មហាវិទ្យាល័យវិស្វកម្ម
FACULTY OF ENGINEERING

# Data Structure & Algorithm II

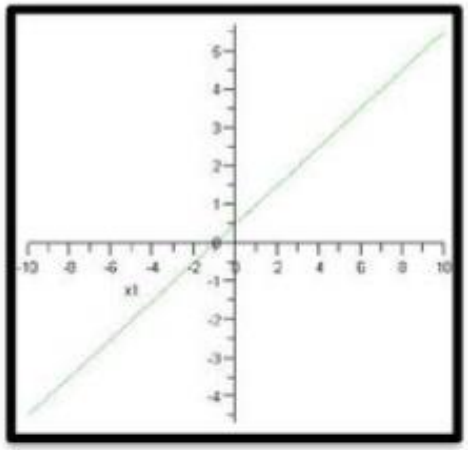
## Lecture 3
## Non-Linear Data Structure

Chhoeum Vantha, Ph.D.
Telecom & Electronic Engineering

# Content

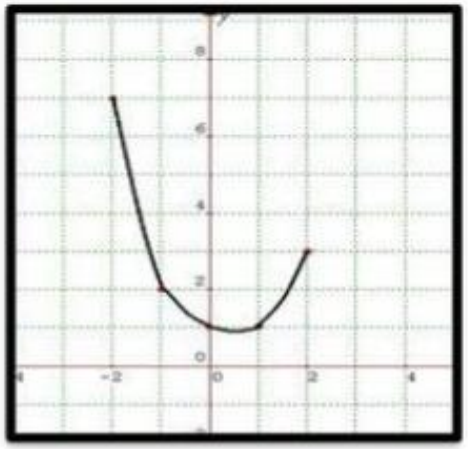
- Graphs
- Tree
- Hash Table

# Data Structure



Linear – Data is in sequential

Non Linear – Data is not in sequential



# Data Structure

**Linear – In Sequence**

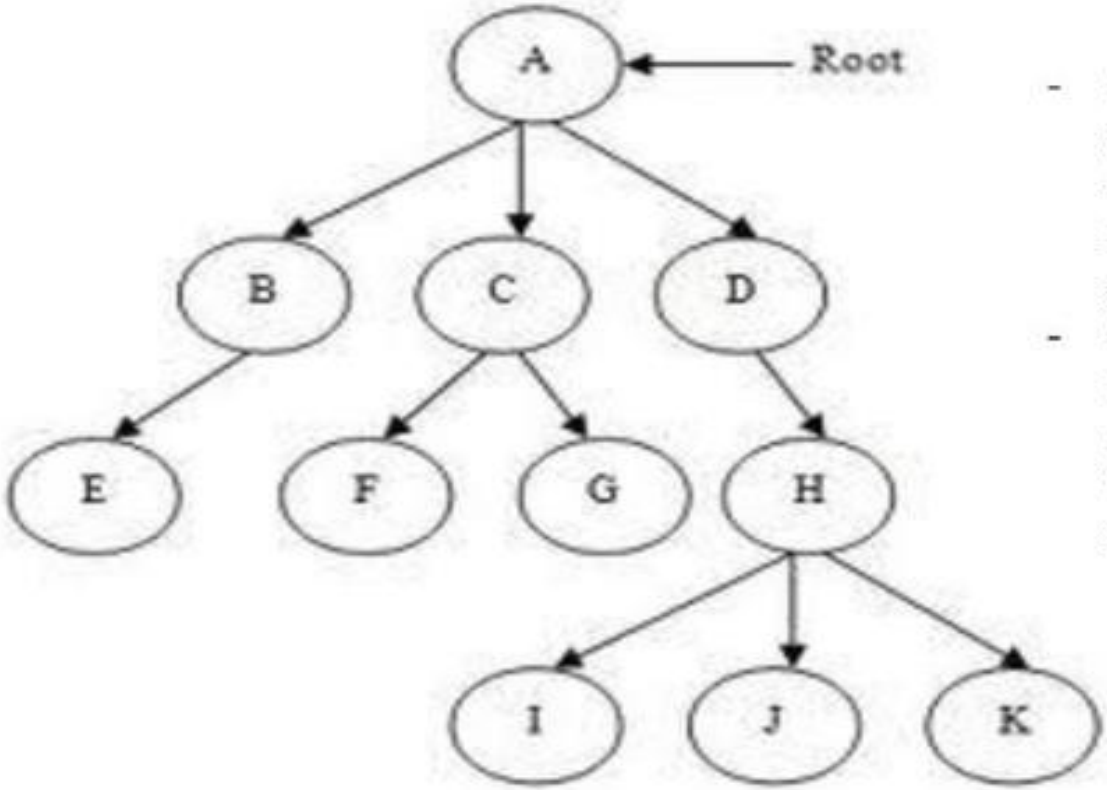
- Insertion/Deletion is possible in linear(sequential) fashion
- Example:
  - Array
  - Stacks
  - Queues
  - Linked List

**Non Linear – Not in sequence**

- Insertion/Deletion is not possible
- Example:
  - Graphs
  - Trees
  - Hash Tree

