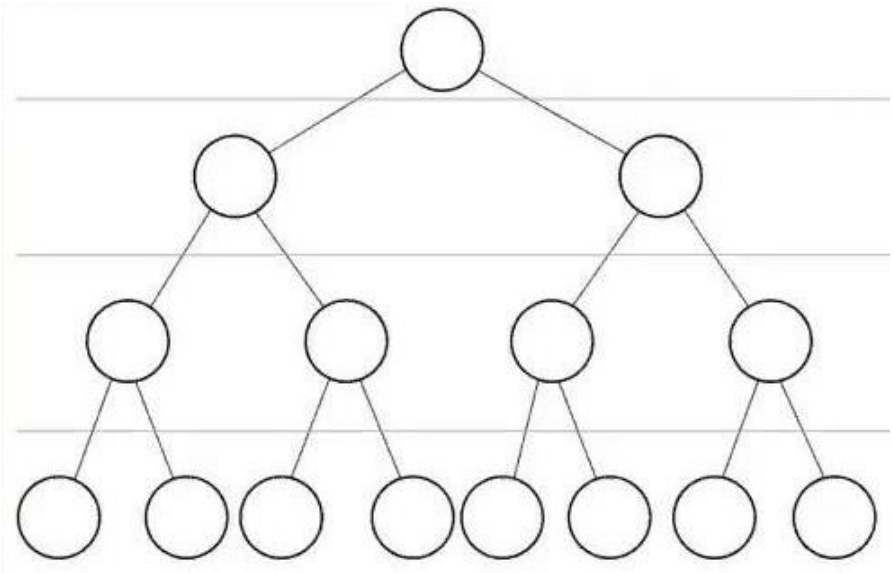
Belajaritma

Data Structure

Sesi V : Trees
(PDF Example)

A. Pengertian

Binary Search Tree (BST) adalah tipe data struktur yang mendukung pencarian dan pengurutan data yang lebih cepat, dan proses memasukkan juga penghapusan data yang mudah. BST dikenal juga sebagai versi terurut dari Binary Tree. Sesuai namanya, pada BST, setiap cabang memiliki maksimal 2 Node.



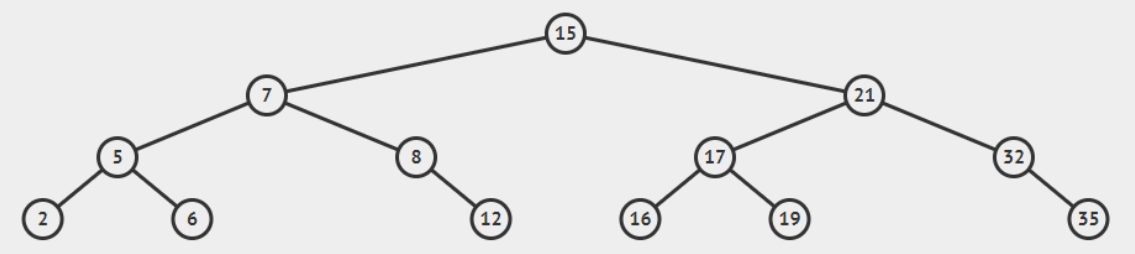
Dari gambar tersebut, dapat disimpulkan hal sebagai berikut :

Jumlah Node maksimum pada sebuah *level* k adalah 2^k
Jumlah Node maksimum pada sebuah BST adalah 2^{h+1}
* dengan h adalah tinggi maksimum dari BST tersebut.

