Introduction to Tree

Unit-3 Lecture-8

Dr. Dillip Rout, Assistant Professor, Dept. of Computer Science and Engineering

Outline

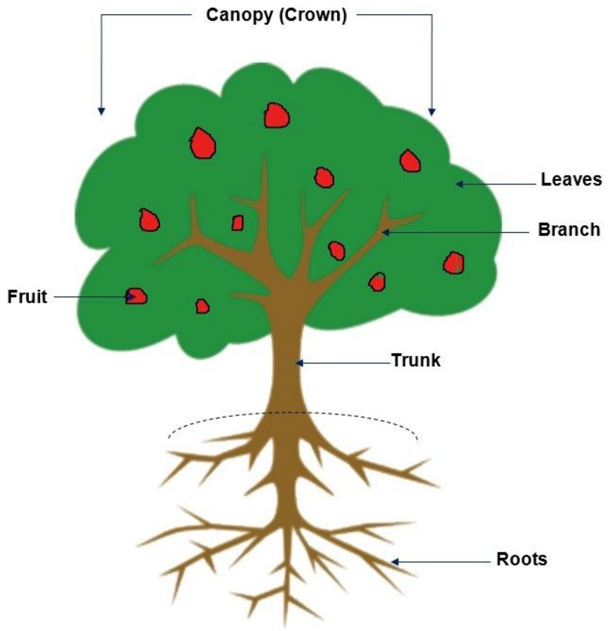
- Fundamentals of Tree
  - Binary Tree, Strictly Binary Tree
  - Complete Binary Tree
- Representation on Memory
- Operations on Tree
- Homework
- Conclusions

Fundamentals of Tree

Overview

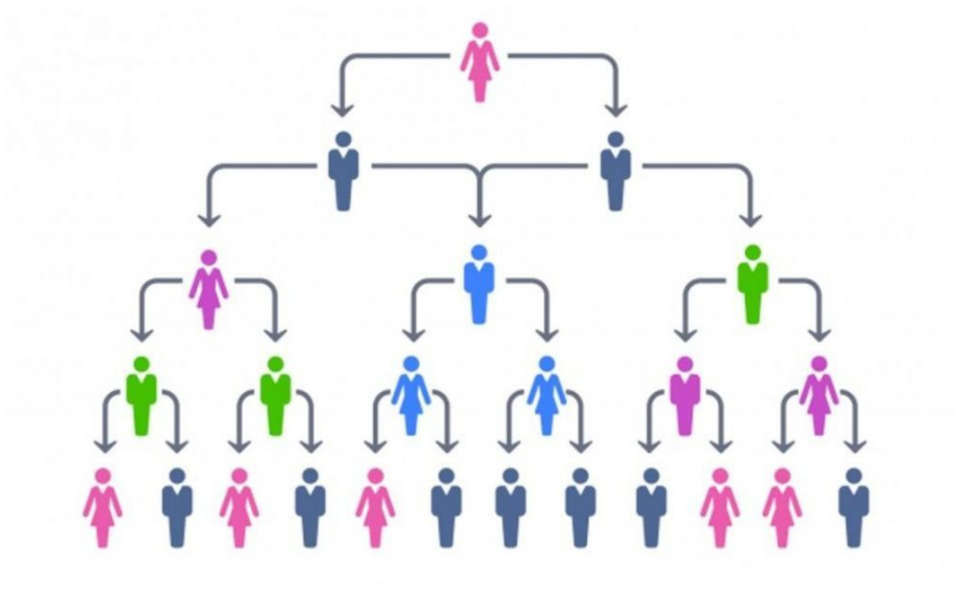
- Tree is a non-linear data structure.
  - Represents hierarchical data structure
  - Traversing up and down, as well as moving side wise or level wise.
- Consists of nodes and edges
  - Each element representing data is called node in a tree
  - Top node is called the root node and it has no parent node
  - Nodes with no children nodes are called leaf node
  - Children node of same parents are called siblings
- Node A, B are called ancestors/predecessors of E if A, B are parents, grandparents of E.
- E is called the successor of A, B.

Real Tree in Real-life

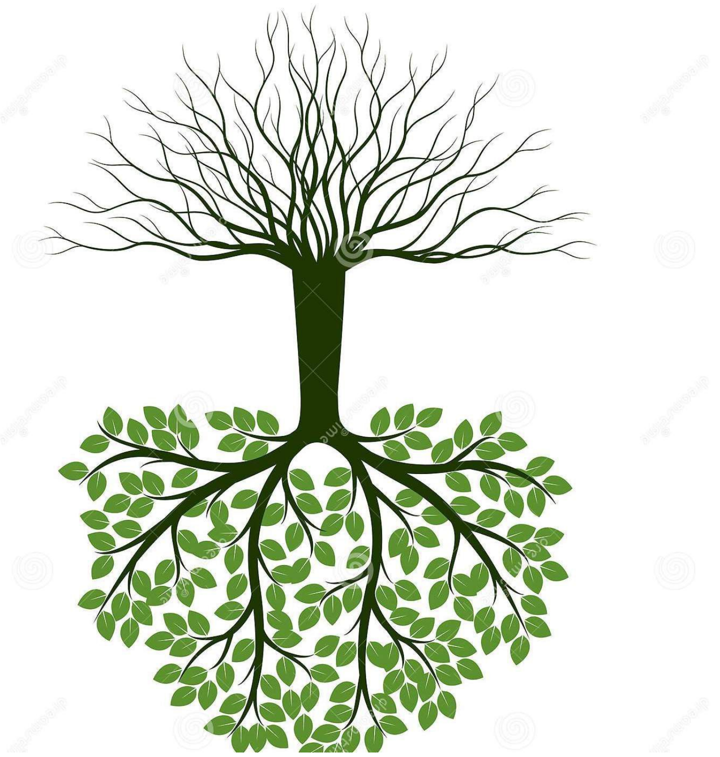


Parts of Tree (Simplified)

