TECHNO INDIA UNIVERSITY
WESTBENGAL

Data Structures & Algorithms
(PCC-CS 301)

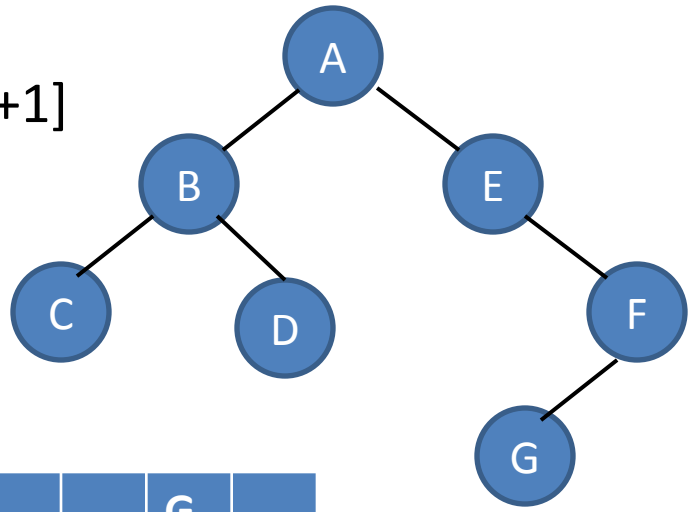
Dr. Debashis Das
Associate Professor
Department of CSE
Techno India University, Kolkata

Topics Covered

1. Binary Tree Representation
2. Threaded Binary Tree
3. Binary Search Tree (BST)
    a. Property
    b. Data insertion

Binary Tree

- Memory Representation
  - Array representation
    - Parent is in i^{th} index position
    - Left child position will be at [(2*i)+1]
    - Right child will be at [(2*i)+2]
    - Root of tree will be stored at the 0^{th} index of the array

