Data Structures

Trees

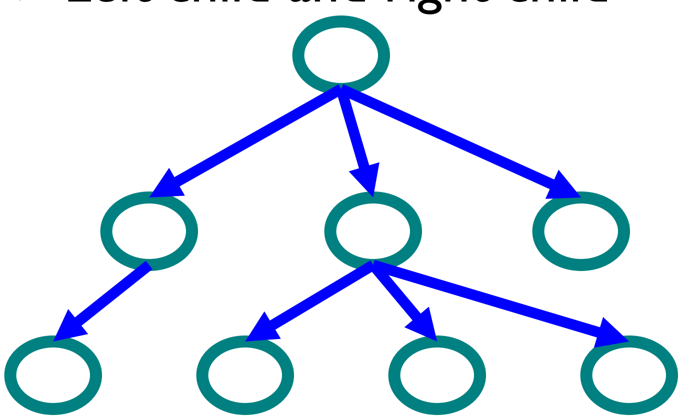
Rutvij H. Jhaveri
Computer Science & Engineering

Outline

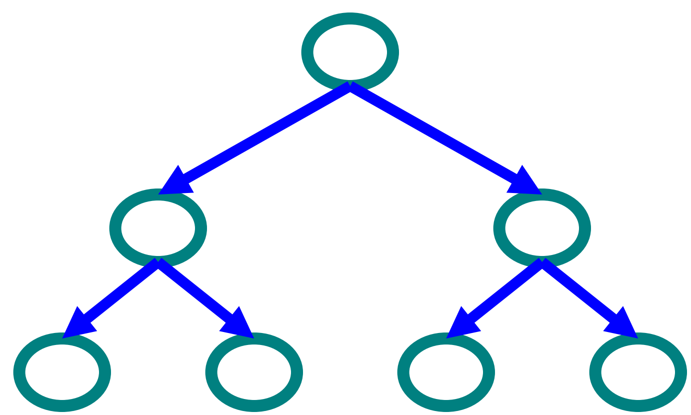
- What is Tree
- Binary Tree
- Representations of Binary Tree
- Applications
- Binary Search Tree Traversals
- Expression Trees (Extra topic)
- Conversion from General to Binary Tree

Tree and Binary Tree

- Tree
  - Set of Nodes
  - Each node can have 0 or more children
  - A node can have at most one parent
- Binary Tree
  - Each node has at most two children
    - Left child and right child



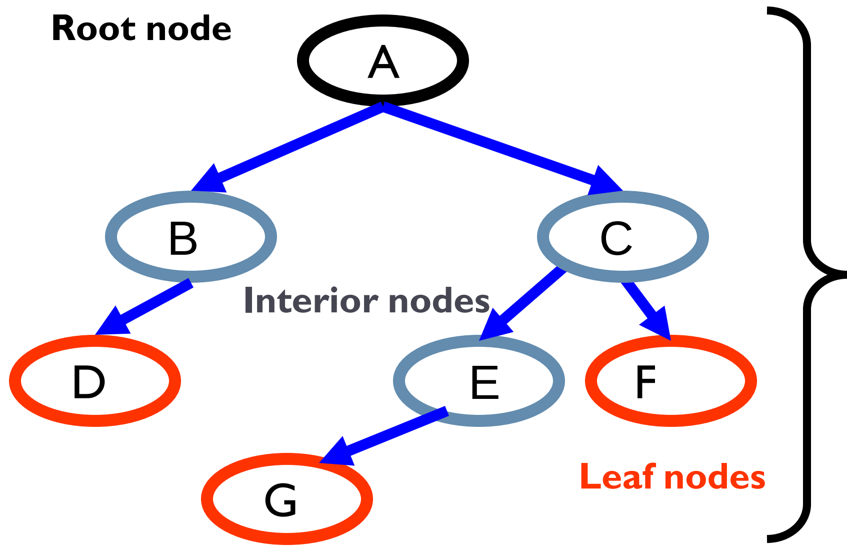
N-ary Tree (N=3)



Binary Tree

Terminology

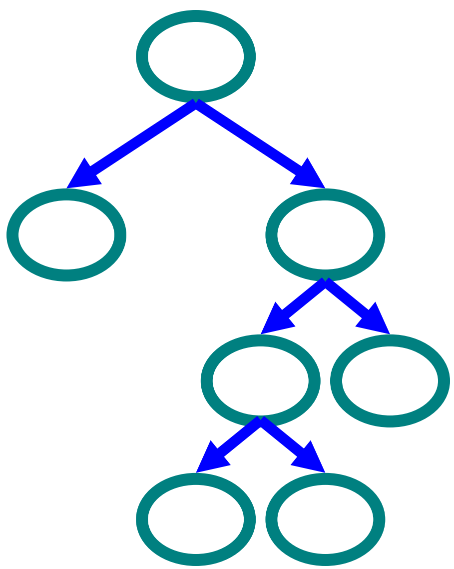
- Terminology
  - Node ⇒ each element
  - Root ⇒no parent
  - Leaf ⇒ no child
  - Interior ⇒ non-leaf

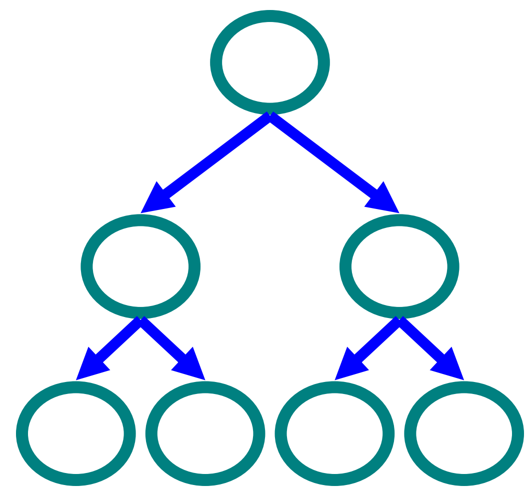


  - Height (Depth) ⇒ distance ‘d’ from root to leaf
  - Level ⇒ Root A at 0 B,C-1 D,E,F-2, G-3
  - Parent ⇒ ??
  - Siblings-Ancestor-Descendents (left & right) ⇒ ??
  - Left-subtree-Right-subtree ⇒ ??

Binary Trees-Types

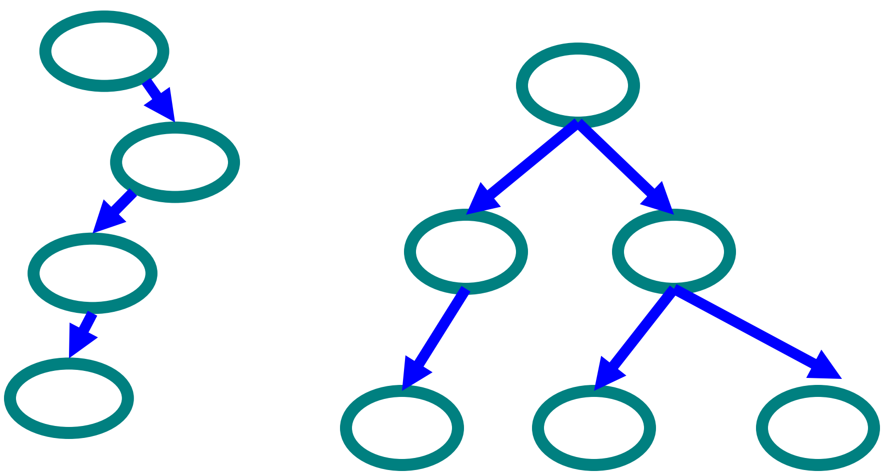
- Strictly Binary Tree
  - Every nonleaf node has non empty left and right subtrees
- Complete Binary Tree
  - Strictly binary tree
  - All leaves are at depth



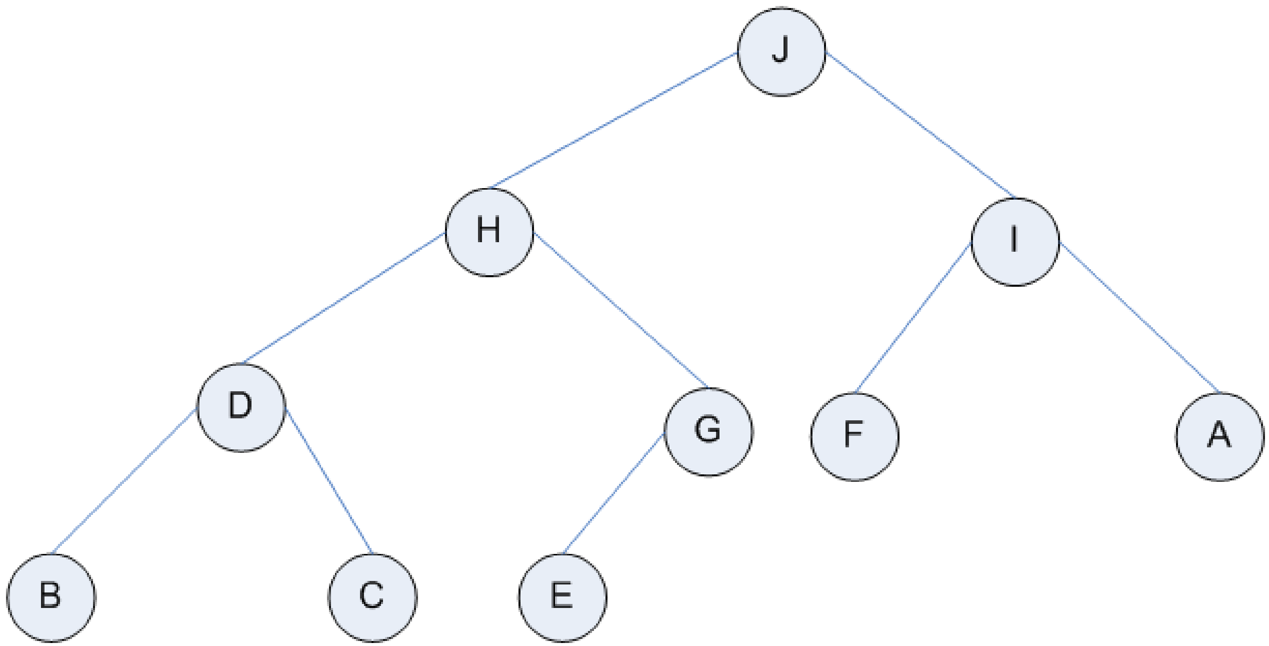


...Binary Trees-Types

- Degenerate
  - each parent node has only one child
- Almost Complete
  - Any node at level less than d-1 has two sons
  - If node has right son, it must have left son
  - Every left descendent of a node is either a leaf at level d or has two sons
  - Is second tree Almost complete??
  - If not then make it, such that it is not complete binary tree