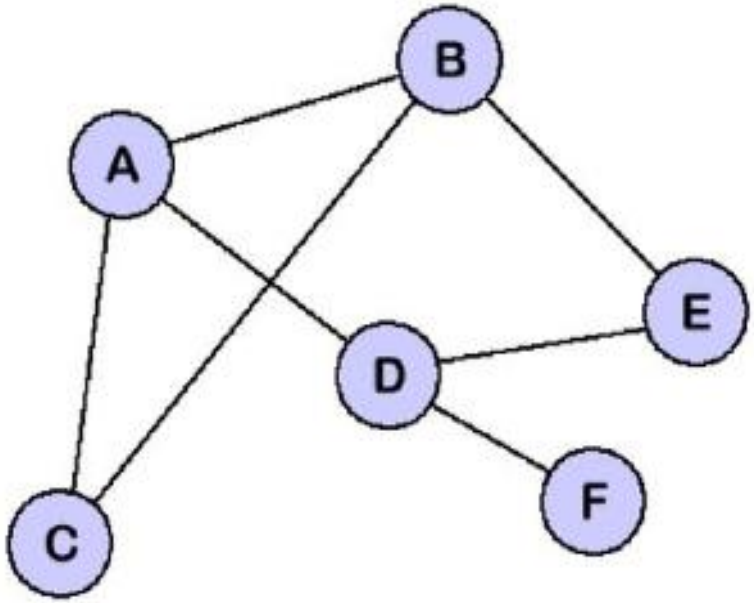
# Non-Linear Data Structure

## Non-linear Data Structures




- Each node may be connected with two or more nodes in a non-linear arrangement
- Removing one of the links could divide the data structure into two different data structures

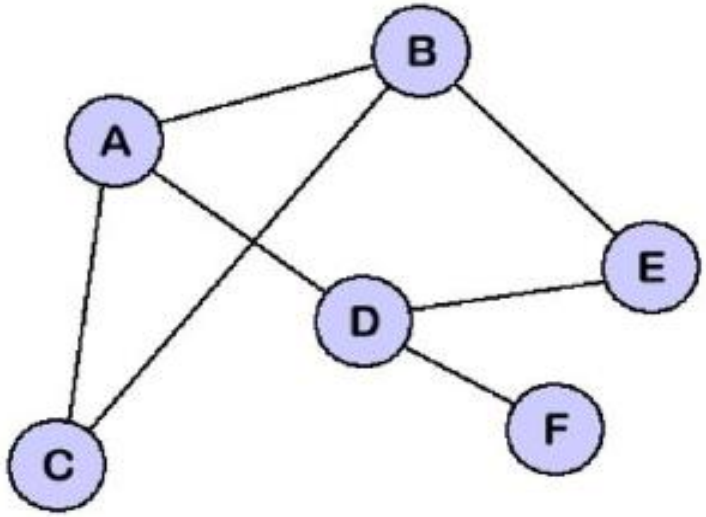
# Graphs




1. It is non linear
2. Set of nodes with set of connections
3. One-way or two-way
4. Directed (with arrow to show direction) or Non-Directed (two way links without any arrow)

# Graphs

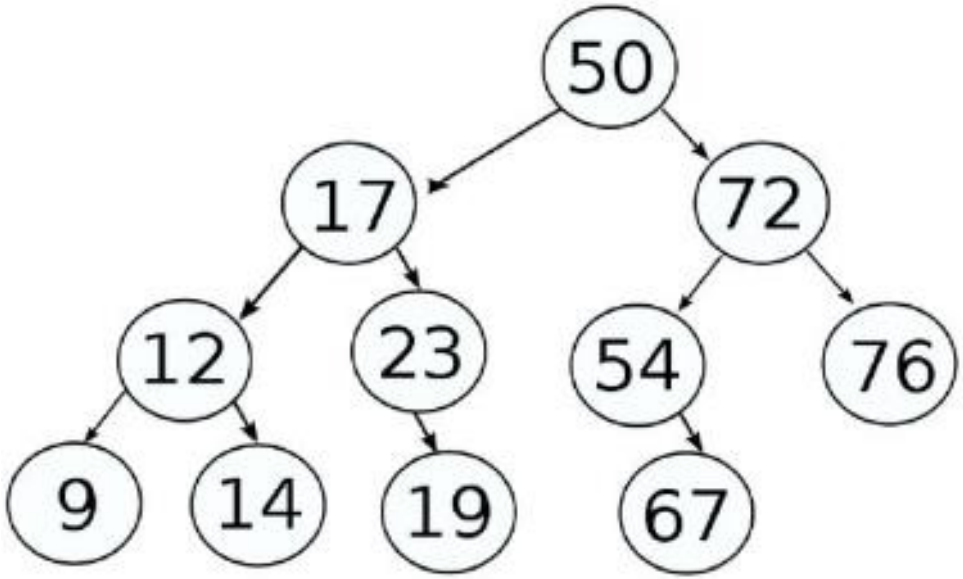
## Graphs Behaviour




1. Node A has three neighbors B,C, and D
2. We will use adjacency matrix for graphs data structures

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| A | -- | 1 | 1 | 1 | -- | -- |
| B | 1 | -- | 1 | -- | 1 | -- |
| C | 1 | 1 | -- | -- | -- | -- |
| D | 1 | -- | -- | -- | 1 | 1 |
| E | -- | 1 | -- | 1 | -- | -- |
| F | -- | -- | -- | 1 | -- | -- |

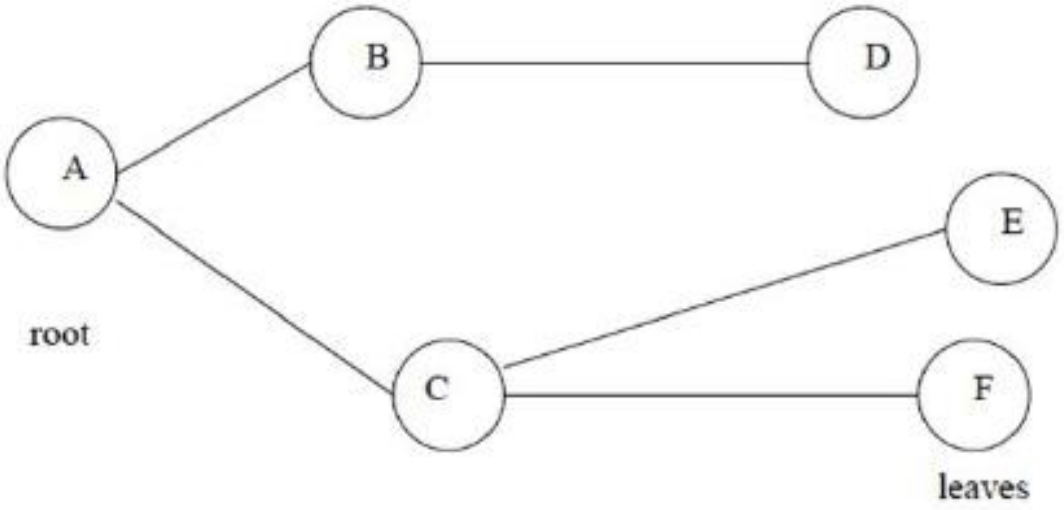
# Trees




- Hierarchical Data Structures
- We have
  - Nodes (root, child nodes, leaf nodes)
  - Branches