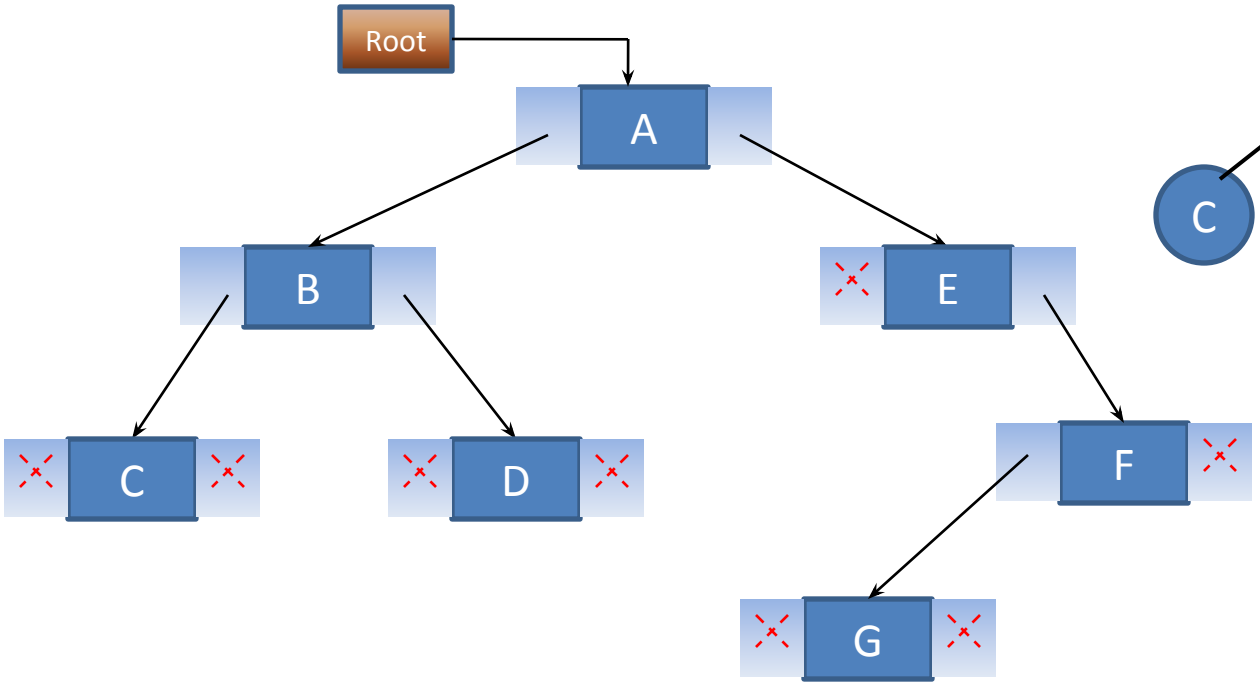


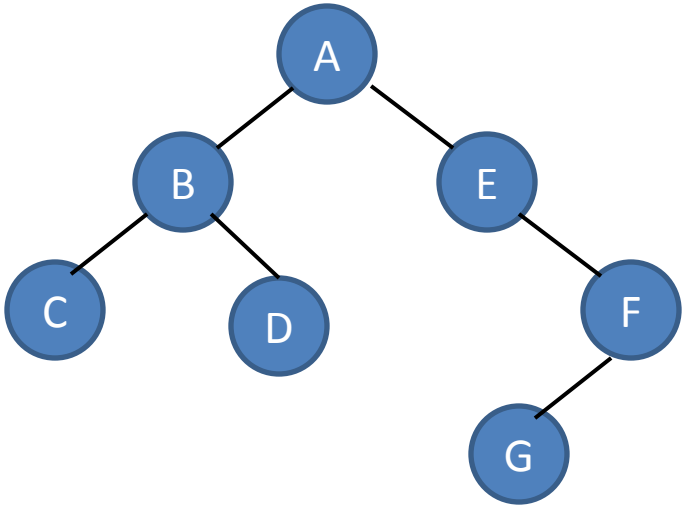
T	A	B	E	C	D	-	F	-	-	-	-	-	-	G	-
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Array size = $2^{(H+1)} - 1$ where *H* is the height of tree

Binary Tree

- Memory Representation
  - Linked representation





Threaded Binary Tree

- Threads
  - Definition

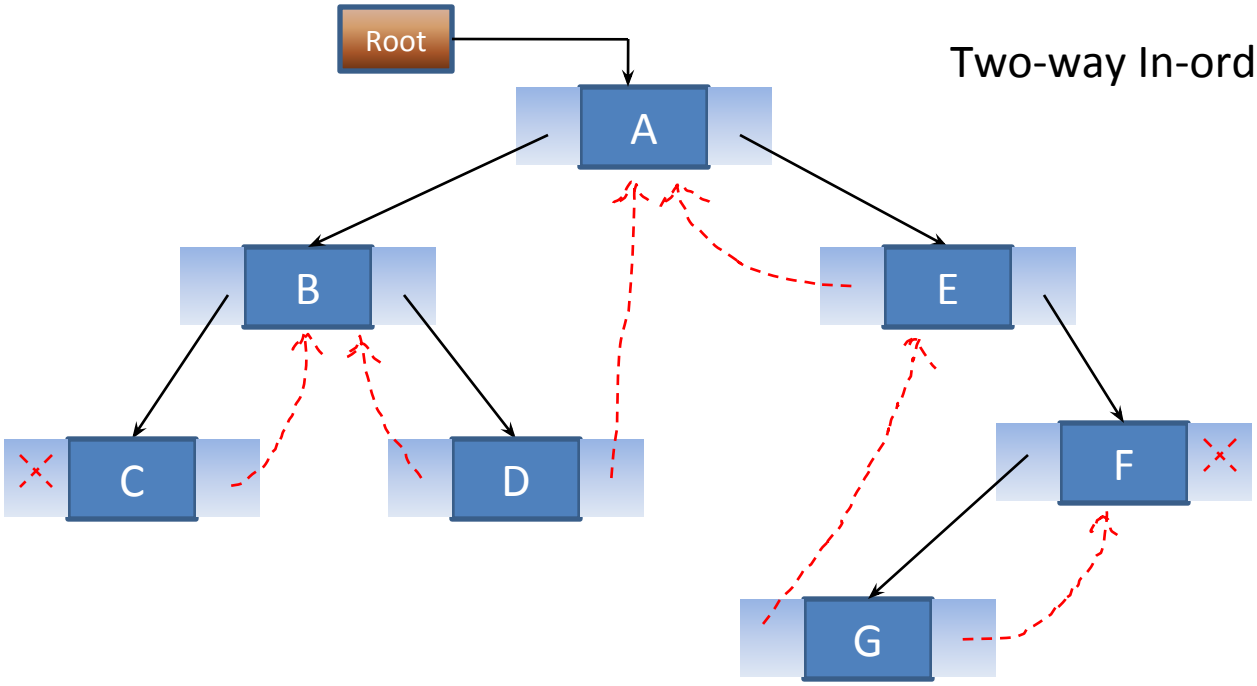
In linked representation of a binary tree, several nodes may contain null value in the pointer fields. These address fields will be replaced by special pointer values which point to the previous node of the tree. These special pointers are called threads

  - In-order Threading

We shall set the threads with tree nodes by following a traversal technique. In-order threading sets the right thread of any node with the address of such node that traversed immediate after the current node.
On the other hand, left thread is set to the immediate past node traversed by in-order traversal

Threaded Binary Tree

- In-order Threading
  - Representation



Two-way In-order threading

Binary Search Tree

- BST
  - Property
    - Every data of the left sub-tree will be lesser than its parent node data
    - Every data of the right sub-tree will be greater than its parent node data
  - Observation
    - In-order traversal of a BST will result a sorted listing of the elements stored