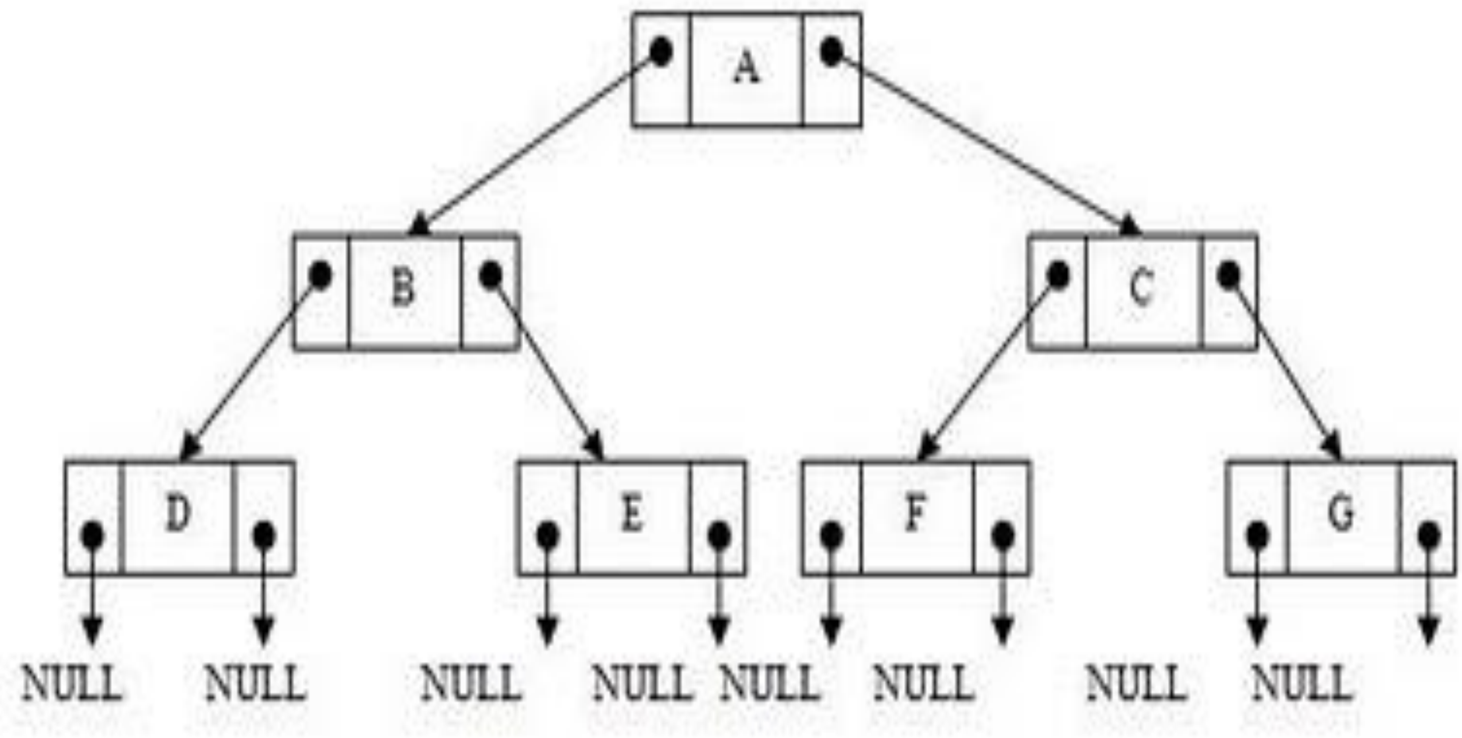


Array Representation of Binary Tree



J	H	I	D	G	F	A	B	C	E
0	1	2	3	4	5	6	7	8	9

Linked List Representation of Binary Tree

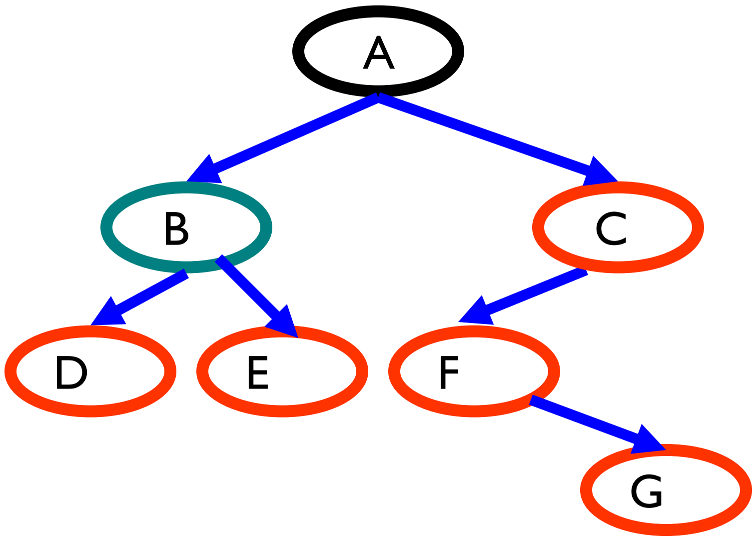


Applications

- Two-way decisions
  - Finding duplicates
  - Comparing a number with sets of numbers
- Binary Search Tree
- Expression tree (Parse tree) –conversion to infix, prefix and postfix
- Huffman Coding
- Heaps
- 2-3 Trees, AVL Trees
- B-Tree, B+-Trees
- Tree Representation of data (like HTML, XML)
- Binary Tries
- Hash Trees
- Binary Space Partitions
- Euler Tour Traversal ……

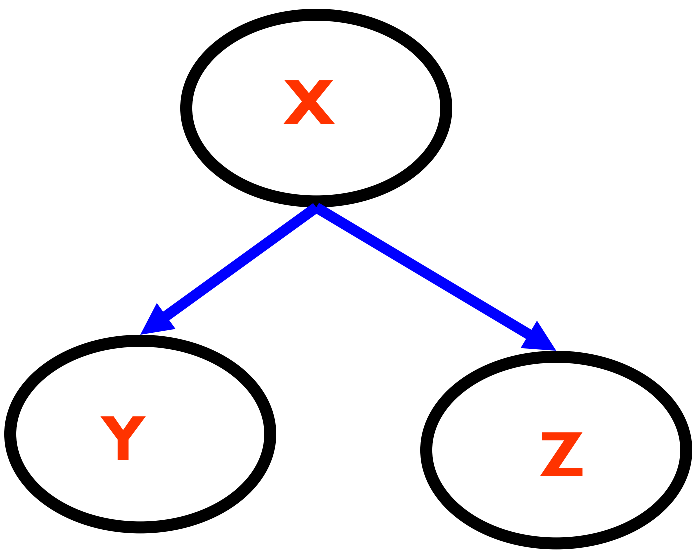
Traversals

- Inorder – Visit left subtree –root- visit right subtree
- Preorder – root- Visit left subtree - visit right subtree
- Postorder – Visit left subtree - visit right subtree -root
- What will be the traversals for…



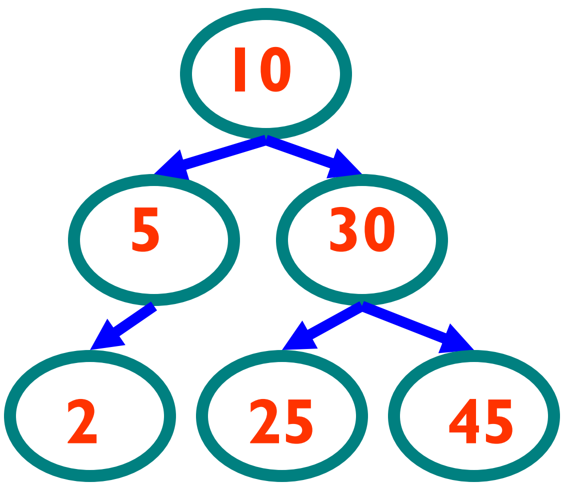
Binary Search Trees (BST)

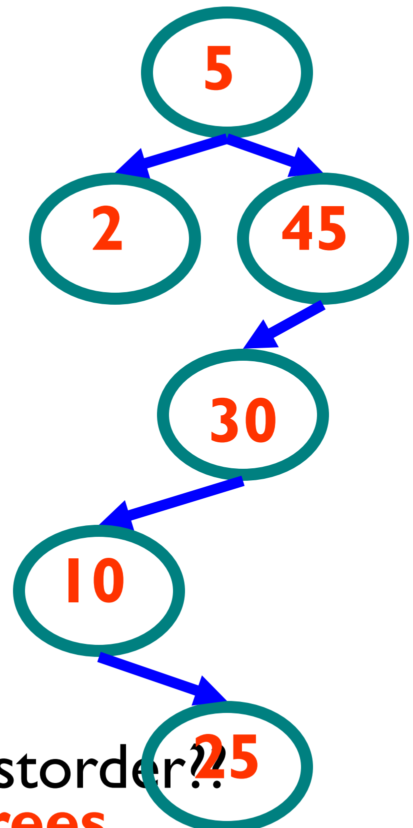
- Key property
  - Value at node
    - Smaller values in left subtree
    - Larger values in right subtree
  - Example
    - X > Y
    - X < Z

