EXPOSER ON TREE DATA STRUCTURE

DETAILED NOTES ON DATA STRUCTURE, TYPE, IT'S PRACTICAL IMPLEMENTATION IN C LANGUAGE, TREE TRAVERSAL ALGO. AND THEIR APP IN REAL LIFE SITUATIONS

TREE DATA STRUCTURES

DEPARTMENT : Computer Science
SPECIALITY: Software Engineering
COURSE: Fundamentals of Algorithms

GROUP MEMBERS :

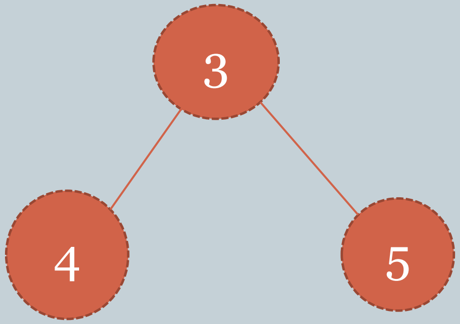
EBENDA ASSENE MARTHE DAVIDA (Leader)
MOMHA JEAN PIERRE
NANA KAMWA HERMAN
NASERI BERTHA BRAINY M BAKATA
NASERI BRYAN NYLE OPHE-BAKATA

Introduction to TREE DATA STRUCTURES

TREE DATA STRUCTURES

1. Introduction :

-
- It is a non-linear data structure that consists of collection of elements known as nodes.
- -Nodes are connected to each other by the use of edges



TREE DATA STRUCTURES

- Take for example a family hierarchy (family tree) , we have grand-parents , parents , kids. When these kids grow up and get married and have their own kids , the tree will grow even more . So this is an example of parent-children relationship and that is exactly what tree data structure is used for. It is used to represent that hierarchy.
- In programming a perfect example is the way folders are organised and structured in your computer.i.e a folder named food for example , inside the folder are other sub folders which are children of the folder "food".
- By doing this , you can organise your data better this way and it is easier to search or find a specific data in the folder.
- It is also used in databases and for different searching and sorting algorithms.

Definition of key terms

- **A root :**

  It is the first node of a tree. We must have only one root on a tree.
- **Edges :**

  Connecting links of any two nodes is called the edge of any data structure. If there are N nodes in a tree , we will have N-1 edges .
- **Parent :**

  The node who is the predecessor of every node is the parent node.
- **Child :**

  Descendant of every node is called the child node.
- **Leaf :**

  A node with no child is known as the leaf node.
- **Degree of a tree :**

  The total number of children of a node is called the degree of the node.
- Level :
- The root node is said to be level 0 and the children of the root level 1.
- Height of a tree :

Definition of key terms

- **Level :**

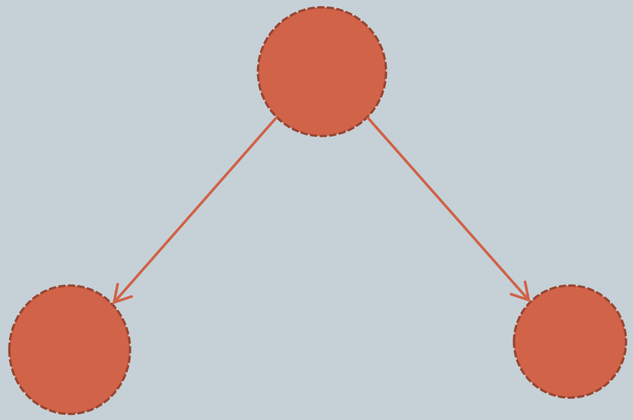
The root node is said to be level 0 and the children of the root level 1.

- **Height of a tree :**

The number of edges from a leaf node to a particular node and the longest path is known as the height of a tree.

Basic things to know on trees

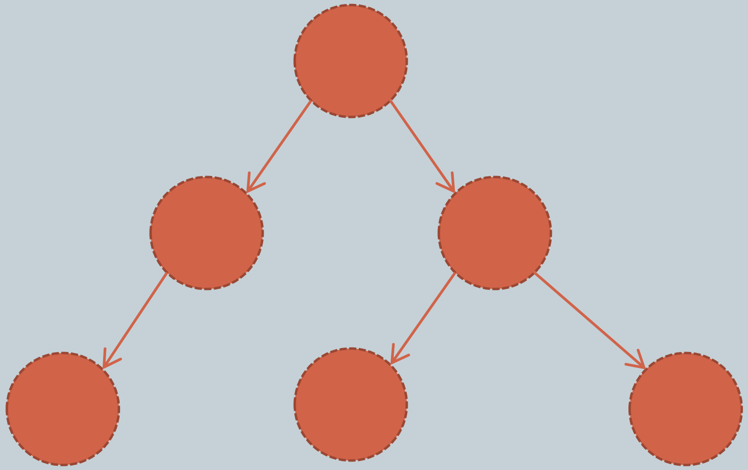
A singular entity of a tree is called a node. It is the building block of a tree. Multiple nodes build a tree data structure. For example;