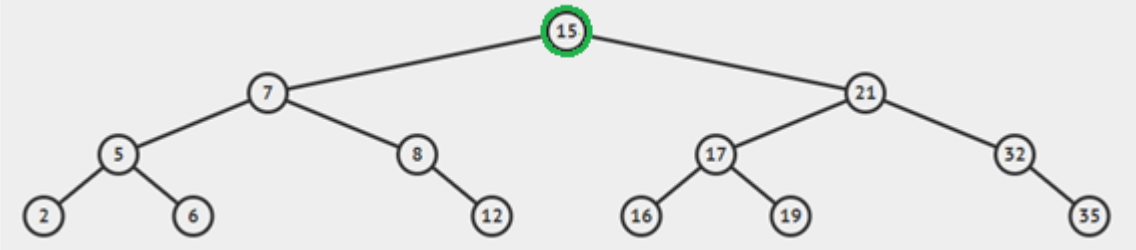
Pada dasarnya, subtree kiri mempunyai nilai yang lebih kecil daripada subtree kanan. Dengan ini memahami konsep dasar dari BST, seharusnya pemahaman fungsi *basic* menjadi lebih mudah.

B. Insert

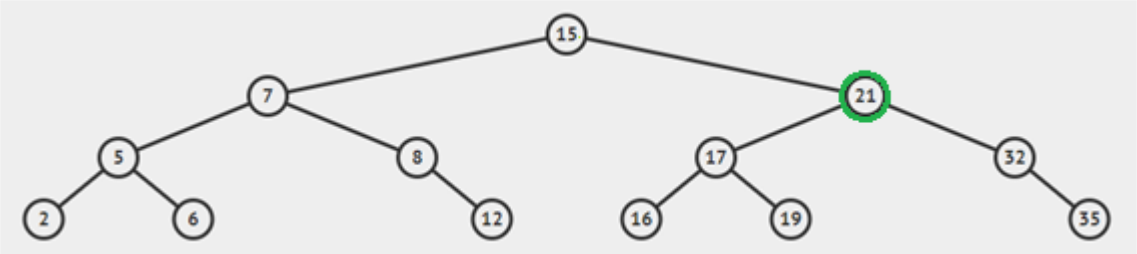
Insert adalah proses memasukkan value pada suatu BST yang di mulai dari root, dan terus membandingkan setiap value dari Node yang ditempuh sampai menemukan lokasi tepat yang kosong.



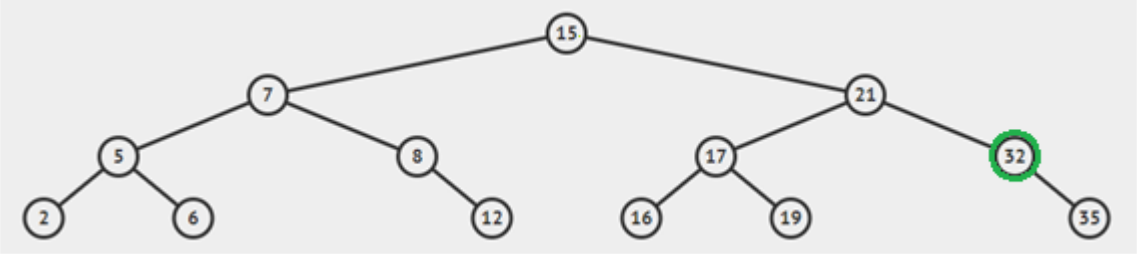
Dalam simulasi ini, kita akan menambahkan Node dengan nilai 27. Maka, dimulai dari root Node, yaitu 15. Lingkaran hijau menunjukkan jalan dan proses yang ditempuh saat ini.



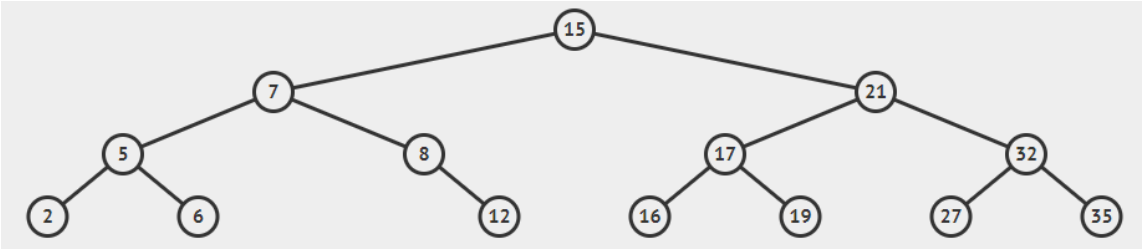
Jika 27 lebih kecil dari 15, maka selanjutnya kita akan membandingkan Node kiri dari Node 15. Namun, 27 lebih besar dari 15, maka kita bergeser pada Node kanan dari Node 15.



Selanjutnya, membandingkan Node 21 dan 27. Nilai 27 lebih besar dari Node 21, sehingga tahap selanjutnya, bergeser lagi pada Node kanan dari Node 21.



Nilai 27 lebih kecil daripada Node 32, jadi seharusnya kali ini kita bergeser pada Node kiri dari Node kiri dan melakukan perbandingan lagi. Namun, ternyata Node 32 hanya memiliki 1 anak Node, yaitu pada sisi kanan. Sehingga, sisi kiri dari Node 32 akan terisi Node 27.

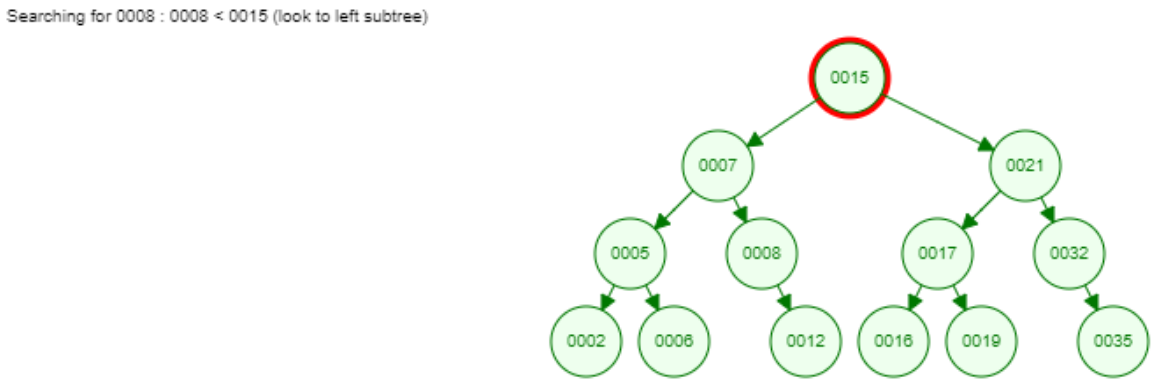


Proses *insert* selesai.

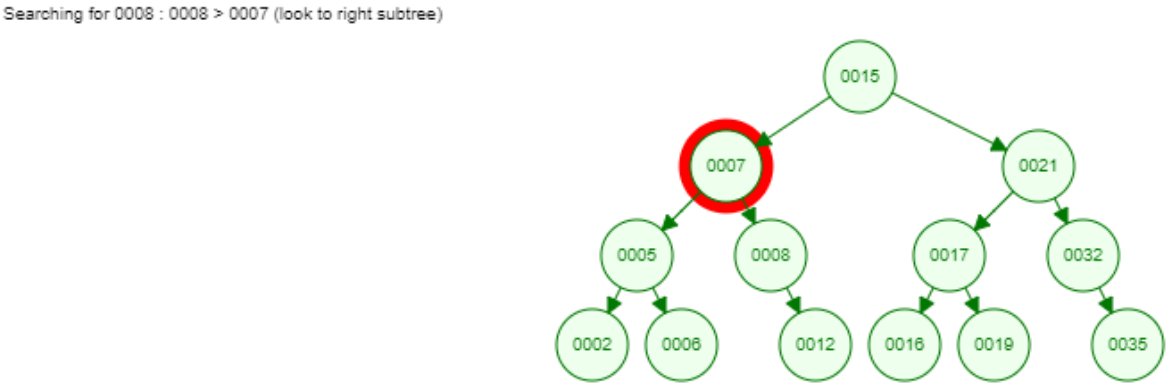
C. Search

Search adalah proses pencarian value yang ingin dicari dalam suatu struktur data *Tree*, dan proses Search ini dimulai dari *Root* atau data pertama atau akar, dan proses Search dalam struktur data *Tree* memiliki kecepatan yang lebih dibandingkan dengan struktur data lainnya.

Dari Tree sebelumnya, berikut gambaran dari proses Search, kita akan search value 8.

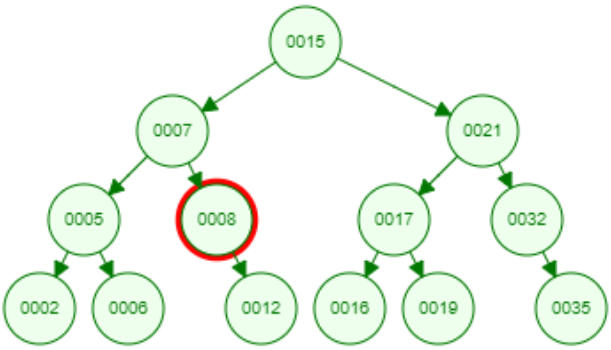


Dari *Root* atau akar akan ditentukan apakah value yang akan dicari lebih besar atau lebih kecil atau bahkan sama dengan value yang dicari, hingga value yang ingin dicari sama dengan value yang ada di *Node* sekarang, maka proses Search tidak akan berhenti. karena lebih kecil, maka node selanjutnya akan menunjuk yang ada di sebelah kiri.



Dan pada *Node* sekarang, akan ditentukan akan ditentukan apakah value yang akan dicari lebih besar atau lebih kecil atau sama dengan value yang dicari, karena lebih besar, maka *Node* selanjutnya akan menunjuk ke sebelah kanan.

Searching for 0008 : 0008 = 0008 (Element found!)

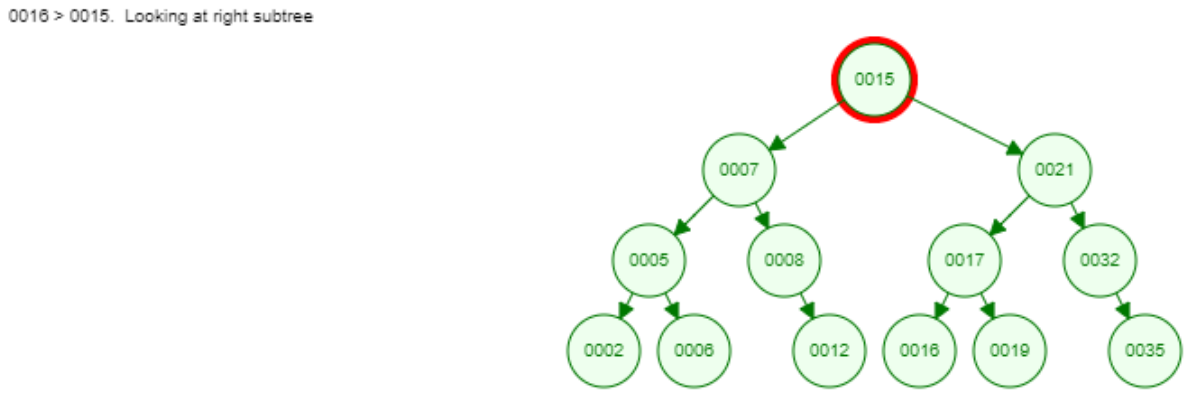


Dan pada *Node* sekarang, akan ditentukan akan ditentukan apakah value yang akan dicari lebih besar atau lebih kecil atau sama dengan value yang dicari, karena sama dengan value yang dicari, maka proses pencarian akan berhenti dan me *return* value yang dicari.

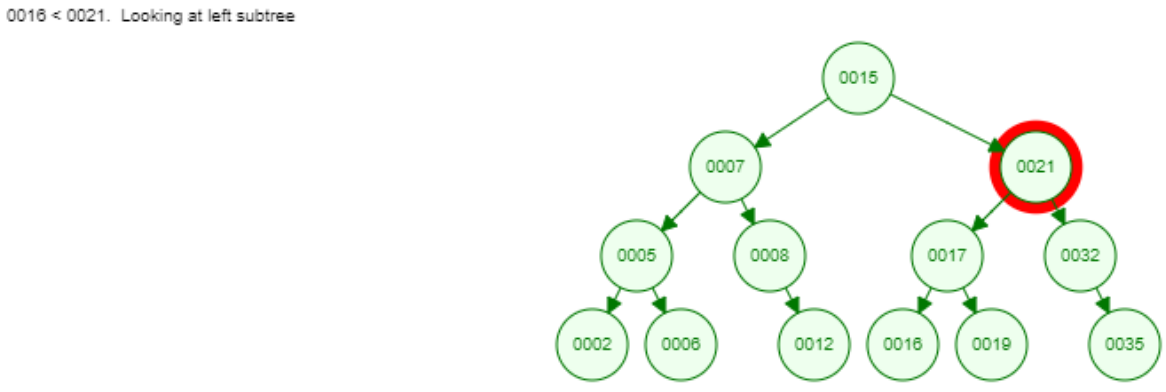
D. Delete

Proses Delete adalah proses menghapus data yang diinginkan di dalam struktur data *Tree*, dan proses Delete ini dimulai dari *Root* atau data pertama atau akar, dan proses Delete dalam struktur data *Tree* memiliki kecepatan yang lebih dibandingkan dengan struktur data lainnya.

Dari Tree sebelumnya, berikut gambaran dari proses Delete, kita akan menghapus value 16.

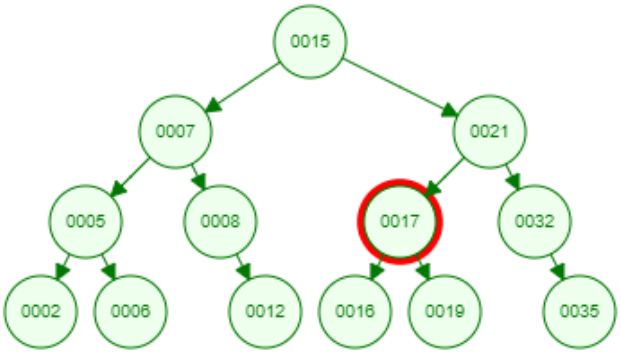


Dari value yang sekarang, proses mencari data yang ingin dihapus tidak akan berheti. Karena 16 lebih besar daripada 15, maka *Node* selanjutnya menunjuk pada value yang ada di kanan.



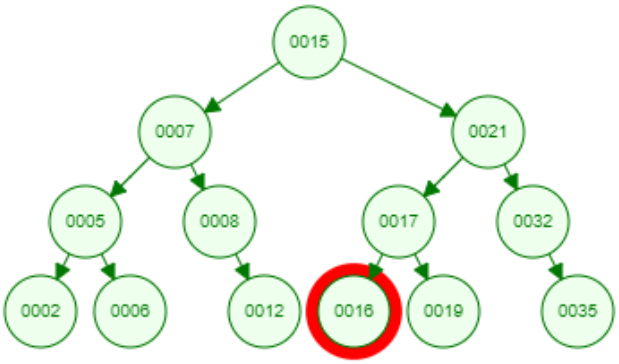
Karena 16 lebih kecil daripada 21, maka *Node* selanjutnya akan menunjuk pada value yang ada di kiri.

0016 < 0017. Looking at left subtree



Karena 16 lebih kecil daripada 17, maka *Node* selanjutnya akan menunjuk pada value yang ada di sebelah kiri.

0016 == 0016. Found node to delete



Ketika Value sekarang sama dengan value yang ingin kita hapus yaitu 16, maka data tersebut akan kita hapus dan proses delete berhenti.

Sumber :

https://www.programiz.com/dsa/binary-search-tree

https://www.cs.usfca.edu/~galles/visualization/BST.html