Data Representation using Tree in Real-life

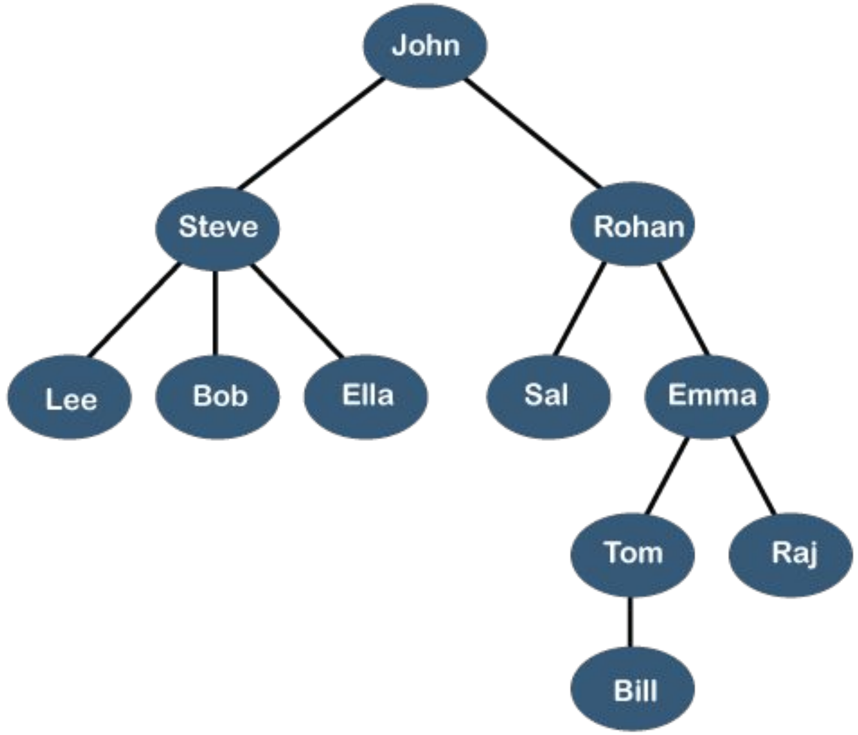


Family Tree looks like an inverted tree

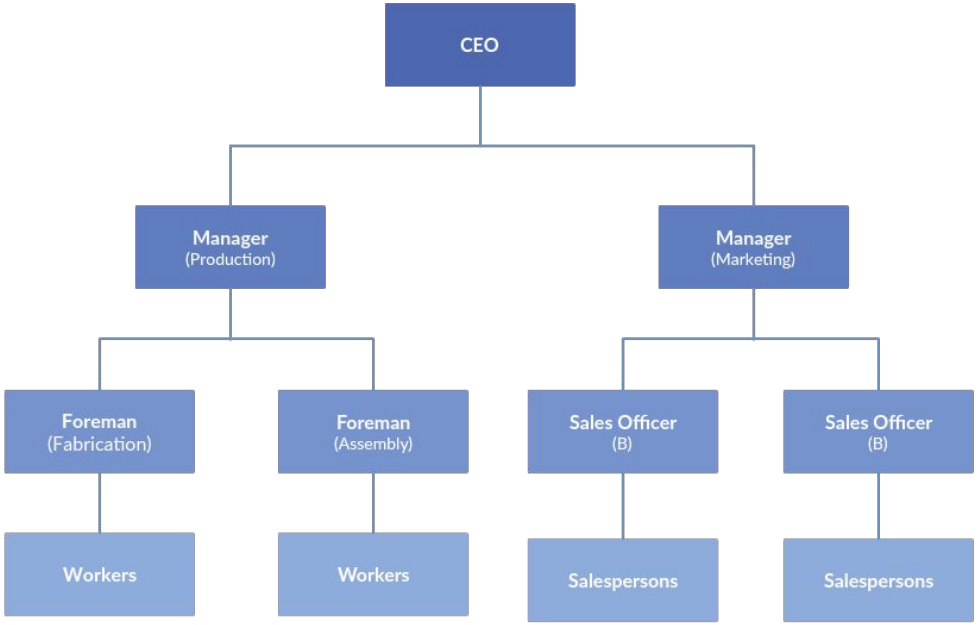


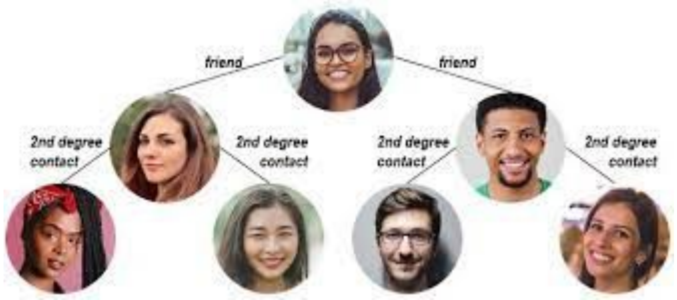
Data Representation using Tree in Real-life

