



Struktur Data

Pendidikan Teknik Informatika

Universitas Negeri Padang

TIM DOSEN ©2022

# JOBSHEET (JS-06)

## Binary Tree

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Pendidikan Teknik Informatika	TIK1.61.2317	4 x 50 Menit

### **TUJUAN PRAKTIKUM**

- 
1. Mahasiswa mampu mengaplikasikan konsep binary tree dalam menyelesaikan kasus.

### **HARDWARE & SOFTWARE**

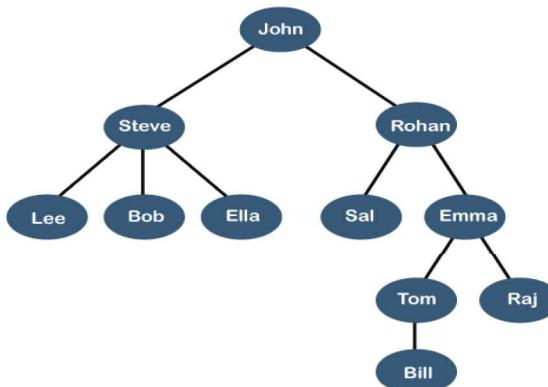
---

1. Personal Computer
  2. IDE: Dev C++
- 

### **TEORI SINGKAT**

#### **A. Tree**

*Tree* merupakan salah satu struktur data yang mewakili data hierarkis. Misalkan kita ingin menunjukkan karyawan dan posisinya dalam bentuk hierarki maka dapat direpresentasikan seperti gambar berikut :



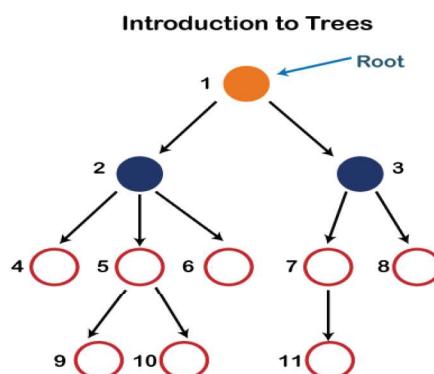
Tree tersebut menunjukkan hierarki organisasi dari beberapa perusahaan. Dalam struktur di atas, John adalah CEO perusahaan, dan John memiliki dua bawahan langsung bernama Steve dan Rohan. Steve memiliki tiga bawahan langsung bernama Lee, Bob, Ella di mana Steve adalah manajernya. Bob memiliki dua bawahan langsung bernama Sal dan Emma. Emma memiliki dua bawahan langsung bernama Tom dan Raj. Tom memiliki satu bawahan langsung bernama Bill. Struktur logis khusus ini dikenal sebagai Tree. Strukturnya mirip dengan pohon asli, sehingga dinamai Tree. Dalam struktur ini, akar berada di atas, dan cabang-cabangnya bergerak ke bawah. Oleh karena itu, kita dapat mengatakan bahwa struktur data Tree adalah cara yang efisien untuk menyimpan data secara hierarkis.

#### **Beberapa poin penting dari struktur data Tree :**

1. Struktur data tree didefinisikan sebagai kumpulan objek atau entitas yang dikenal sebagai node yang dihubungkan bersama untuk mewakili atau mensimulasikan hierarki.
2. Struktur data tree adalah struktur data non-linier karena tidak menyimpan secara berurutan. Ini adalah struktur hierarkis karena elemen dalam Tree disusun dalam berbagai level.
3. Dalam struktur data Tree, node paling atas dikenal sebagai root node. Setiap node berisi beberapa data, dan data dapat berupa jenis apa pun. Dalam struktur tree di atas, node berisi nama karyawan, sehingga tipe datanya adalah string.
4. Setiap node berisi beberapa data dan link atau referensi dari node lain yang dapat disebut anak.

