Laporan Praktikum
Algoritma dan Struktur Data

AVL Tree

Nama: Ruth Aulya Silalahi
NIM: 11S20018
Program Studi: Informatika

INSTITUT TEKNOLOGI DEL
FAKULTAS INFORMATIKA DAN TEKNIK ELEKTRO

Tugas

1. Tulis kode program AVL Tree (Anda bisa menggunakan program BST), gunakan kode program AvlNode yang ada di slide presentasi dan implementasikan kode program insert dan remove pada AVL Tree. Gunakan penjelasan yang sudah diberikan pada slide presentasi dan juga buku Allan Weiss Bab 19.4.
2. Tulis kelas untuk menguji setiap method yang ada pada kedua kelas di atas. Jika misalnya method A sudah dipakai oleh method B, maka anda bisa menguji method B saja.
3. Tulis dalam bentuk laporan ilustrasi kelakuan setiap method ketika dipanggil. Gunakan satu contoh binary tree saja.

Jawaban

1. Kode program AVL Tree
   a) BinaryNode_11S20018

      AVLTree_11S20018\BinaryNode_11S20018.java
   b) AVLNode_11S20018

      AVLTree_11S20018\AVLNode_11S20018.java
   c) AVLTree_11S20018

      AVLTree_11S20018\AVLTree_11S20018.java

2. Kode program test AVL Tree

   AVLTree_11S20018\Test_11S20018.java

   Output:

   output_11S20018

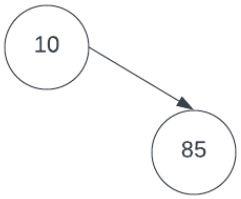
3. Ilustrasi implentasi

   - insert 10, 85, 15, 70, 20, 60, 30,
   - delete 15, 10,
   - insert 50, 65, 80
   - delete 20, 60,
   - insert 90, 40, 5, 55
   - delete 70

Tahap 1: at.insert(10)

Tahap 2: at.insert(85)

Nilai 85 > 10 maka disebelah kanan

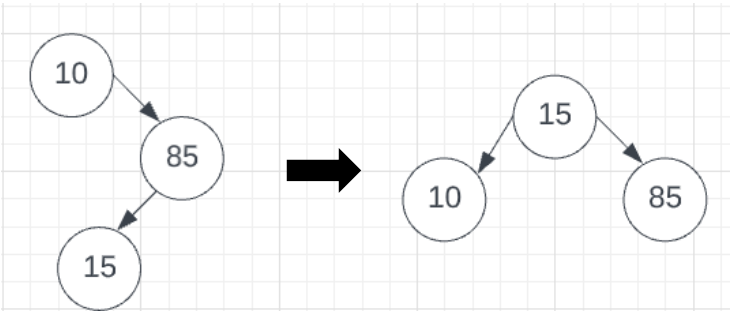


Tahap 3: at.insert(15)

Nilai 15 > 10 maka disebelah kanan

Nilai 15 < 85 maka disebelah kiri

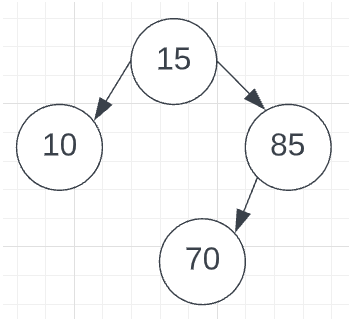
Double rotation: memindahkan anak bagian kiri dari node yang tidak balance ke posisi parent dan pindahkan anak bagian kanan dari node parent ke posisi parent



Tahap 4: at.insert(70)

