Pertemuan 12

Binary Search Tree

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- **mendemonstrasikan operasi pada Binary Search Tree.**
- **Menerapkan Binary Search Tree pada program aplikasi komputer.**

Outline Materi

- Karakteristik
- Operasi
  - Insert
  - Delete
  - Update
- Representasi
- Implementasi

Karakteristik

Binary Search Tree (BST) adalah Binary Tree yang terurut, dengan ketentuan :

- **semua LEFT CHILD harus lebih kecil dari PARENT dan RIGHT CHILD.**
- **semua RIGHT CHILD harus lebih besar dari PARENT dan LEFT CHILD.**

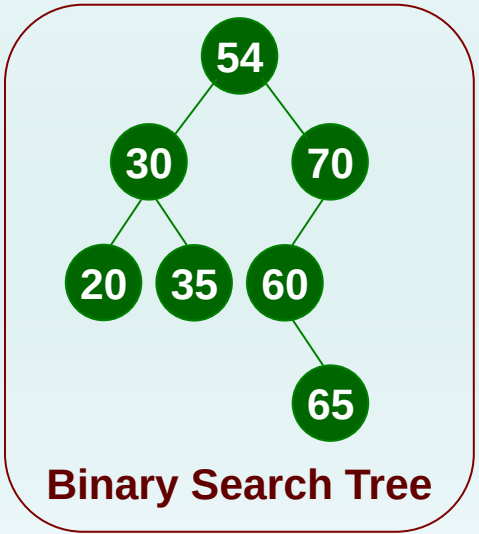
Target NODE :

Node yang diinginkan/dicari

Keuntungan :

Searching/Pencarian Target Node menjadi lebih efisien dan cepat.

Contoh :



Binary Search Tree

Operasi

Semua operasi pd Binary Tree bisa diimplementasikan langsung pd BST,kecuali:

- **INSERT()**
- **UPDATE()**
- **DELETEKEY()**

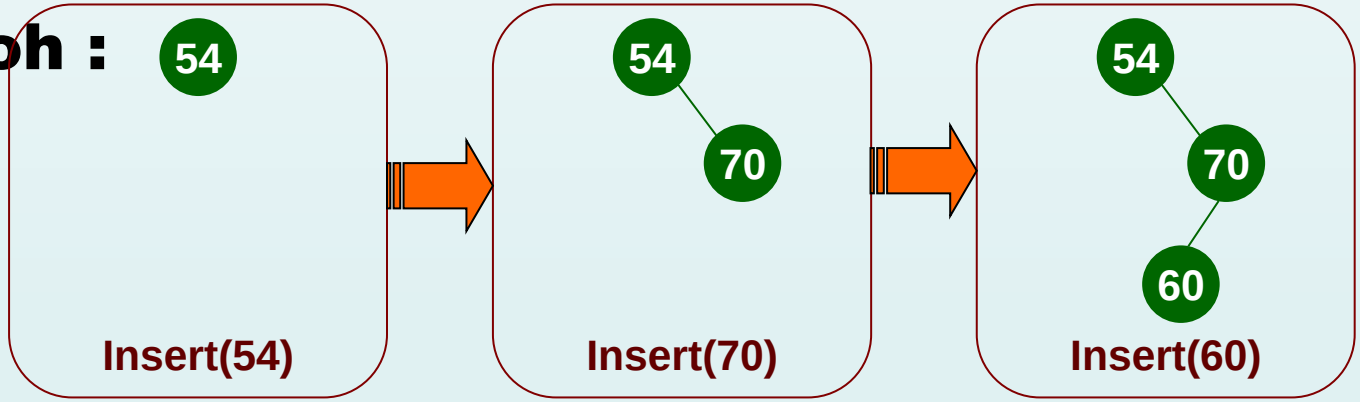
Untuk ketiga operasi tsb perlu dilakukan modifikasi terhadap posisi node sehingga BST tetap terurut.

