Sunbeam Institute of Information Technology
Pune and Karad
PG - DESD

Module – Data Structures

Trainer - Devendra Dhande
Email – devendra.dhande@sunbeaminfo.com

BST - Types

- **Skewed Binary tree**
  - In binary tree if only left or right links are used, then tree grows in only one direction and such tree is called as skewed binary tree.
    - Left skewed binary tree
    - Right skewed binary tree
  - Tree has maximum height that is same as number of elements
  - Time complexity of searching is O(n). (Like linked list)
- **Balanced BST**
  - If nodes in BST are arranged so that its height is kept as less as possible, is called as balanced BST.
  - Balance Factor = Height of left subtree – Height of right subtree
  - In balanced BST, balance factor of each node is -1, 0 or +1
  - A tree can be balanced by applying series of left or right rotations.
- **AVL Tree**
  - AVL tree is a self balancing binary search tree.
  - Node are balanced on each insert and delete operation.
  - Difference between heights of left and right subtrees can not be more than 1 for all nodes.

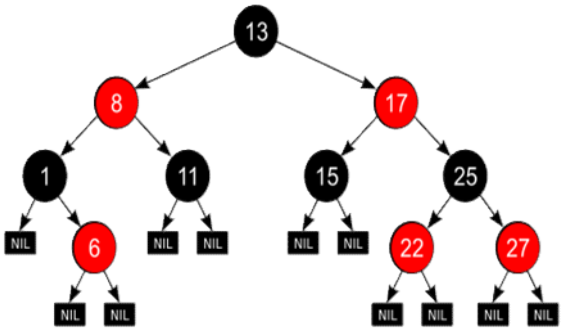
BST - Types

- **AVL Tree**
  - AVL tree is a self balancing binary search tree.
  - Node are rebalanced on each insert and delete operation.
  - Difference between heights of left and right subtrees can not be more than 1 for all nodes.
  - Most of BST operations are done in O(h) i.e. O(log n) time.
  - Need more number of rotations as compared to Red & Black tree.

  - Construct an AVL Tree by inserting nodes : 40, 20, 10, 25, 30, 22, 50

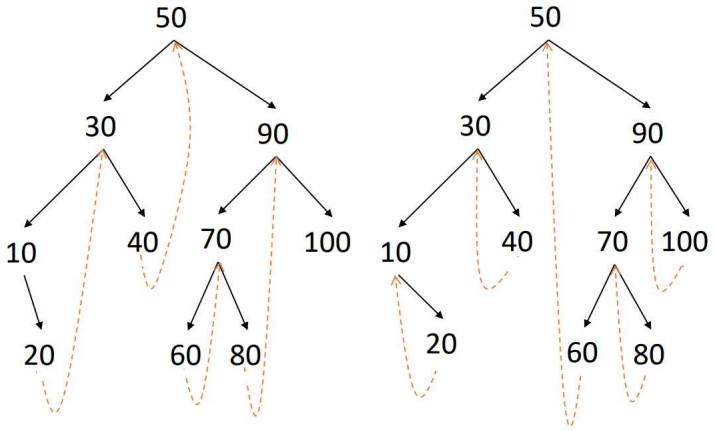
BST - Types

- **Red and Black Tree**
  - Red and Black tree is a self balancing binary search tree.
  - Each node follows some rules:
    - Every node has a color either red or black.
    - Root of tree is always black.
    - Two adjacent cannot be red nodes (Parent color should be different than child).
    - Every path from a node (including root) to any of its descendant NULL node has the equal number of black nodes.
  - Most of BST operations are done in O(h) i.e. O(log n) time.
  - For frequent insert/delete, RB tree is preferred over AVL tree.



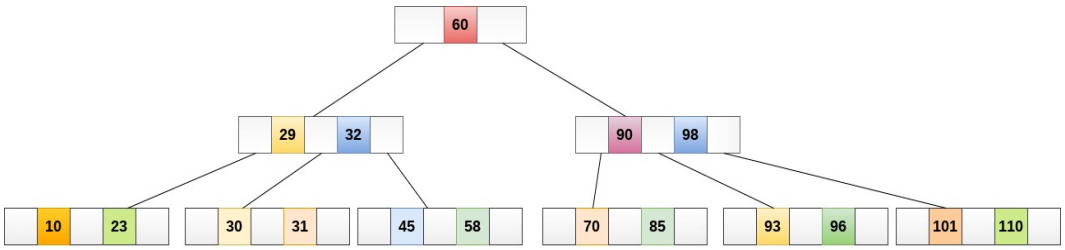
BST - Types

- **Threaded Tree**
  - Typical BST in-order traversal involves recursion or stack. It slows execution and also need more space.
  - Threaded BST keep address of in-order successor or predecessor addresses instead of NULL to speed up in-order traversal (using a loop).
    - Left threaded BST
    - Right threaded BST
    - In-threaded BST



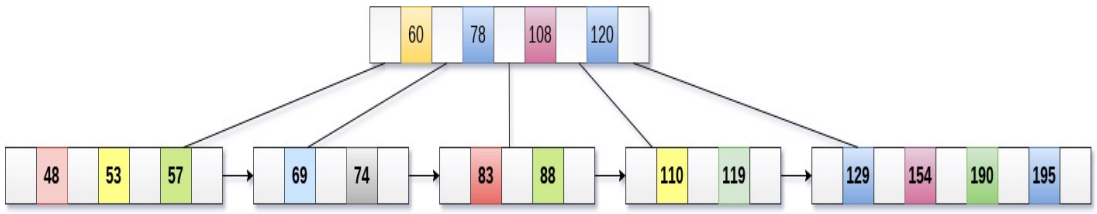
Tree Types

- **B Tree (Tree with degree m/m-way tree)**
  - A B-Tree of order m can have at most m-1 keys and m children.
  - B tree store large number of keys in a single node. This allows storing number of values keeping height minimal.
  - Note that in B-Tree all leaf nodes are at same level.
  - B-Tree is commonly used for indexing into file systems and databases. It ensures quick data searching and speed up disk access.



