Session 6: Trees

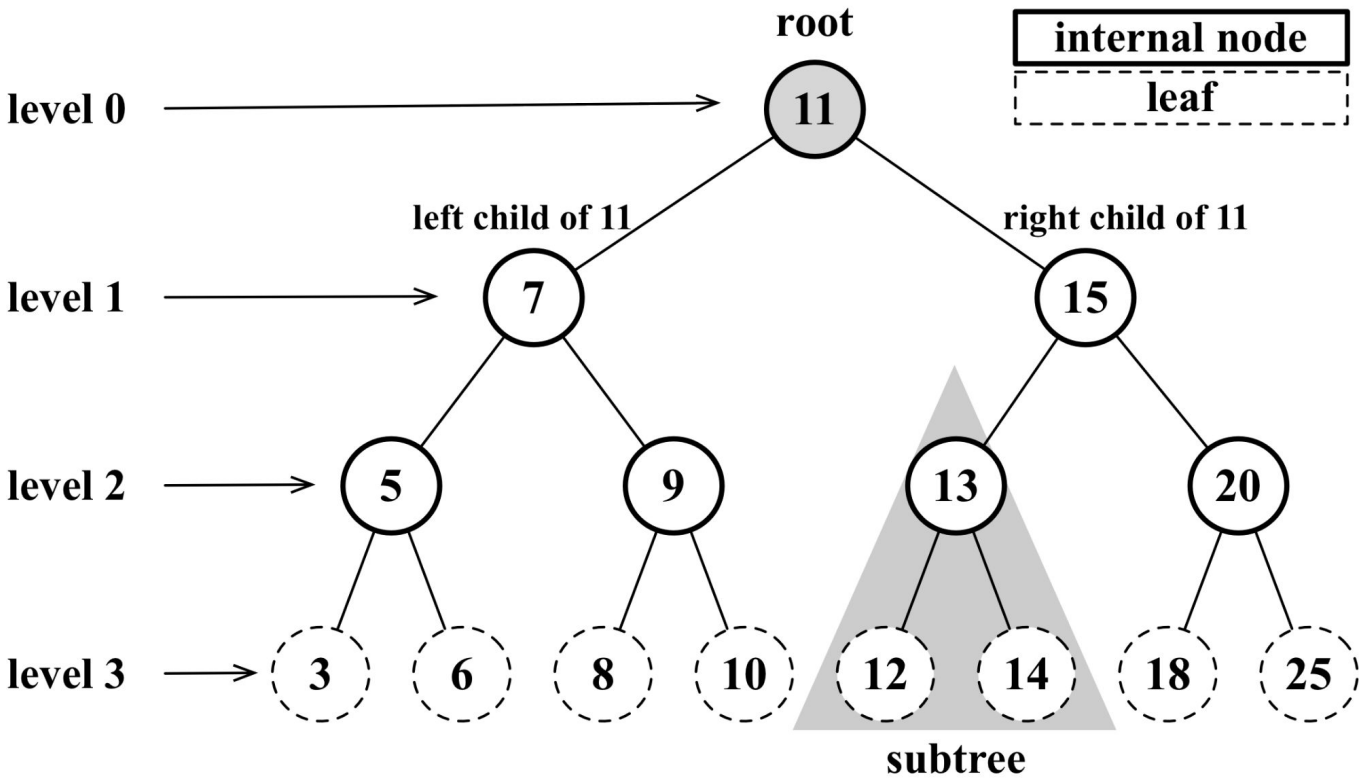
Data Structures and Algorithm 1 - Lab

Yahya Tawil
05 Nov 2021

This Session

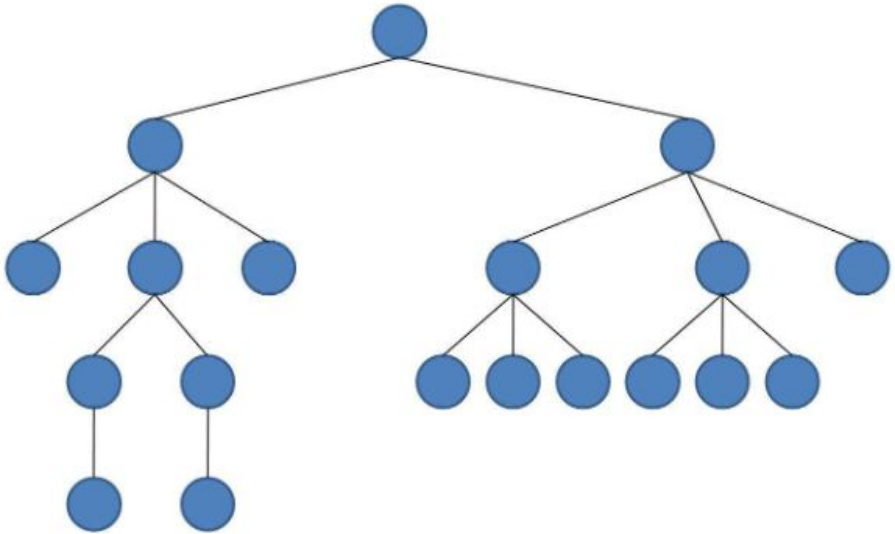
- Review the implementation of binary tree in C/C++.
