Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

CET2001B Advanced Data Structures

S. Y. B. Tech CSE

Semester – IV

SCHOOL OF COMPUTER ENGINEERING AND TECHNOLOGY

CET2001B: Advanced Data Structures

·**requisites**: Fundamentals of Data Structures

Course Objectives:

1. Knowledge

i. Learn the nonlinear data structure and its fundamental concept.

2. Skills

i. Understand the different nonlinear data structures such as Trees and Graph.
ii. Study the concept of symbol table, heap, search tree and multiway search tree.
iii. Study the different ways of file organization and hashing concepts.

3. Attitude

i. Learn to apply advanced concepts of nonlinear data structure to solve real world problems.

Course Outcomes:

After completion of the course the students will be able to :-

1. To choose appropriate non-linear data structures to solve a given problem.
2. To apply advanced data structures for solving complex problems of various domains.
3. To apply various algorithmic strategies to approach the problem solution.
4. To compare and select different file organization and to apply hashing for implementing direct access organization.

CET2001B Advanced Data Structures: Assessment Scheme:

Class Continuous Assessment (CCA) - 30 Marks

Mid Term	Active Learning	Theory Assignment
15 Marks	10 Marks	5 Marks

Laboratory Continuous Assessment (LCA) - 30 Marks

Practical Performance	Additional Implementation/ On paper Design	End term Practical Examination
10 Marks	10 Marks	10 Marks

Term End Examination: 40 Marks

Syllabus

1. Hashing - Concepts-hash table, hash function, basic operations, bucket, collision, probe, synonym, overflow, open hashing, closed hashing, perfect hash function, load density, full table, load factor, rehashing, issues in hashing, hash functions- properties of good hash function, division, multiplication, extraction, mid-square, folding and universal, Collision resolution strategies- open addressing and chaining, Hash table overflow- open addressing and chaining.

2. Tree - Basic Terminology, Binary Tree- Properties, Converting Tree to Binary Tree, Representation using Sequential and Linked organization, Binary tree creation and Traversals, Operations on binary tree.
Binary Search Tree (BST) and its operations,
Threaded binary tree- Creation and Traversal of In-order Threaded Binary tree.
Case Study- Expression tree

3. Graph - Basic Terminology, Graphs (Directed, Undirected), Various Representations, Traversals & Applications of graph- Prim's and Kruskal's Algorithms, Dijsktra's Single source shortest path, Analysis complexity of algorithm, topological sorting.

Continued…

4. Heap - Heap as a priority queue, Heap sort. Symbol Table-Representation of Symbol Tables- Static tree table and Dynamic tree table, Weight balanced tree - Optimal Binary Search Tree (OBST), OBST as an example of Dynamic Programming, Height Balanced Tree- AVL tree.
Search trees: Red-Black Tree, AA tree, K-dimensional tree, Splay Tree.

5. Multiway search trees, B-Tree - insertion, deletion, B+Tree - insertion, deletion, use of B+ tree in Indexing, Trie Tree.

Files: concept, need, primitive operations. Sequential file organization - concept and primitive operations, Direct Access File- Concepts and Primitive operations, Indexed sequential file Organization-concept, types of indices, structure of index sequential file, Linked Organization - multi list files.

List of Assignments

1. Implement following polynomial Operations using Circular Linked List :
   1) Create 2) Display 3) Addition

2. Implement binary tree and perform following operations: Creation of binary tree and traversal recursive and non-recursive.

3. Implement a dictionary using a binary search tree where the dictionary stores keywords & its meanings. Perform following operations:
- Insert a keyword
- Delete a keyword
- Create mirror image and display level wise
- Copy
- Create mirror image and display level wise

4. Implement threaded binary tree. Perform inorder traversal on the threaded binary tree.

List of Assignments contd…

5. Consider a friend’s network on Facebook social web site. Model it as a graph to represent each node as a user and a link to represent the friend relationship between them using adjacency list representation and perform DFS traversal. Perform BFS traversal for the above graph.

6. A business house has several offices in different countries; they want to lease phone lines to connect them with each other and the phone company charges different rent to connect different pairs of cities. (Create & display of Graph). Solve the problem using Prim’s algorithm.

7. Read the marks obtained by students of second year in an online examination of a particular subject. Find the maximum and minimum marks obtained in that subject. Use heap data structure and heap sort.

8. Implement direct access file using hashing (linear probing with and without replacement) perform following operations on it a) Create Database b) Display Database c) Add a record d) Search a record e) Modify a record

9.Design a Project to implement a Smart text editor.

Learning Resources

Text Books:

1. Fundamentals of Data Structures, E. Horowitz, S. Sahni, S. A-Freed, Universities Press.
2. Data Structures and Algorithms, A. V. Aho, J. E. Hopperoft, J. D. UIlman, Pearson.

Reference Books:

1. The Art of Computer Programming: Volume 1: Fundamental Algorithms, Donald E. Knuth.
2. Introduction to Algorithms, Thomas, H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, The MIT Press.
3. Open Data Structures: An Introduction (Open Paths to Enriched Learning), (Thirty First Edition), Pat Morin, UBC Press.

Supplementary Readings:

1. Aaron Tanenbaum, “Data Structures using C”, Pearson Education.
2. R. Gilberg, B. Forouzan, "Data Structures: A pseudo code approach with C", Cenage Learning, ISBN 9788131503140
3. R.G.Dromy, “How to Solve it by Computers”, Prentice Hall.