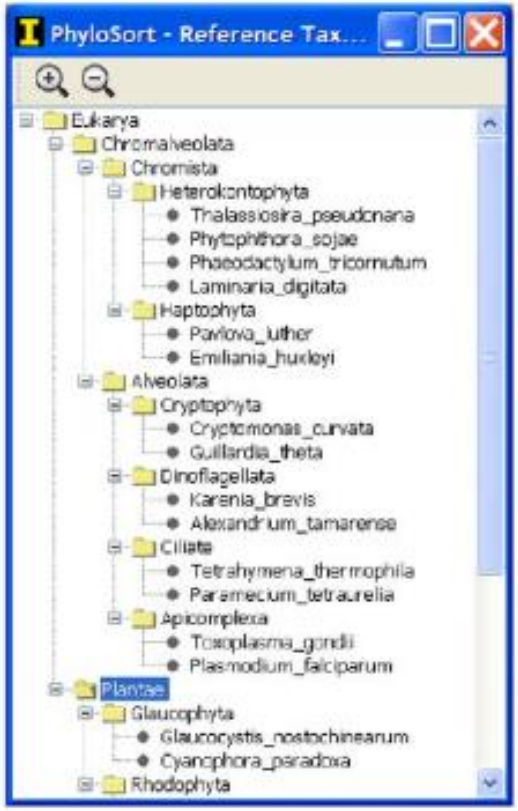
# Trees

- Trees are a special case of a graph data structure.
- The tree has nodes (shown with circles) that are connected with branches. Each node will have a parent node (except for the root) and may have multiple child nodes.