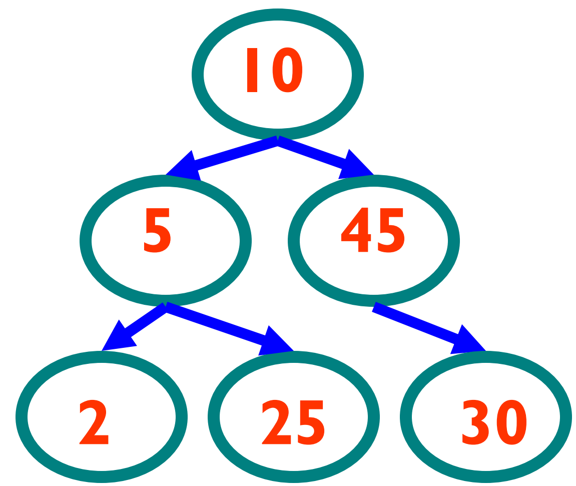


…Binary Search Trees

- Examples







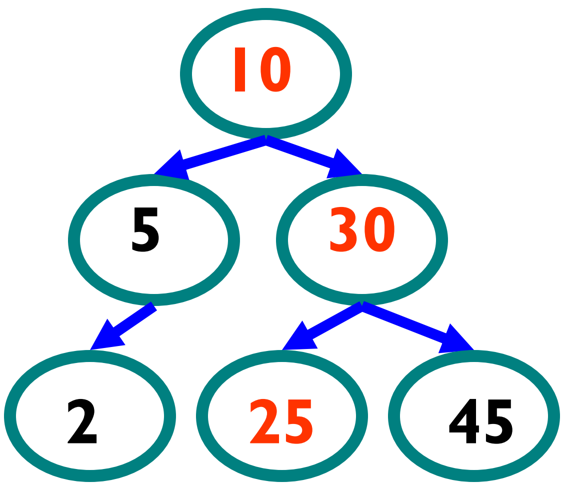
- Inorder, Preorder, Postorder??

Binary search trees

Not a binary search tree

Example Binary Searches

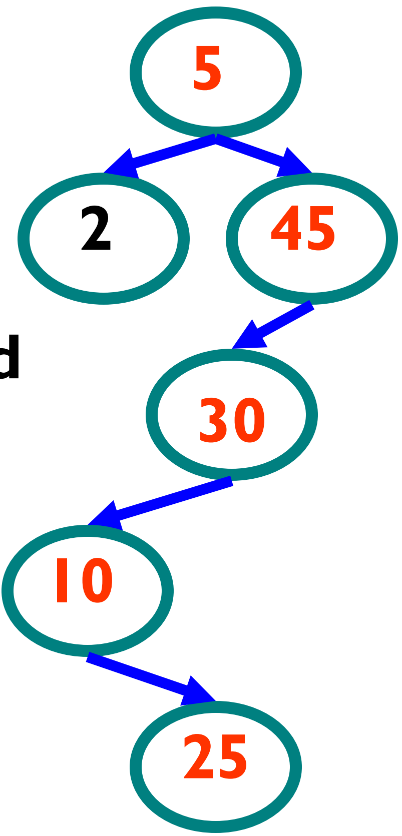
- Find (root, 25)



10 < 25, right

30 > 25, left

25 = 25, found



5 < 25, right

45 > 25, left

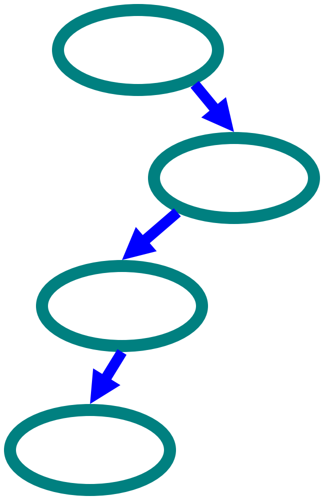
30 > 25, left

10 < 25, right

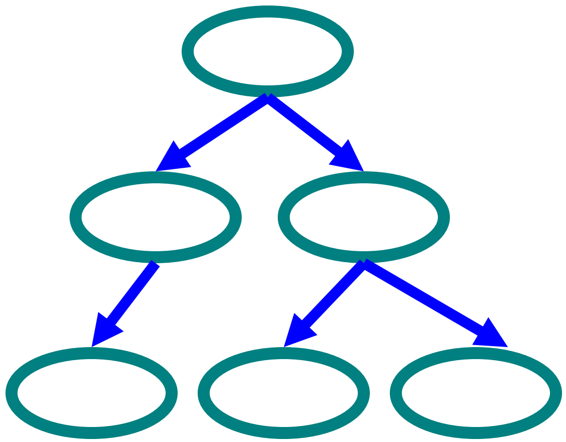
25 = 25, found

Binary Trees Properties

- Degenerate
  - Height = O(n) for n nodes
  - Similar to linked list
- Balanced
  - Height = O(log(n)) for n nodes
  - Useful for searches



Degenerate binary tree



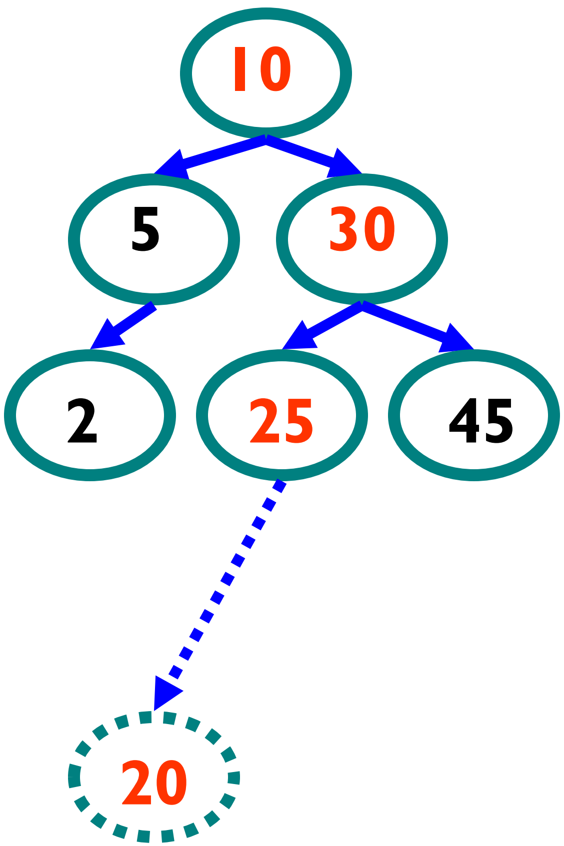
Balanced binary tree

BST Properties

- Time of search
  - Proportional to height of tree
  - Balanced binary tree
    - O(log(n)) time
  - Degenerate tree
    - O(n) time
    - Like searching linked list / unsorted array

Example Insertion

- Insert (20)