NB : The interconnection between children in a tree data structure is not possible because it will become a graph data structure which consists of vertices and edges

Basic things to know on trees

- Child nodes come from parent nodes and this child nodes can have their own children. For example;

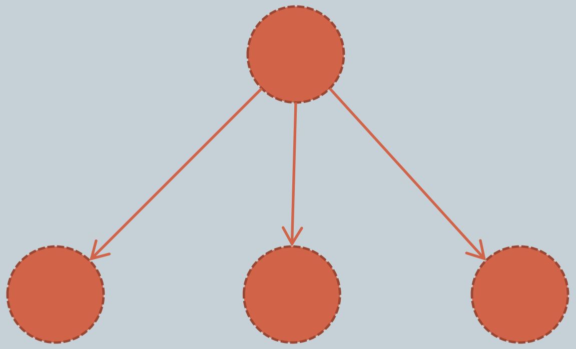


Inside these nodes we can store any data you need e.g numbers , strings , etc.

TYPES OF TREES

- **Regular tree :**

  It has a parent-child relationship. It has a fluctuating amount of children i.e it can have more or less than the original number of children



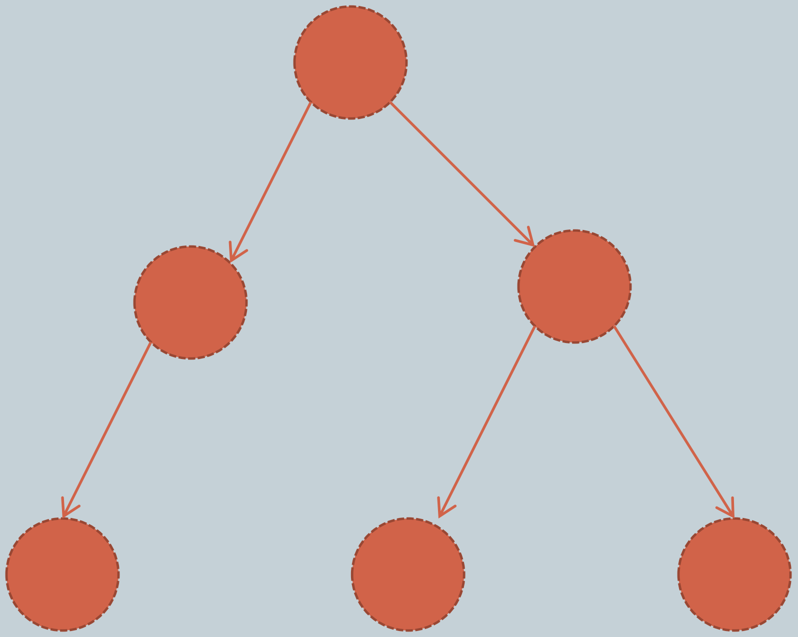
Binary tree :
It also has a parent-child relationship but here, each node is allowed to have atmost 2 children and atleast 0 child i.e it can have 0 , 1 or 2 children.

TYPES OF TREES

- **Binary tree :**

It also has a parent-child relationship but here, each node is allowed to have

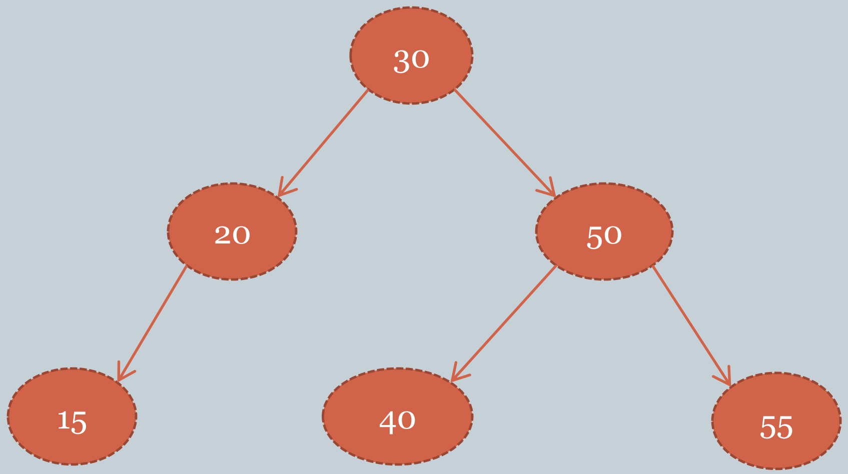
atmost 2 children and atleast 0 child i.e it can have 0 , 1 or 2 children.