- Compute the size of the subtree rooted at a node.
- Compute the depth/height of a node.

Trees Termonology



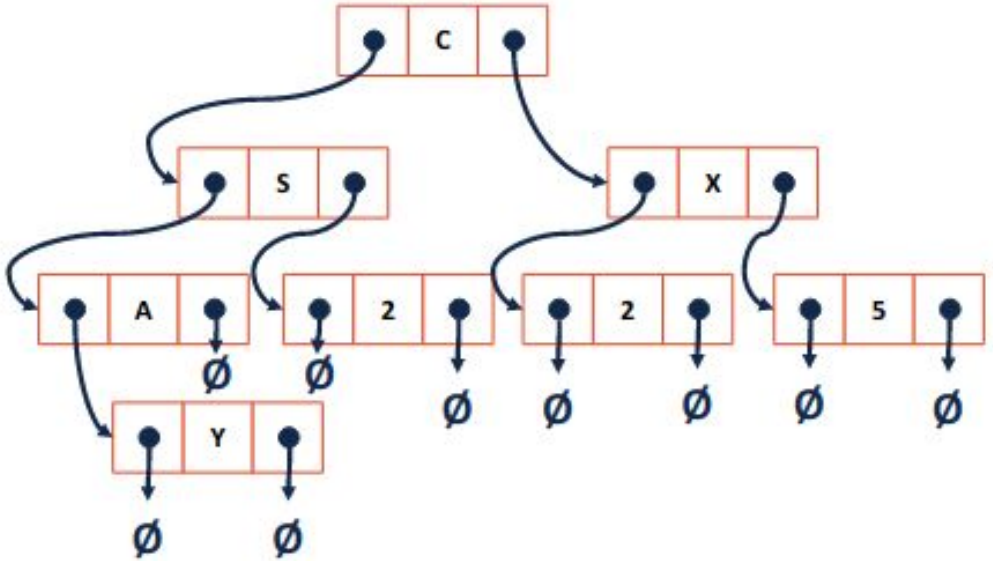
Example

Explain the values of the main characteristics of the tree shown (height, number of nodes, leafs and internal nodes)



