**RANGKUMAN**

**Binery Search Tree**



**Nama:**

**Taufik Akbar 201610370311224**

**TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS MUHAMMADIYAH MALANG**

**2017/2018**

1. ADT

Tipe data abstrak (ADT) dapat didefinisikan sebagai model matematika dari objek data yang menyempurnakan tipe data dengan cara mengaitkannya dengan fungsi-fungsi yang beroprasi pada data yang bersangkutan. Merupakan hal yang sangat penting untuk mengenali bahwa operasi-operasi yang akan dimanipulasi data pada objek yang bersangkutan termuat dalam spesifikasi ADT. Sebagai contoh, ADT HIMPUNAN didefinisikan sebagai koleksi data yang diakses oleh operasi-operasi himpunan seperti penggabungan (UNION), irisan (INTERSECTION), dan selisih antar-himpunan (SET DIFFERENCE).

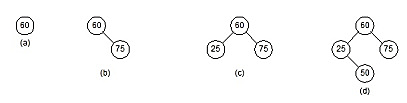
1. BINARY SEARCH TREE

Pohon cari biner adalah pohon biner yang dirancang untuk menskemakan urutan data yang akan dimasukkan ke dalam memori agar proses pencarian, penghapusan dan penambahan data dapat berjalan secara efisien dibanding dengan pemasukan data secara array maupun link.

Sifat dari skema pohon cari biner adalah : (1) setiap elemen yang berada di left substrees selalu lebih kecil dari elemen yang ada di right substrees, (2) setiap elemen yang berada di right substrees selalu lebih besar atau sama dengan elemen yang berada di left substrees.

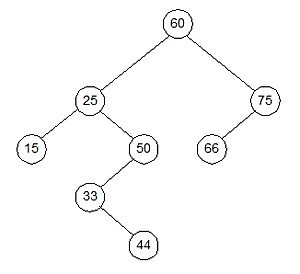
Contoh : diketahui sekumpulan elemen sebagai berikut :

60, 75, 25, 50, 15, 66, 33, 44

[](https://saragusti22.files.wordpress.com/2015/05/tree.jpg)

Gambar 1.1 Pembentukan awal Binary Search Tree

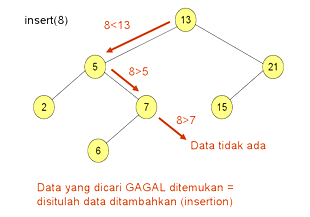
dan, hasil akhirnya sebagai berikut :

[](https://saragusti22.files.wordpress.com/2015/05/tree2.jpg)

Gambar 1.2 Binary Search Tree

1. INSERTING

Penyisipan sebuah node baru, didahului dengan operasi pencarian posisi yang sesuai. Dalam hal ini node baru tersebut akan menjadi daun/leaf.

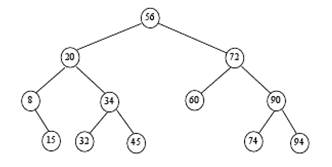
[](http://1.bp.blogspot.com/-hTpkECzSrDI/VJE7OGFg23I/AAAAAAAAAd8/d8ykBM-dmuA/s1600/bst+insert.PNG)

Gambar 1.3 Insert Binary Search Tree

1. DELETING

Operasi delete memiliki 3 kemungkinan :

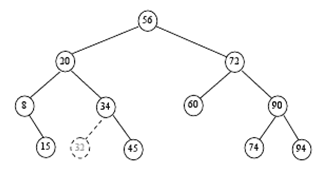
* Delete terhadap node tanpa anak/child (leaf/daun) : node dapat langsung dihapus
* Delete terhadap node dengan satu anak/child : maka node anak akan menggantikan posisinya.
* Delete terhadap node dengan dua anak/child : maka node akan digantikan oleh node paling kiri dari Sub Tree Kanan atau dapat juga digantikan oleh anak paling kanan dari Sub Tree Kiri.

[](http://3.bp.blogspot.com/-bI-1FPrUMuY/VJE4Y2uhiyI/AAAAAAAAAdE/_Xrg7x9RU_E/s1600/bst+delete.PNG)

Gambar 1.4 Delete Binary Search Tree

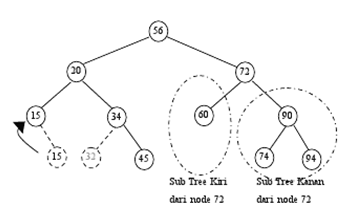
Misalnya ingin dihapus :

1. Node (32) : dapat langsung dihapus sehingga akan dihasilkan tree sebagai berikut:

[](http://4.bp.blogspot.com/-l2yuN2hovuo/VJE465DT0TI/AAAAAAAAAdQ/SIP9wuIvTKI/s1600/bst+delete1.PNG)

Gambar 1.5 Delete Binary Search Tree

2. Node (8) : node dengan satu child

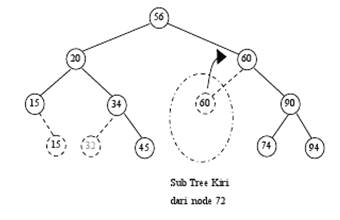
[](http://1.bp.blogspot.com/-SzohIUQVXbo/VJE5harxBzI/AAAAAAAAAdY/LWnWwrbSvkA/s1600/bst+delete2.PNG)

Gambar 1.6 Delete Binary Search Tree

3. Node (72) : node dengan 2 child

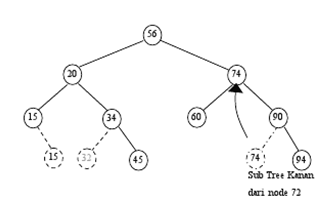
Node akan digantikan oleh anak paling kanan dari Sub Tree Kiri (node(60))

Atau anak paling kiri dari Sub Tree Kanan (node(74))

[](http://3.bp.blogspot.com/-eEqAe1Asmes/VJE6ABDaQ3I/AAAAAAAAAdg/pROxFwYe1nA/s1600/bst+delete3-1.PNG)

Gambar 1.7 Delete Binary Search Tree

Atau

[](http://3.bp.blogspot.com/-zlZ_D5S7aKE/VJE6dXyGD_I/AAAAAAAAAds/5rCmQuTgTk0/s1600/bst+delete3-2.PNG)

Gambar 1.8 Delete Binary Search Tree

1. SEARCHING

Operasi searching memiliki 3 kemungkinan :

* Jika yang dicari = parent, pencarian selesai.
* Jika yang dicari lebih besar dari parent, search akan berjalan ke bagian kanan parent, berlaku sebaliknya
* Jika tidak ditemukan dibagian left sub tree, melakukan pencarian di right sub tree secara rekursif, berlaku sebaliknya