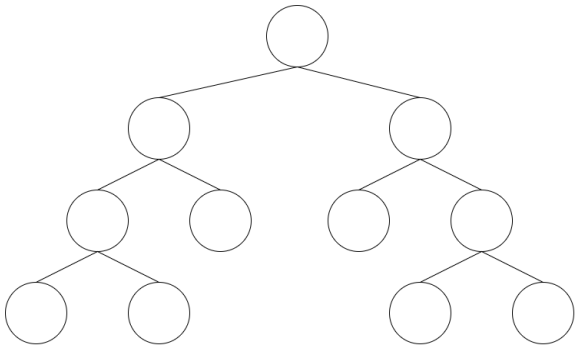
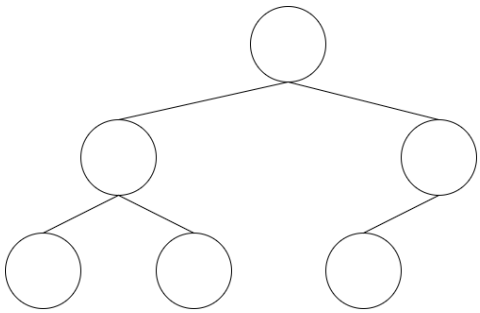
UTS Bootcamp Data Structure

Binary search tree

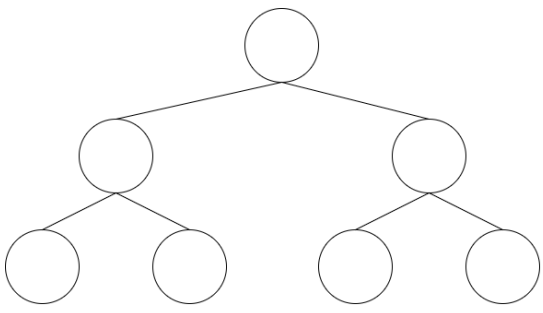
1. **Full binary tree**
   Full binary tree adalah binary tree yang mempunyai 0 anak atau 2 anak, sehingga tidak ada node yang mempunyai 1 anak



Complete binary tree
Adalah binary tree di setiap level mempunyai node kecuali level yang terakhir.

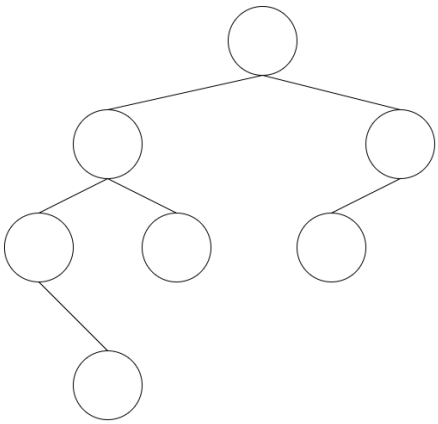


Perfect binary tree
Adalah dimana semua internal node hanya mempunyai 2 anak, sehingga tidak boleh kurang dan lebih dan setiap leaf node harus di level yang sama dalam sebuah tree.



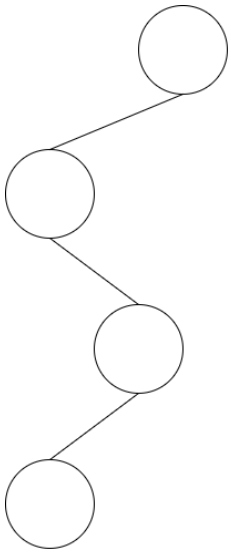
Balanced binary tree

Balanced binary tree mempunya tinggi O(log N), dimana N adalah nomor sebuah node. Jika tinggi subtree kiri dan kanan selisih maksimal satu, maka dapat disebut balanced binary tree.

