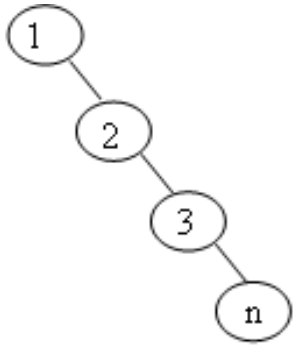
AVL TREE

Struktur Data

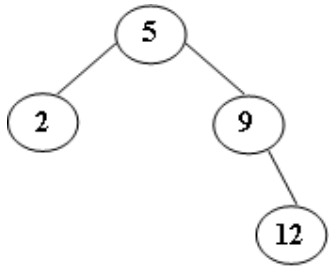
AVL TREE

- Walaupun BINARY SEARCH TREE sudah dapat mengatasi kelemahan pada BINARY TREE dengan cara mengurutkan / sort node yang di insert, di update dan di delete, tetapi masih ada kendala lain yang dihadapi BST, yaitu masih ada kemungkinan terbentuk SKEWED BINARY TREE (TREE MIRING) yang mempunyai PERBEDAAN HEIGHT (HEIGHT BALANCED) antara subtree kiri dengan subtree kanan sampai AVL (Adelson Velskii dan Landis) tree mengatasi hal ini dengan cara membatas HEIGHT BALANCED maksimum 1. AVL TREE dapat didefinisikan sebagai BST yang mempunyai ketentuan bahwa **"Maksimum perbedaan Height antara subtree kiri dan subtree kanan adalah 1"**. AVL TREE juga sering disebut dengan HEIGHT BALANCED 1-TREE.

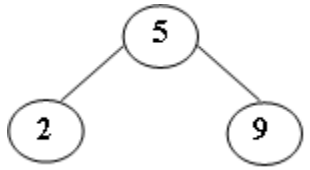
- Gambar dibawah ini memperlihatkan BST setelah dilakukan operasi INSERT sebagai berikut : +1, +2, +3,…, +n



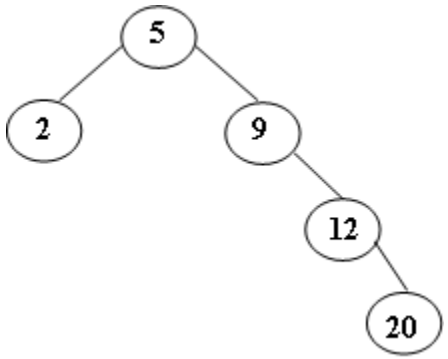
- Bila dilakukan pencarian terhadap node n diatas, maka pencarian sama seperti dilakukan pada BST, yaitu pencarian menjadi sekuensi, yang memakan waktu lama, hal ini tidak mungkin terjadi pada AVL TREE karena perbedaan Height dibatasi maksimal hanya 1. Berikut ini adalah contoh AVL TREE dan bukan AVL TREE:



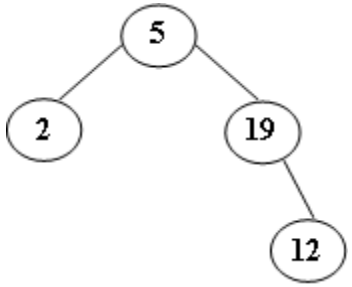
- a. AVL TREE



b. AVL TREE



- C. Bukan AVL TREE karena perbedaan >1



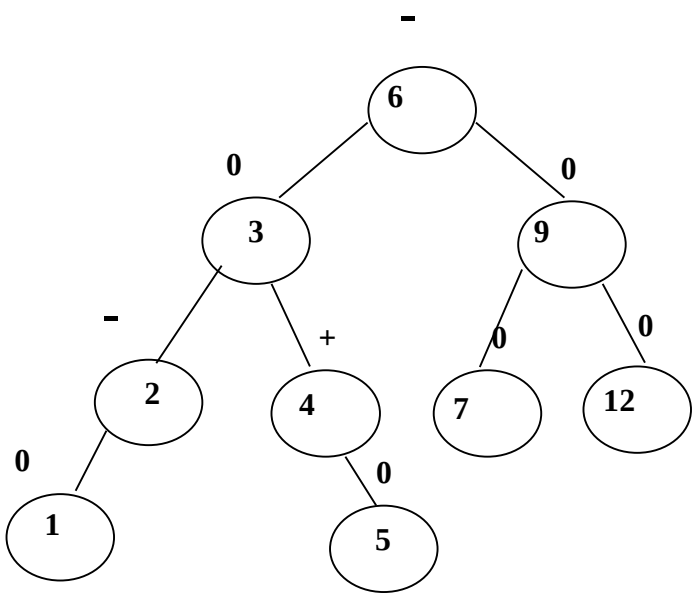
d. Bukan AVL TREE karena bukan BST

Status node

Setiap node dalam AVL TREE diberikan simbol untuk mengetahui tentang statusnya (lihat gambar dibawah ini), yaitu :

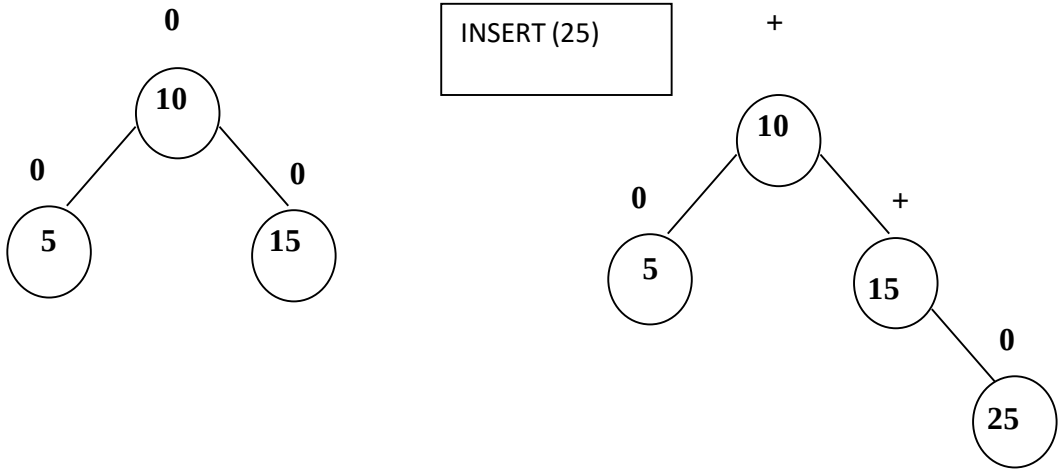
- Node diberi simbol - dan disebut TAllLeft bila subtree kiri lebih panjang dari subtree kanan
- Node diberi simbol + dan disebut TAllRight bila subtree kanan lebih panjang dari subtree kiri
- Node diberi simbol **0** dan disebut Balance bila subtree kiri dan kanan mempunyai height yang sama

- Contoh :