Family Tree / Ancestor Hierarchy



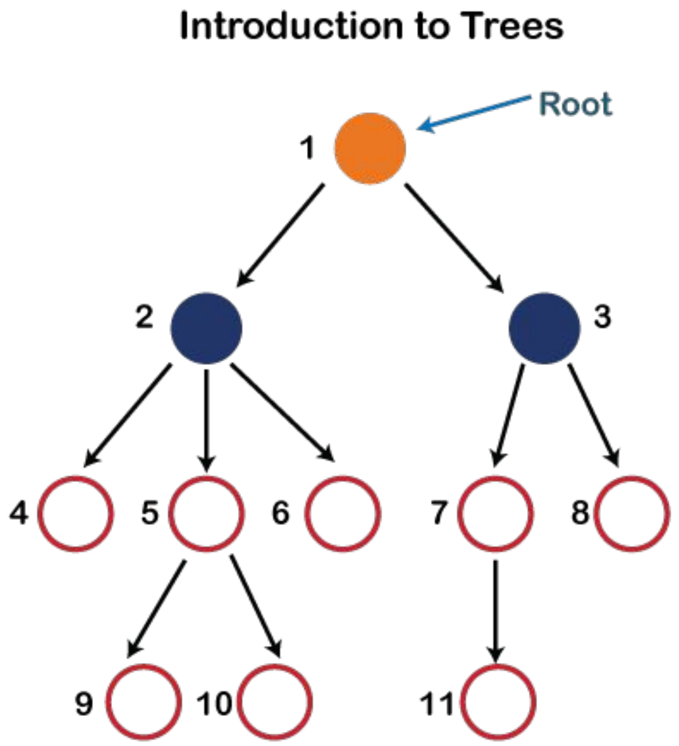
Data Representation using Tree in Real-life





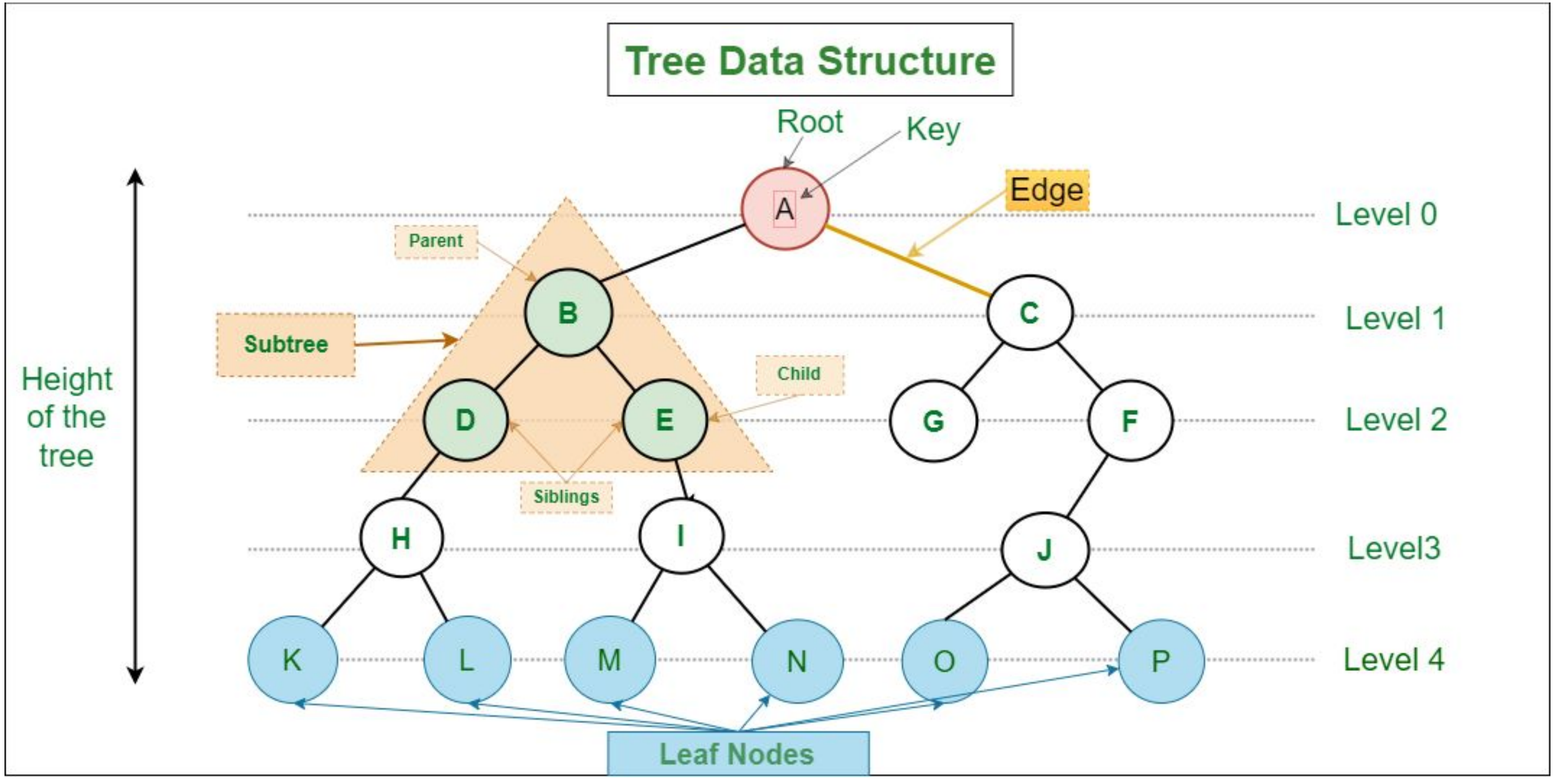
- Organizational Hierarchy
- Friends Connections

Tree (Generalized) in Computers



- A simplified tree is given
- There are several nodes and edges (branches)
- A hierarchy is maintained

Attributes of Tree



Attributes of Tree...