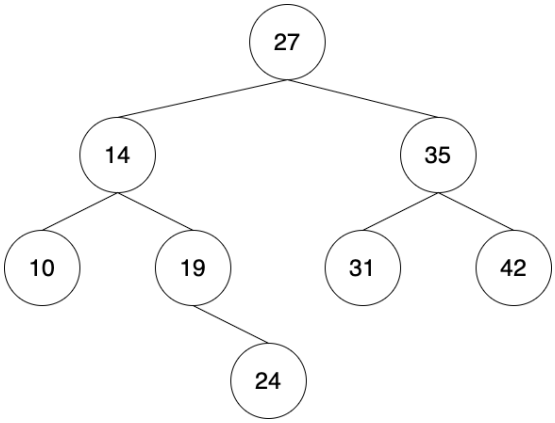


Degenerate binary tree

Adalah sebuah binary tree dimana setiap internal node hanya memiliki satu anak. Jika suatu tree serupa dengan linked list, maka dapat disebut degenerate binary tree.



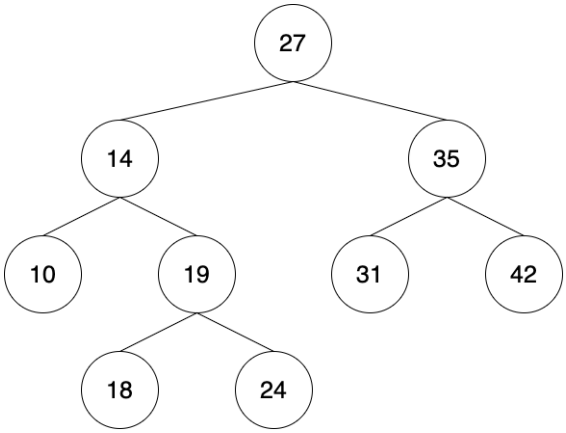
2. **Insertion: 24**



Langkah-langkah:

1. Saat melakukan insert 24, maka akan masuk melewati root atau angka 27.
2. 24 akan dicek, apakah 24 itu < atau > dari 27 karena 24 < 27 maka akan masuk ke subtree kiri, yaitu 14.
3. Kemudian node 24 dicek kembali apakah node 24 < atau > node 14, karena 24 > 14, maka akan masuk ke kanan, yaitu node 19.
4. Node 24 akan melakukan pengecekkan lagi apakah node 24 < atau > node 19, karena 24 > 19 maka akan masuk ke kanan atau di push ke kanan dan di situlah node 24 berakhir .

Insertion: 18

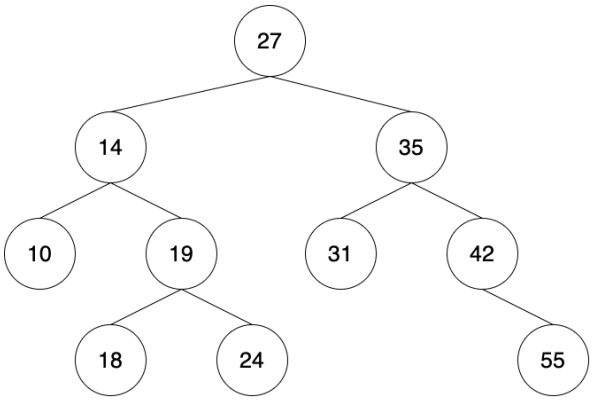


Langkah-langkah:

1. Saat melakukan insert 18, maka akan masuk melewati root atau angka 27.
2. 18 akan dicek, apakah 18 itu < atau > dari 27 karena 18 < 27 maka akan masuk ke subtree kiri, yaitu node 14.
3. Kemudian node 18 dicek kembali apakah node 18 < atau > node 14, karena 18 > 14, maka akan masuk ke kanan, yaitu node 19.

4. Node 18 akan melakukan pengecekkan lagi apakah node 18 < atau > node 19, karena 18 < 19 maka akan masuk ke kiri atau di push ke kiri dan di situlah node 18 berakhir.