Nilai 70 > 15 maka disebelah kanan

Nilai 70 < 85 maka disebelah kiri



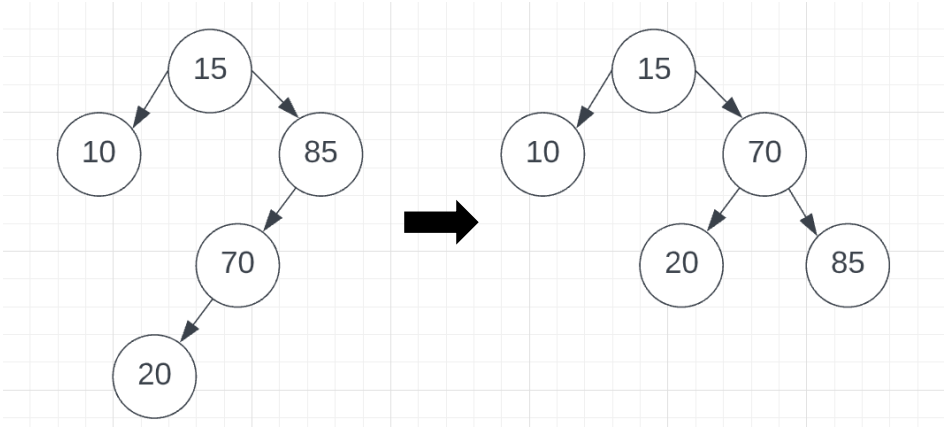
Tahap 5: at.insert(20)

Nilai 20 > 15 maka disebelah kanan

Nilai 20 < 85 maka disebelah kiri

Nilai 20 < 70 maka disebelah kiri

Single rotation-right: pindahkan anak bagian kiri dari node yang tidak balance ke posisi parent



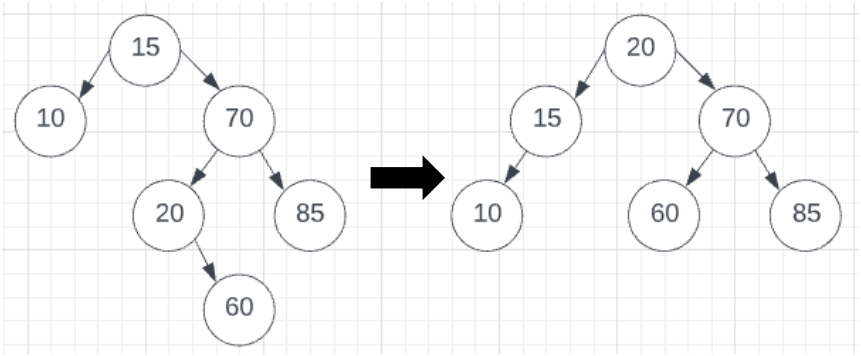
Tahap 6: at.insert(60)

Nilai 60 > 15 maka disebelah kanan

Nilai 60 < 70 maka disebelah kiri

Nilai 60 > 20 maka disebelah kanan

Double rotatation-left

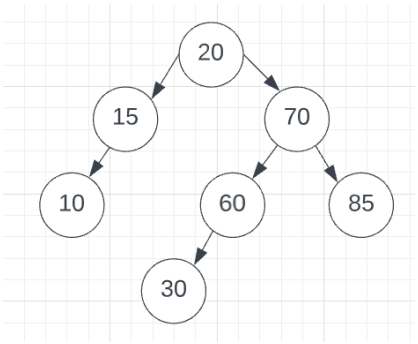


Tahap 7: at.insert(30)

