Introduction to trees

Introduction to trees

- Trees and terminology
- Tree traversals
- Binary trees
- Complete binary trees / Almost complete binary tree (ACBT)
- Array implementation of ACBT
- Binary search trees
- AVL tree
- Multi-way tree
- Brief introduction and uses cases of B-Tree /B+Tree.

Non Linear Data Structures

- un order --- no fixed sequence.
- have one-to-many or many-to-many relationship
- e.g. Tree, graph

Tree

finite set of nodes with one specially designated node called the root.

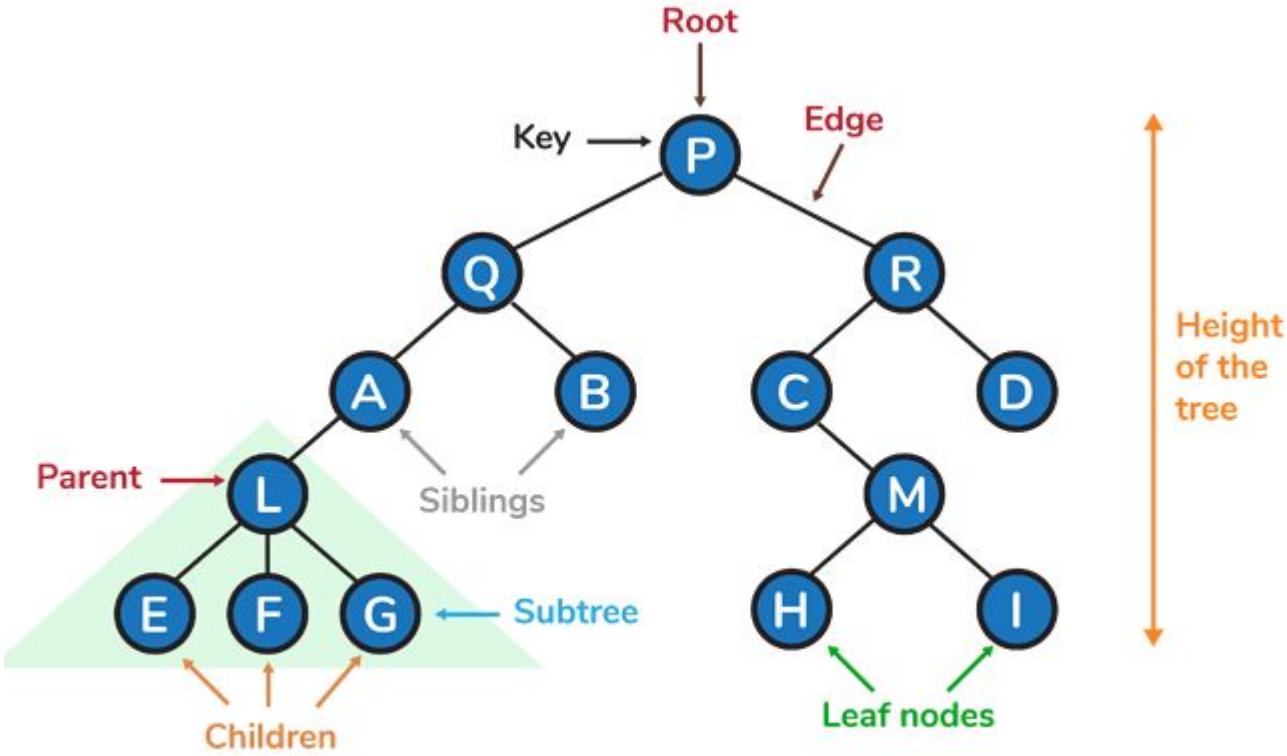
root - Starting point of Tree - does not have parent

remaining nodes are partitioned into n>=0 disjoint sets T1, T2, ...,Tn called subtrees of the root.