- Each Node represents an information
  - Value/Key: either single or multiple
  - Reference/Address: Children information
- Each Edge represents a relation (information)
  - Edges can only be from one level to another level
- Depth of node is the number of edges from root node to that node (example for A to F)
- Height of the tree is the number of edges from root node to the deepest node.
- Set of all nodes at a given depth is called level {B, C are in level 1}

Attributes of Tree...

- Root:
  - A special node at level 0 which is the starting node.
- Relations:
  - Parent: Any node having children
  - Child: Node originating from the parent. Parent and Child differ by a level.
  - Siblings: Children of the same parent node
- Leaf Node:
  - A special node which has no children
- SubTree:
  - A group of Parent and Children combination with hierarchy.

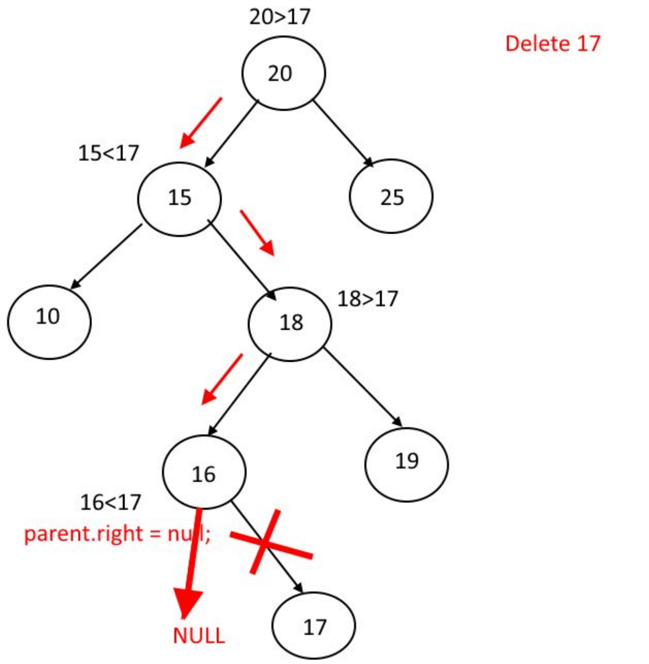
Operations in Tree

Native

- Insertion (R, x)
- Deletion (R, x)
- Search (R, x)
- Traversal (R)
  - Pre-order
  - In-order
  - Post-order

Auxiliary

- Extract Root
- Find Leaves
- Count



Case 1 : Node to be deleted is a leaf node (No Children).

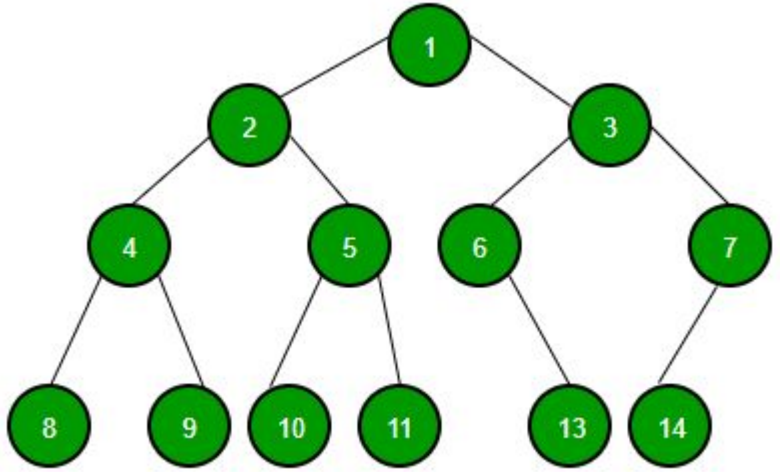
Binary Tree

A tree whose elements have at most 2 children is called a binary tree.

Typically the children nodes names are referred as the left and right child Since each element in a binary tree can have only 2 children.

How many edges are there in this tree?

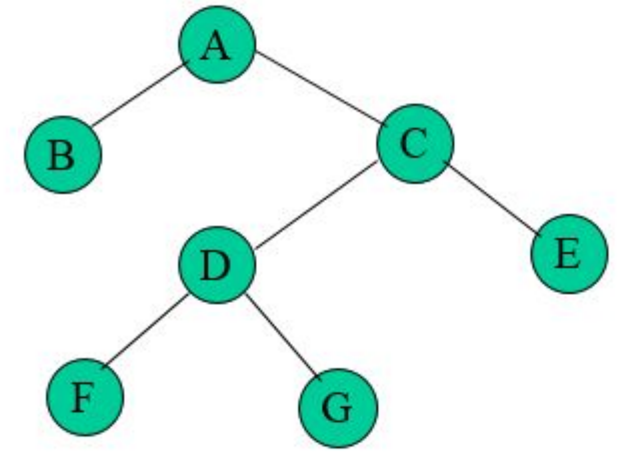
If there are n nodes in a binary tree then how many edges are there?



Strictly Binary Tree

- Each node must contain 0 or 2 children other than the leaf nodes.
- The nodes having 2 children are called as internal nodes.
- If a node is not an internal node then what type of node is this?
- n internal nodes can have (n+1) possible leaf nodes

n nodes can produce how many number of edges?