10 < 20, right

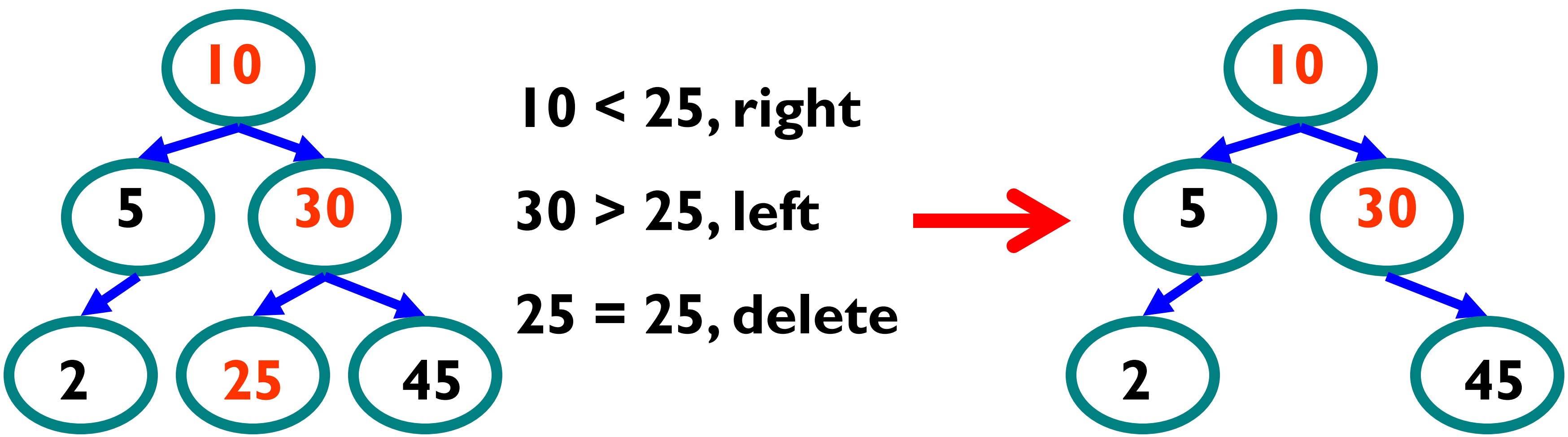
30 > 20, left

25 > 20, left

Insert 20 on left

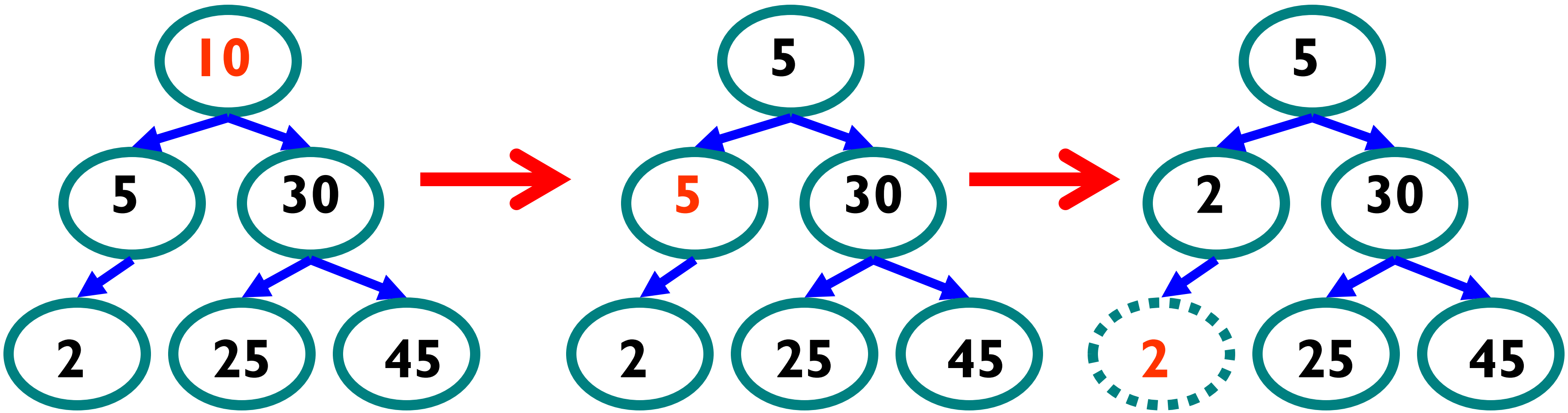
Example Deletion (Leaf)

- Delete (25)



Example Deletion (Internal Node)

- Delete (10)



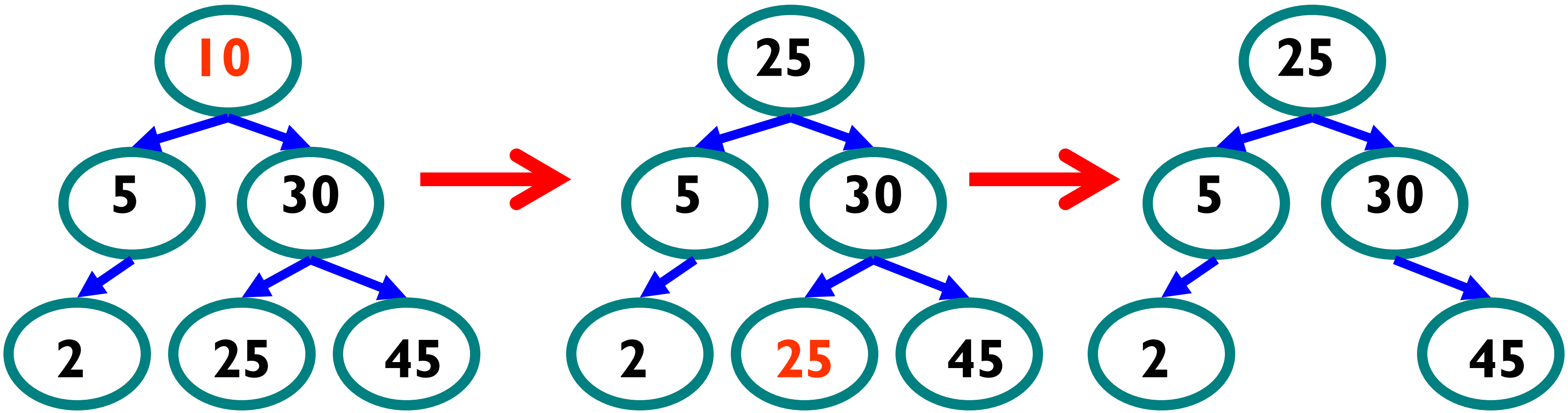
Replacing 10 with largest value in left subtree

Replacing 5 with largest value in left subtree

Deleting leaf

…Example Deletion (Internal Node)

- Delete (10)



Replacing 10 with smallest value in right subtree

Deleting leaf

Resulting tree

Program for binary search tree

binary_search_tree.c

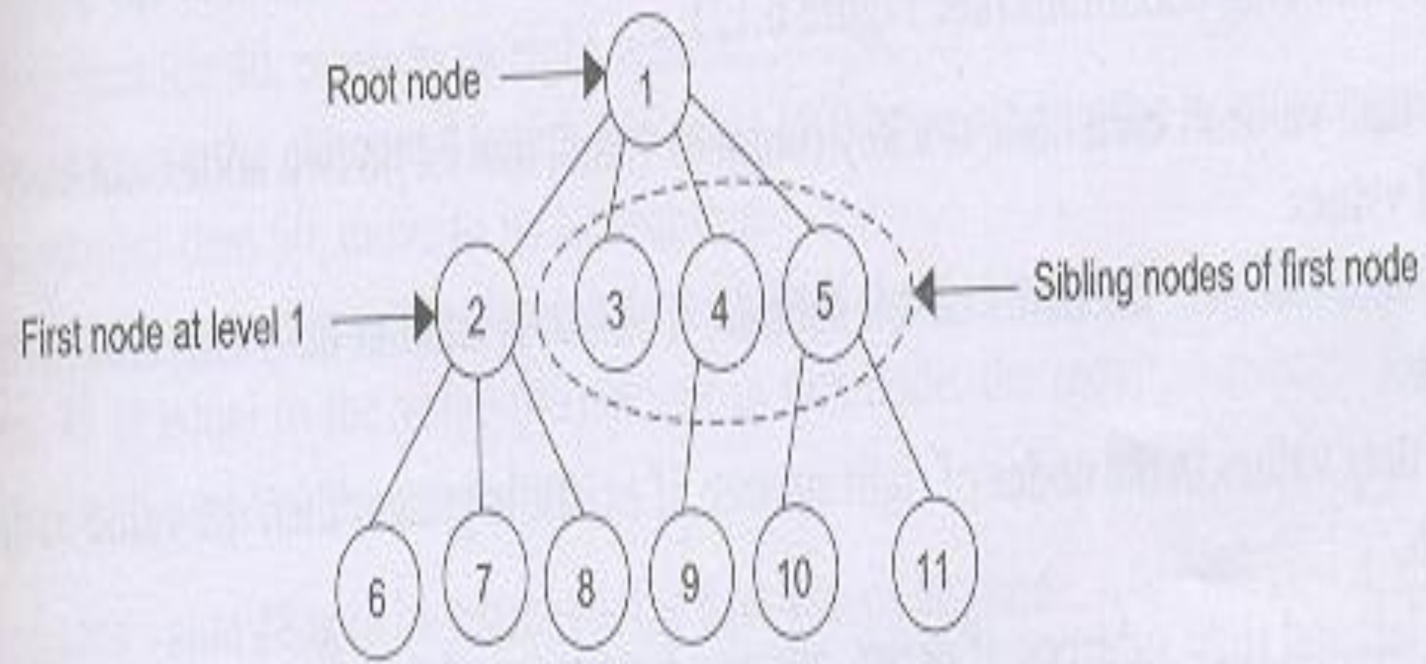
Expression Trees

- Binary expression trees capture the precedence and associative nature of arithmetic operators
- Let's represent following expressions as binary trees and convert the following expressions into pre-fix and post-fix:
  - A+B*C-D
  - A+B*(C-D)^E
  - 2*3/(2-1)+5*(4-1)