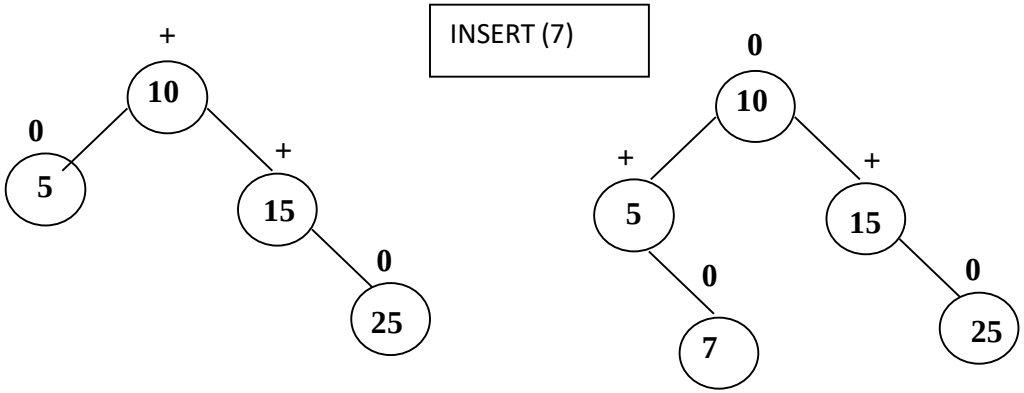


Operasi Insert

- Agar AVL TREE dapat tetap mempertahankan HEIGHT-BALANCED 1-TREE maka setiap kali pelaksanaan operasi INSERT, jika diperlukan maka harus dilakukan ROTASI. Operasi INSERT dalam AVL TREE a da 3 kondisi / kasus yaitu :
- 1. Tidak ada pivot point dan setiap node adalah Balance, maka bisa langsung diinsert sama seperti pada BST (tanpa perlu di REGENERATE)



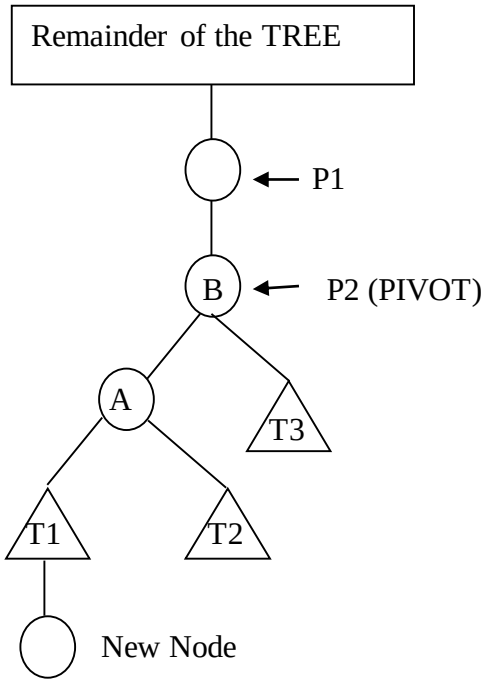
- 2. Jika ada pivot point tetapi subtree akan ditambahkan node baru memiliki height yang lebih kecil, maka bisa langsung di INSERT.

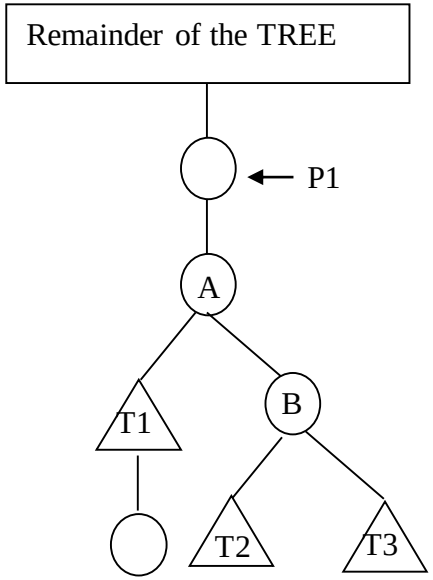


- 3. Jika ada pivot point dan subtree yang akan ditambahkan node baru memiliki height yang lebih besar, maka TREE harus di REGENERATE, supaya tetap menghasilkan AVL TREE.