Complete Binary Tree

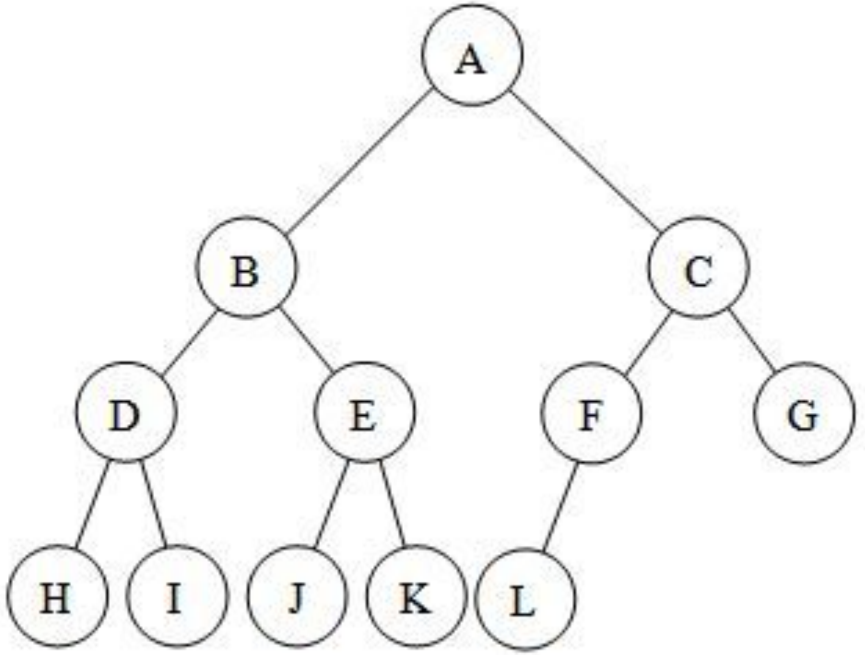
- A Binary Tree is a Complete Binary Tree if all the levels are completely filled except possibly the last level, and the last level has all keys as left as possible.
- The nodes are going to be inserted every time from left to right manner.
- A half filled is sometimes refers to as Almost Complete Binary Tree.

What is the height of this tree?

How many nodes are there? How many edges are there?

How many maximum nodes it can have without increasing the height?

How many maximum edges it can have without increasing the height?

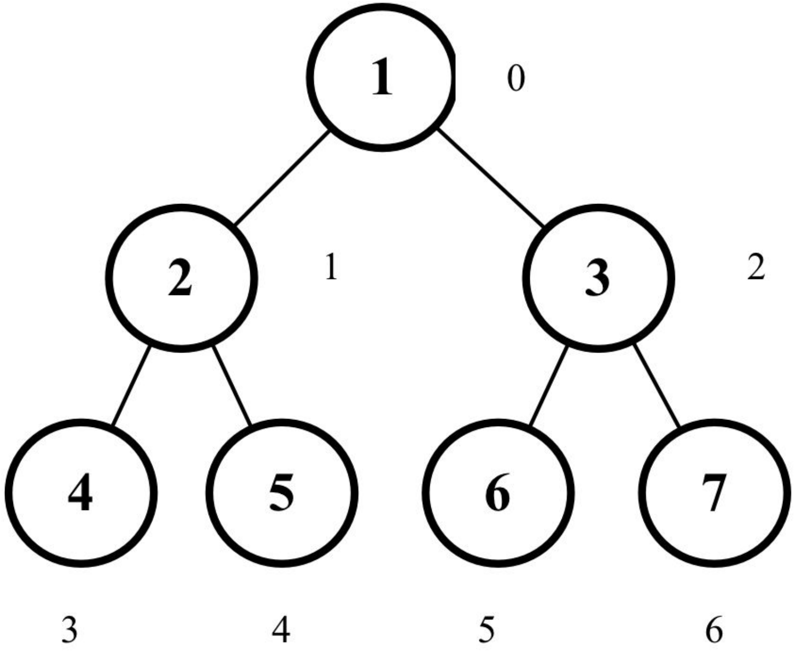


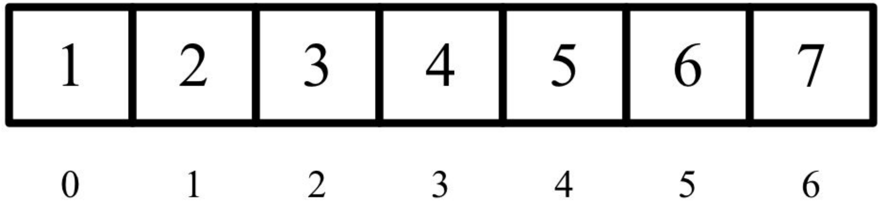
Usages of Tree as Data Structure

- Store hierarchical data: like folder structure, organization structure, XML/HTML data.
- Binary Search Tree is a tree that allows fast search, insert, delete on a sorted data. It also allows finding closest item
- Heap: is a tree data structure which is implemented using arrays and used to implement priority queues.
- B-Tree and B+ Tree: They are used to implement indexing in databases.
- Syntax Tree: Scanning, parsing , generation of code and evaluation of arithmetic expressions in Compiler design.
- K-D Tree: A space partitioning tree used to organize points in K dimensional space.
- Trie: Used to implement dictionaries with prefix lookup.