Nilai 30 > 20 maka disebelah kanan

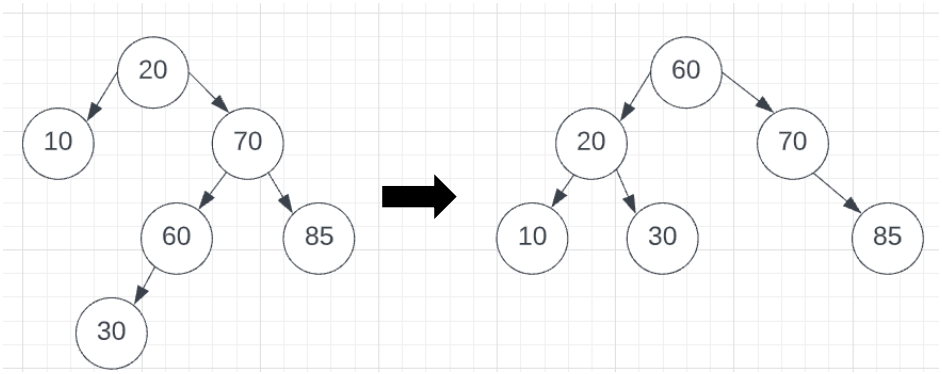
Nilai 30 < 70 maka disebelah kiri

Nilai 30 < 60 maka disebelah kiri



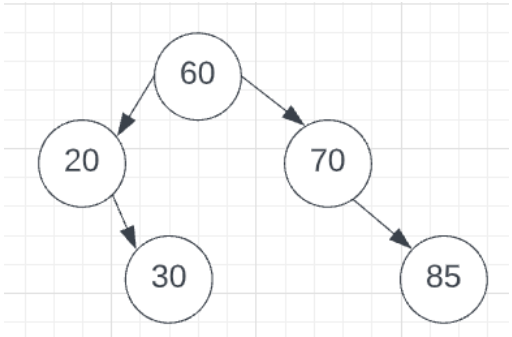
Tahap 8: at.remove(15)

Kasus pada tahap ini node(15) tidak memiliki anak disebelah kanan (1 anak) maka anak akan menjadi parent



Tahap 9: at.remove(10)

Kasus pada tahap ini node(10) adalah leaf sehingga dapat langsung dihapus



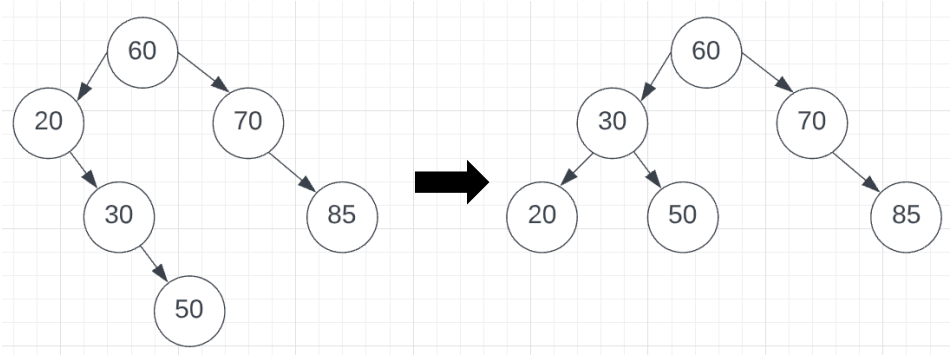
Tahap 10: at.insert(50)

Nilai 50 < 60 maka disebelah kiri

Nilai 50 > 20 maka disebelah kanan

Nilai 50 > 30 maka disebelah kanan

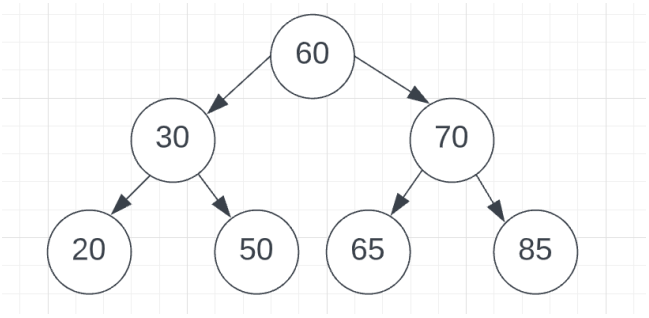
Singe rotation-left



Tahap 11: at.insert(65)

Nilai 65 > 60 maka disebelah kanan

Nilai 65 < 70 maka disebelah kiri

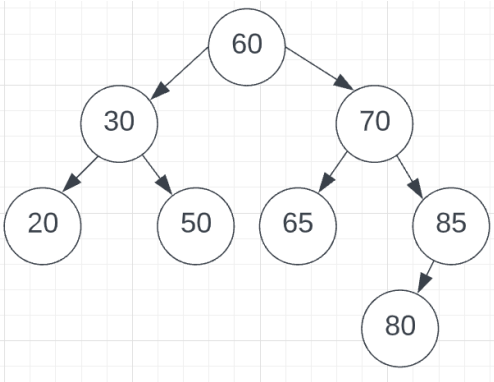


Tahap 12: at.insert(80)