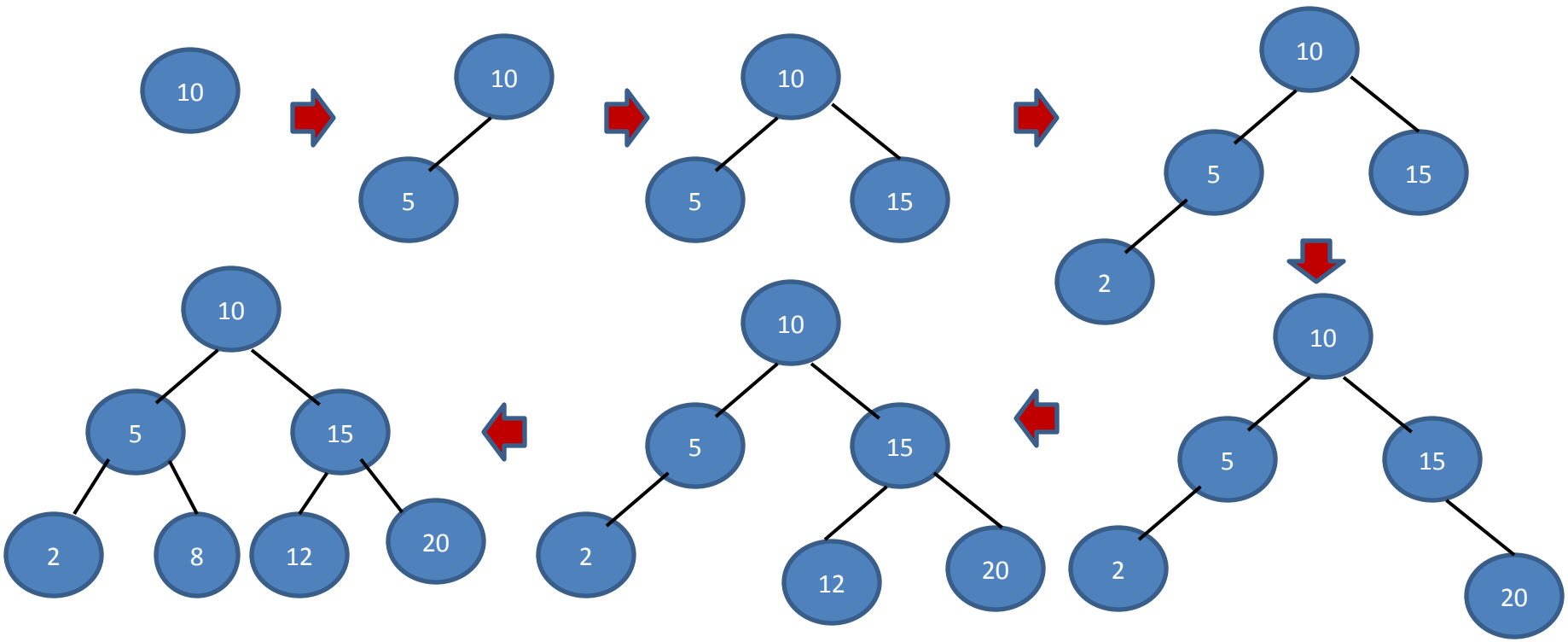
Binary Search Tree

- BST
  - Data Insertion
    - If the input data is lesser than the root (or parent) move towards its left sub-tree
    - If the input data is larger than the root (or parent), move towards right sub-tree
    - Input data position is searched starting from the root node of the tree
    - Follow steps 1 and 2 until a vacant place is found
    - If proper place found, the new data is inserted into the tree

Binary Search Tree

- BST
  - Data Insertion (example)

10 , 5 , 15 , 2 , 20 , 12 , 8



Binary Search Tree

- BST
  - Data Insertion (algorithm)

```
Insert_data(root, new) // root is the head pointer which holds the entire
{                              //  tree and 'new' is the new node to insert
 if root = NULL
   root=new
 else
   set ptr =root        // 'ptr' is an augmented pointer used to traverse the tree
   while ptr != NULL
        if new->data > ptr->data
           ptr = ptr->right
        else
           ptr = ptr->left
   set ptr = new
}
```

Binary Search Tree

- BST
  - Data Insertion (complexity)

✔ In a BST, every parent has maximum 2 children
✔ Each level of the tree, will contain the total number of nodes in power of 2
  - level zero will contain 2^0 node = 1 node (root)
  - level one will contain 2^1 nodes = 2 nodes
  - level 'i' will contain 2^i nodes
✔ To insert a new data, maximum we need to traverse the height of the tree
  - we should follow any particular path from root to leaf node
✔ If 'h' is the height of tree which will be the i^{th} level
✔ Assume that there are 'n' nodes in the level 'i' of the tree
✔ Hence, $2^i = n \Rightarrow \log_2 2^i = \log_2 n \Rightarrow i = \log_2 n$ (height of tree in terms of n)

✔ Time complexity to insert a data into tree is $O(\log_2 n)$

Queries?

Problem

Q. Insert following data in a BST and print the data in pre-order sequence.

[10 , 3 , 15 , 20 , 5 , 2 , 12 , 30 , 14 , 8 , 25 , 18]