Representation on Memory

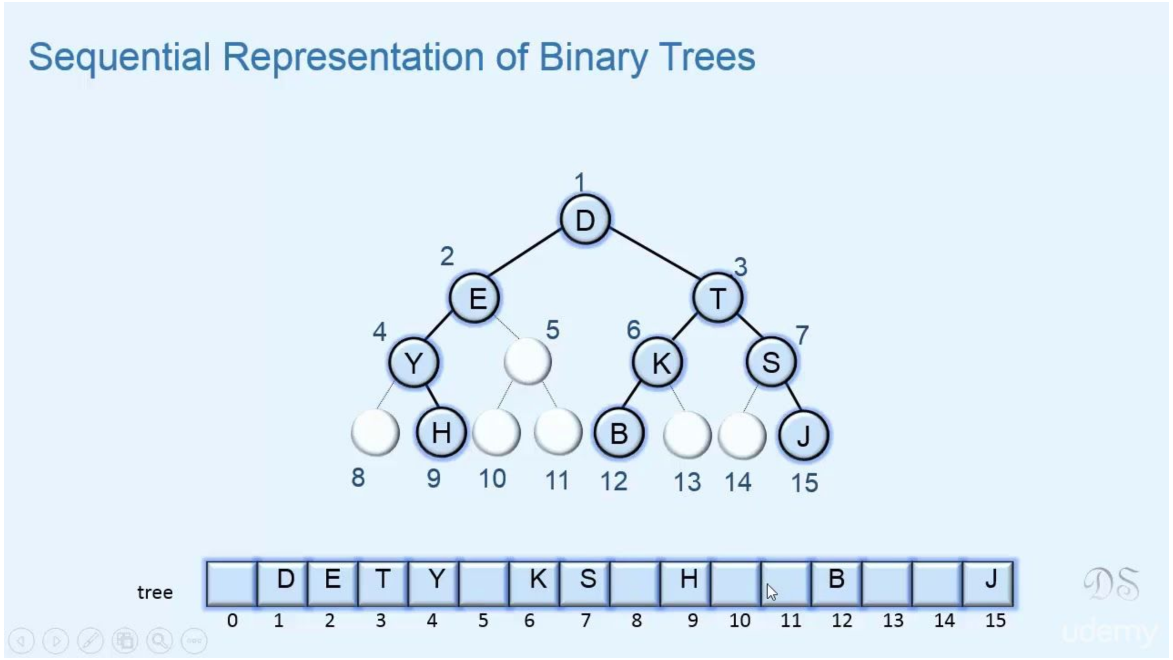
Representation using Array





What is the relation of the index of any child with the index of its parent?

Representation using Array...



This is not a complete binary tree.

What is the disadvantage here in this representation?

Representation using Linked List

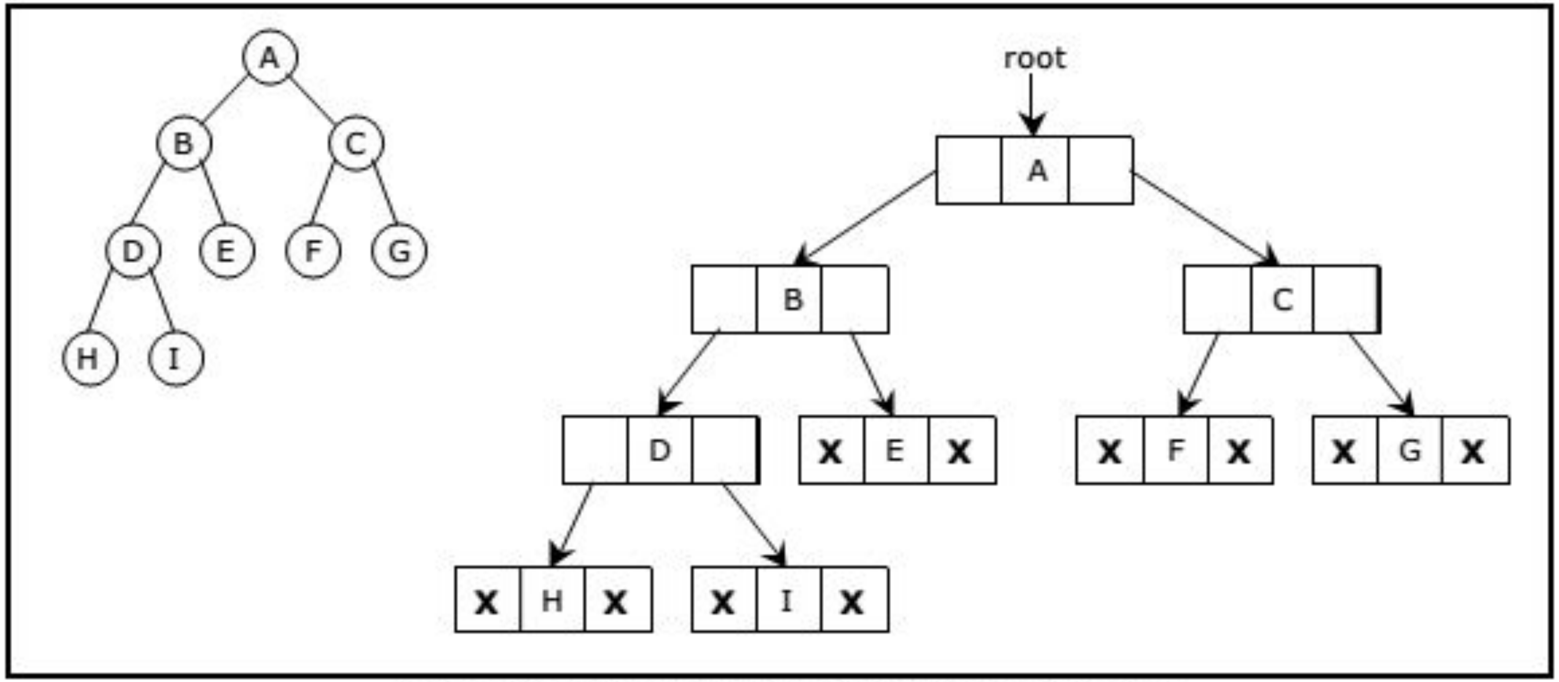
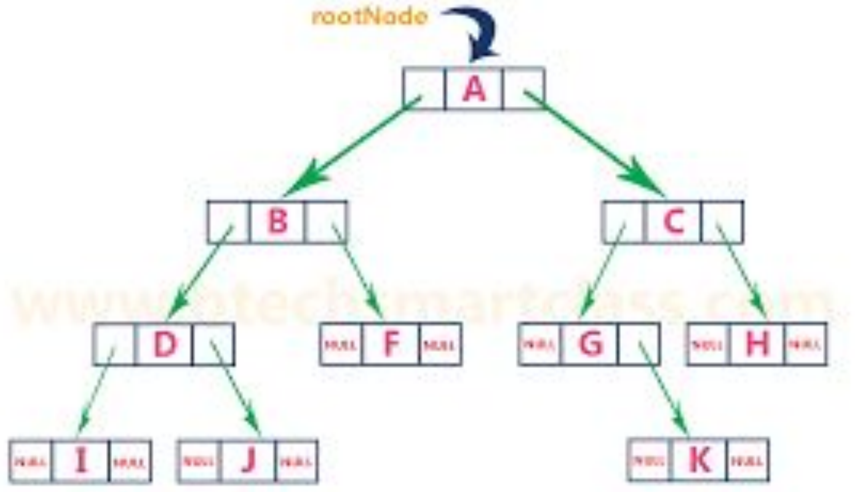