Binary Tree Implementation

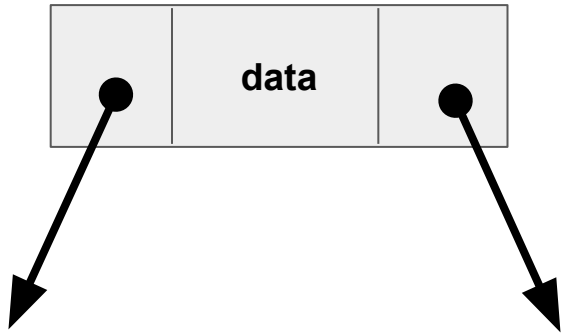
Trees aren’t new



Binary Tree Implementation

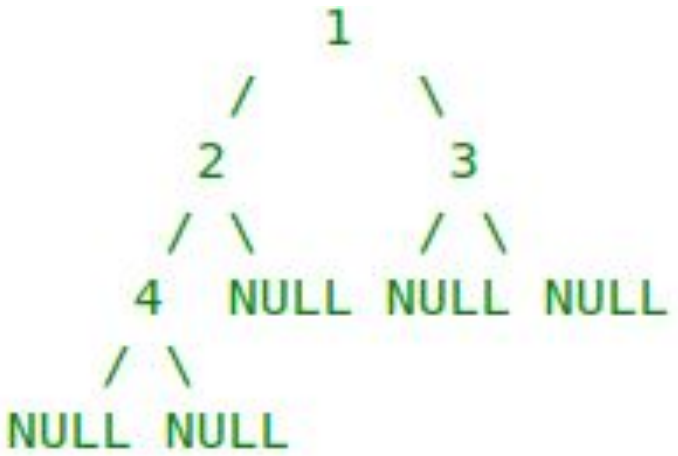
A tree node:

```
struct Node {
    int data;
    struct Node* left;
    struct Node* right;


    Node(int val)
    {
        data = val;

        left = NULL;
        right = NULL;
    }
};
```



Binary Tree Implementation

Create the following Tree:



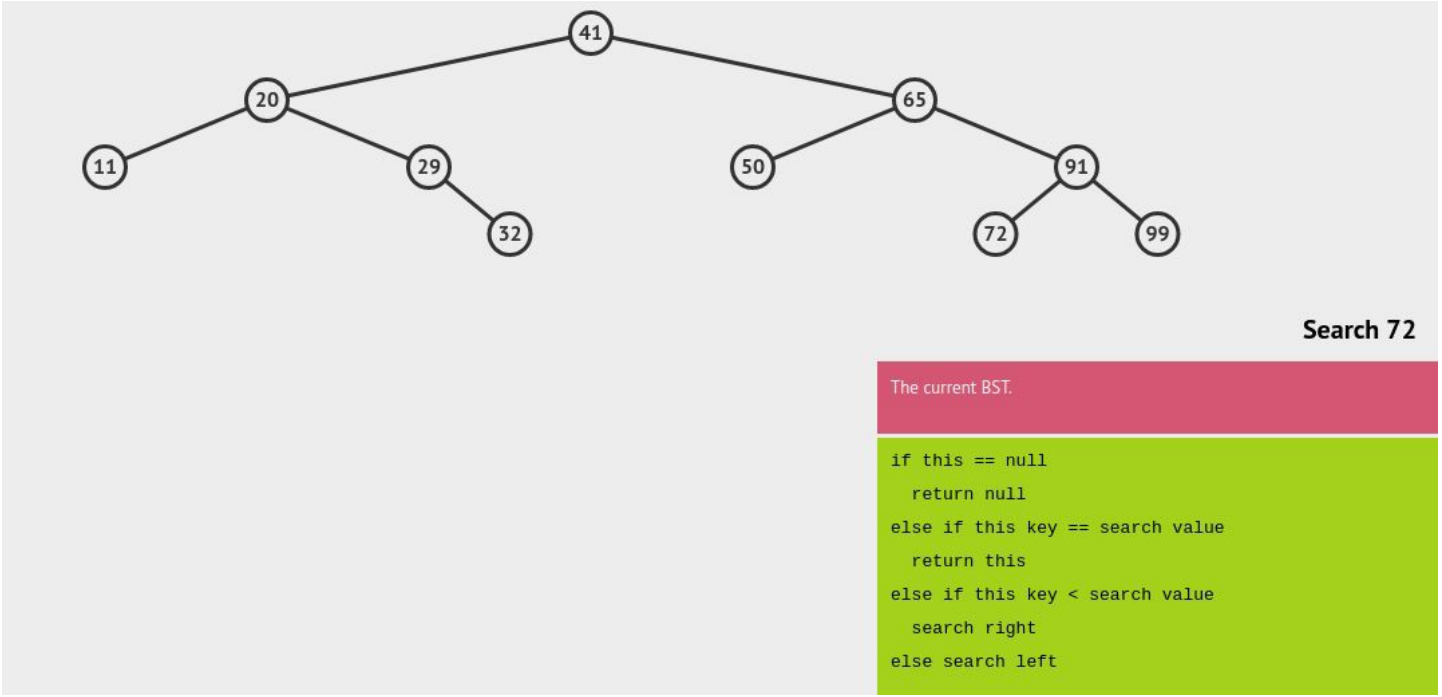
Binary Tree Implementation

```
Node* root = new Node(1);

root->left = new Node(2);
root->right = new Node(3);

root->left->left = new Node(4);
```