Converting General Tree to Binary Tree

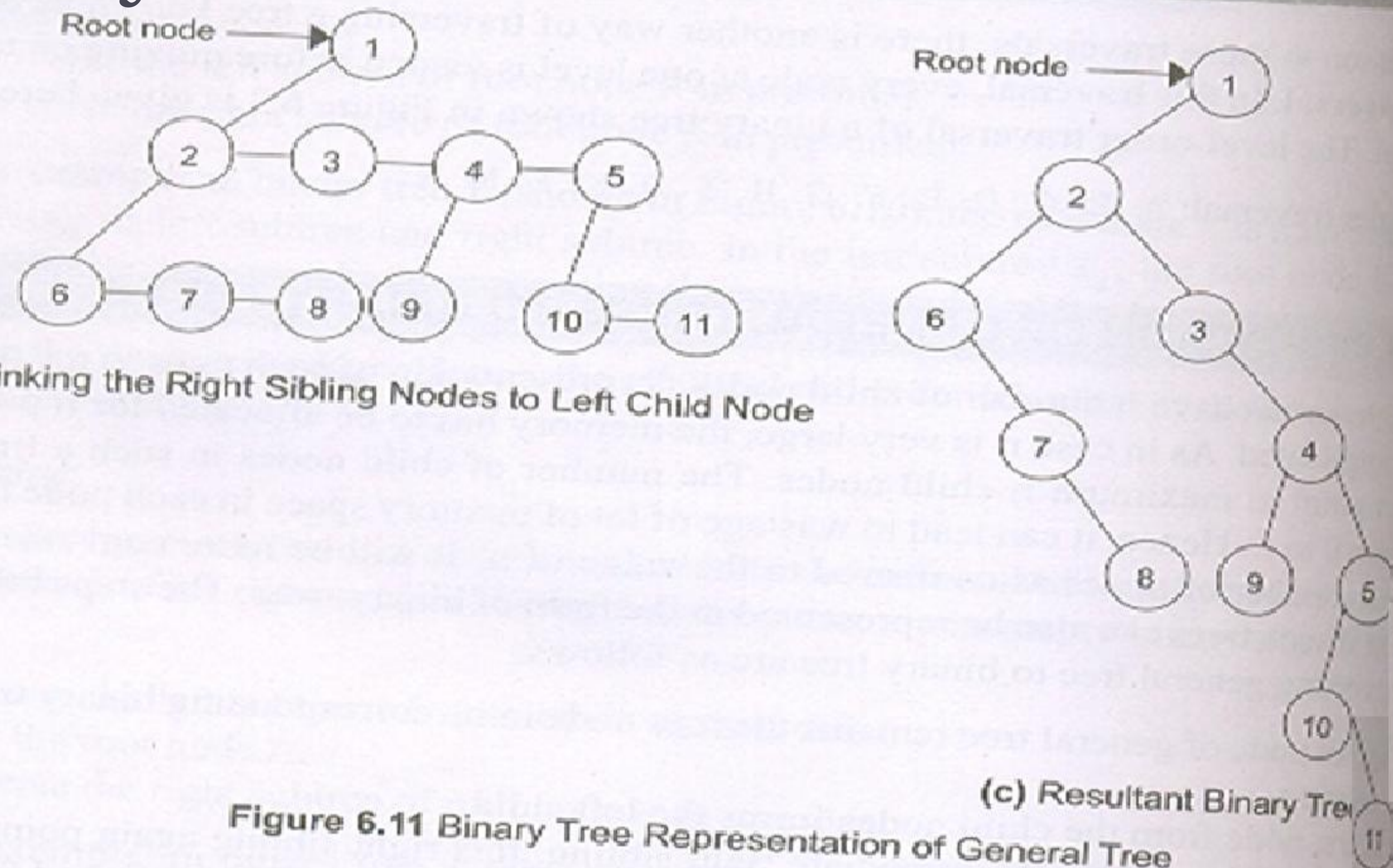
- Steps:
  - Root of general tree (N-ary tree) remains root for binary tree
  - First node from child nodes forms left child
  - Delete all pointers except left pointer
  - Link all siblings of the tree
  - Rotate the tree clockwise by 45º

...Converting General Tree to Binary Tree



(a) General Tree

…Converting General Tree to Binary Tree



(b) Linking the Right Sibling Nodes to Left Child Node

(c) Resultant Binary Tree

Figure 6.11 Binary Tree Representation of General Tree

Threaded Binary Trees

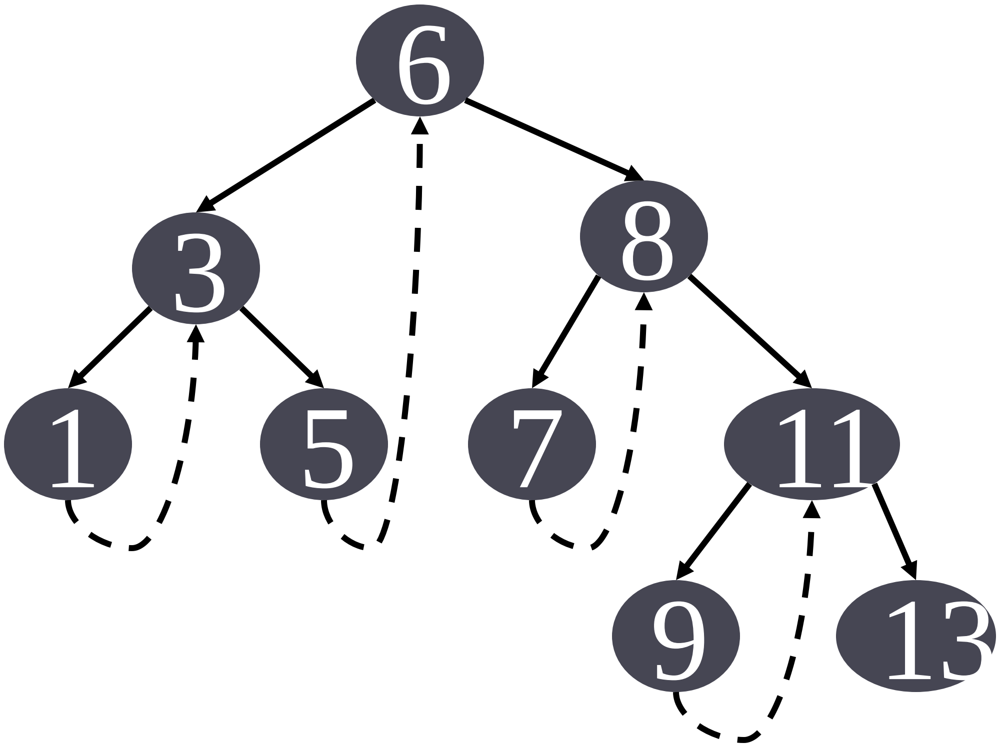
- Binary trees have a lot of wasted space: the leaf nodes each have 2 null pointers
- We can use these pointers to help us in inorder traversals
- We have the pointers reference the next node in an inorder traversal; called *threads*
- We need to know if a pointer is an actual link or a thread, so we keep a boolean for each pointer

…Threaded Binary Tree

```
struct nodetype
{
 int info;
 struct nodetype *left;
 struct nodetype *right;
 int lthread;
 int rthread
};
```

- Value 1 for lthread/rthread indicates the left/right pointers are normal pointers and 0 indicates that they are threads

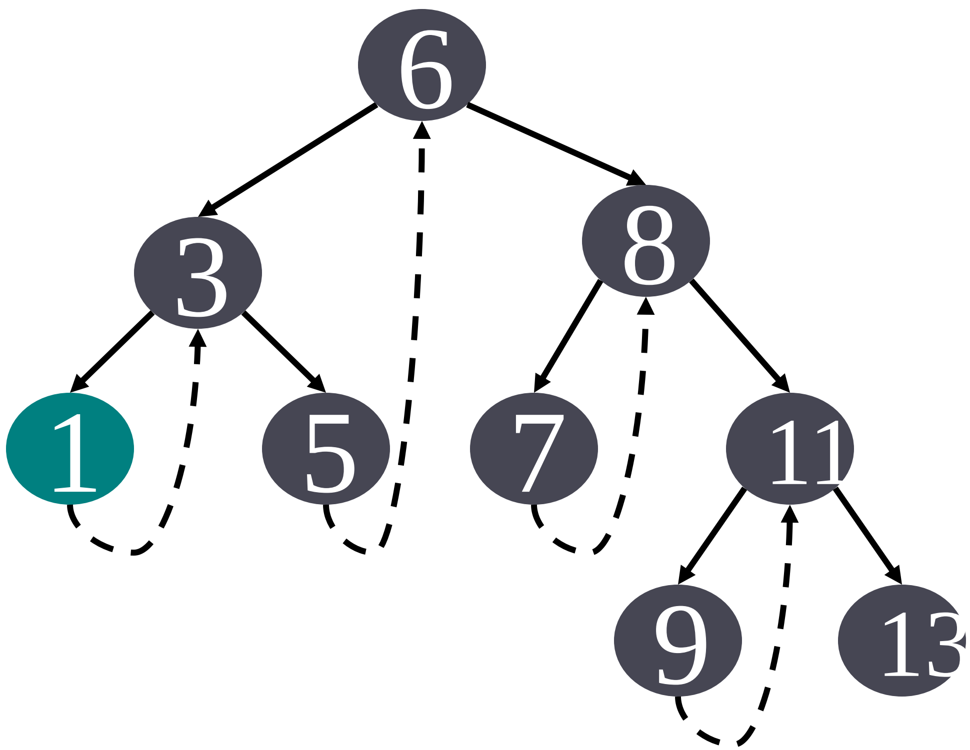
Right In-Threaded Tree Example



Right In-Threaded Tree Traversal

- We start at the leftmost node in the tree, print it, and follow its right thread
- If we follow a thread to the right, we output the node and continue to its right
- If we follow a link to the right, we go to the leftmost node, print it, and continue