#### **Beberapa istilah dasar yang digunakan dalam struktur data Tree :**

Mari kita perhatikan struktur tree, yang ditunjukkan di bawah ini:



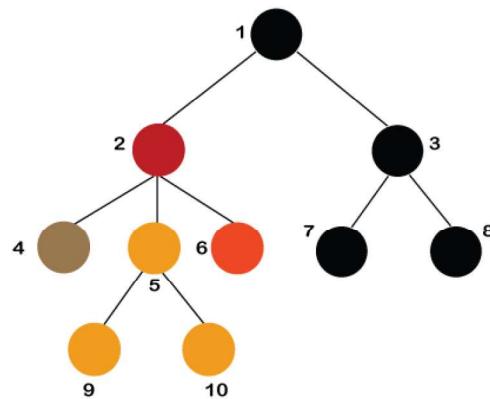
Dalam struktur tersebut, setiap node diberi label dengan beberapa nomor. Setiap panah yang ditunjukkan pada gambar di atas dikenal sebagai ***penghubung*** antara dua simpul.

1. **Root:** Node root adalah node paling atas dalam hierarki tree. Dengan kata lain, simpul akar adalah simpul yang tidak memiliki orang tua. Pada struktur di atas, simpul bernomor 1 adalah **the root node of the tree**. Jika sebuah simpul terhubung langsung ke beberapa simpul lain, itu akan disebut hubungan induk-anak.
2. **Child node:** Jika node adalah keturunan dari setiap node, maka node tersebut dikenal sebagai node anak.
3. **Parent:** Jika node berisi sub-node, maka node tersebut dikatakan sebagai parent dari sub-node tersebut.
4. **Sibling:** Node yang memiliki orang tua yang sama dikenal sebagai saudara kandung.
5. **Leaf Node:** - Node tree, yang tidak memiliki node anak, disebut leaf node. Simpul daun adalah simpul paling bawah dari tree. Bisa ada sejumlah simpul daun yang ada di tree umum. Node daun juga bisa disebut node eksternal.
6. **Internal nodes:** Sebuah node memiliki setidaknya satu node anak yang dikenal sebagai ***internal***
7. **Ancestor node:** - An ancestor dari sebuah node adalah setiap node pendahulu pada sebuah path dari root ke node tersebut. Node root tidak memiliki ancestor. Pada tree yang ditunjukkan pada gambar di atas, node 1, 2, dan 5 adalah nenek moyang dari node 10.
8. **Descendant:** Penerus langsung dari node yang diberikan dikenal sebagai keturunan dari sebuah node. Pada gambar di atas, 10 adalah turunan dari simpul 5.

## B. Properti struktur data Tree

1. **Struktur data rekursif:** Tree ini juga dikenal sebagai ***struktur data rekursif***. Sebuah tree dapat didefinisikan secara rekursif karena simpul yang dibedakan dalam struktur data tree dikenal sebagai ***simpul akar***. Node akar tree berisi tautan ke semua akar subtreenya. Subtree kiri ditunjukkan dengan warna kuning pada gambar di bawah, dan subtree kanan ditunjukkan dengan warna

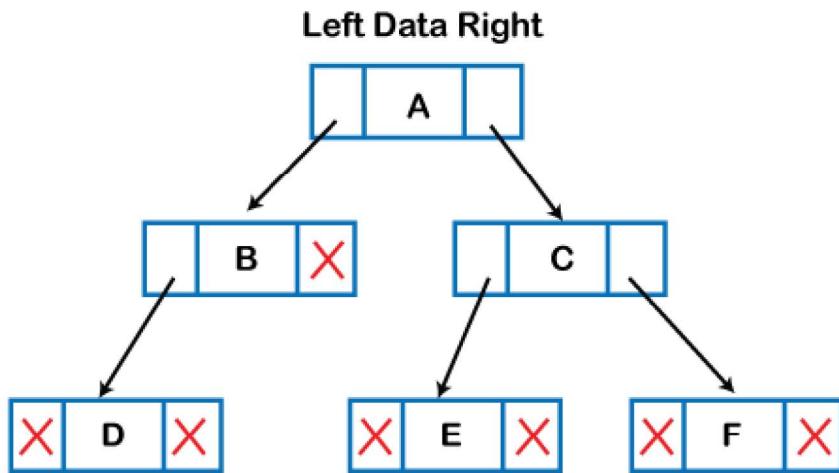
merah. Subtree kiri dapat dibagi lagi menjadi subtree yang ditampilkan dalam tiga warna berbeda. Rekursi berarti mengurangi sesuatu dengan cara yang serupa. Jadi, properti rekursif dari struktur data tree ini diimplementasikan dalam berbagai aplikasi.



