Binary tree traversal

Binary Tree traversals

- In whatever way we store the data, either as a linked list or array, we should be able to traverse all the data or access all the data
- Linked list and array are linear data structures and we can visit all the elements in a sequence without the possibility of visiting an element twice or more
- But, in a Binary Tree, there is no such linear order
- So, how to enumerate or visit or traverse all the elements of a binary tree such that we pass through the nodes only once???

Binary Tree traversals

Three methods of traversing

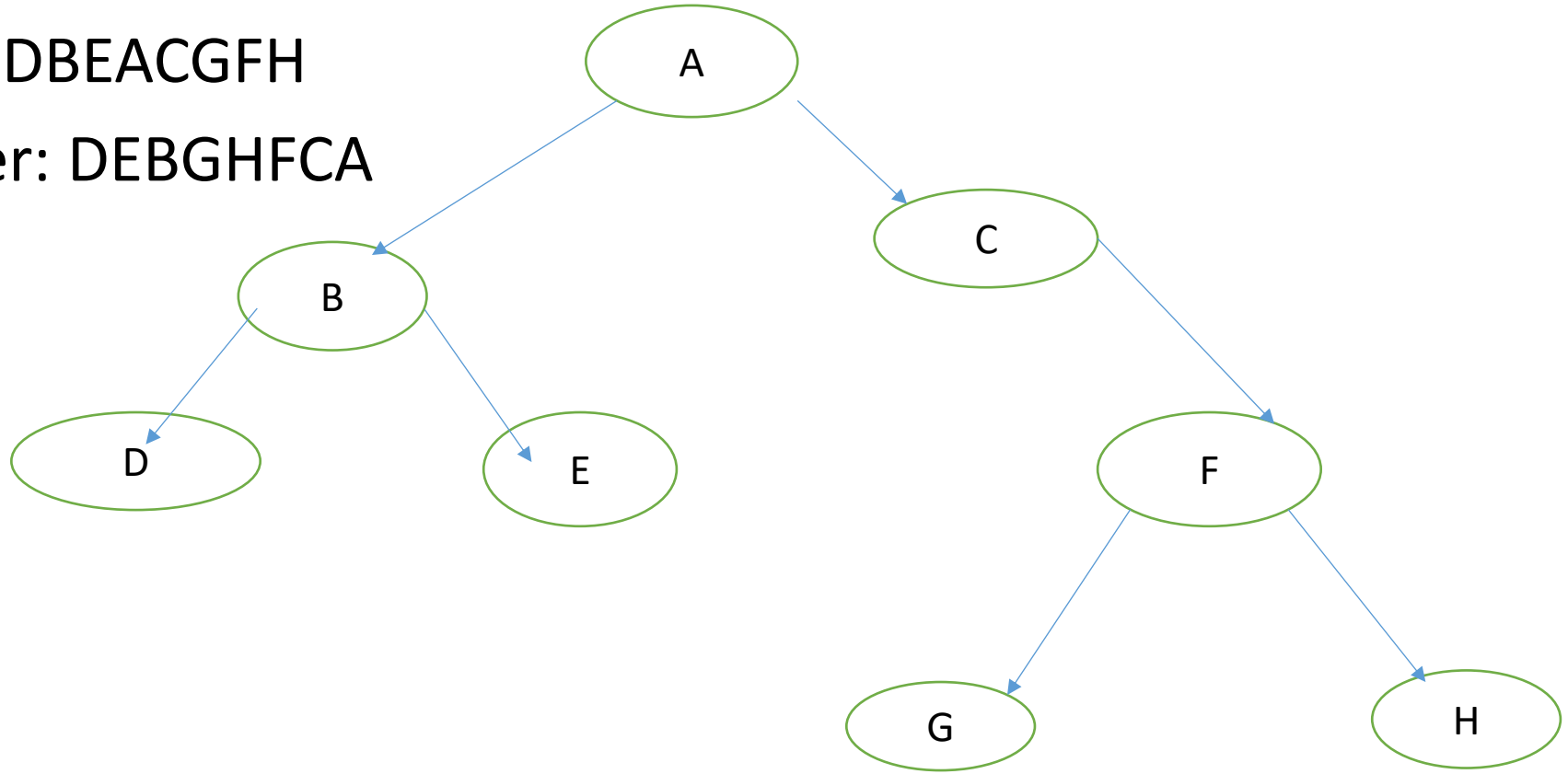
- **Preorder:** **1.** Visit the root
  2. Traverse left subtree in preorder
  3. Traverse right subtree in preorder
- **Inorder:** **1**. Traverse left subtree in inorder
  2. Visit the root
  3. Traverse right subtree in inorder
- **Postorder:** **1.** Traverse left subtree in postorder
  2. Traverse right subtree in postorder
  3. Visit the root