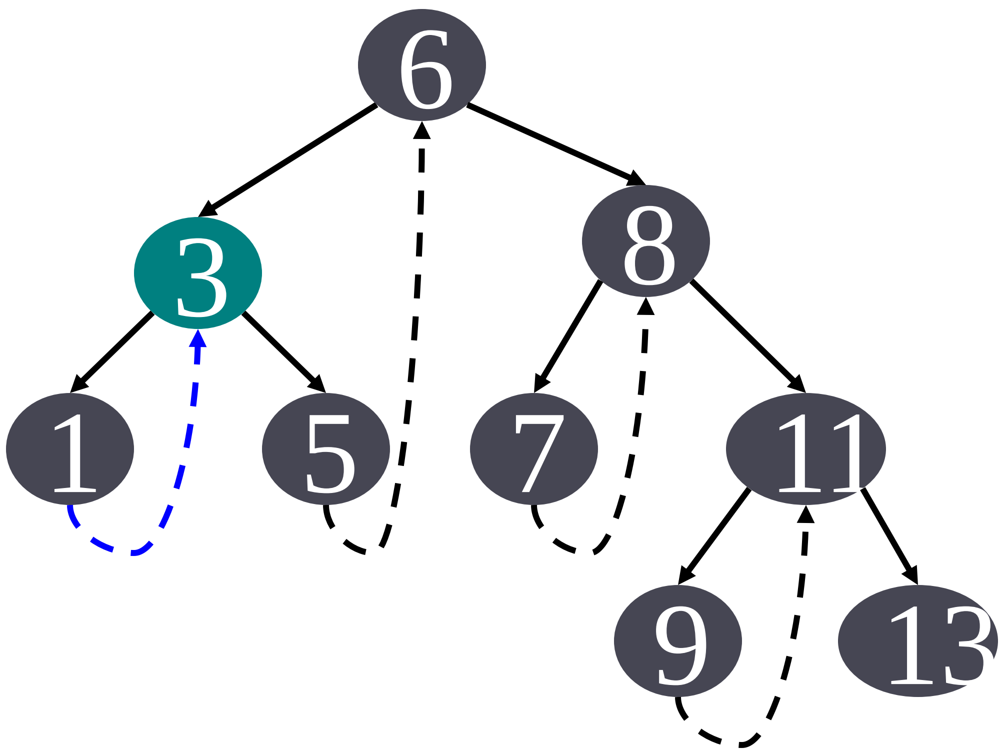
…Right In-Threaded Tree Traversal



Output
1

Start at leftmost node, print it

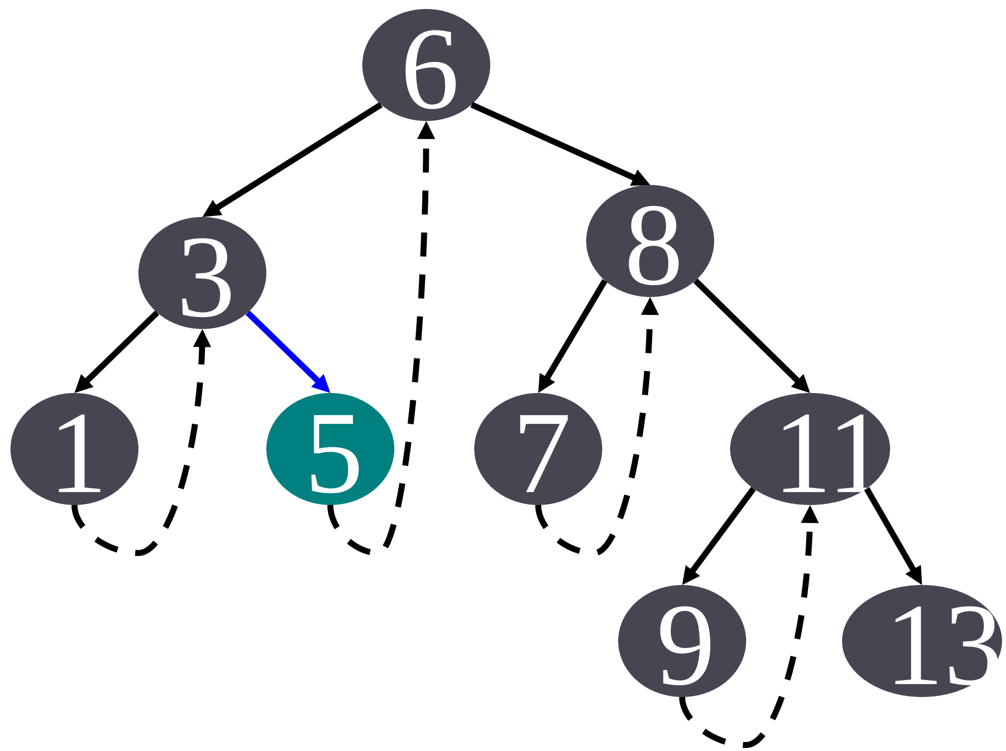
…Right In-Threaded Tree Traversal



Output
1
3

Follow thread to right, print node

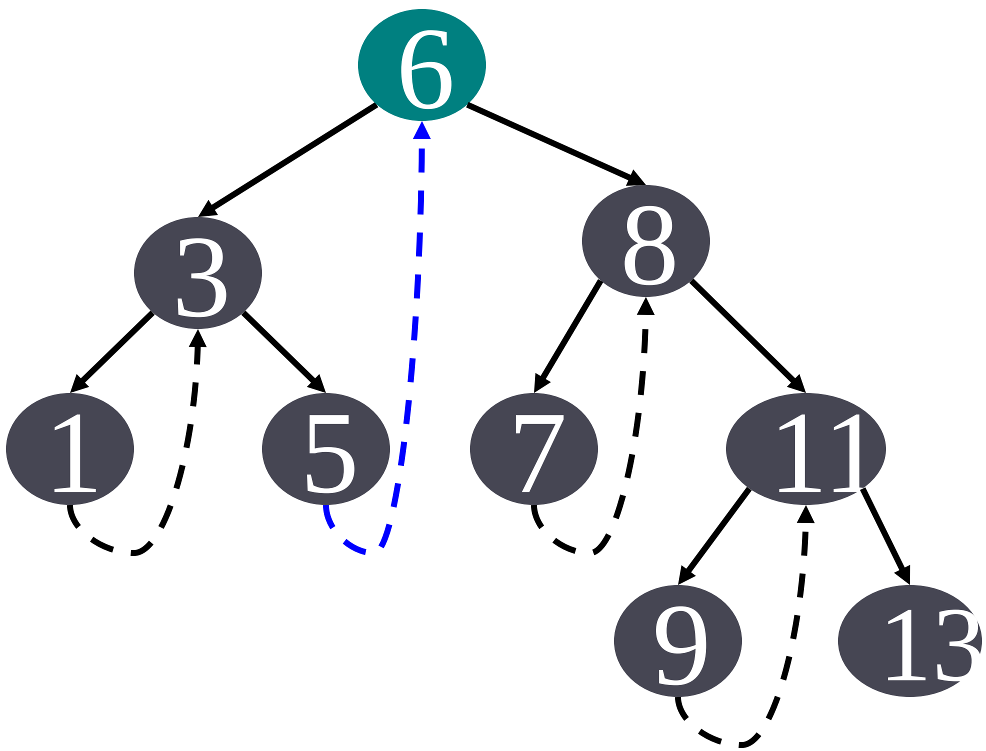
…Right In-Threaded Tree Traversal



Output
1
3
5

Follow link to right, go to leftmost node and print

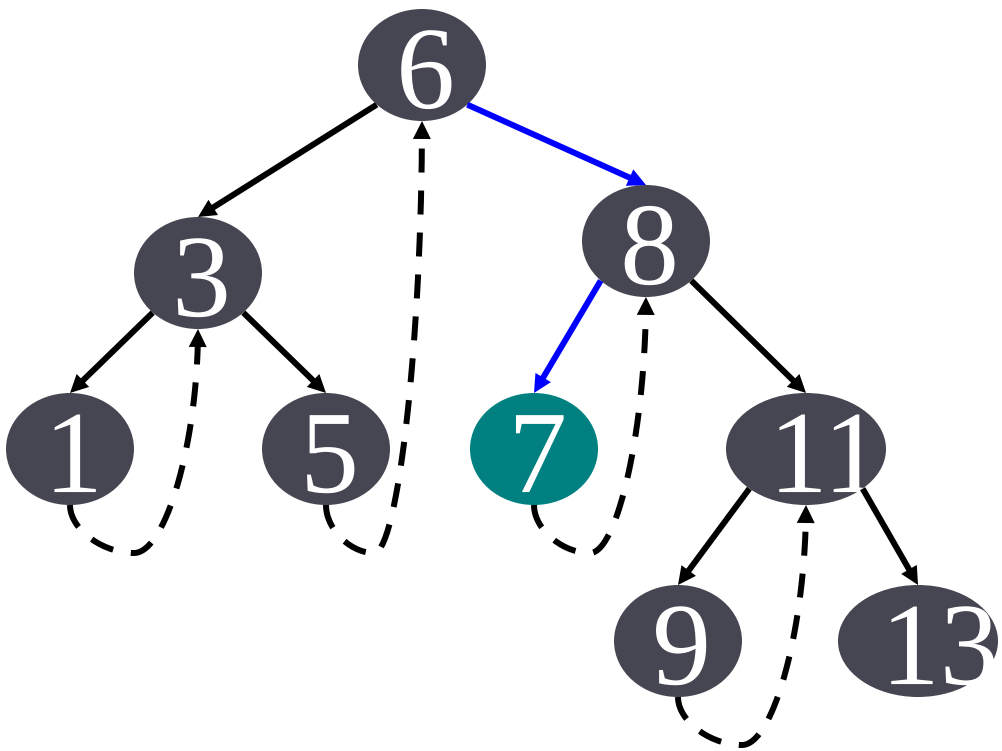
…Right In-Threaded Tree Traversal



Output
1
3
5
6

Follow thread to right, print node

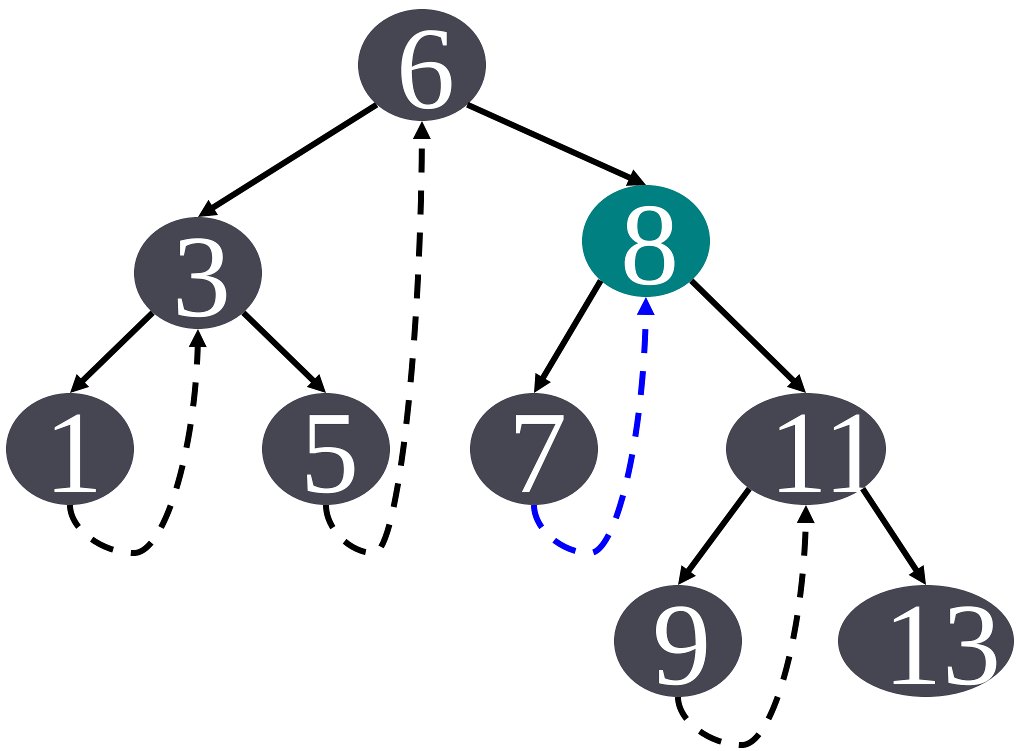
…Right In-Threaded Tree Traversal



<u>Output</u>
1
3
5
6
7

Follow link to right, go to leftmost node and print

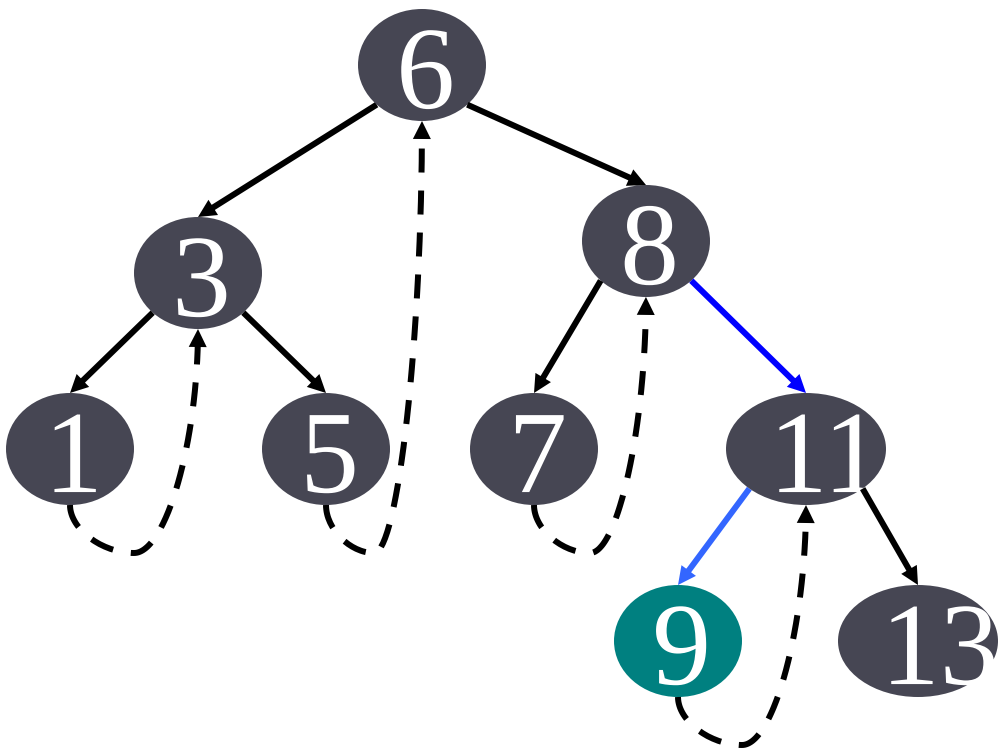
…Right In-Threaded Tree Traversal



Output
1
3
5
6
7
