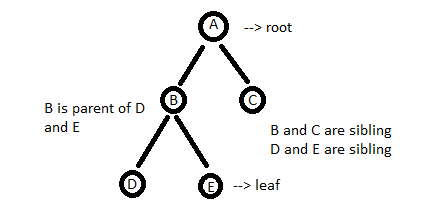
Binary Search Tree

adalah sebuah binary tree dimana dalam penempatan simpulnya tidak bebas, melainkan melalui aturan tertentu, seperti sebelah kiri nilainya harus lebih kecil dari root, dan kanan lebih besar. Sehingga simpul tersebut nilainya sudah tersorting otomatis. Dalam BST, setiap simpulnya tidak boleh memiliki nilai yang sama. Setiap pohon memiliki root dan leaf, root adalah simpul yang tidak memiliki parent dan leaf adalah simpul yang tidak memiliki anak. Berikut ilustrasinya :



Pada BT terdapat istilah Ancestor dan Descendent, dimana Ancestor adalah induk dari suatu simpul. Sedangkan Descendent adalah turunan dari suatu simpul. Contoh pada gambar diatas :

* Ancestor dari E adalah B dan A.
* Descendent dari A adalah B, D dan E.

untuk menulis notasi prefix, infix, atau postfix dari tree, dapat menggunakan cara berikut :

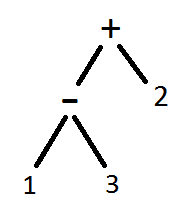
Prefix = VLR

Infix = LVR

Postfix = LRV

dengan v = cetak, r = right, l = kiri

contohnya :



dari tree diatas,

prefix = +-132,

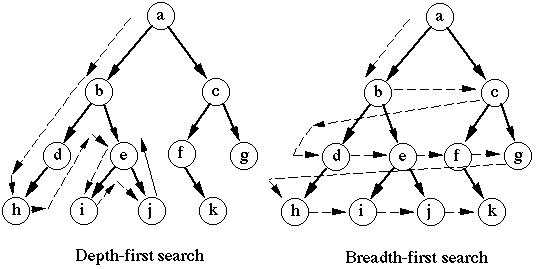
infix = 1-3+2,

postfix = 13-2+

Dalam BST ada 2 jenis atau cara pencarian, yaitu :

* Depth First Search (DFS): DFS adalah pencarian berdasarkan kedalamannya.
* Breadth First Search (BFS): BFS adalah pencarian berdasarkan perluasannya.

Ilustrasinya :



Dapat disimpulkan bahwa Binary Search Tree mempunyai aturan :

* Setiap child node sebelah kiri harus lebih kecil nilainya daripada root nodenya.
* Setiap child node sebelah kanan harus lebih besar nilainya daripada root nodenya.

1. Inserting

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

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.

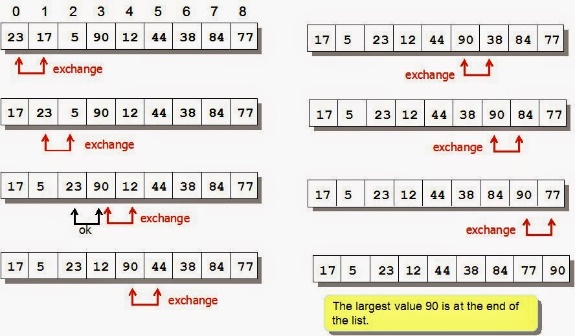
1. Searching

Operasi searching memiliki 3 kemungkinan :

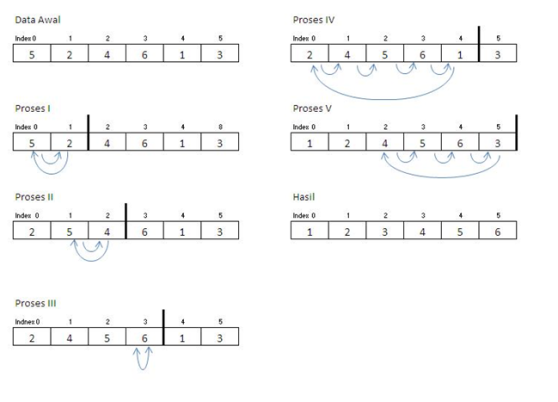
* Jika yang dicari = parent, pencarian selesai.
* Jika yang dicari lebih besar dari parent, search akan berjalan ke bagian kanan parent, berlaku sebaliknya
* Jika tidak ditemukan dibagian left sub tree, melakukan pencarian di right sub tree secara rekursif, berlaku sebaliknya

Sorting

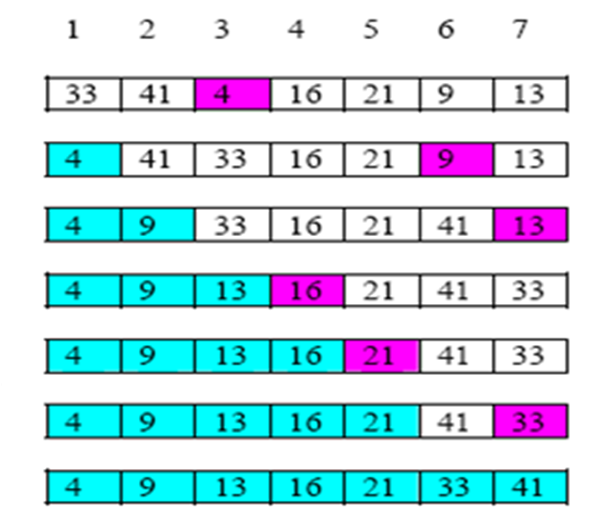
1. Bubble Sort

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

1. Insertion Sort

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

1. Selection Sort

Algoritma insertion sort pada 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.