2. **Jumlah tepi:** Jika ada  $n$  simpul, maka akan ada  $n-1$  tepi. Setiap panah dalam struktur mewakili tautan atau jalur. Setiap node, kecuali node root, akan memiliki setidaknya satu link masuk yang dikenal sebagai edge. Akan ada satu tautan untuk hubungan orangtua-anak.
3. **Kedalaman simpul  $x$ :** Kedalaman simpul  $x$  dapat didefinisikan sebagai panjang jalur dari akar ke simpul  $x$ . Satu sisi memberikan kontribusi satu satuan panjang di jalan. Jadi, kedalaman simpul  $x$  juga dapat didefinisikan sebagai jumlah sisi antara simpul akar dan simpul  $x$ . Node akar memiliki 0 kedalaman.
4. **Ketinggian simpul  $x$ :** Ketinggian simpul  $x$  dapat didefinisikan sebagai jalur terpanjang dari simpul  $x$  ke simpul daun.

### C. Implementasi Tree

Struktur data tree dapat dibuat dengan membuat node secara dinamis dengan bantuan pointer. Tree dalam memori dapat direpresentasikan seperti yang ditunjukkan di bawah ini:



Gambar di atas menunjukkan representasi struktur data tree dalam memori. Dalam struktur di atas, simpul berisi tiga bidang. Bidang kedua menyimpan data; bidang pertama menyimpan alamat anak kiri, dan bidang ketiga menyimpan alamat anak kanan. Dalam pemrograman, struktur node dapat didefinisikan sebagai:

```
struct node
{
    int data;
    struct node *left;
    struct node *right;
}
```

Struktur di atas hanya dapat didefinisikan untuk tree biner karena tree biner dapat memiliki paling banyak dua anak, dan tree generik dapat memiliki lebih dari dua anak. Struktur simpul untuk tree generik akan berbeda dibandingkan dengan tree biner.

#### D. Aplikasi tree

Berikut ini adalah aplikasi dari tree:

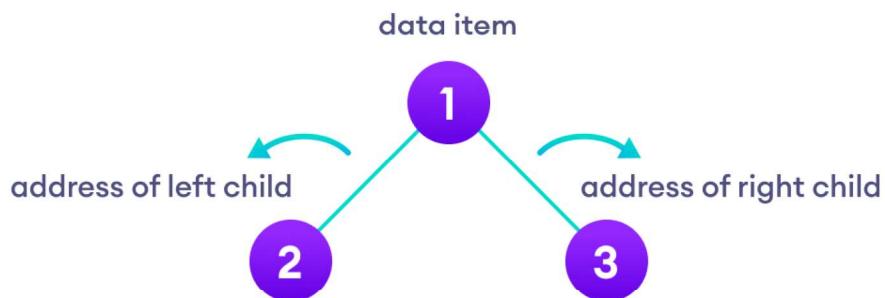
1. **Menyimpan data hierarkis secara alami:** Tree digunakan untuk menyimpan data dalam struktur hierarkis. Misalnya, sistem file. Sistem file yang disimpan di drive disk, file dan folder dalam bentuk data hierarkis alami dan disimpan dalam bentuk tree.
2. **Atur data:** Digunakan untuk mengatur data untuk penyisipan, penghapusan, dan pencarian yang efisien. Misalnya, tree biner memiliki waktu  $\log N$  untuk mencari elemen.

3. **Trie:** Ini adalah jenis tree khusus yang digunakan untuk menyimpan kamus. Ini adalah cara yang cepat dan efisien untuk pemeriksaan ejaan dinamis.
4. **Heap:** Ini juga merupakan struktur data tree yang diimplementasikan menggunakan array. Ini digunakan untuk mengimplementasikan antrian prioritas.
5. **B-Tree dan B+Tree:** B-Tree dan B+Tree adalah struktur data tree yang digunakan untuk mengimplementasikan pengindeksan dalam database.
6. **Tabel perutean:** Struktur data tree juga digunakan untuk menyimpan data dalam tabel perutean di router.