TYPES OF TREES

Binary search tree :

It is exactly the same as binary tree except that its data is organised in a very specific way i.e;



TYPES OF TREES

- The tree must be implemented such that the left node child should be less than the root node and the right node child should be greater as seen on the diagram above.

- This is coming from the binary search property which allows us to explore all the nodes in sorted order by performing as in order tree transversal.

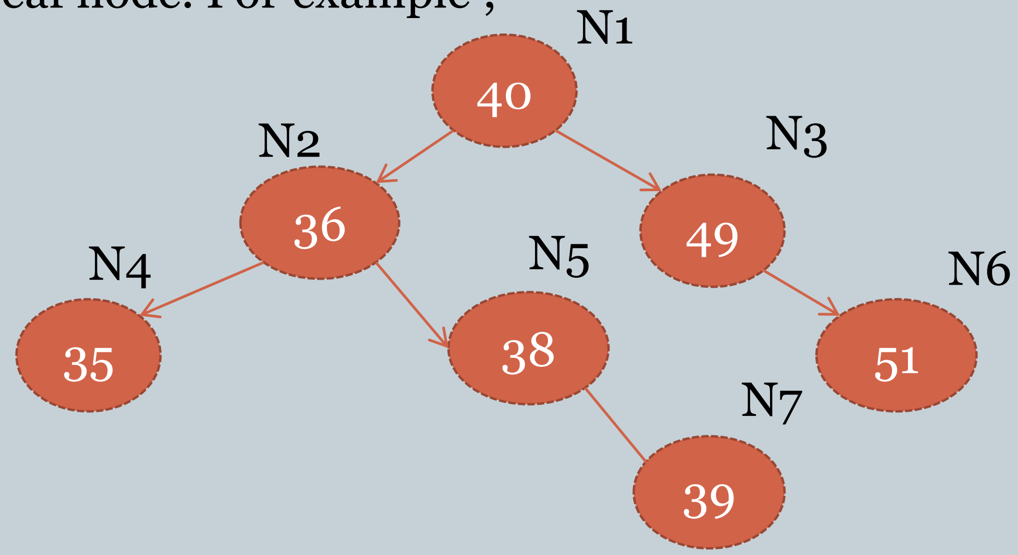
- This means we first explore all the nodes in the left subtree then the root node itself, and finally all the nodes in the right subtree .

TYPES OF TREES

- **Balanced binary search trees ;**

It is also known as height balanced tree. It is defined as a binary tree when the difference between the height of the left subtree and the right subtree is more than m, where m is usually equal to 1

The height of a tree is the number of edges on the longest path between the root node and the leaf node. For example ;



TYPES OF TREES

- The above tree is a binary search tree. In the above tree, N1 is the root node, and N4, N6, N7 are the leaf nodes. The N7 node is the farthest node from the root node. The N4 and N6 contain two edges and there exist three edges between the root node and N7 node. Since N7 is the farthest from the root node, therefore, the height of the above tree is 3.

TYPES OF TREES

- Now we will see whether the above tree is balanced or not. The left subtree contains the nodes N2, N4, N5 and N7, while the right subtree contains the nodes N3 and N6.The left subtree has 2 leaf nodes i.e N4 and N7.There is only one node edge between the node N2 and N4 and 2 edges between the nodes N2 and N7, therefore , node N7 is the farthest from the root node.

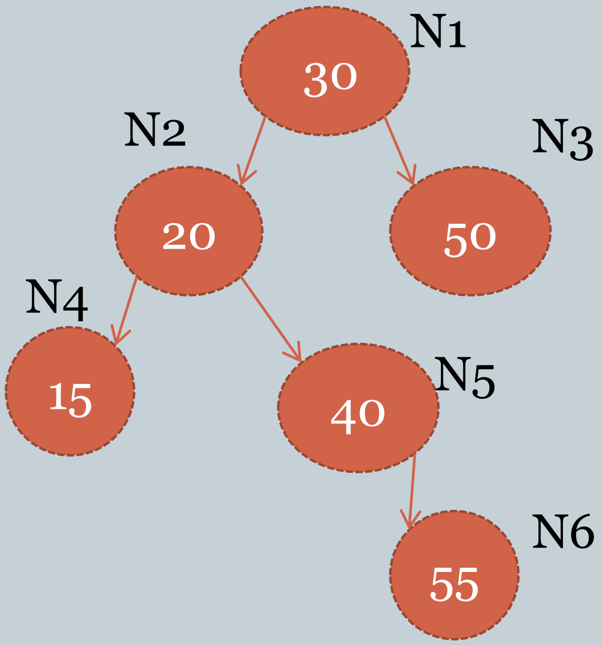
- The height of the left subtree is 2. The right subtree contains only 1 leaf node, therefore the height of the right subtree is 1. The difference between the heights of the left subtree and right subtree is 1. Since we got the value 1 so we can say that the above tree is a height-balanced tree.