Learning Resources contd…

Web Resources:

Web links:

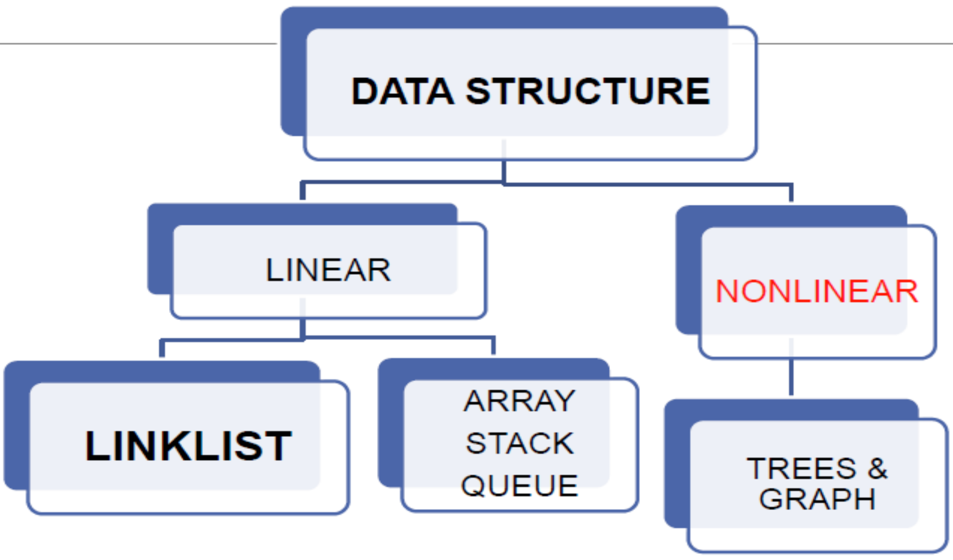
1. https://www.tutorialspoint.com/data_structures_algorithms/

MOOCs:

1. http://nptel.ac.in/courses/106102064/1

2. https://nptel.ac.in/courses/106103069/

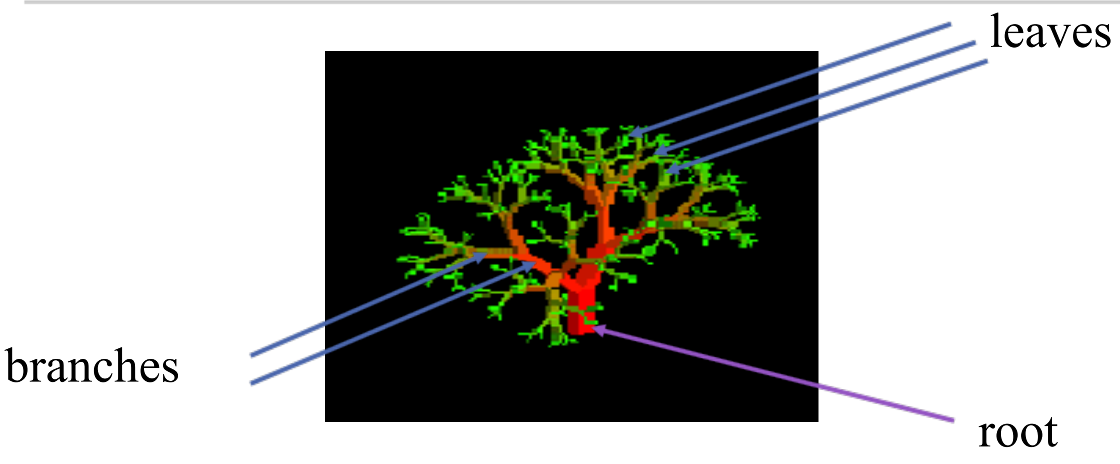
Types of Data Structures



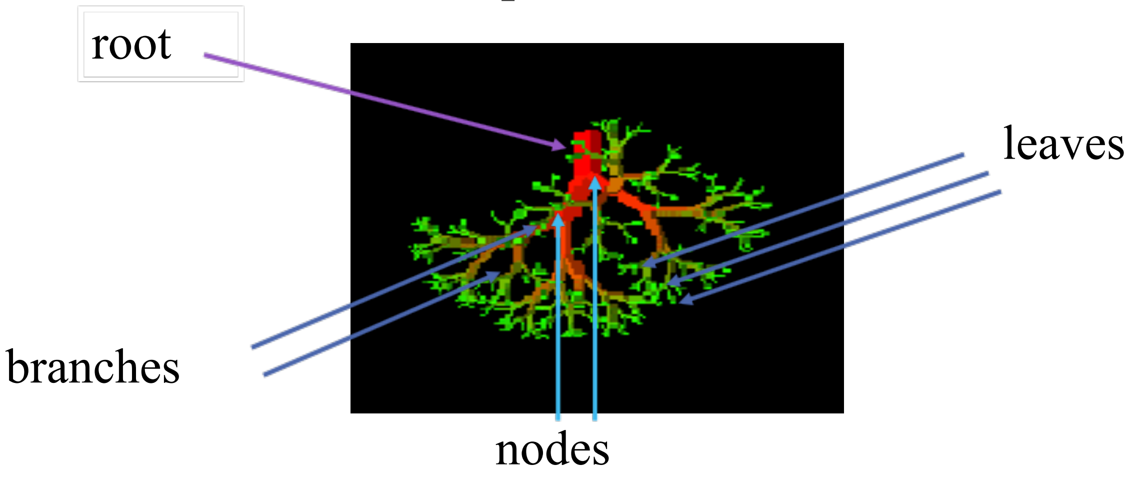
Tree

- Basic Terminology, Binary Tree- Properties
- Converting Tree to Binary Tree.
- Representation using Sequential and Linked organization .
- Binary tree creation and Traversals, Operations on binary tree.
- Binary Search Tree (BST) and its operations
- Threaded binary tree- Creation and Traversal of inorder Threaded Binary tree.
- **Case Study**- Expression tree.