Tree Terminology

1. Tree
2. Null tree
3. Node
4. Parent node
5. Leaf node
6. Siblings
7. Degree of a node
8. Degree of a tree

9. Descendents
10. Ancestors
11. Level of a node
12. Height or depth of a tree
13. Forest


Root
Key
P
Edge
Q
R
A
B
C
D
Parent
L
Siblings
M
E
F
G
Subtree
H
I
Height of the tree
Children
Leaf nodes

Types of Tree

1. Binary Tree
2. M way Tree
3. B Tree
4. B+ Tree

Types of Binary Tree

Strictly binary tree

- Either 0 or 2 child

Full binary tree -

- All levels must be full total elt = 2^level-1

Complete binary tree -

- If n levels then n-1 levels must be full and last level partially full from left to right

Skewed binary tree -

- A binary tree in which every node has only one child,

Heap tree

Complete binary tree parent node contain either greater value than its children or smaller value than its children.

Two types of heap trees -

Max heap tree : every parent will have greater value than its children.

Therefore

root node of max heap tree is the largest value among all the values.

Min heap tree : every parent will have smaller value than its children.

Therefore

root node of min heap tree is the smallest value among all the values.

Binary search tree

- binary tree
- Child nodes are placed based on data comparison
- Left child has less value than its parent and right child will have greater or equal to value than parent
- If we insert the data in the BST in ascending order then BST becomes right skewed
- if we insert the data in descending order then BST becomes left skewed.

Expression tree -

- Binary tree
- Used to represent mathematical expression of operator
- All leaf node are operands and all internal nodes are operator

AVL Tree -

- AVL tree is a self-balancing binary search tree
- Height balanced Binary search tree.
- balance factor is maintained for every node

- Balance factor has values either -1, 0 or +1.
- Balance Factor = (Height of Left Subtree - Height of Right Subtree) or (Height of Right Subtree - Height of Left Subtree)
- Search efficient tree less comparisons will be needed
- While constructing AVL tree need to maintain balance factor for every node
- If bf goes out or has other than the fixed values from range need to perform some operations to get balance factor from the range

2 types of Operations

1. Single Rotation
    - LL Rotation
    - RR Rotation
2. Double Rotation
    - LR Rotation
    - RL rotation

1. LL/RR Rotation
    a. By adding node at right/left in tree node in tree gets unbalanced so need to apply LL rotation which makes tree balance
2. LR/RL Rotation
    a. In this type of rotation we need to apply 2 operations at one place
    b. LL then RR rotation // RR then LL rotation

Types Representation

1. By Array
2. By Linked List

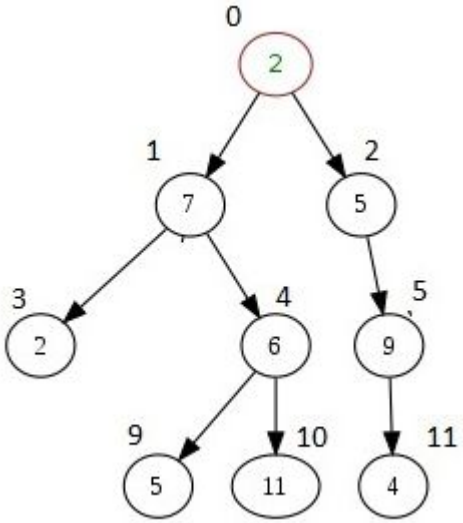
Representation of Complete binary trees / Almost complete binary tree (ACBT) in array