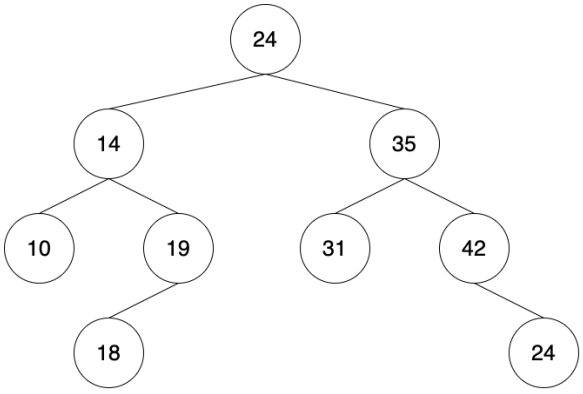
Insertion: 55



Langkah-langkah:

1. Saat melakukan insert 55, maka akan masuk melewati root atau angka 27.
2. 55 akan dicek, apakah 55 itu < atau > dari 27 karena 55 > 27 maka akan masuk ke subtree kanan, yaitu node 35.
3. Kemudian node 55 dicek kembali apakah node 55 < atau > node 35, karena 55 > 35, maka akan masuk ke kanan, yaitu node 42.
4. Node 55 akan melakukan pengecekkan lagi apakah node 55 < atau > node 42, karena 55 > 19 maka akan masuk ke kanan atau di push ke kanan dan di situlah node 55 berakhir .

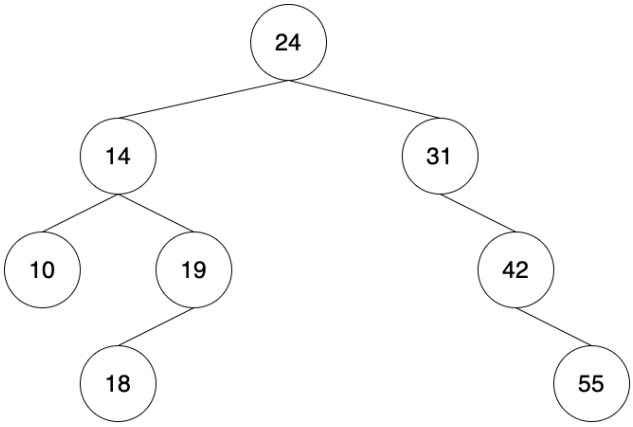
Delete: 27



Langkah-langkah:

1. Karena angka 27 mau dihilangkan, maka cari angka yang paling besar di subtree kiri.
2. Karena angka paling terbesar di subtree kiri < angka terbesar di subtree kanan, maka angka yang terbesar di subtree adalah 24, maka 24 menggantikan 27.

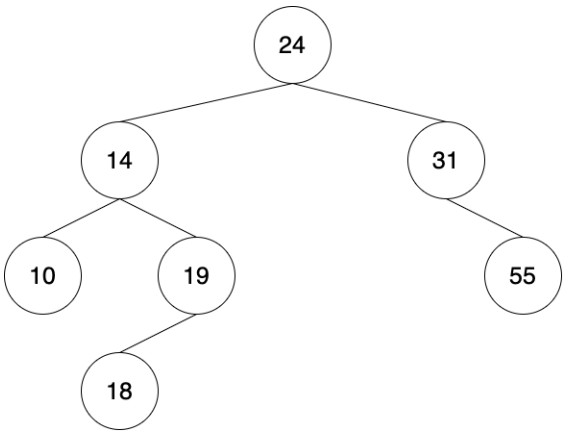
Delete: 35



Langkah-langkah:

1. Karena kita ingin menghilangkan angka 35, maka akan melakukan pengecekkan terhadap anaknya.
2. Apakah 31 < atau > 42, karena 31 < 42, maka 31 menggantikan angka 35.

Delete: 42



Langkah-langkah:

1. Karena kita ingin menghilangkan 42 maka harus melakukan pengecekkan terhadap anaknya.
2. Karena 42 hanya memiliki satu anak saja, yaitu 52, maka 52 bisa langsung menggantikan 42.