Nilai 80 > 60 maka disebelah kanan

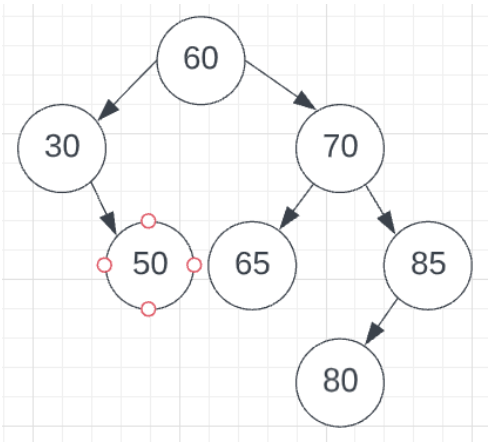
Nilai 80 > 70 maka disebelah kanan

Nilai 80 < 85 maka disebelah kiri



Tahap 13: at.remove(20)

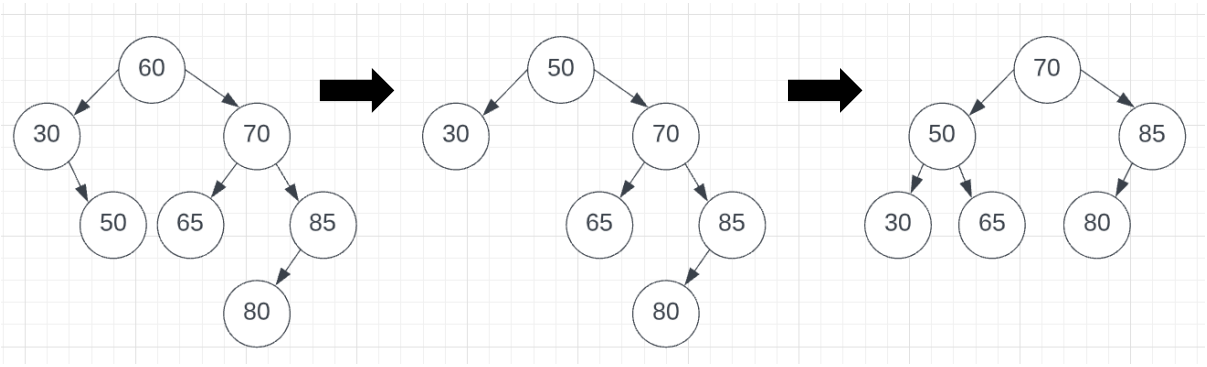
Kasus pada tahap ini node(20) adalah leaf sehingga dapat langsung dihapus



Tahap 14: at.remove(60)

Kasus pada tahap ini node(60) memiliki 2 anak sehingga node(60) secara rekursif dengan predecessor-nya secara inorder.

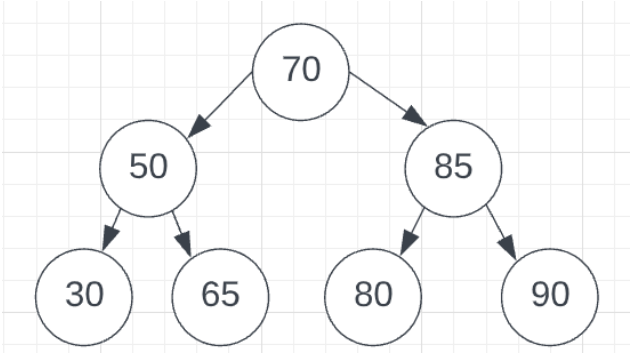
Single rotation-left



Tahap 15: at.insert(90)