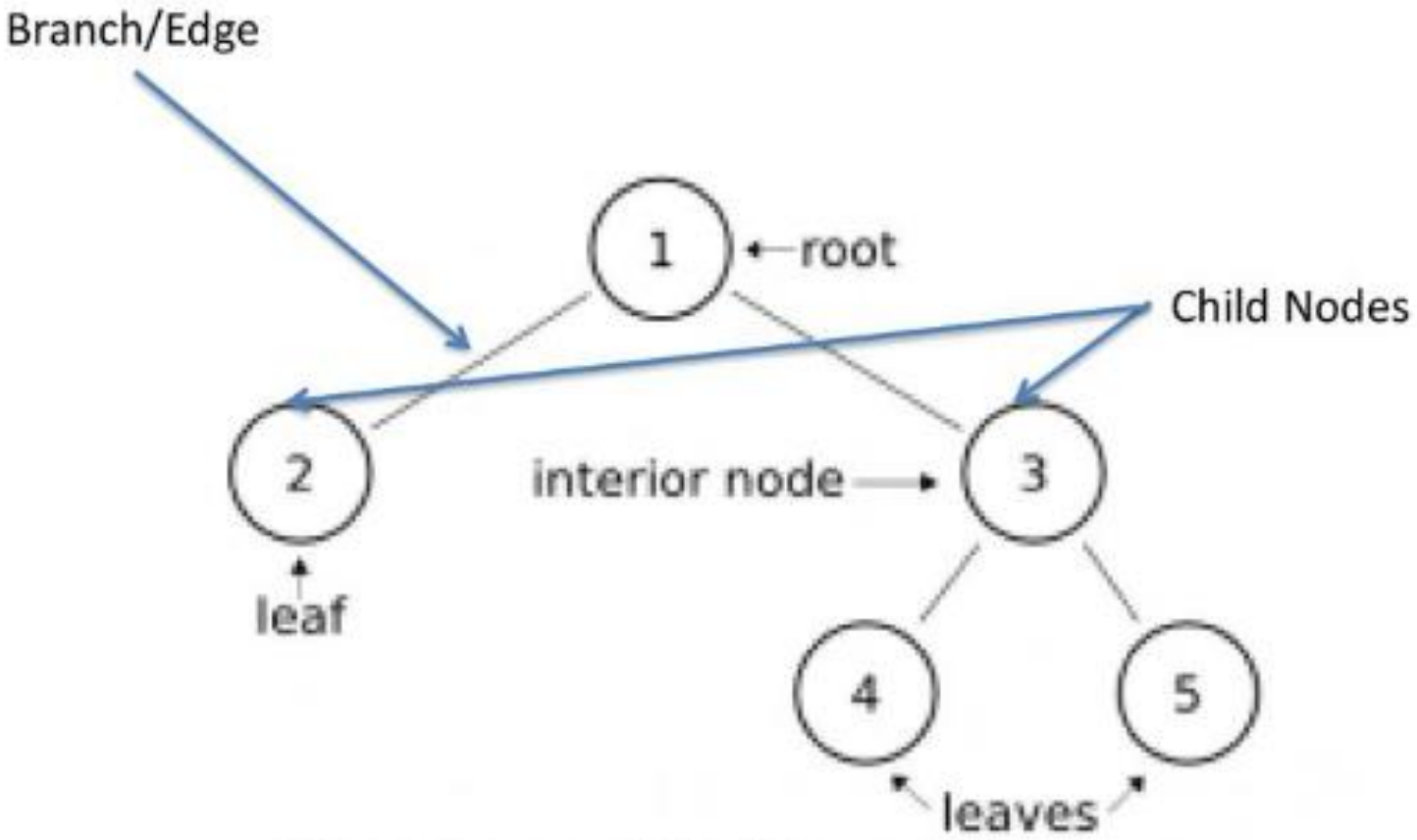


# Trees

## “Trees” in real application

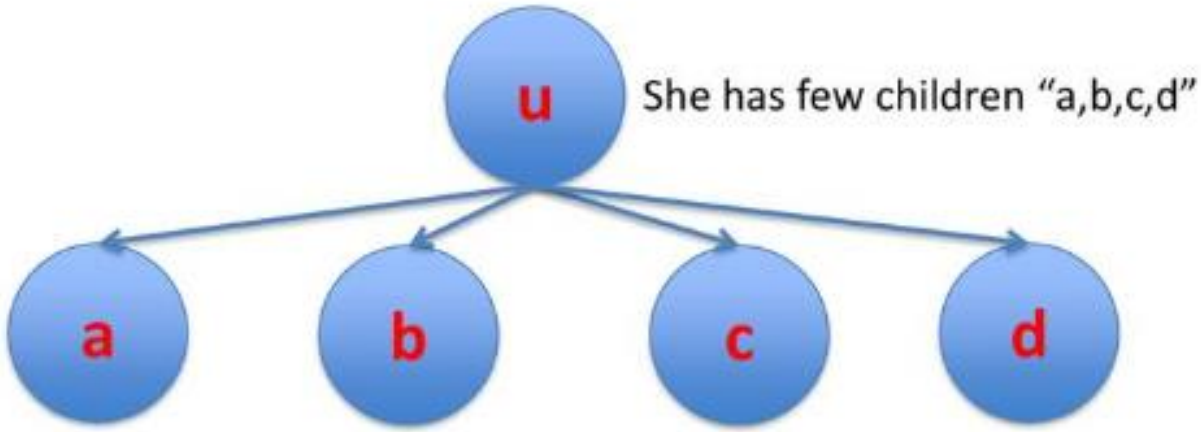



# Trees Terminology



# Trees

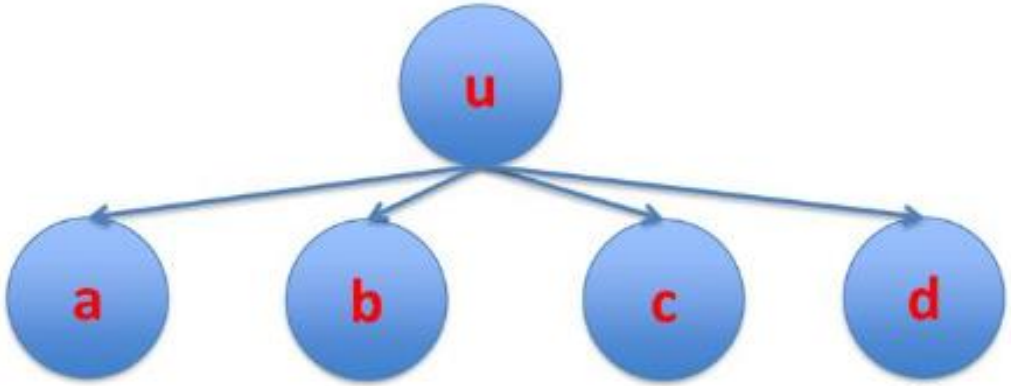
Child of a node **u**



# Trees

Parent node

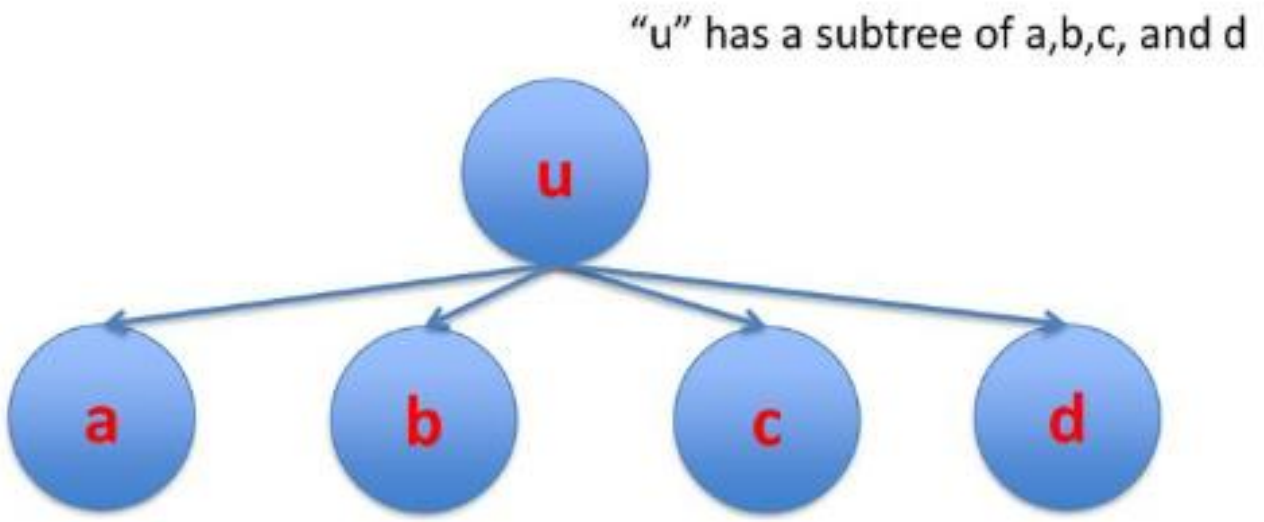
But root node does not have parent. Poor root




d has a parent node “u”