Insert

Langkah

- **Pencarian lokasi utk node yg akan diinsert (baru) selalu dimulai dr ROOT.**
- **Jika node baru < ROOT,maka insert pd LEFT SUBTREE.**
- **Jika node baru > ROOT,maka insert pd RIGHT SUBTREE.**

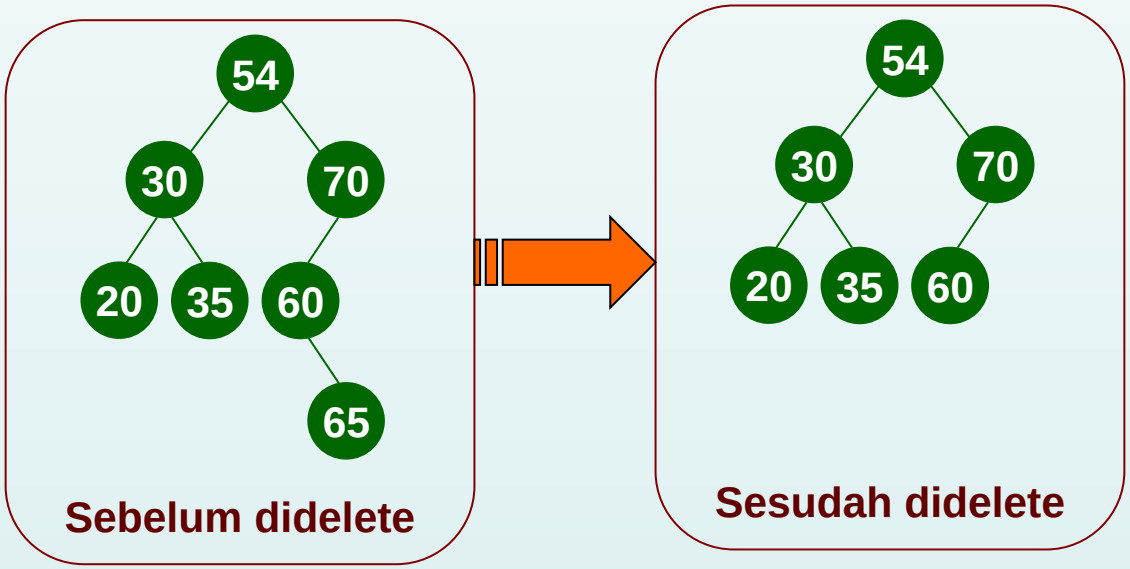
Contoh :



Delete

Jika yang dihapus adalah LEAF maka tidak perlu dilakukan modifikasi terhadap lokasi.

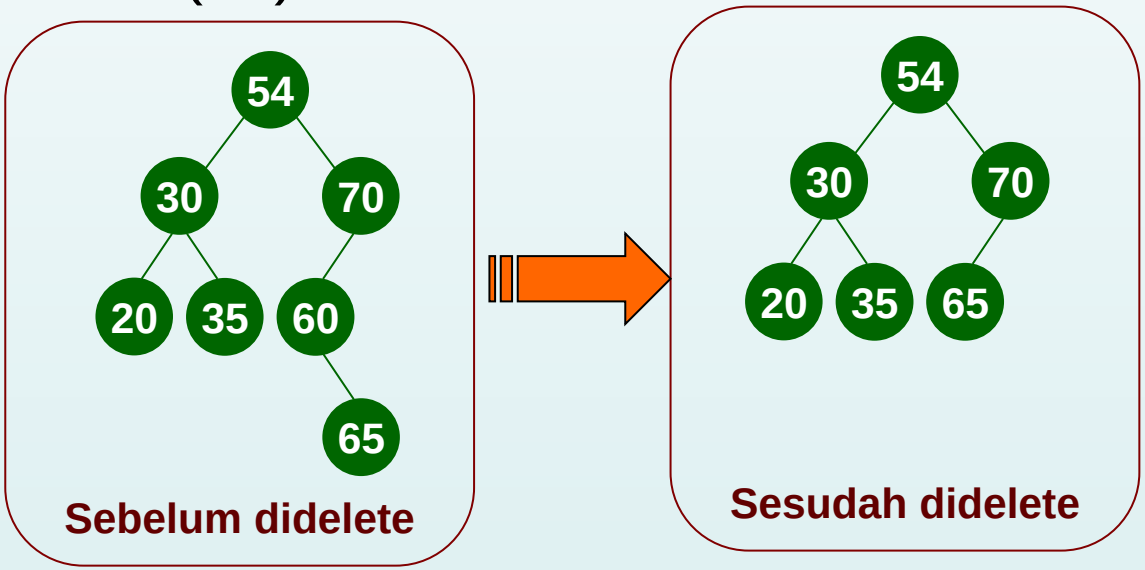
Contoh : Delete(65)