8

Follow thread to right, print node

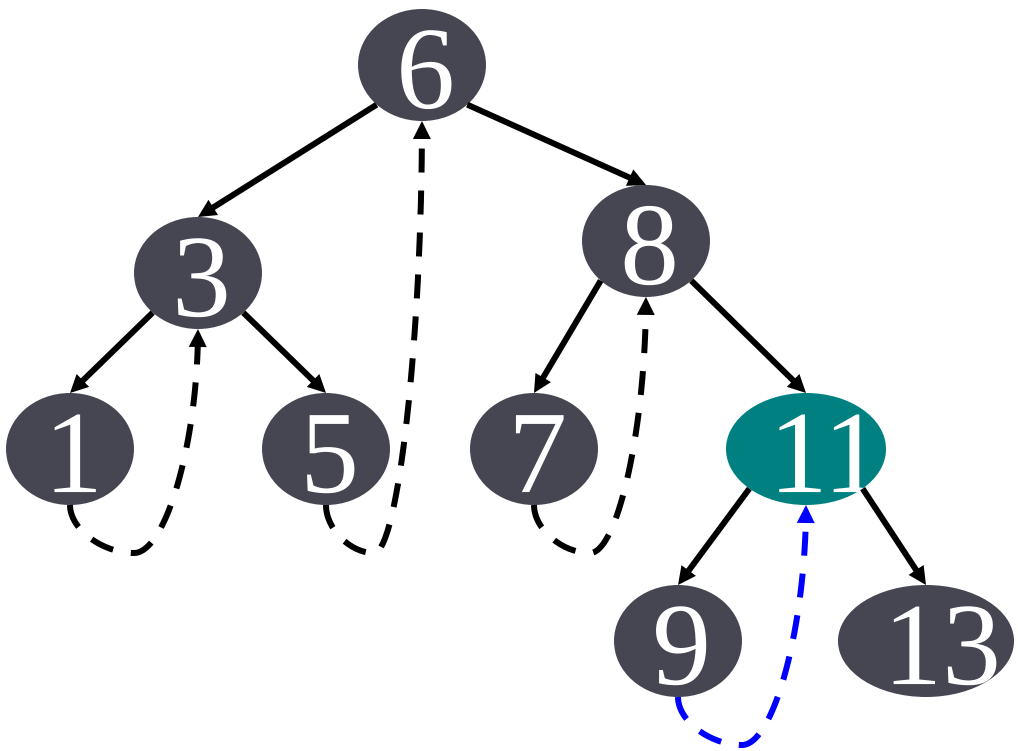
…Right In-Threaded Tree Traversal



Output
1
3
5
6
7
8
9

Follow link to right, go to leftmost node and print

…Right In-Threaded Tree Traversal

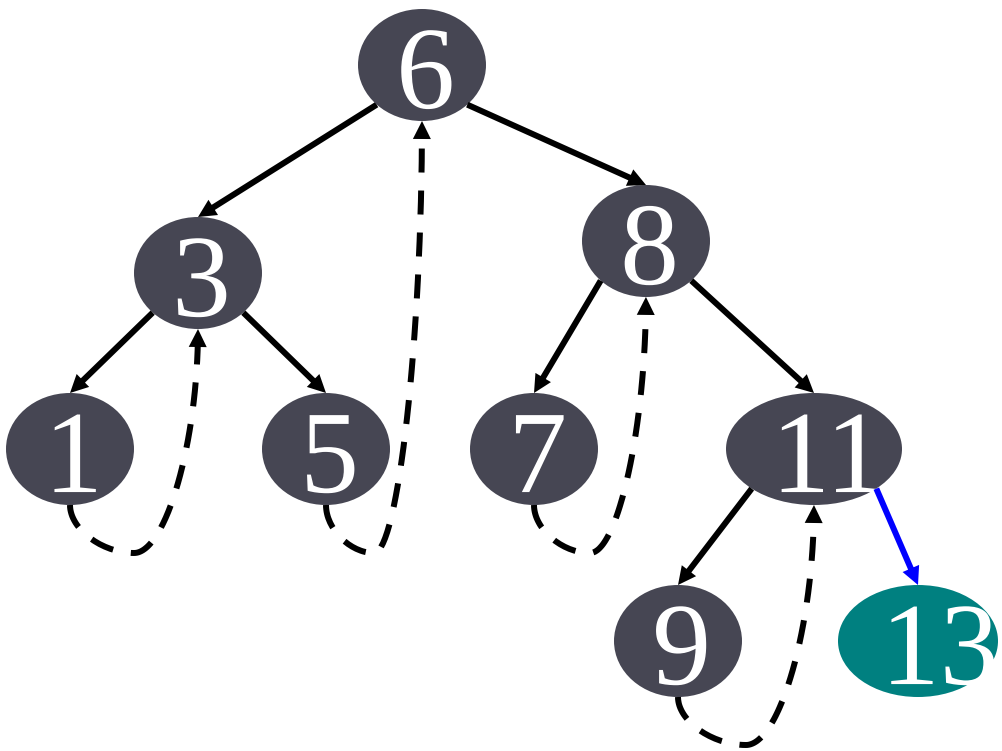


Output

1
3
5
6
7
8
9
11

Follow thread to right, print node

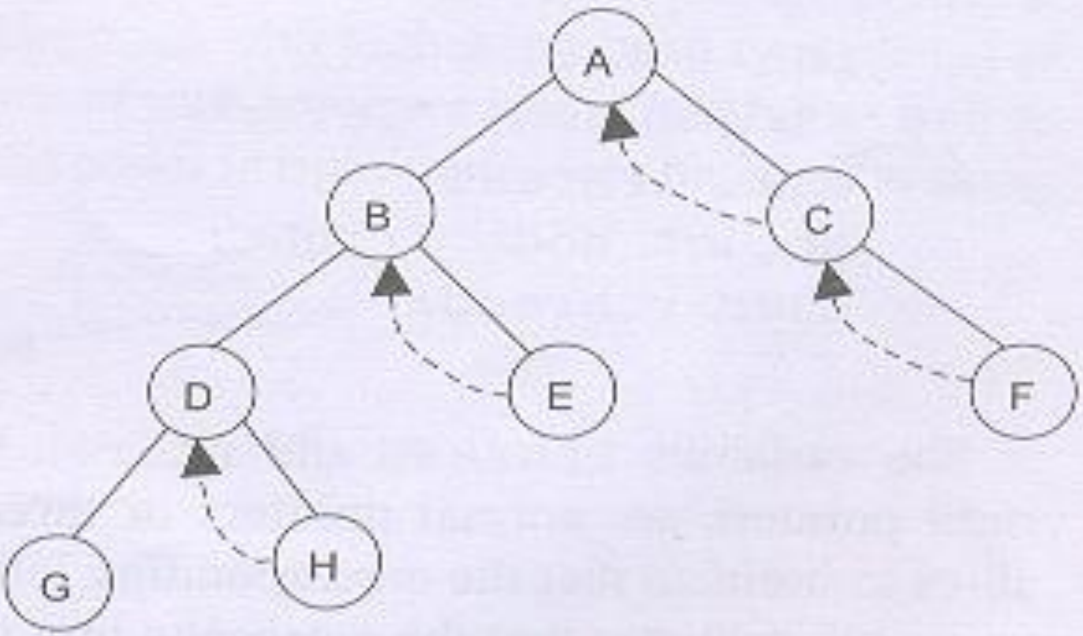
…Right In-Threaded Tree Traversal



Output
1
3
5
6
7
8
9
11
13

Follow link to right, go to leftmost node and print

Left In-Threaded Tree Example

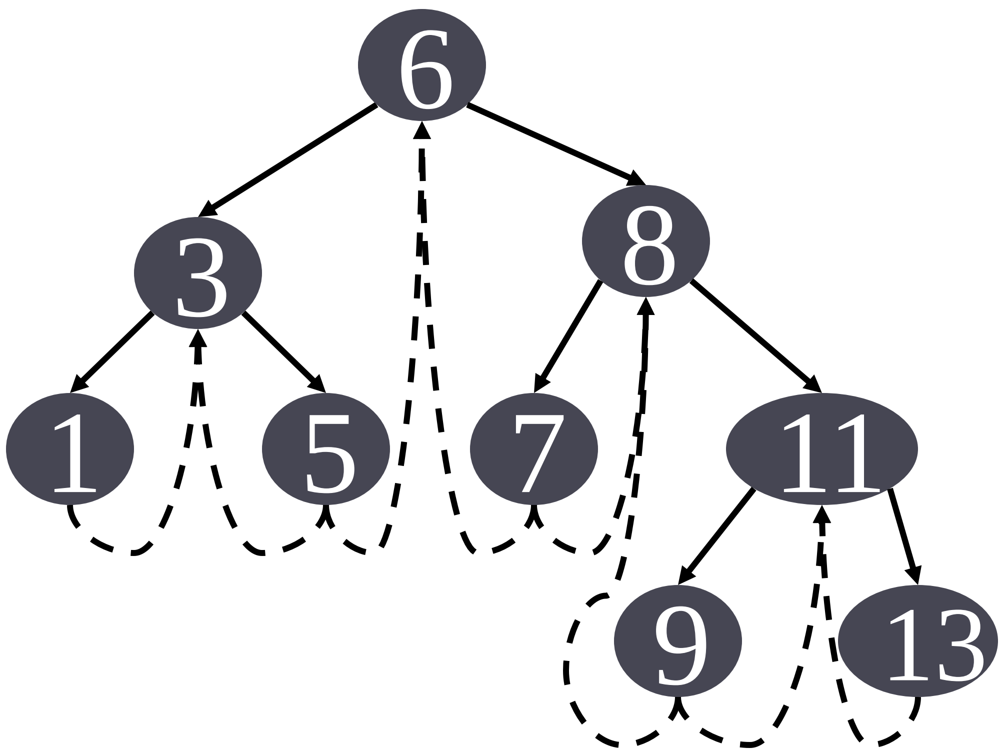


(b) Left-threaded Binary Tree

Full In-Threaded Tree

- We're still wasting pointers, since half of our leafs' pointers are still NULL
- Combine Right threaded and Left threaded tree

Full In-Threaded Tree Example



