode<T> t) : untuk mengkopi sebuah Binary Tree
- clearTree(TNode<T> t) : untuk menghapus Binary Tree
- height(TNode<T> t) : untuk menghitung kedalaman Binary Tree

Class Test BinaryTree untuk menguji class Binary Tree yang sudah dibuat.



Gambar 6. Class Diagram dari praktikum Binary Tree

Percobaan 1 : Membangun Binary Tree seperti gambar 6 secara manual



Gambar 6. Contoh Binary Tree

```
public class BinaryTree<T> {
    public static TNode<Integer> buildTreeExample() {
        TNode<Integer> root, p, q, r;
        p = new TNode<Integer>(20);
        q = new TNode<Integer>(40);
        r = new TNode<Integer>(30, q, null);
        root = new TNode<Integer>(10, p, r);
        return root;
    }
}
```

Percobaan 2 : Untuk mendapatkan kedalaman dari Binary Tree

```
public class BinaryTree<T> {

    public static <T> int height(TNode<T> t) {
        int heightLeft, heightRight, heightval;

        if (t == null) // tinggi dari tree kosong adalah -1
```

```
        {
            heightval = -1;
        } else {
            // mendapatkan kedalaman dari subtree kiri dari t
            heightLeft = height(t.left);
            // mendapatkan kedalaman dari subtree kanan dari t
            heightRight = height(t.right);
            //kedalaman dari tree dari root t adalah 1 + maximum
            // dari kedalaman dua subtree t
            heightval = 1
                    + (heightLeft > heightRight ? heightLeft
                    : heightRight);
        }

        return heightval;
    }
}
```

Percobaan 3 : Untuk mengkopi Binary Tree

```
public class BinaryTree<T> {
    public static <T> TNode<T> copyTree(TNode<T> t) {
         TNode<T> newLeft, newRight, newNode;
        if (t == null) {
            return null;
        }
        newLeft = copyTree(t.left);
        newRight = copyTree(t.right);
        newNode = new TNode<T>(t.nodeValue, newLeft, newRight);
        return newNode;
    }
}
```

Percobaan 4: Untuk menghapus Binary Tree

```
public class BinaryTree<T> {

   public static <T> void clearTree(TNode<T> t) {
        if (t != null) {
            clearTree(t.left);
            clearTree(t.right);
            t = null;
        }
    }
}
```

Selanjutnya ujilah method height(), copyTree(TNode<T> t), clearTree(TNode<T> t). Sebagai contoh untuk menghitung kedalaman Binary Tree pada gambar 6 adalah sebagai berikut:

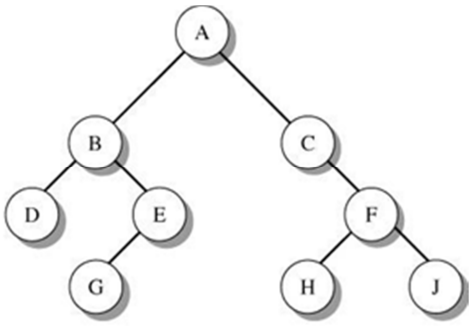
```
public class TestBinaryTree {
    public static void main(String[] args) {
     TNode<Character> root = BinaryTree.buildTree0();
```

```
     System.out.println("Kedalaman Binary Tree : " +
BinaryTree.height(root));
    }
}
```

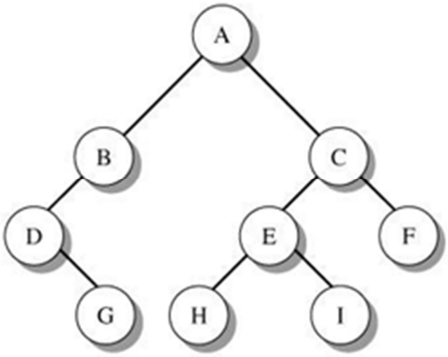
E. LATIHAN

Latihan 1 : Buatlah diagram rekursif untuk menghitung kedalaman Binary Tree pada gambar 6.

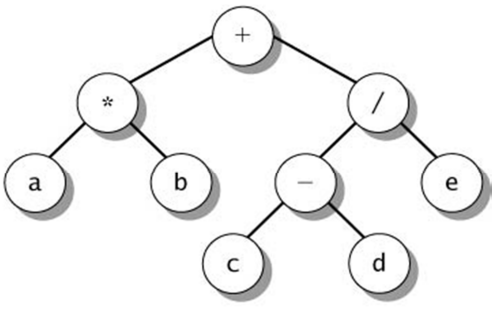
Latihan 2 : Untuk studi kasus pada gambar 7, gambar 8 dan gambar 9, buatlah Binary Tree secara manual



Gambar 7. Binary Tree Studi Kasus 1



Gambar 8. Binary Tree Studi Kasus 2



Gambar 9. Binary Tree Studi Kasus 3

Latihan 3 : Untuk studi kasus 1, 2 dan 3. Buatlah Class Main untuk menguji method height(), copyTree(TNode<T> t), clearTree(TNode<T> t).

F. LAPORAN RESMI

Kerjakan hasil percobaan(D) dan latihan(E) di atas dan tambahkan analisa.