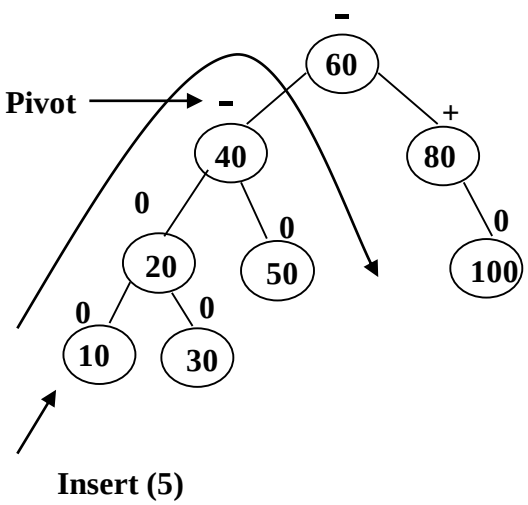
Cara melakukan RE-GENERATE

- Single Rotation
- Double Rotation

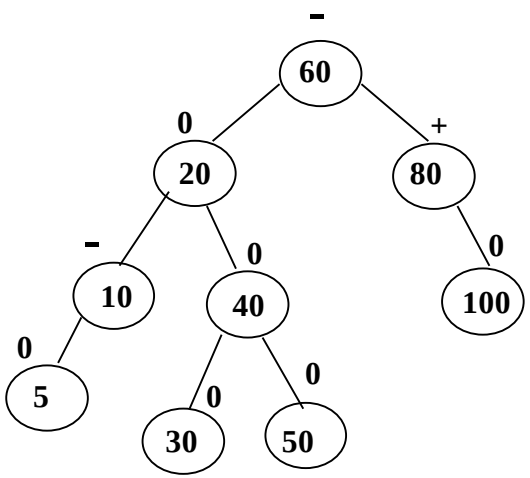




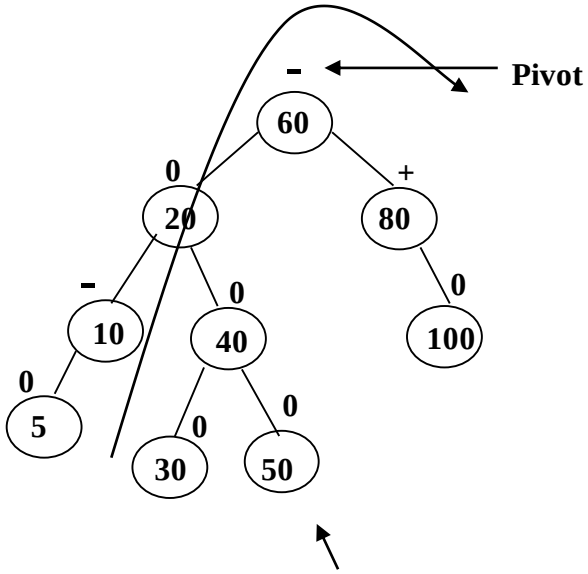
- Contoh :



- Insert (5)

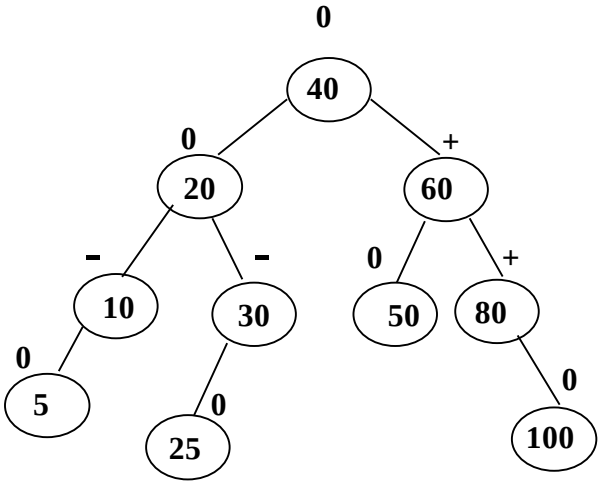


(sesudah dilakukan Single Rotation)



Insert (25)

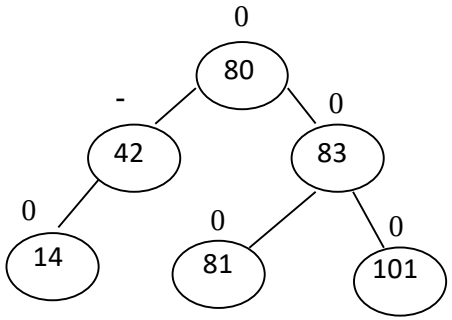
- Insert (25)



(Sesudah dilakukan Double Rotation)

- **Latihan**

  Diketahui Tree :



  Berdasarkan prinsip AVL Tree, buatlah operasi :

  - Insert (5)
  - Insert(36)
  - Insert (39)
  - Insert(43)