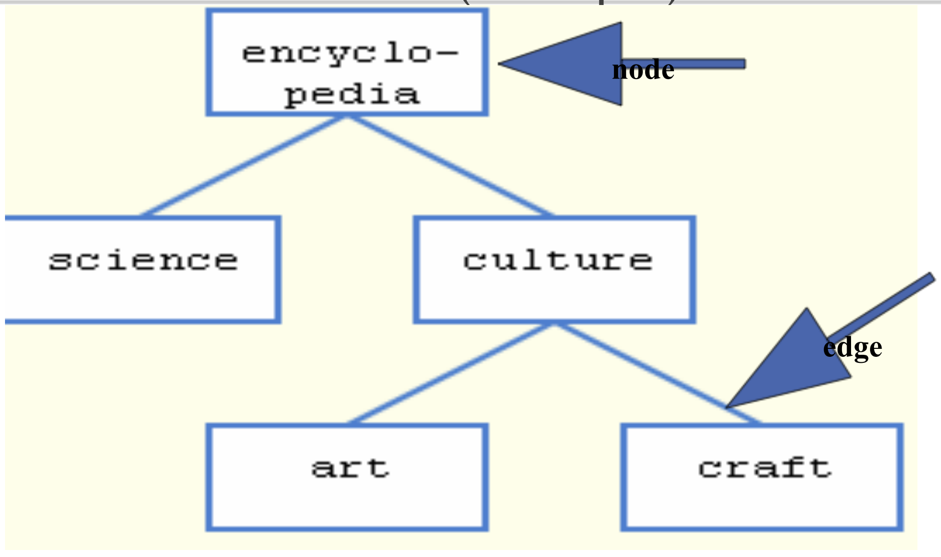
Natural environment Tree



Computer Scientist's View



Tree (example)



General tree

A tree is a finite set of one or more nodes such that:

(i)There is a specially designated node called the root;

(ii) The remaining nodes are partitioned into n >=0 disjoint sets T1, ...,Tn where each of these sets is a tree. T1, ...,Tn are called the subtrees of the root.

Sample Tree

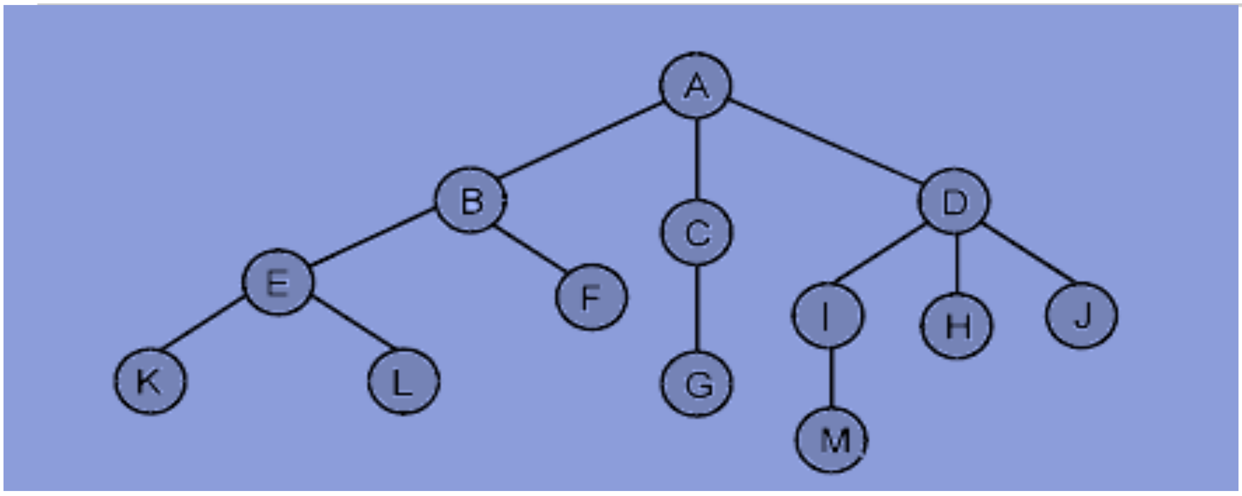


Figure 8: Sample Tree

Tree Terminology

Root: Node without parent (A)

Siblings: Nodes share the same parent

Ancestors of a node: all the nodes along the path from root to that node

Descendant of a node: child, grandchild, grand-grandchild, etc.

The height or depth of a tree is defined to be the maximum level of any node in the tree.(4)

Degree of a node: the number of subtrees(children) of a node is called degree

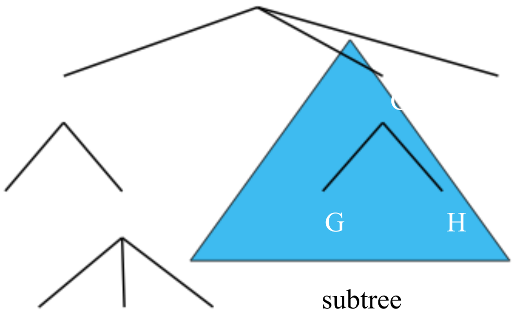
Degree of a tree: the maximum of the degree of the nodes in the tree.

Nonterminal nodes: other nodes

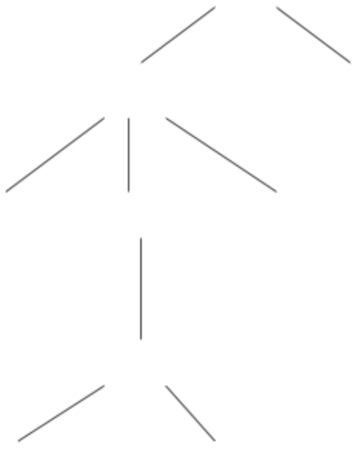
leaf or terminal node: Node that have degree zero (E, I, J, K, G, H, D)

The level of a node is defined by initially letting the root be at level one. If a node is at level l, then its children are at level l + 1.