#### E. Binary Tree

Pohon biner adalah struktur data pohon di mana setiap simpul induk dapat memiliki paling banyak dua anak. Setiap node dari pohon biner terdiri dari tiga item:

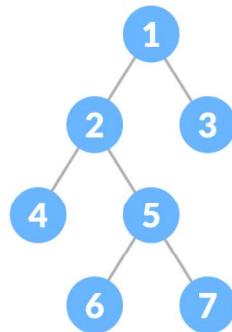
1. data item
2. address of left child
3. address of right child



### Jenis Binary Tree :

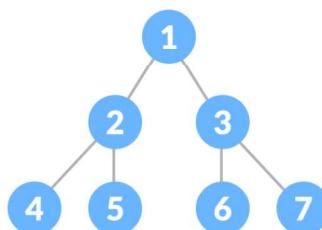
#### 1. Full Binary Tree

Full binary tree adalah jenis binary tree khusus di mana setiap simpul induk/simpul internal memiliki dua atau tidak memiliki anak.



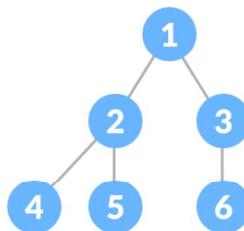
#### 2. Perfect Binary Tree

Perfect binary tree adalah jenis binary tree di mana setiap simpul internal memiliki tepat dua simpul anak dan semua simpul daun berada pada level yang sama.



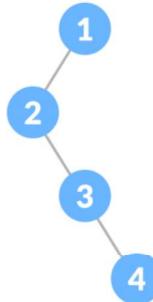
#### 3. Complete Binary Tree

Complete binary tree seperti full binary tree, tetapi dengan dua perbedaan utama yaitu setiap level harus terisi penuh dan semua elemen daun harus condong ke kiri. Elemen daun terakhir mungkin tidak memiliki saudara kandung yang benar yaitu complete binary tree tidak harus full binary tree.



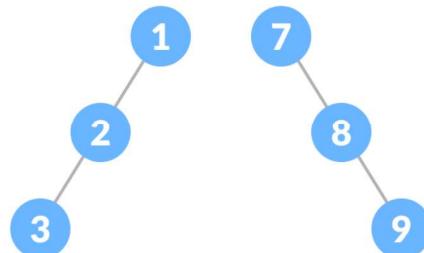
#### 4. Degenerate or Pathological Binary Tree

Degenerate or pathologi tree adalah pohon yang memiliki anak tunggal baik kiri atau kanan.



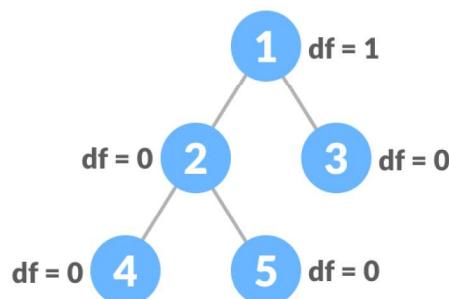
#### 5. Skewed Binary Tree

Skewed binary tree adalah pohon patologis/degenerasi di mana pohon tersebut didominasi oleh simpul kiri atau simpul kanan. Jadi, ada dua jenis pohon biner miring: pohon biner miring **kiri** dan pohon biner miring **kanan**.



#### 6. Balanced Binary Tree

Ini adalah jenis pohon biner di mana perbedaan antara ketinggian kiri dan kanan subpohon untuk setiap node adalah 0 atau 1.



## LATIHAN

### 1. Binary tree

```
/* Nama File : binary tree 1
Pembuat : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/

#include <stdio.h>
#include <malloc.h>

struct nod
{
    struct nod *left;
    char data;
    struct nod *right;
};

typedef struct nod NOD;
typedef NOD POKOK;

NOD *NodBaru(char item)
{
    NOD *n;
    n = (NOD*) malloc(sizeof(NOD));
    if(n != NULL)
    {
        n->data = item;
        n->left = NULL;
        n->right = NULL;
    }
    return n;
}

void BinaPokok(POKOK **T)
{
    *T = NULL;
}

typedef enum
{
    FALSE = 0, TRUE = 1
} BOOL;
```

```

BOOL PokokKosong(POKOK *T)
{
    return((BOOL)(T == NULL));
}

void TambahNod(NOD **p, char item)
{
    NOD *n;
    n = NodBaru(item);
    *p = n;
}

void preOrder(POKOK *T)
{
    if(!PokokKosong(T))
    {
        printf("%c ", T->data);
        preOrder(T->left);
        preOrder(T->right);
    }
}

void inOrder(POKOK *T)
{
    if(!PokokKosong(T))
    {
        inOrder(T->left);
        printf("%c ", T->data);
        inOrder(T->right);
    }
}

void postOrder(POKOK *T)
{
    if(!PokokKosong(T))
    {
        postOrder(T->left);
        postOrder(T->right);
        printf("%c ", T->data);
    }
}

int main()