Delete(2)

Jika yang dihapus adalah NODE yang hanya memiliki 1 child, maka child tersebut langsung menggantikan posisi dari parentnya..

Contoh : Delete(60)

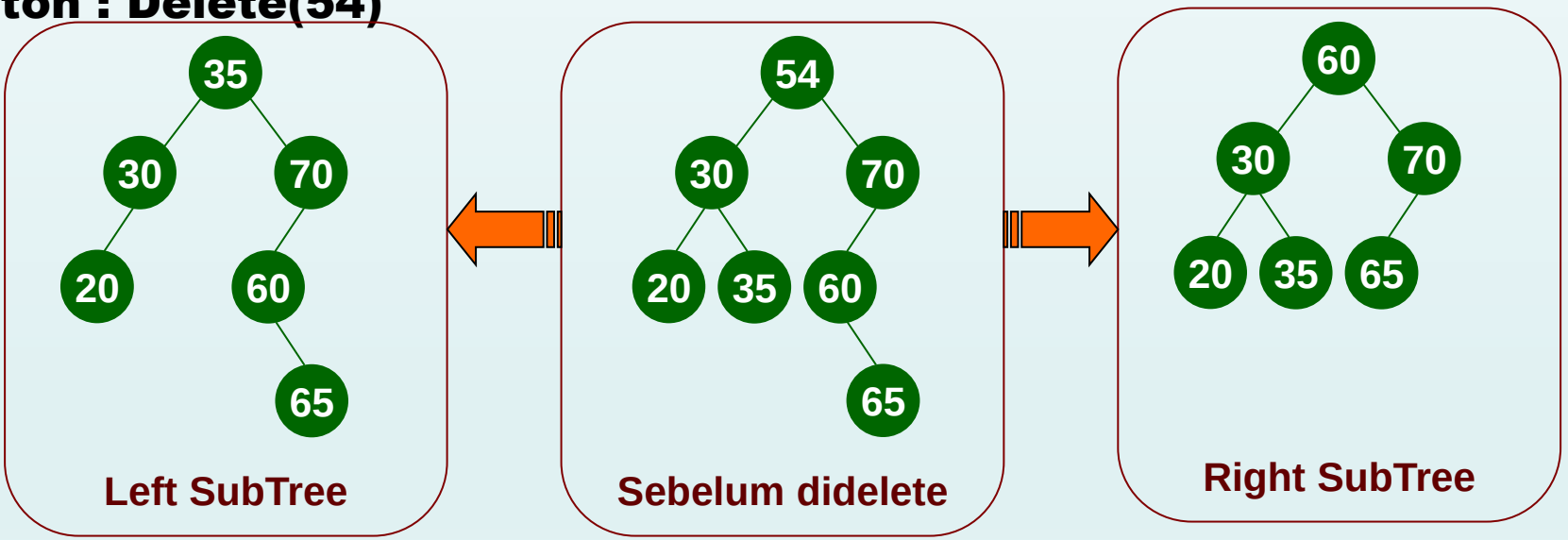


Delete (3)

Jika yang dihapus adalah NODE dengan 2 children (2 SUBTREE), maka node yang diambil untuk menggantikan posisi node yang dihapus adalah :

- **berasal dari LEFT SUBTREE, yang diambil adalah node yang paling kanan (yang mempunyai nilai terbesar).**
- **atau dari RIGHT SUBTREE, yang diambil adalah node yang paling kiri (yang mempunyai nilai terkecil).**

Contoh : Delete(54)