No. of elements in tree - depends on level of tree

Total no. of elts in tree = 2^(level+1) - 1

If level = 2 no. of elt = 2^3-1 ⇒ 7

Level = 3 n= 2^4-1 ⇒ 15



2	7	5	2	6	9				5	11	4			

level = 3 n= 2^4-1 ==15

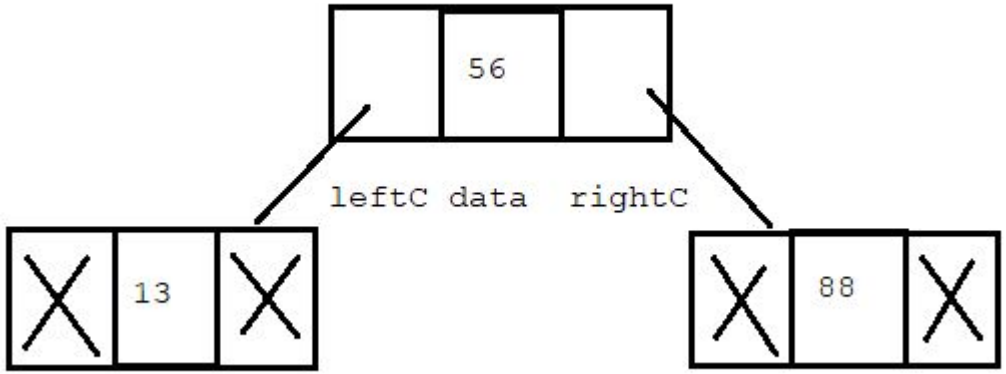
1. By Linked List

Every node will be storing value , links to it’s childs

Structure of node -

Operations on tree

1. Insert
2. Delete
3. Traversal
4. Search
5. …..



Tree Traversals

2 types of traversals

1. Breadth first Traversal
2. Depth first Traversal
    a. Inorder Traversal
    b. Preorder Traversal
    c. Postorder Traversal

M way Tree

multiway search tree of order m (or an m-way search tree).

Definition -

m-way search tree should satisfy -

Each node has 0 .. m subtrees

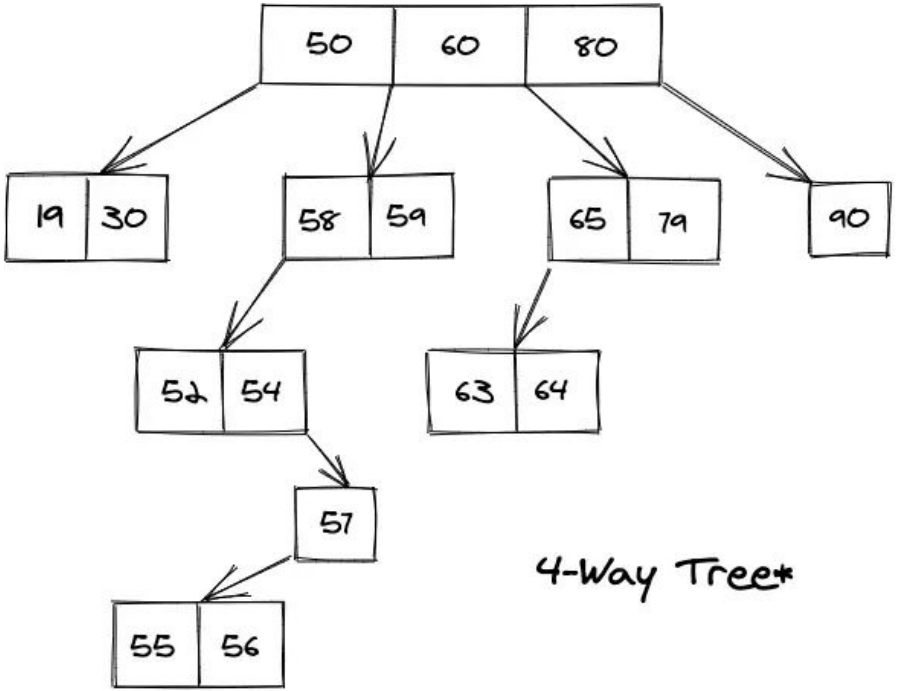
A node with k < m subtrees, contains k-1 keys.