Binary Tree Traversal

Preorder: ABDECFGH

Inorder: DBEACGFH

Postorder: DEBGHFCA

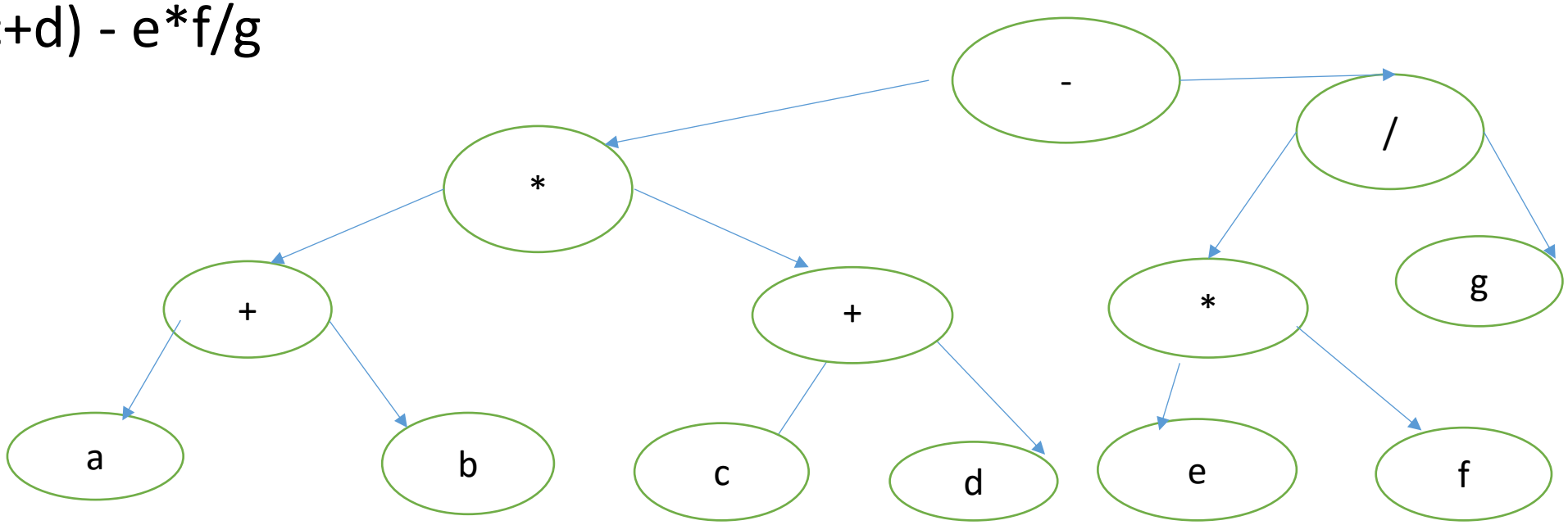


Applications of Binary Tree

- We can represent mathematical expression containing operands and binary operators by a strictly binary tree
- The root of the strictly binary tree contains an operator that is to be applied to the results of expressions represented by left and right subtrees
- A node representing operator is always a non leaf node whereas node representing the operand will be a leaf node
- For the expression (a+b) *(c+d) - e*f/g, the tree will be

Binary Tree for mathematical expression

(a+b) *(c+d) - e*f/g



Traversals of expression tree

- Preorder is equivalent to prefix
- Postorder is postfix

Few functions of binary tree

- The number of nodes in a binary tree
- The sum of the contents of all nodes of a binary tree
- The depth or height of a binary tree
- Whether we should have recursive function for above operations or we can have non recursive also??

Count the nodes of a Binary tree

```
int count_nodes( binarytree * root)
{
   if (root == null)
      return 0;
   else
        return(1 + count_nodes(root->left) + count_nodes(root->right));
}
```

Height of a Binary tree

```
int height( binarytree * root)
{ int h1,h2;
   if (root == null)
      return 0;
   else
       { h1 = 1 + height(root->left);
        h2 = 1+ height(root->right);
       }
     If(h1>h2) return h1;
    else return h2;
}
```