Storage Representation of Full In-Threaded Tree

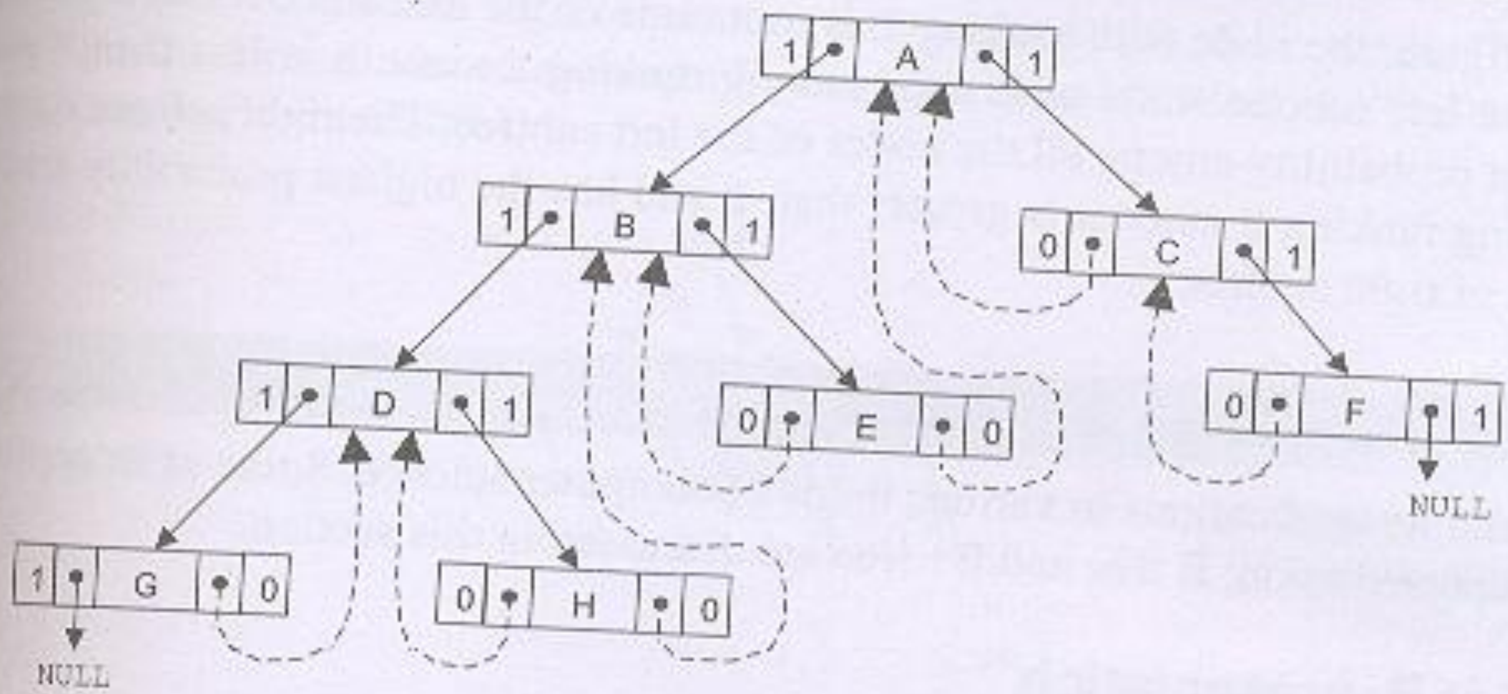


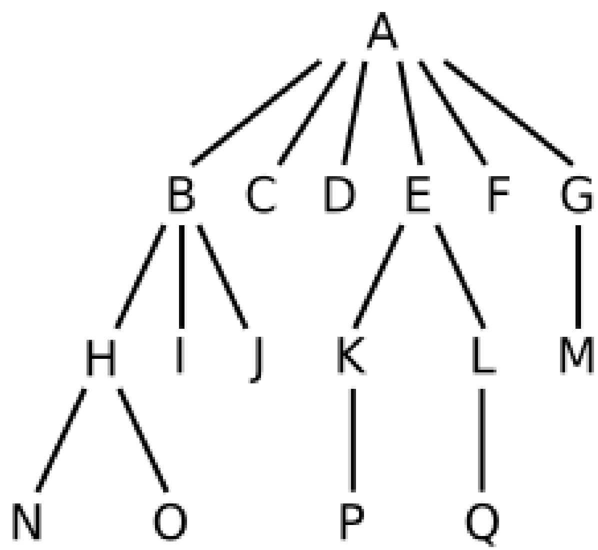
Figure 6.26 Storage Representation of Full-threaded Binary Tree

Assignments

- Algorithm for evaluating an expression using tree
- Draw Binary Search Trees and Traverse In-order, Pre-order and Post-order:
  - 50, 30, 0, 20, 70, 90, 80, 10, 40
  - 40, 20, 0, 30, 80, 90, 70, 10, 50
- Draw Expression Trees and Traverse Pre-order and Post-order:
  - A+B-C+D/F*G/H-I+J
  - A/B/C^D-E^(F^G)^H
  - A*(B*C)*D/(F/G^H)^I^J

…Assignments

- Convert the following tree to binary tree:



Summary

- Binary tree types
- Traversals
- Applications
- Binary search trees
- Expression trees
- General to binary tree conversion
- Threaded binary trees