Figure 5.2.7. Linked representation for the binary tree

Representation using Linked List

- Each node is represented as a node in a doubly linked list.
    - Information/Key/Value
    - Left pointer: points to the left child
    - Right pointer: points to the right child

Is there any advantage with such representations?



Knowledge Test

1. Which representation of a Binary Tree is efficient?
2. Is it specific to some context or scenario?

Example Problem-1

Consider a binary tree that has 200 leaf nodes. Then, the number of nodes in that have exactly two children are ________.

A. 200
B. 199
C. 198
D. Cannot Say

Operations of Binary Tree

- Search
- Insertion
- Deletion

Example of Operations

Insertion cases

- Root
- Leaf
- Internal

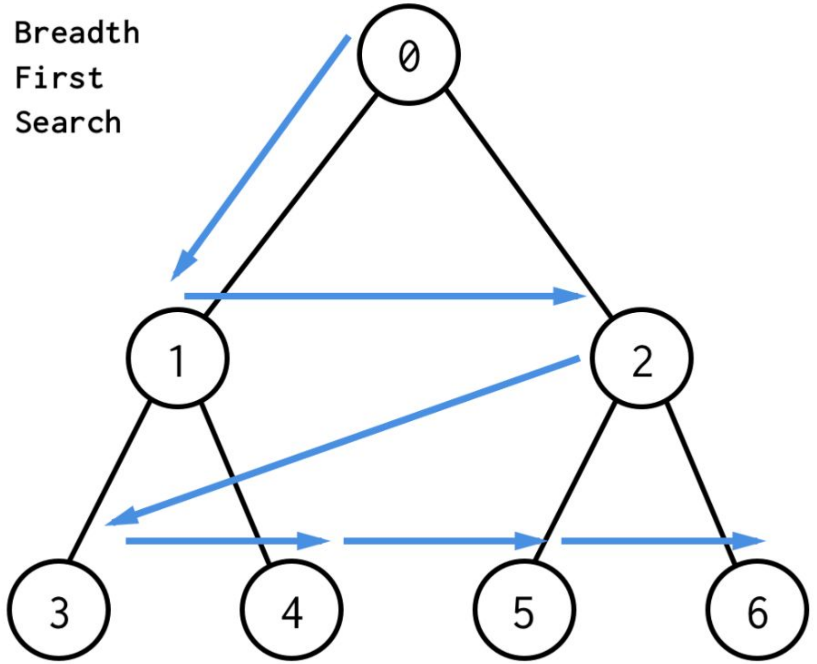
Deletion cases

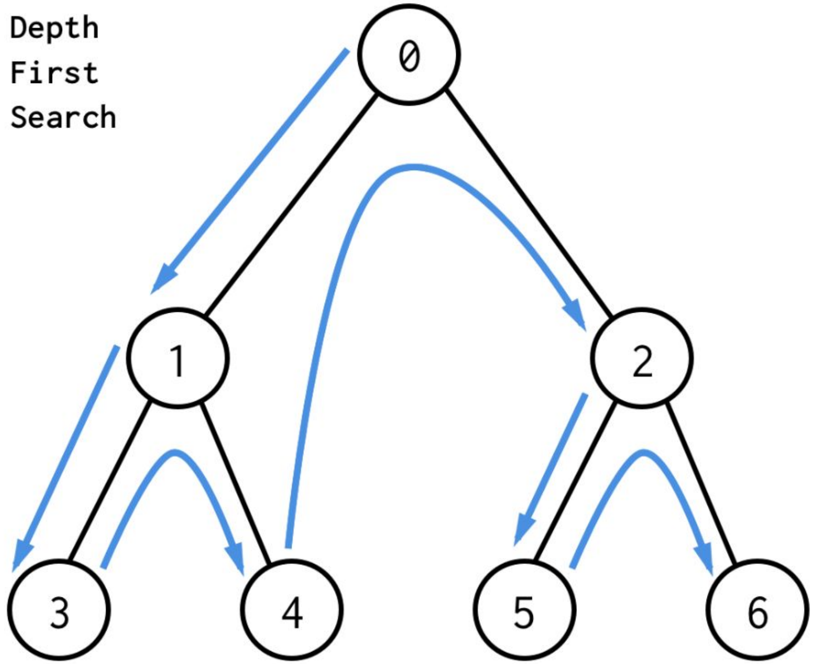
- Root
- Leaf
- Internal

Search cases

- Node
- Node’s right successor
- Node’s left successor
- Node’s predecessor

Searching in Tree





Breadth First Search (BFS)