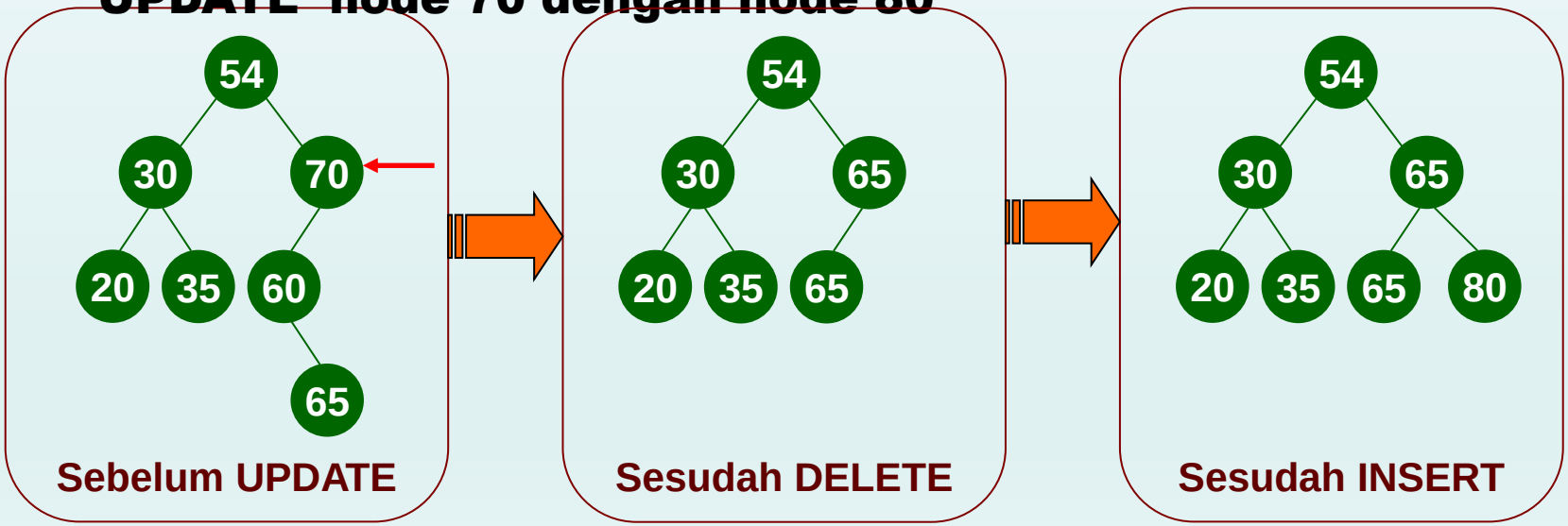
Update

Update terhadap suatu node akan mempengaruhi lokasi node tersebut setelah diupdate. Bila node tersebut setelah diupdate menimbulkan Tree yang bukan BST, maka harus diregenerate Tree tersebut.

Pendekatan yang lebih sederhana untuk menyelesaikan operasi UPDATE adalah dengan penggabungan antara operasi DELETE dengan INSERT.