```

```
{  
    POKOK *kelapa;  
    char buah;  
    BinaPokok(&kelapa);  
    TambahNod(&kelapa, buah = 'M');  
    TambahNod(&kelapa->left, buah = 'E');  
    TambahNod(&kelapa->left->right, buah = 'I');  
    TambahNod(&kelapa->right, buah = 'L');  
    TambahNod(&kelapa->right->right->left, buah = 'O');  
    TambahNod(&kelapa->right->right->right, buah = 'D');  
    printf("Tampilan secara PreOrder: ");  
    preOrder(kelapa);  
    printf("\nTampilan secara InOrder: ");  
    inOrder(kelapa);  
    printf("\nTampilan secara PreOrder: ");  
    postOrder(kelapa);  
    printf("\n\n");  
  
    return 0;  
}
```

## 2. Binary tree

```
/* Nama File : binary tree 2  
Pembuat : tuliskan nama dan NIM anda  
Tgl pembuatan : tuliskan tanggal hari ini*/  
  
#include <stdio.h>  
#include <stdlib.h>  
  
struct data{  
    int number;  
    data *left, *right;  
}*root;  
  
void push(data **current, int number){  
    if((*current)==NULL){  
        (*current) = (struct data *)malloc(sizeof  
(struct data));  
        (*current)->number=number;  
        (*current)->left = (*current)->right = NULL;  
    }else if(number < (*current)->number){  
        push(&(*current)->left, number);  
    }else if(number >= (*current)->number){
```

```

        push(&(*current)->right, number);
    }
}

void inOrder(data **current) {
    if((*current)!=NULL) {
        inOrder(&(*current)->left);
        printf("%d -> ", (*current)->number);
        inOrder(&(*current)->right);
    }
}

void preOrder(data **current) {
    if((*current)!=NULL) {
        printf("%d -> ", (*current)->number);
        preOrder(&(*current)->left);
        preOrder(&(*current)->right);
    }
}

void postOrder(data **current) {
    if((*current)!=NULL) {
        postOrder(&(*current)->left);
        postOrder(&(*current)->right);
        printf("%d -> ", (*current)->number);
    }
}

void search(data **current, int number) {
    if((*current)!=NULL) {
        if(number<(*current)->number) {
            search(&(*current)->left,number);
        }else if(number>(*current)->number) {
            search(&(*current)->right,number);
        }else{
            printf("Found : %d", (*current)->number);
        }
    }else{
        printf("Not Found.");
    }
}

int main(){
    push(&root, 11);
}

```

```
    push(&root, 22);
    push(&root, 13);
    push(&root, 15);
    push(&root, 9);
    inOrder(&root);
    printf("\n");
    preOrder(&root);
    printf("\n");
    postOrder(&root);
    printf("\n");
    search(&root, 91);
    getchar();
}
```

## TUGAS

Buatlah program untuk menampilkan node baru ke dalam pohon dengan menggunakan prosedur preorder, inorder, dan postorder.

Sehingga akan didapatkan hasil :

Tampilan secara PreOrder : R A S I T E

Tampilan secara InOrder : I S T A R E

Tampilan secara PostOrder : I T S A F E R

## DAFTAR PUSTAKA

Kelley, Al and Pohl, Ira. 2003. *C by Dissection: The Essentials of C Programming*. Addison-Weasley.

Khannedy, Eko Kurniawan. 2007. *Diktat Pemrograman C*. Bandung : Unikom.

Liem, Ingriani. 2003. *Catatan Singkat Bahasa C*. Jurusan Teknik Informatika. Institut Teknologi Bandung.

Reema Thareja. 2014. *Data Structures Using C*. India : Oxford University Press.

Solichin, Achmad. 2003. *Pemrograman Bahasa C dengan Turbo C*. Kuliah Berseri ilmukomputer.com