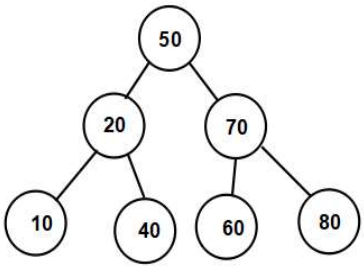
Tree Types

- **B+ Tree**
  - Extension of B-Tree for efficient insert, delete and search operation.
  - Data is stored in leaf nodes only and all leaf nodes are linked together for sequential access.
  - Faster searching, simplified deletion (as only from leaf nodes).
  - B+ Tree is commonly used for indexing into file systems and databases. It ensures quick data searching and speed up disk access.



Tree : Types

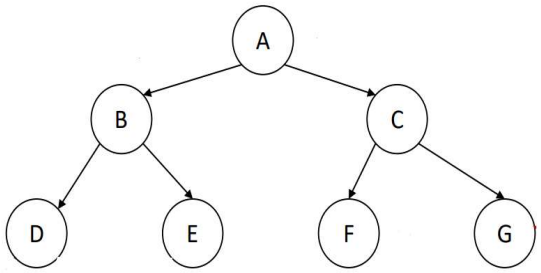
- **Strictly Binary Tree**
  - Binary tree in which each non leaf node has exact two child nodes.
- **Full Binary Tree**
  - Binary tree with its full capacity foe the given height.
  - In other words, adding one more node will increase height of the tree.
  - It is always complete as well as strictly binary tree.
  - Number of elements = 2^h – 1
- **Complete Binary Tree**
  - The binary tree which follows two conditions
    - All leaf nodes are at level h or h-1.
    - All leaf nodes at last level (h) are aligned to left as much as possible.



Heap

- Heap is array implementation of complete binary tree.
- Parent child relation is maintained through index calculations

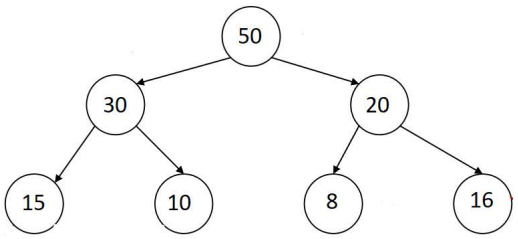
- If a node is at index - i
- Its left child at index = 2 * i
- Its right child at index = 2 * i + 1
- Its parent at index = i / 2



A	B	C	D	E	F	G
1	2	3	4	5	6	7

Heap Types – Max and Min

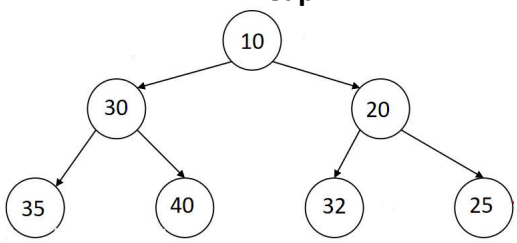
Max Heap



50	30	20	15	10	8	16
1	2	3	4	5	6	7

- Max heap is a heap data structure in which each node is greater than both of its child nodes.

Min Heap



10	30	20	35	40	32	25
1	2	3	4	5	6	7

- Min heap is a heap data structure in which each node is smaller than both of its child nodes.

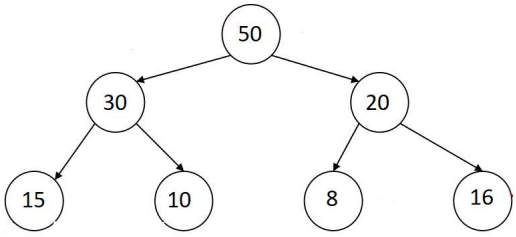
Heap Sort

- Heap sort is 2 step process
  - Create heap
  - Delete heap

10	20	15	30	40
1	2	3	4	5

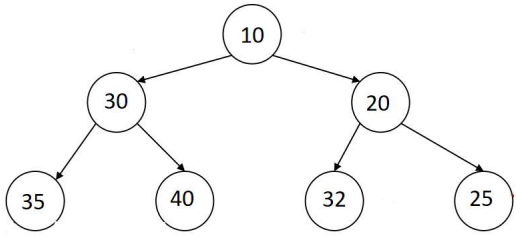
Priority Queues

Higher number
Higher priority



- In Max heap always root element which has highest value is removed

Lower number
Higher priority



- In Min heap always root element which has lowest value is removed

Thank you!

Devendra Dhande
<devendra.dhande@sunbeaminfo.com>