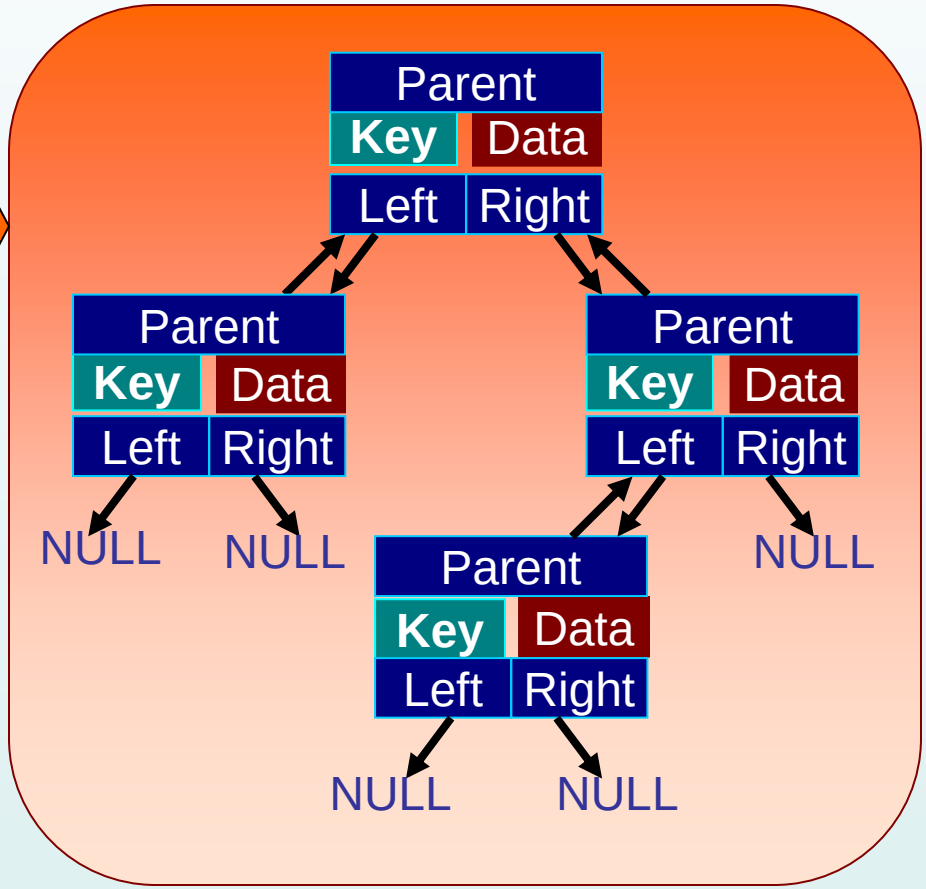
Contoh :

UPDATE node 70 dengan node 80



Representasi

```
struct node {
        Key_Type Key;
        Elemen_Type Data;
        struct node *Left;
        struct node *Right;
        struct node *Parent;
};
```



Implementasi

```
#include <stdio.h>
#include <stdlib.h>

#define compLT(a,b) (a < b)
#define compEQ(a,b) (a == b)

typedef enum {
    STATUS_OK,
    STATUS_MEM_EXHAUSTED,
    STATUS_DUPLICATE_KEY,
    STATUS_KEY_NOT_FOUND
} statusEnum;

typedef int keyType;
typedef int ElmDataType;
```

```
typedef struct nodeTag {
    struct nodeTag *left;
    struct nodeTag *right;
    struct nodeTag *parent;
    keyType key;
    ElmDataType ElmData;
} nodeType;

nodeType *root = NULL;
```

Dikutip dari :
http://ciips.ee.uwa.edu.au/~morris/Year2/PLDS210/niemann/s_man.htm

Implementasi(2)

```
statusEnum insert(keyType key, ElmDataType *ElmData) {
       nodeType *x, *current, *parent;

/* find future parent */
    current = root;
    parent = 0;
    while (current) {
        if (compEQ(key, current->key))
            return STATUS_DUPLICATE_KEY;
        parent = current;
        current = compLT(key, current->key) ?
            current->left : current->right;    }

    /* setup new node */
    if ((x = malloc (sizeof(*x))) == 0) { return STATUS_MEM_EXHAUSTED;  }

    x->parent = parent;
    x->left = NULL;
    x->right = NULL;
    x->key = key;
    x->ElmData = *ElmData;

    /* insert x in tree */
    if(parent)
        if(compLT(x->key, parent->key)) parent->left = x; else parent->right = x;
    else        root = x;
    return STATUS_OK; }
```

Latihan

Jika + berarti Insert, maka buatlah Binary Search Tree (BST) dari operasi-operasi berikut ini :

+ 76 + 35 + 90 + 50 + 25

+ 150 + 100 + 80 + 40 + 50

+ 125 + 60 + 75 + 140 + 55

+ 140 + 20 + 120 + 66 + 82

Latihan

Jika + berarti Insert, dan – berarti Delete, maka buatlah Binary Search Tree (BST) dari operasi-operasi berikut ini :

+ 76 + 35 + 90 + 50 + 25

+ 150 + 100 + 80 + 40 - 50

+ 125 + 60 - 76 + 140 + 55

+ 140 - 25 + 120 - 60 - 80

Selesai