- This process of calculating the difference between heights should be performed for each node like N2, N3, N4, N5, N6 and N7. When we process each node, then we will find that the value of m is not more than 1, so we can say that the above tree is a balanced binary tree.

TYPES OF TREES

- **Unbalanced Binary search tree :**

It is one where all the data is overwhelming either greater than or less than the root node



TYPES OF TREES

- In the above tree , N6, N4, and N3 are the leaf nodes, where N6 is the farthest node from the root node.
- Three edges exist between the root node and the leaf node; therefore, the height of the above tree is 3.
- When we consider N1 as the root node, then the left subtree contains the nodes N2, N4, N5 and N6, while right subtree contains the node N3.In the left subtree, N2 is a root node, and N4 and N6 are leaf nodes. Among N4 and N6 nodes, N6 is the farthest node from its root node, and N6 has 2 edges; therefore, the height of the left subtrees is 2.
- The right subtree does not have any child on its left and right, therefore the height of the right subtree is 0. Since the height of the left subtree is 2 and the height of the right subtree is 0, so the difference between the height of the left subtree and the right subtree is 2.
- According to the definition of a balance binary search tree, the difference between the height of the left subtree and the right subtree must not be greater than 1. In this case the difference comes to be 2, which is greater than 1, therefore, the above tree is an unbalanced binary search tree.

TYPES OF TREES

- **<u>Why do we need balancing :</u>**

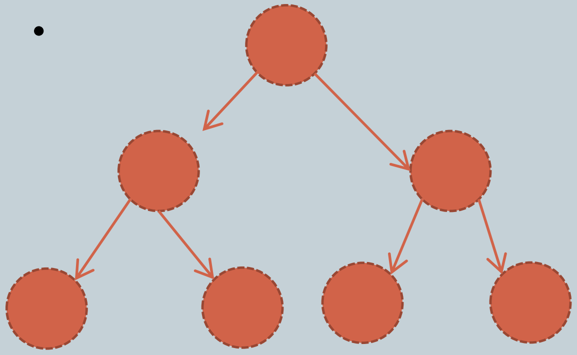
Tree balancing is a crucial aspect of managing binary trees in computer science. It ensures that operations on tree are as efficient as possible, which can significantly improve the performance of a computer program.

- **<u>Balanced vs Unbalanced trees :</u>**

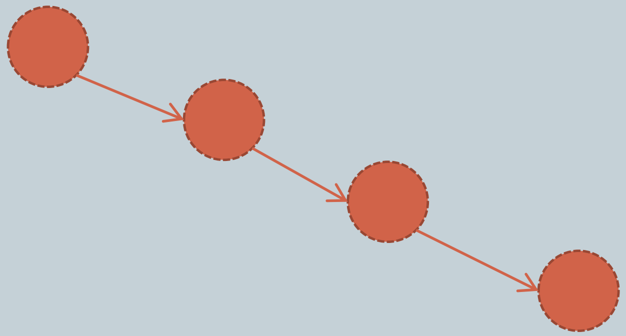
Binary search trees can easily become unbalanced if they are not managed correctly. e.g imagine the root node has a value of 1 and therefore each time we insert a node, it will just form on the right hand side of the root node.

TYPES OF TREES

•



Balanced BST



Unbalanced BST

This is not good… why? Well if we have an unbalanced BST then lose efficiency.
 The more unbalanced the tree becomes the more the **time complexity** of the lookup,
insertion, deletion etc becomes **O(n).**

TYPES OF TREES

- With a balanced binary search tree, we can search efficiently as the tree is ordered on each level by value(left is lower, right is higher). This means that as the data set grows exponentially as we go down the tree, we just search the tree for one particular node before moving onto the next step. This makes the time complexity of balanced binary search trees: **O(log n).**

TYPES OF TREES

- **Tree traversal :**

This is visiting each node in a binary tree and printing its value. Tree traversal algorithms can be classified broadly in the following 2 categories by the order in which nodes are visited; Depth-first search(DFS) and Breadth-first search(BFS) algorithms.

- **Depth-first search :**

It starts with the root node and the first visits all nodes of one branch as deep as possible before backtracking.

- **Breadth-first search :**

This also starts from the root node and visits all nodes of current depth before moving to the next depth in the tree.

Ways of traversing binary tree

- There are 3 ways of traversing a binary tree; **Preorder, inorder, postorder traversal**
- **Preorder traversal :**

Visits the current node before visiting any nodes inside left or right subtrees.

- **Inorder traversal :**

Visits the current node after visiting all nodes inside left subtree, but before visiting any node within the right subtree.

- **Postorder traversal :**

Visits the current node after visiting all the nodes of left and right subtrees