Binary Tree: Search

```
struct node* search(struct node* root, int key)
{

    if (root == NULL || root->key == key)
       return root;


    if (root->key < key)
       return search(root->right, key);


    return search(root->left, key);
}
```

Binary Tree: Search



https://visualgo.net/en/bst

Binary Tree: Size of tree

Size of Tree = Number of nodes

```
int size(Node* node)
{
    if (node == NULL)
        return 0;
    else
        return(size(node->left) + 1 + size(node->right));
}
```

Binary Tree: Depth and Height of Node

The depth of a node is the number of edges from the node to the tree's root node.

```
int depth(Node* root, int data)
{
    if (root == NULL || root->data == data)  return 0;

    if (root->data < data)
        return 1 + depth(root->left, data) ;
    else if(root->data > data)
        return 1 + depth(root->right, data) ;
}
```

Binary Tree: Depth and Height of Node

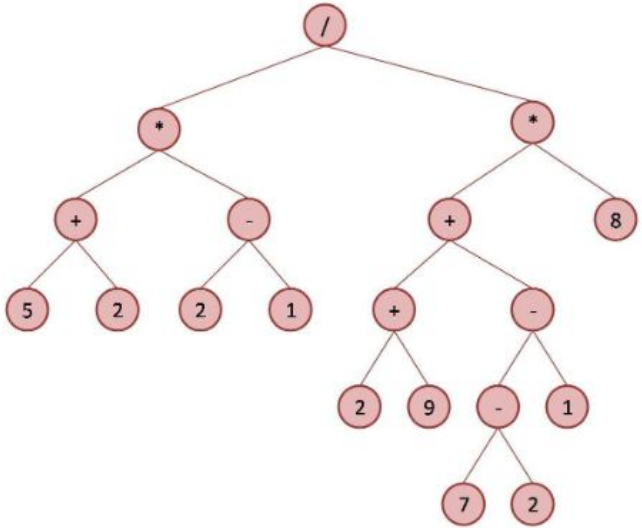
The **height** of a node is the number of edges on the longest path from the node to a leaf

```
int maxDepth(Node* node)
{
    if (node == NULL) return 0;
    else
    {
        int lDepth = maxDepth(node->left);
        int rDepth = maxDepth(node->right);
        if (lDepth > rDepth)
            return(lDepth + 1);
        else return(rDepth + 1);}}
```

Binary Tree

Draw the binary tree representation of the following

$$(((5 + 2) * (2 - 1))/((2 + 9) + ((7 - 2) - 1)) * 8)$$



Assignment 6: Exercise 1

Write a function returns the sum of all the data in a binary tree.

Assignment 6: Exercise 2

Write a function returns the maximum value in a binary tree, and return -1 if the tree is empty.

Assignment 6: Exercise 3

Write a C function prints all the data less than a given value v in a binary tree. (Write it in recursive style)