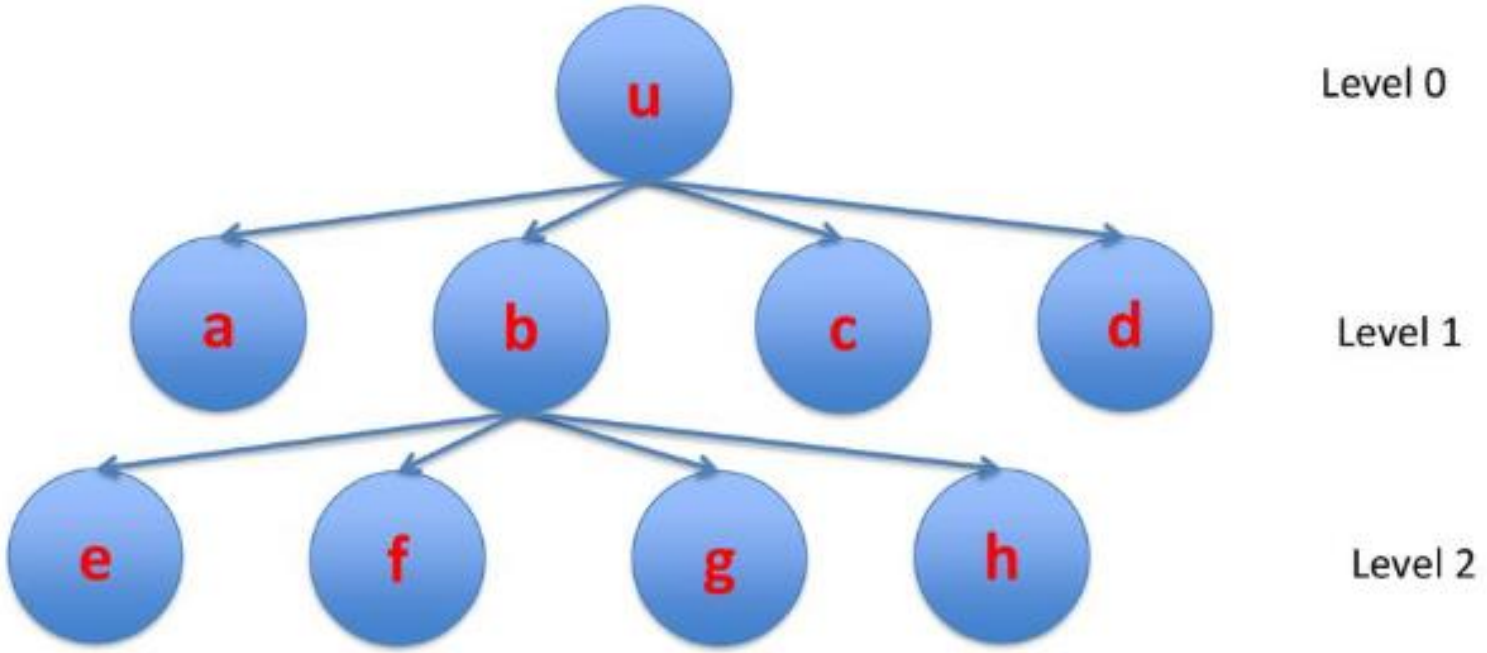
# Trees

subtree



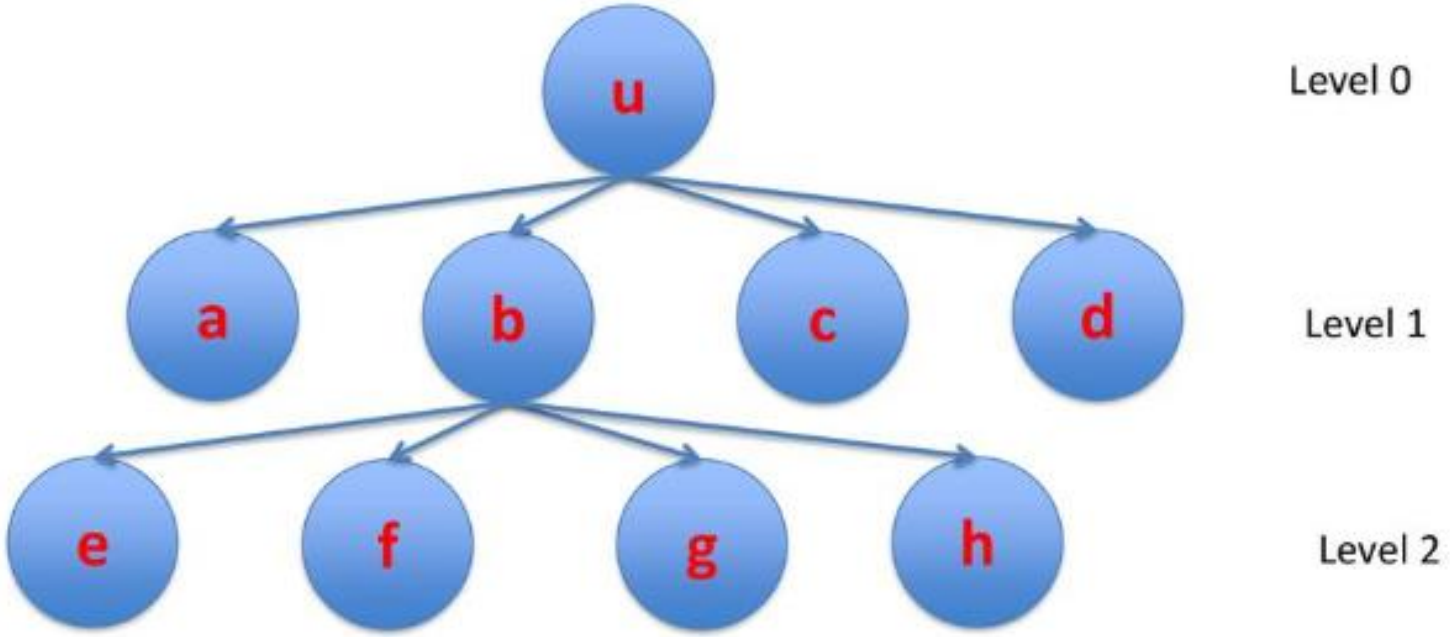
# Trees

Depth of a node



# Trees

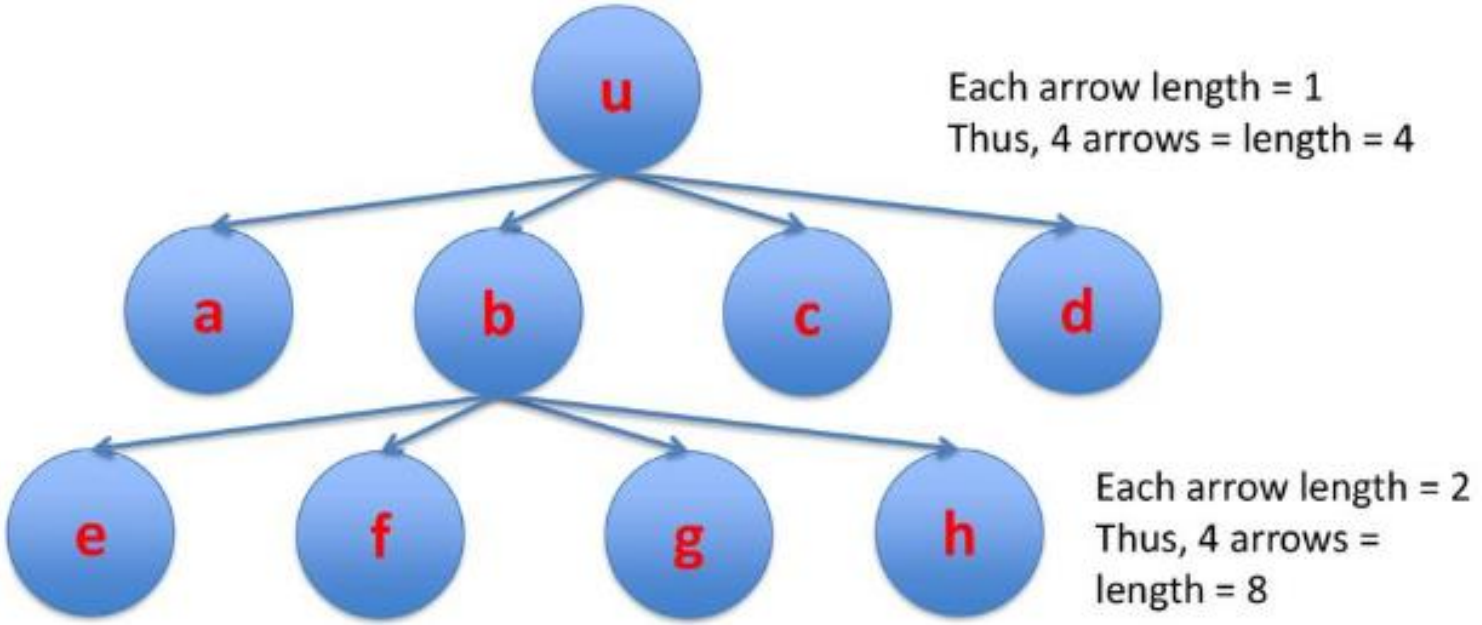
## Height of the tree




Height of the tree = 2

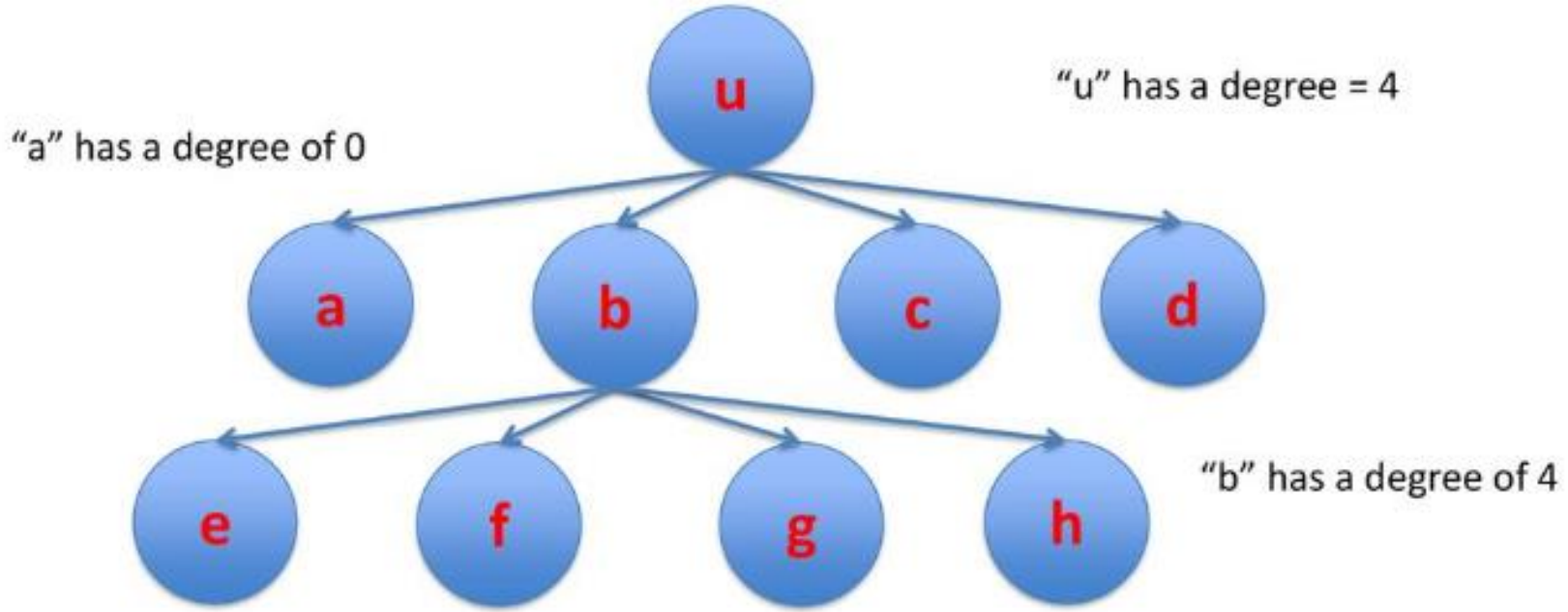
# Trees

## Path Length



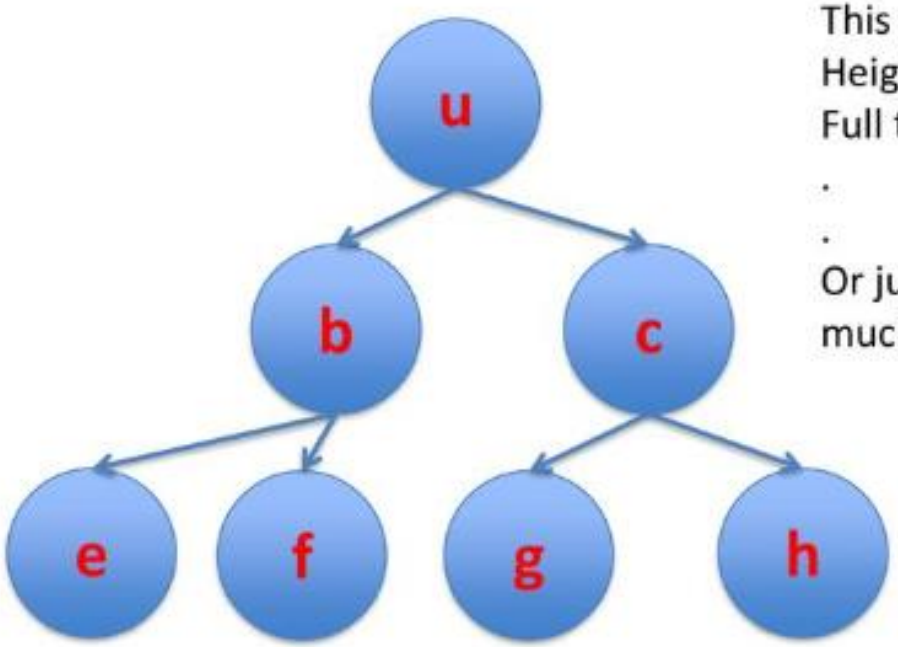