Subtree: Tree consisting of a node and its descendants



Tree Properties



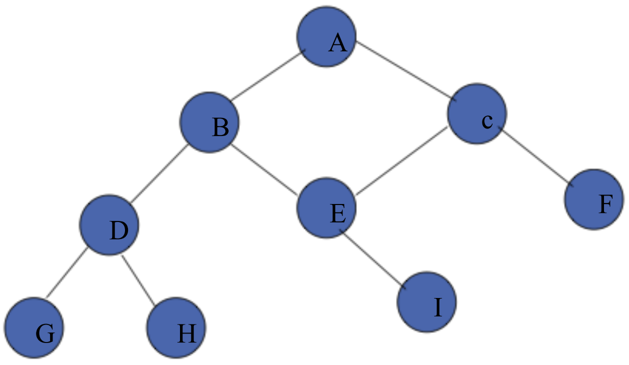
Property	Value
Number of nodes	9
Height	5
Root Node	A
Leaves	C,D,F,H,I
Interior nodes	B,E,G
Ancestors of H	A,B,E,G
Descendants of B	D,E,G,H,I,F
Siblings of E	D,F
Right subtree of A	A,C
Degree of this tree	3

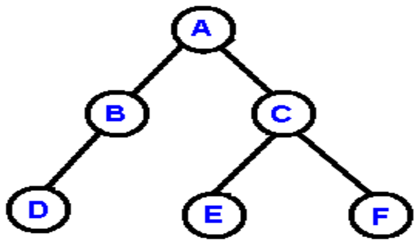
Binary Tree

- Every node in a binary tree can have at most two children.
- A binary tree is a finite set of nodes that is either empty or consists of a root and two disjoint binary trees called *the left subtree* and *the right subtree*.

Structures that are not binary trees

Binary tree





Maximum Number of Nodes in BT

The maximum number of nodes on level i of a binary tree is 2i-1, i>=1.

The maximum number of nodes in a binary tree of depth k is 2k-1, k>=1.

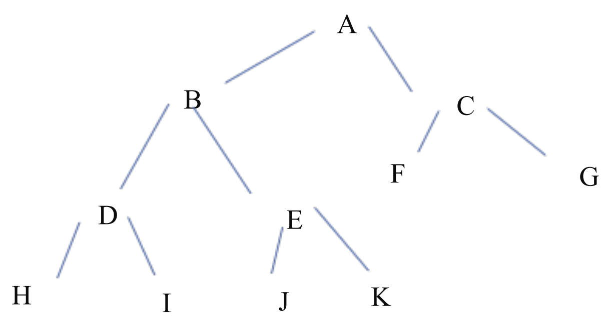
Binary Trees

- **A Full binary tree** of depth K is a binary tree of depth having 2k-1 nodes k>=0

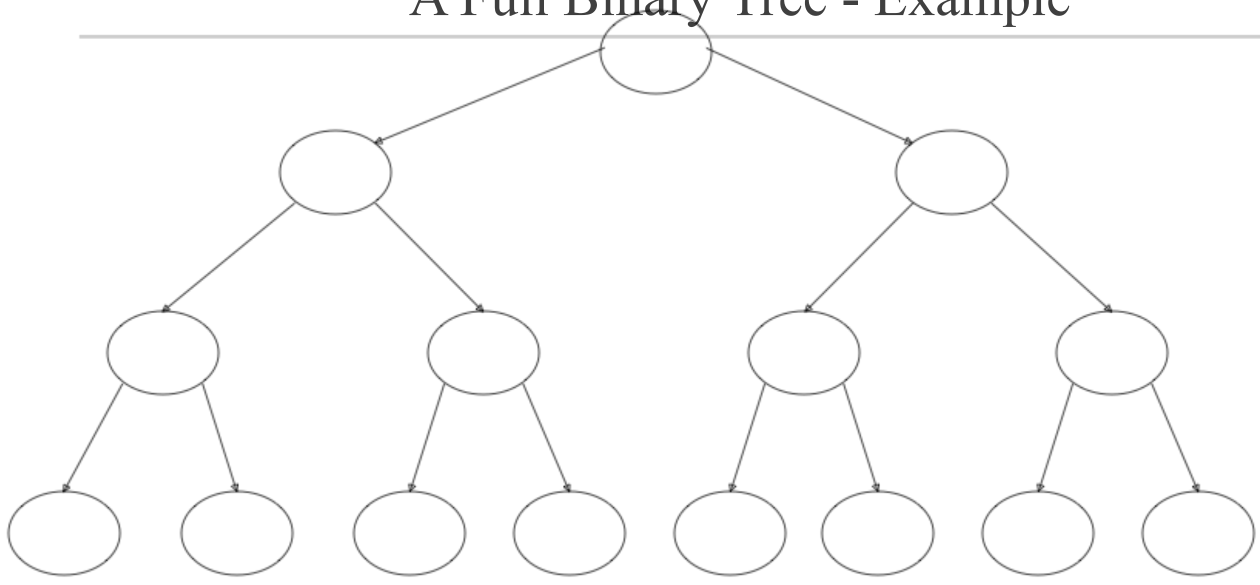
- **Complete Binary Tree**

  A binary tree T with n levels is *complete if all* levels except possibly the last are completely full, and the last level has all its nodes to the left side.

Complete Binary Trees - Example



A Full Binary Tree - Example



Complete Binary Trees

The second node of a complete binary tree is always the left child of the root...

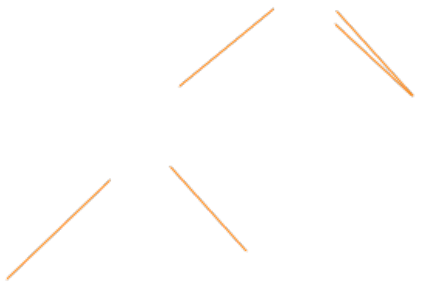
... and the third node is always the right child of the root.

Complete Binary Trees

The next nodes must always fill the next level from left to right.

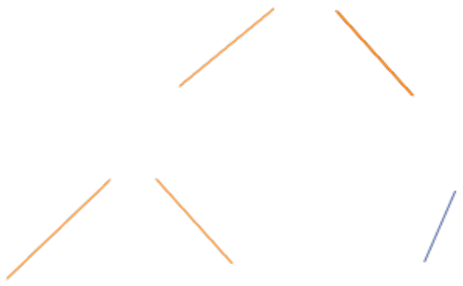
Complete Binary Trees

The next nodes must always fill the next level from left to right.