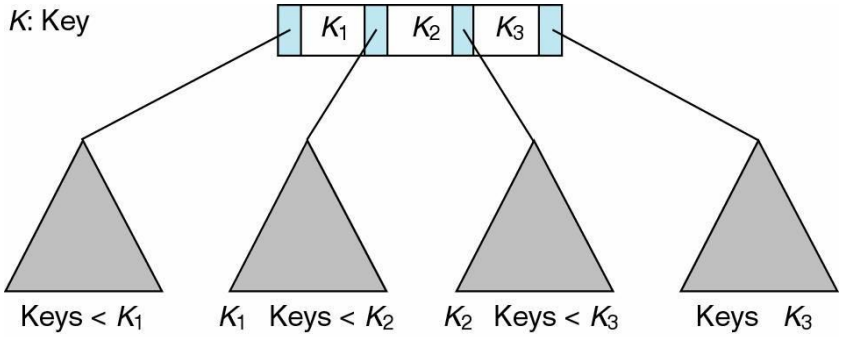
The key values of the first subtree are all less than the key value.

The data entries are ordered.

All subtrees are m-way trees.

Example


50
60
80
19
30
58
59
65
79
90
52
54
63
64
57
55
56
4-Way Tree*


K: Key
K1
K2
K3
Keys < K1
K1 Keys < K2
K2 Keys < K3
Keys K3

B Tree

The m-way tree is not balanced.

In 1970 Bayer and McCreight created the B-tree data structure.

In B-Tree a node can store more than one value (key) and it can have more than two children.

B-Tree is a self-balanced search tree with multiple keys in every node and more than two children for every node.

B-tree is an extension of m-way tree with the following additional properties:

Property #1 - All leaf nodes must be at same level.

Property #2 - All nodes except root must have at least [m/2]-1 keys and maximum of m-1 keys.

Property #3 - All non leaf nodes except root (i.e. all internal nodes) must have at least m/2 children.

Property #4 - If the root node is a non leaf node, then it must have at least 2 children.

Property #5 - A non leaf node with n-1 keys must have n number of children.

Property #6 - All the key values in a node must be in Ascending Order.

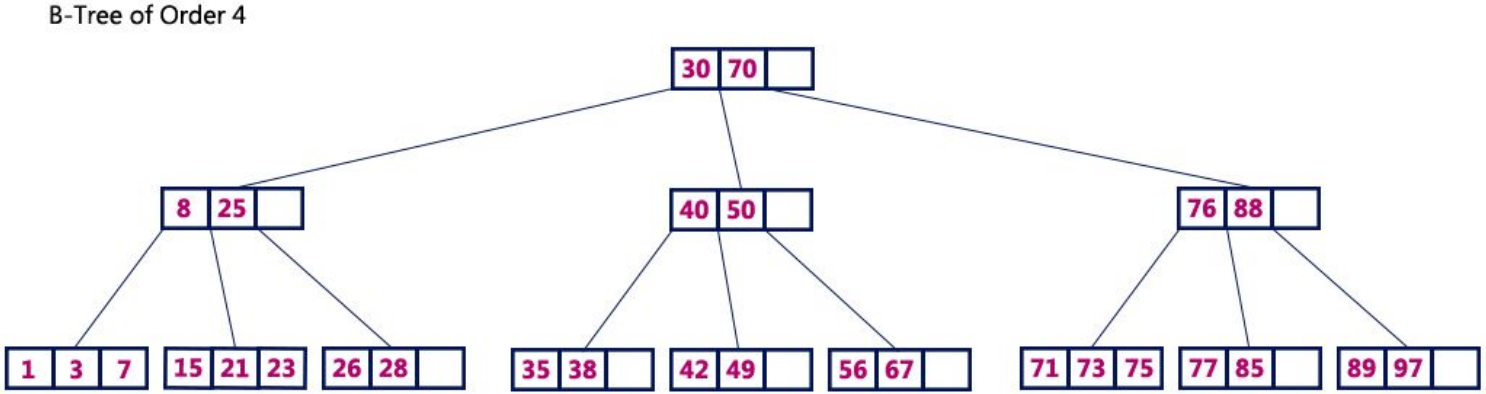
Operations on a B-Tree

The following operations are performed on a B-Tree...