# Trees

## Degree



# Trees

Full Tree :



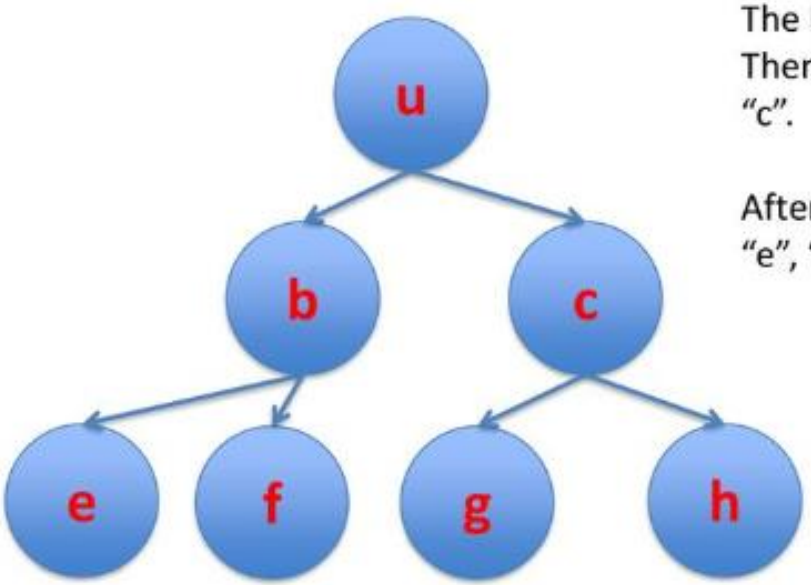

This tree has a degree of 2
Height = 2
Full tree = (2^3 – 1)/(2-1) = 7
.
.
Or just calculate yourself how much node this tree has..

# Trees

## Traversal Algorithm

We have multiple traverse patterns

**Let's go for Breath First Search**




The Mr.Algo visit "u" first
Then Mr. Algo visit "b", and "c".

After that, Mr. Algo back to "e", "f", "g", and "h"

# Trees

## Rooted Tree/Ordered Tree

**Rooted Tree**

- Any node can become a root for its own subnode
- General term “Free Tree”
- They have more family tree definition such as siblings or grandparents.