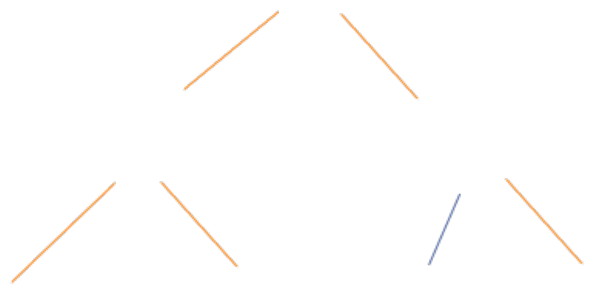
Complete Binary Trees

The next nodes must always fill the next level from left to right.



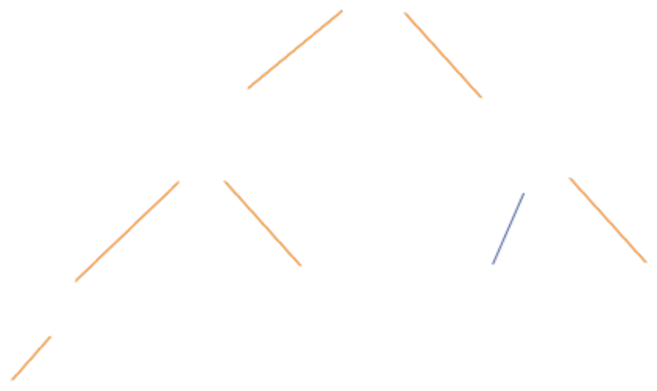
Complete Binary Trees

The next nodes must always fill the next level from left to right.



Complete Binary Trees

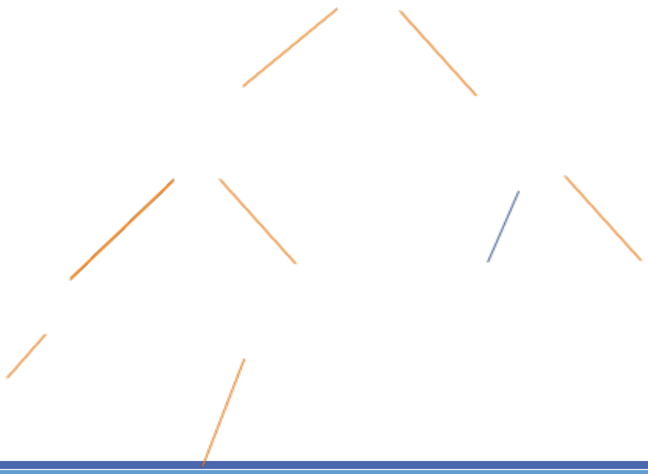
The next nodes must always fill the next level from left to right.



Complete Binary Trees

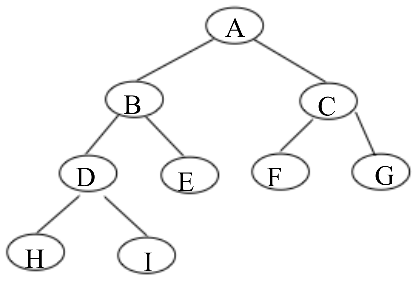
The next nodes must always fill the next level from left to right.

Is This Complete?

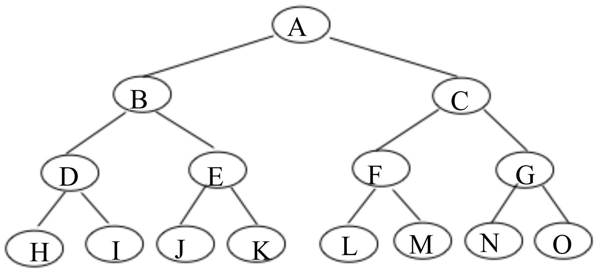


Is This Complete?

Full BT VS Complete BT

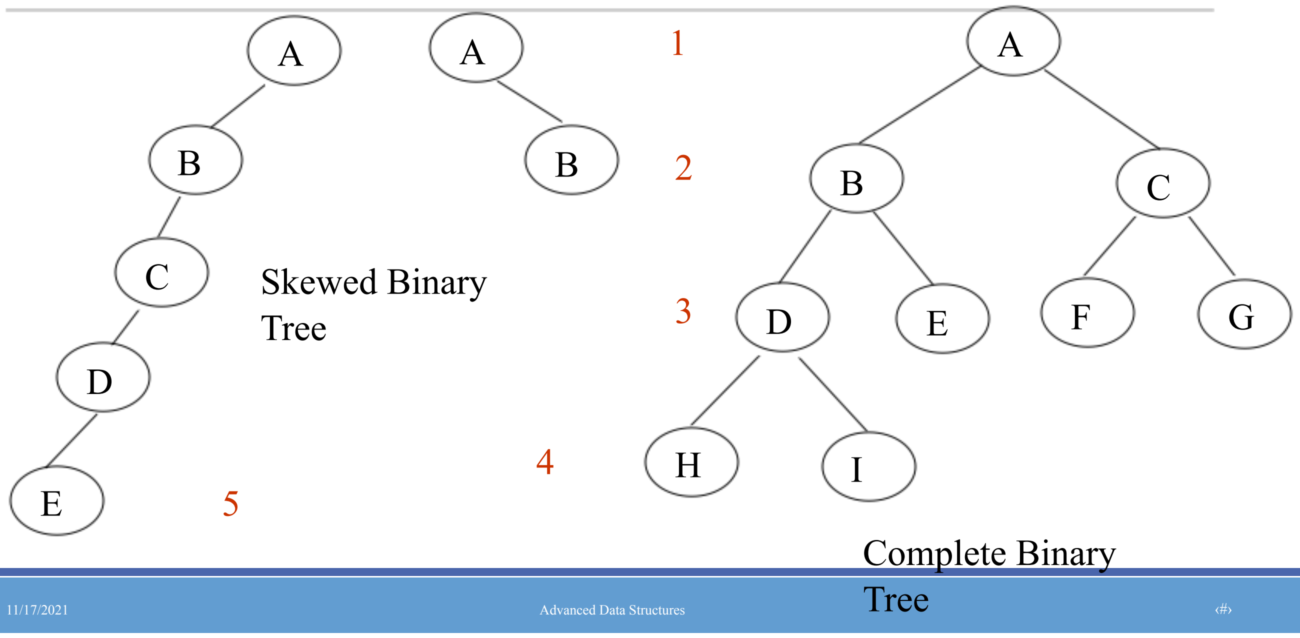


Complete binary tree



Full binary tree of depth 4

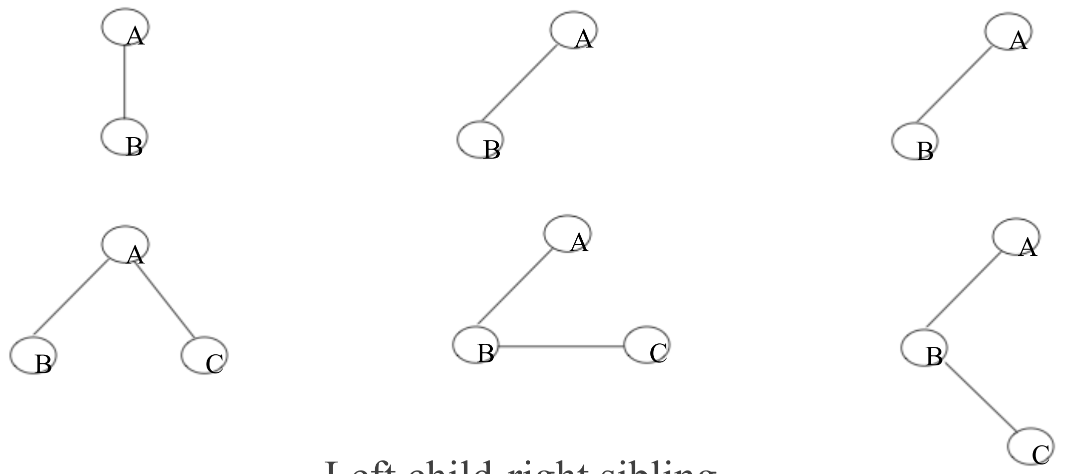
Samples of Trees



Converting tree to binary tree

- Any tree can be transformed into binary tree.
- by left child-right sibling representation
- The left subtree and the right subtree are distinguished.

Tree Representations



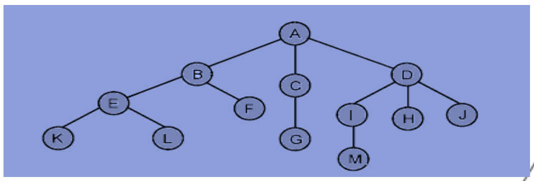
Left child-right sibling

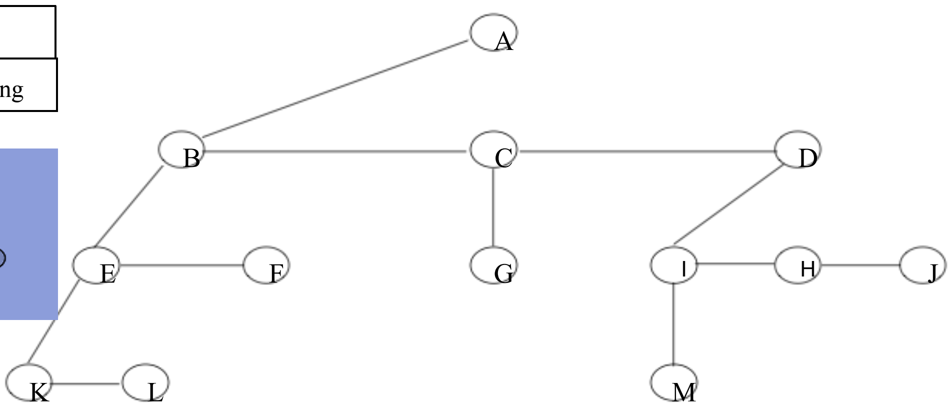
Binary tree

Representation of Trees

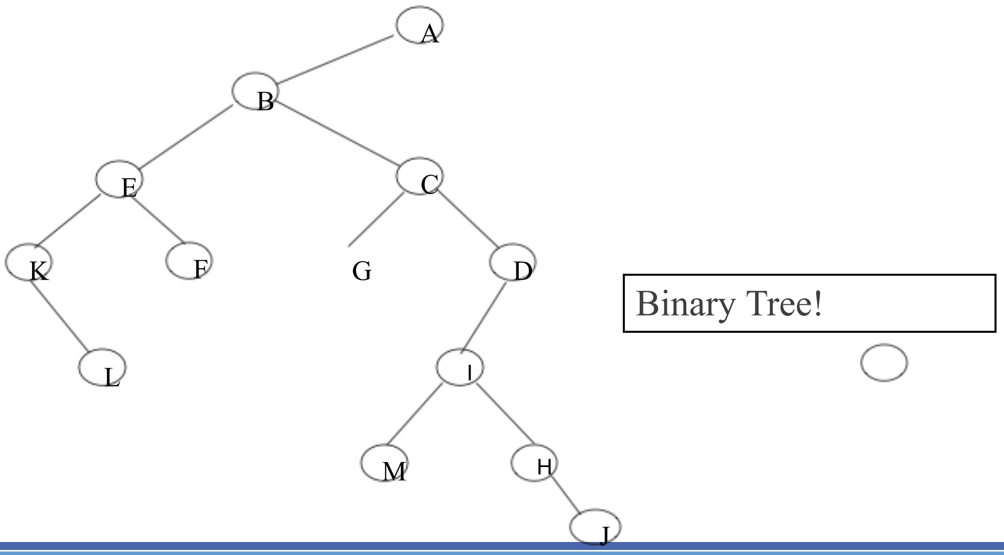
- Left Child-Right Sibling Representation
- Each node has two links (or pointers).
- Each node only has one leftmost child and one closest sibling.

data	
left child	right sibling



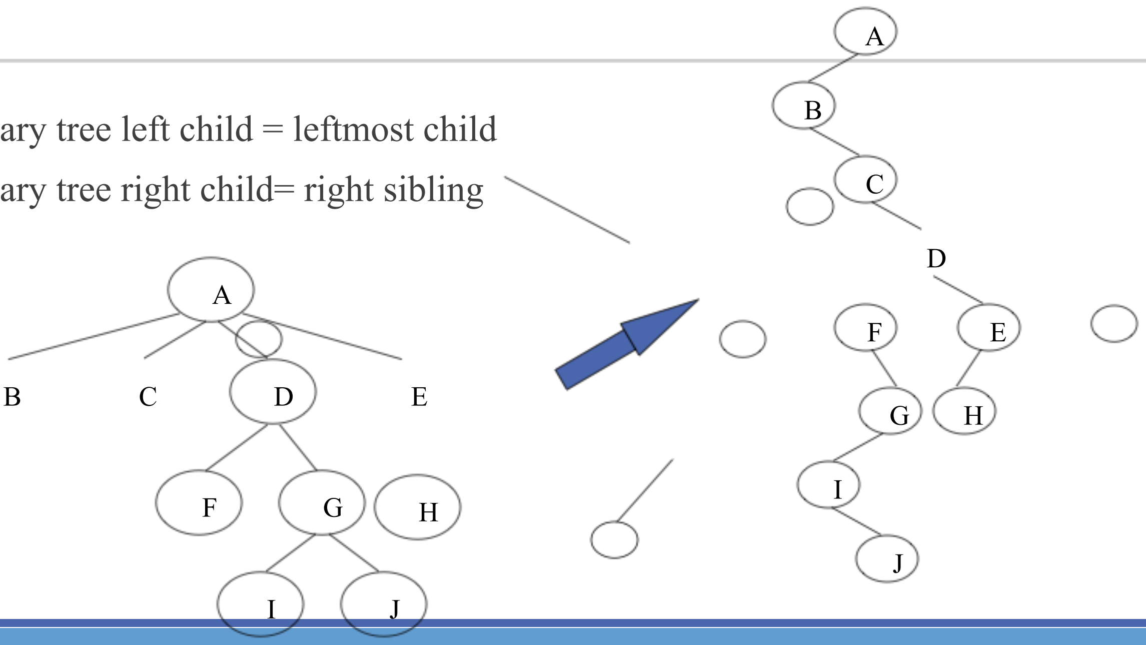


Degree Two Tree Representation

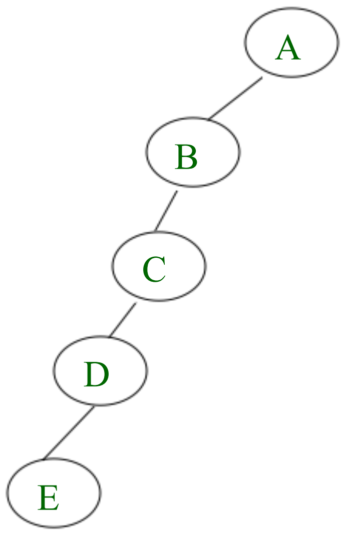


Converting to a Binary Tree

- Binary tree left child = leftmost child
- Binary tree right child= right sibling

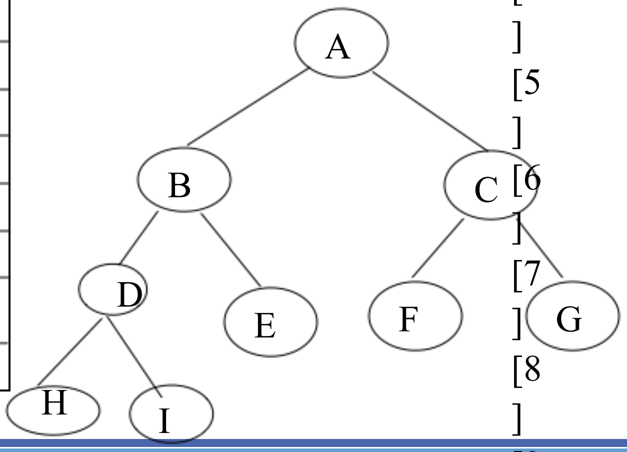


Sequential Representation



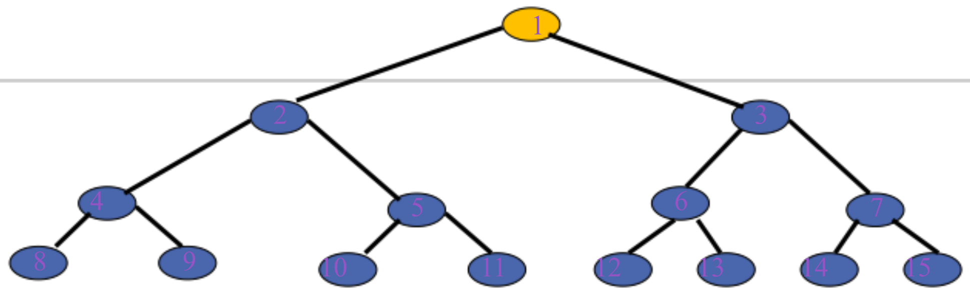
[1]	A
[2]	B
[3]	-
[4]	-
[5]	C
[6]	-
[7]	-
[8]	-
[9]	-
.	-
[16]	-
	D
	-
	-

(1) waste space
(2) insertion/deletion problem



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

Node Number Properties



Parent of node i is node i/2

- But node 1 is the root and has no parent

Left child of node i is node 2i if 2i is <=n

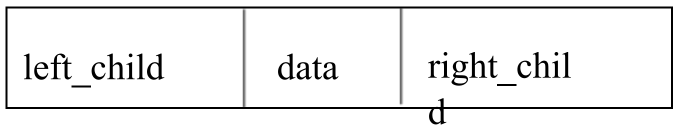
- But if 2i > n, node i has no left child

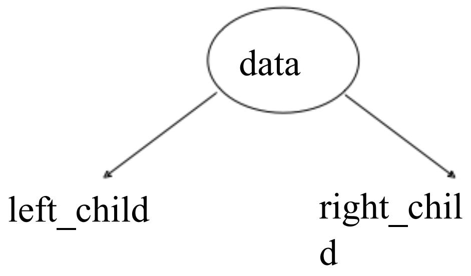
Right child of node i is node 2i+1 if 2i+1 is <=n

- But if 2i+1 > n, node i has no right child

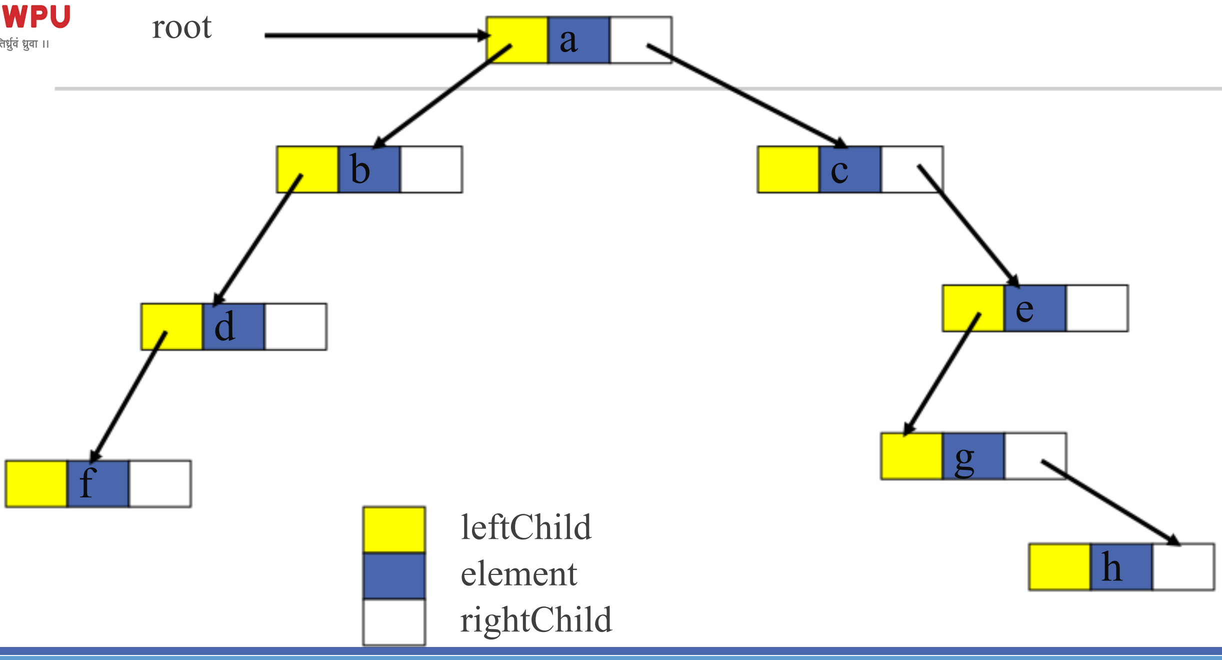
Linked Representation

```
class node{
  int data;
  node *lchild;
  node *rchild;
};
```





Linked Representation Example



Binary Tree Creation

```
class treenode
{
   char data[10];
   treenode *left;
   treenode *right;
   friend class tree;
}
class tree
{
   treenode *root;
   public:
     tree();
      void create_r();
      void create_r(treenode *);
 }
```

```
Algorithm create_r()     //Driver for creation
{
   Allocate memory for root and accept data;
    create_r(root);
}
```

```
int main()
{
  tree bt;
  bt.create_r();
}
```

```
tree::tree()      //constructor
{
  root=NULL;
}
```

Algorithm create_r(treenode * temp)   //workhorse for creation

```
{
  Accept choice whether data is added to left  of temp->data;
   if ch='y'
  {
   Allocate a memory for curr and accept data;
   temp->left=curr;
   create_r(curr);
  }

  Accept choice whether data is added to right  of  temp->data;
   if ch='y'
  {
   Allocate a memory for curr and accept data;
   temp->right=curr;
   create_r(curr);
  }
}
```

```
Algorithm create_nr()
{
  if root=NULL
  {
    Allocate memory for root and accept the data;      }
do
  {
    temp=root;
    flag=0;
    allocate memory for curr and accept data;
    while(flag==0)
    {
      Accept choice to add node(left or right);
      if  ch='l'
      {
          if temp->left=NULL
          {    temp->left=curr;
               flag=1;
          }
               temp=temp->left;
      }
      else {
           if ch='r'
           {
                if temp->right=NULL
                {
                 temp->right=curr;
                 flag=1;
                 }
                temp=temp->right;
           }
         }          //else end
    }//while flag
    Accept choice for continuation;
  }  // do while end
}// algo end
```

Binary Tree Traversals

- Let L, V/D and R stand for moving left, visiting the node, and moving right.
- There are six possible combinations of traversal
  – LVR, LRV, VLR, VRL, RVL, RLV
- Adopt convention that we traverse left before right, only 3 traversals remain
  – LVR, LRV, VLR
  – inorder, postorder, preorder

Binary Tree Traversals

- A traversal is where each node in a tree is visited once
- There are two very common traversals
  - Breadth First
  - Depth First

Breadth First

- In a breadth first traversal all of the nodes on a given level are visited and then all of the nodes on the next level are visited.
- Usually in a left to right fashion
- This is implemented with a queue