Search

insertion

Deletion



B+ Tree

B+ tree is a variation of B-tree data structure.

In a B+ tree, data pointers are stored only at the leaf nodes of the tree.

Structure of a leaf node differ from the structure of internal nodes.

The leaf nodes have value of the search field, along with a data pointer to the record (or to the block that contains this record).

The leaf nodes of the B+ tree are linked together to provide ordered access on the search field to the records.

The drawback of B-tree used for indexing stores the data pointer (a pointer to the disk file block containing the key value), corresponding to a particular key value, along with that key value in the node of a B-tree.

reduces the number of entries that can be packed into a node of a B-tree, increase in the number of levels in the B-tree,

increasing the search time of a record.

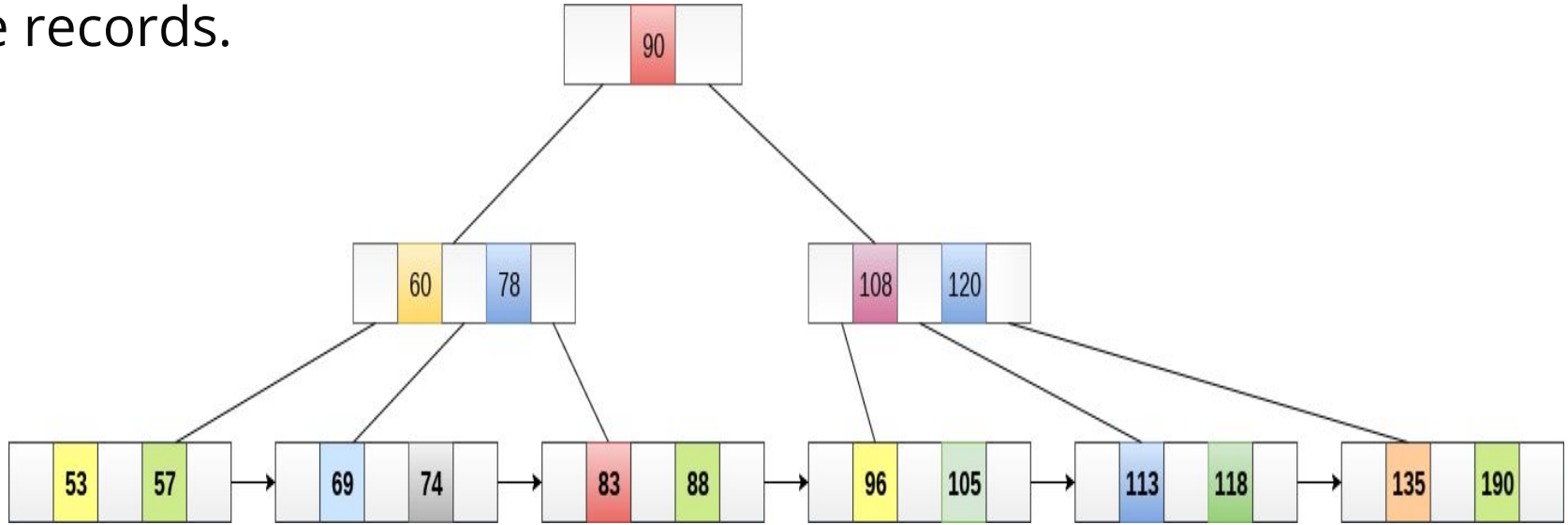
Why B+ tree

storing data pointers only at the leaf nodes of the tree.

structure of leaf nodes of a B+ tree is quite different

leaf nodes must necessarily store all the key values along with their corresponding data pointers to the disk file block, in order to access them.

Moreover, the leaf nodes are linked to provide ordered access to the records.



Advantages of B+ Tree

Records can be fetched in equal number of disk accesses.

Height of the tree remains balanced and less as compare to B tree.

We can access the data stored in a B+ tree sequentially as well as directly.

Keys are used for indexing.

Faster search queries as the data is stored only on the leaf nodes.