Algorithm BFS(root, x: element to be searched)

```
1.  If root == NULL then return
2.  Let Q is a queue
3.  Q.enqueue(root)
4.  While (Q != ϕ) do
5.      node = Q.dequeue()
6.      If  node.data == x then
7.          return node
8.      Q.enqueue(node.left)
9.      Q.enqueue(node.right)
10. Print “No elements found”
11. Exit
```

The order in which the elements are dequeued is level ordered

Is it recursive?

What is a node here; pointer or data?

What is node.left; pointer or data?

What kind of traversal algorithms be defined using BFS?

Depth First Search (BFS)

Algorithm DFS(root, x: element to be searched)

```
1.   If root == NULL then return
2.   Let S is a stack
3.   S.push(root)
4.   While (S != $\phi$) do
5.       node = S.pop()
6.       If  node.data == x then
7.           return node
8.       S.push(node.right)
9.       S.push(node.left)
10.  Print "No elements found"
11.  Exit
```

Is it recursive?

Which lines can be modified to make it more efficient?

What kind of traversal algorithms be defined using DFS?

BFS and DFS Difference

- Extra Space required for Level Order Traversal is O(w) where w is maximum width of Binary Tree. In level order traversal, queue one by one stores nodes of different level.
- Extra Space required for Depth First Traversals is O(h) where h is maximum height of Binary Tree. In Depth First Traversals, stack (or function call stack) stores all ancestors of a node.
- Depth First Traversals are typically recursive and recursive code requires function call overheads.
- The most important points is, BFS starts visiting nodes from root while DFS starts visiting nodes from leaves. So if our problem is to search something that is more likely to closer to root, we would prefer BFS. And if the target node is close to a leaf, we would prefer DFS.

Insertion Example

I. Insert 5 as root
II. Insert 4 and 7 as left and right children of 5
III. Insert 2 as left child of 4
IV. Insert 3 as right child of 2
V. Insert 6 and 9 as left and right children of 7
VI. Insert 13 as right child of 9
VII. Insert 11 as left child of 13
VIII. Insert 10 and 12 as left and right children of 11
IX. Insert 8 as left child of 9

Which cases are insertion at root?

Which cases are insertion at leaf?

Which cases are insertion at internal?

Deletion Example

I. Delete 1
II. Delete right child of 2
III. Delete 13
IV. Delete 7
V. Delete left child of 11
VI. Delete 6
VII. Delete 12
VIII. Delete 5

Convention:

- Right child will take parent's position if the latter is deleted
- It will be done recurrently until stability is achieved

Which cases are deletion at root?

Which cases are deletion at leaf?

Which cases are deletion at internal?

Which is simpler Insertion or Deletion?

Classwork: Insertion & Deletion

I. Insert E as root
II. Insert D and G as left and right children of E
III. Insert B as left child of D
IV. Insert C as right child of B
V. Insert F and I as left and right children of G
VI. Insert M as right child of I
VII. Insert K as left child of M
VIII. Insert J and L as left and right children of K
IX. Insert H as left child of I

Which cases are insertion at root?

Which cases are insertion at leaf?

Which cases are insertion at internal?

Classwork: Deletion

I. Delete A
II. Delete right child of B
III. Delete M
IV. Delete G
V. Delete left child of K
VI. Delete 6
VII. Delete L
VIII. Delete E

Convention:

- Right child will take parent’s position if the latter is deleted
- It will be done recurrently until stability is achieved

Which cases are deletion at root?

Which cases are deletion at leaf?

Which cases are deletion at internal?

Example Problem-1

Suppose there is a complete binary tree with 97 nodes in it.

1. What is the height of the tree?
   - ○ 10
   - ○ 20
   - ○ 8
   - ○ None of the above
2. How many leaf nodes are there?
3. How many nodes are there having only one child?

Example Problem-2

Suppose there is a binary tree with 97 nodes in it.

1. What is the height of the tree?
2. What is the maximum possible height of the tree?
3. What is the minimum possible height of the tree?

Homework

Homework

- Discuss the similarities and differences among Tree, Binary Tree, Full Binary Tree and Complete Binary Tree. Show some examples.
- Compare between insertion and deletion operation in a Tree. Show the difference using appropriate examples.
- Construct a complete binary tree taking number from 1 to 50 in sequence. Delete the nodes which are having keys as non-prime numbers. Now measure the height of the newly formed binary tree. Is it still a complete binary tree? What is the longest path from the root to any leaf?
- Differentiate between BFS and DFS. Demonstrate the difference with examples. Metrics for stating the differences are: Data Structure, Definition, Technique/Strategy, Concept, Approach/Principle, Suitability, Time Complexity, Visiting of Siblings/ Children, Removal of Visited Nodes, Backtracking, Applications, Memory, Optimality, Space complexity, Speed, Tapping in loops (Safety).
- Which traversal should be used to print leaves of Binary Tree efficiently?
- Which traversal should be used to print nodes at k'th level where k is much less than total number of levels?

Conclusions