Nilai 90 > 70 maka disebelah kanan

Nilai 90 > 85 maka disebelah kanan

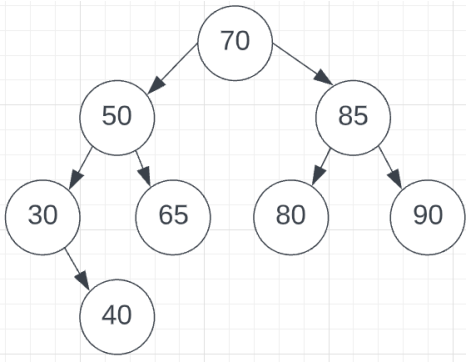


Tahap 16: at.insert(40)

Nilai 40 < 70 maka disebelah kiri

Nilai 40 < 50 maka disebelah kiri

Nilai 30 > 30 maka disebelah kanan

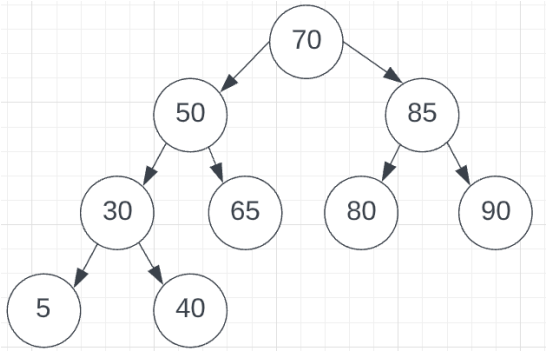


Tahap 17: at.insert(5)

Nilai 5 < 70 maka disebelah kiri

Nilai 5 < 50 maka disebelah kiri

Nilai 5 < 30 maka disebelah kiri

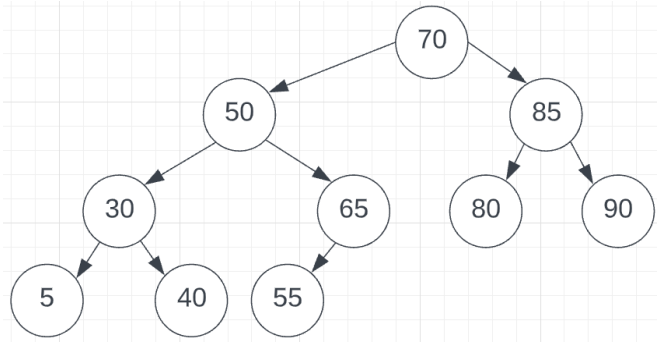


Tahap 18: at.insert(55)

Nilai 55 < 70 maka disebelah kiri

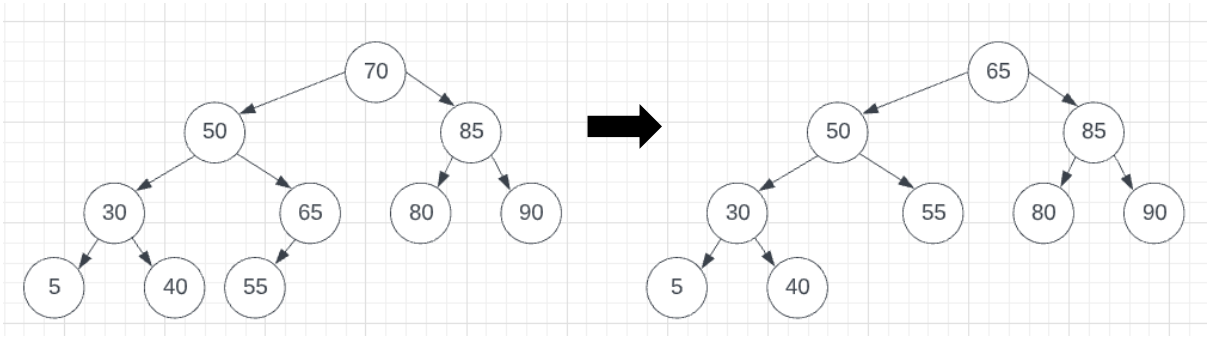
Nilai 55 > 50 maka disebelah kanan

Nilai 55 < 60 maka disebelah kiri



Tahap 19: at.remove(70)

Kasus pada tahap ini node(70) memiliki 2 anak sehingga node(70) secara rekursif dengan predecessor-nya secara inorder.



Hasil: