**RANGKUMAN**

**BINARY TREE & SORTING**



**Nama:**

**Muhammad Zain Iqbal Muzakki**

**201610370311237**

**TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS MUHAMMADIYAH MALANG**

**2017/2018**

**BINARY TREE**

1. **BINARY SEARCH TREES (POHON CARI BINER)**
2. **Binary Search Tree**

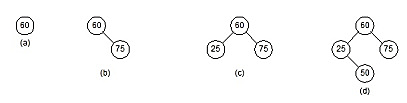
Binary search tree adalah pohon biner yang dirancang untuk menskemakan urutan data yang akan dimasukkan ke dalam memori agar proses pencarian, penghapusan dan penambahan data dapat berjalan secara efisien dibanding dengan pemasukan data secara array maupun link.

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 :

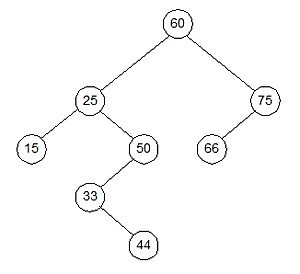
60, 75, 25, 50, 15, 66, 33, 44

Pembentukan awal skema pohon binernya berturut-turut sebagai berikut :

[](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. **OPERASI BINARY SEARCH TREE**
2. **INSERT**

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

|  |
| --- |
| [https://1.bp.blogspot.com/-hTpkECzSrDI/VJE7OGFg23I/AAAAAAAAAd8/d8ykBM-dmuA/s1600/bst%2Binsert.PNG](http://1.bp.blogspot.com/-hTpkECzSrDI/VJE7OGFg23I/AAAAAAAAAd8/d8ykBM-dmuA/s1600/bst+insert.PNG) |
| Gambar 1.3 Insert Binary Search Tree |

1. **DELETE**

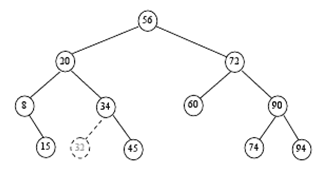
Operasi delete memiliki 3 kemungkinan :

* 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.

|  |
| --- |
| [https://3.bp.blogspot.com/-bI-1FPrUMuY/VJE4Y2uhiyI/AAAAAAAAAdE/_Xrg7x9RU_E/s1600/bst%2Bdelete.PNG](http://3.bp.blogspot.com/-bI-1FPrUMuY/VJE4Y2uhiyI/AAAAAAAAAdE/_Xrg7x9RU_E/s1600/bst+delete.PNG) |
| Gambar 1.4 Delete Binary Search Tree |

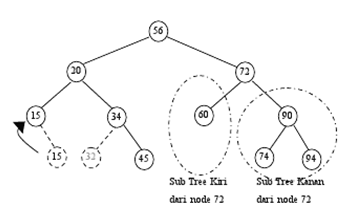
Misalnya ingin dihapus

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

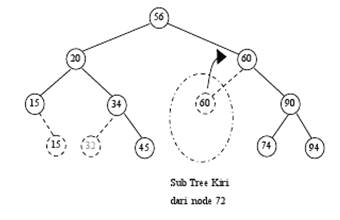
[](http://1.bp.blogspot.com/-SzohIUQVXbo/VJE5harxBzI/AAAAAAAAAdY/LWnWwrbSvkA/s1600/bst+delete2.PNG)

Gambar 1.6 Delete Binary Search Tree

3. Node (72) : node dengan 2 child

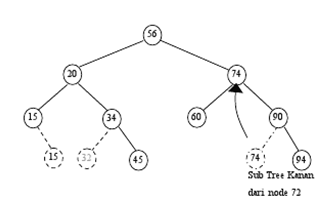
Node akan digantikan oleh anak paling kanan dari Sub Tree Kiri (node(60))

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

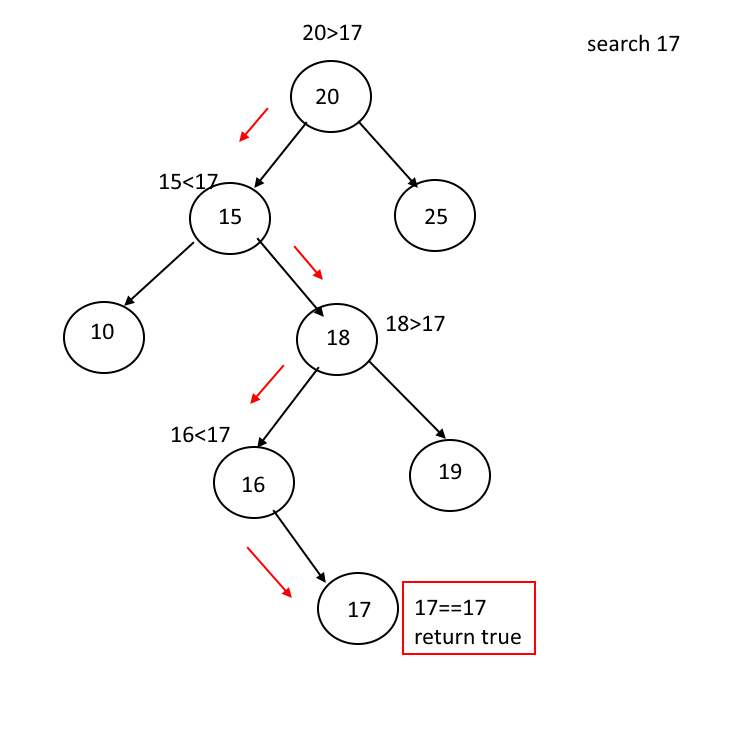
[](http://3.bp.blogspot.com/-zlZ_D5S7aKE/VJE6dXyGD_I/AAAAAAAAAds/5rCmQuTgTk0/s1600/bst+delete3-2.PNG)

Gambar 1.8 Delete Binary Search Tree

1. **SEARCHING**

Operasi searching memiliki 3 kemungkinan :

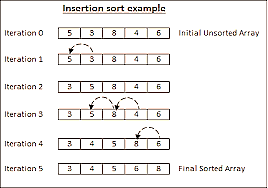
* Jika YANG DICARI = PARENT, pencarian selesai.
* Jika YANG DICARI lebih BESAR dari PARENT, search akan berjalan ke bagian kanan parent, berlaku sebaliknya
* Jika tidak ditemukan dibagian left sub tree, melakukan pencarian di right sub tree secara rekursif, berlaku sebaliknya



1. **SORTING**
2. **INSERTION SORT**

Algoritma insertion sortpada dasarnya memilah data yang akan diurutkan menjadi dua bagian, yang belum diurutkan dan yang sudah diurutkan. Elemen pertama diambil dari bagian array yang belum diurutkan dan kemudian diletakkan sesuai posisinya pada bagian lain dari array yang telah diurutkan. Langkah ini dilakukan secara berulang hingga tidak ada lagi elemen yang tersisa pada bagian array yang belum diurutkan.

Contoh :



Gambar 2.1 Insertion Sort

1. **SELECTION SORT**

Ide utama dari algoritma selection sortadalah memilih elemen dengan nilai paling rendah dan menukar elemen yang terpilih dengan elemen ke-i. Nilai dari idimulai dari 1 ke n, dimana nadalah jumlah total elemen dikurangi 1.

Contoh :

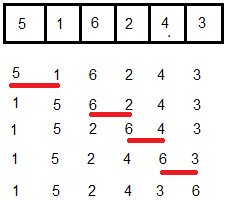


Gambar 2.2 Selection Sort

1. **BUBLE SORT**

Merupakan algoritma pengurutan paling tua dengan metode pengurutan paling sederhana. Pengurutan yang dilakukan dengan membandingkan masing-masing itemdalam suatu list secara berpasangan, menukar itemjika diperlukan, dan mengulaginya sampai akhir list secara berurutan, sehingga tidak ada lagi item yang dapat ditukar.

Contoh :



Gambar 2.3 Buble Sort