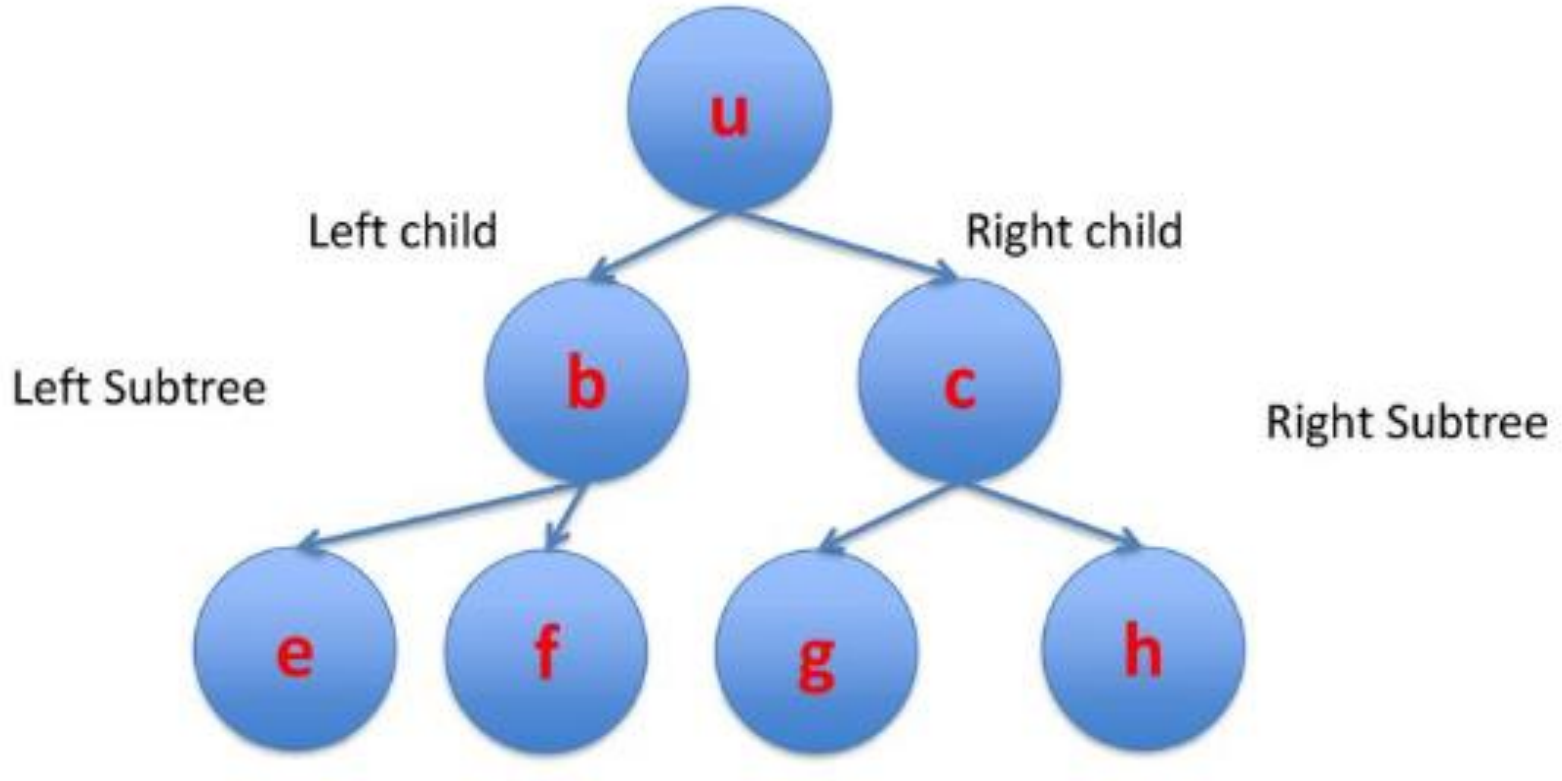
**Ordered Tree**

- Rooted tree where order of each child node is specified.

*[English] It is where you determined yourself where you put your node.*

- But if we have spesific order, then we could have so called n-tree (binary,quad, oct-tree)

# Binary Trees



# Binary Trees: Code

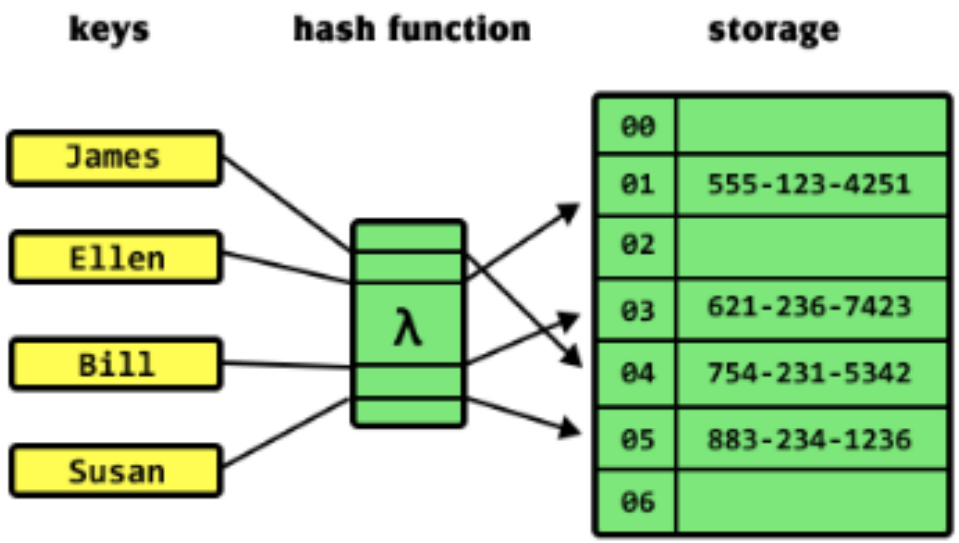
```
struct TreeNode {
    int item;       //The data in this node.
    TreeNode *left; //Pointer to the left subtree.
    TreeNode *right;//Pointer to the right subtree.
}
```

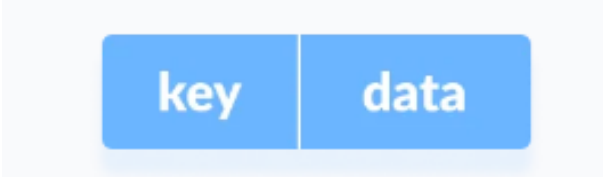
# Binary Trees: Code

```
int countNodes( TreeNode *root ) {
      // Count the nodes in the binary tree to which
      // root points, and return the answer.
    if ( root == NULL )
      return 0;  // The tree is empty.  It contains no nodes.
    else {
      int count = 1;   // Start by counting the root.
      count += countNodes(root->left);  // Add the number of nodes
                                        //    in the left subtree.
      count += countNodes(root->right); // Add the number of nodes
                                        //    in the right subtree.
      return count;  // Return the total.
    }
  } // end countNodes()
```

# Hash Table

- The Hash table data structure stores elements in key-value pairs where
  - Key- unique integer that is used for indexing the values
  - Value - data that are associated with keys.



# W3 – Lab

# Exercise

1. Find a few examples regarding the implementation of a Non-Linear Data Structure:
   - Graphs
   - Trees
   - Hash Table

# Thanks!