Data Structures (15B11CI311)

Odd Semester 2020

3rd Semester , Computer Science and Engineering

Jaypee Institute Of Information Technology (JIIT), Noida

Lecture: 25

Topics to be covered:

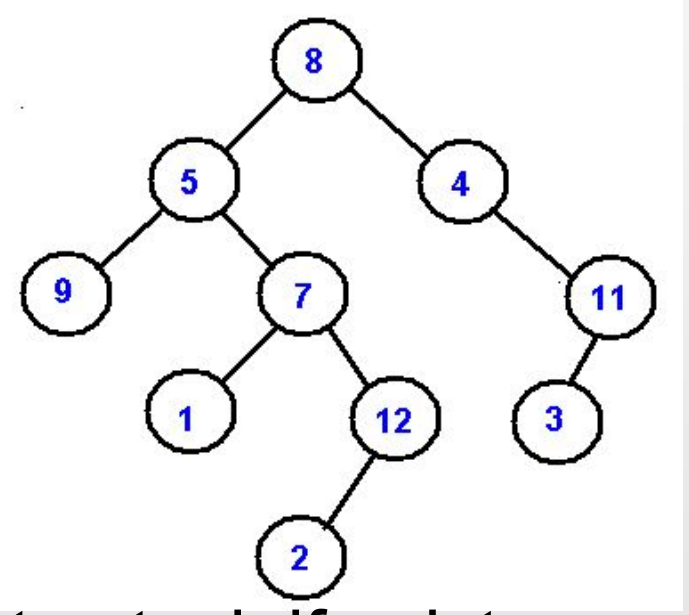
- Non-recursive Traversal of Binary tree
- Construction of Binary Tree

Binary Tree Traversal

- A traversal is a process that visits all the nodes in the tree exactly once.
  - Depth-first traversal
    - Preorder traversal
    - Inorder traversal
    - Postorder traversal
  - Breadth-first traversal
    - Level order traversal

Non-recursive Preorder Traversal

- Traverse the nodes in the following order
  - Push the root into Stack
  - Repeat till Stack is not empty
    - Display top of the Stack
    - Pop the node from Stack
    - Push right child of the popped node into stack if exist
    - Push left child of the popped node into stack if exist

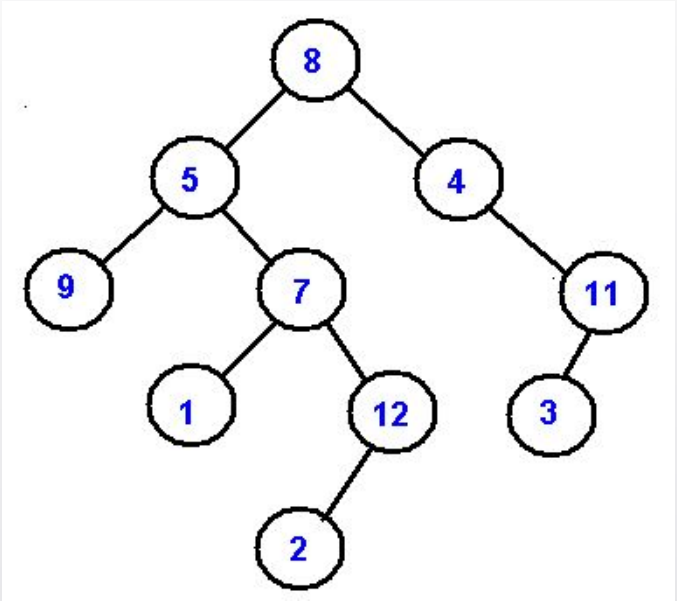


Preorder Traversal: Non-recursive Implementation

```
void preorder_non(node *root) {
   if (root == NULL)
   return;
   stack<node *> Stack;
   Stack.push(root);
   while (!Stack.empty()) {
        node *temp = Stack.top();
        cout<< temp->data;
        Stack.pop();
        if (temp->rchild)
         Stack.push(temp->rchild);
        if (temp->lchild)
         Stack.push(temp->lchild);
     }
}
```

Non-recursive Inorder Traversal

- Traverse the nodes in the following order
  - temp=root
  - Repeat till temp is not NULL or Stack is not empty
    - Repeat till temp is not NULL
      - Push temp into Stack
      - temp=Left_child(temp)
    - Display top of the Stack
    - Pop the node from Stack and store it in temp
    - Push right child of the temp into stack

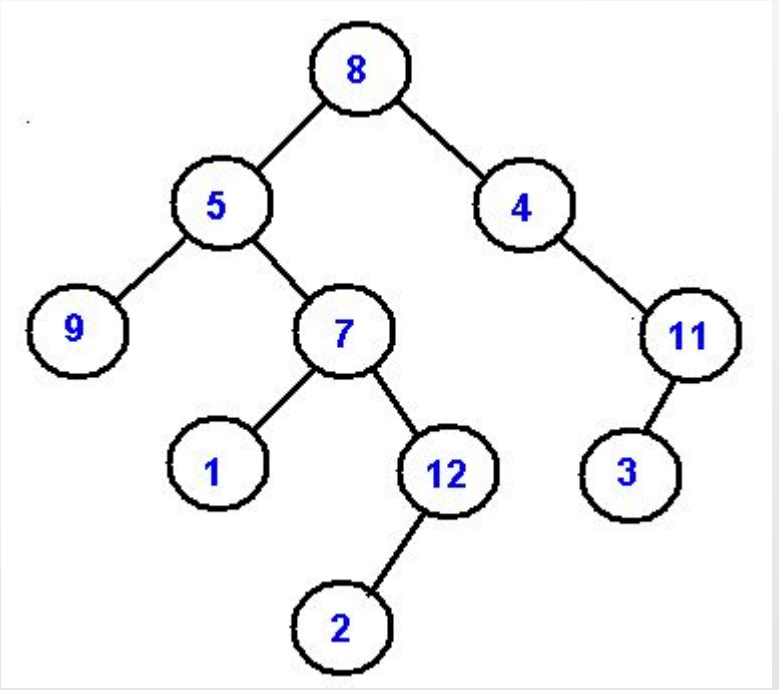


Inorder Traversal: Non-Recursive Implementation

```
void inorder_non(node *root) {
  stack<node *> Stack;
  node *temp = root;
  while (temp != NULL || !Stack.empty()) {
      while (temp != NULL) {
          Stack.push(temp);
          temp = temp->lchild;
      }
      temp = Stack.top();
          Stack.pop();
          cout << temp->data;
          temp = temp->rchild;
    }
}
```

Non-recursive Postorder Traversal (Using 2 Stacks)

- Push root to first stack.
- Repeat till first stack is not empty
  - Pop a node from first stack and push it to second stack
  - Push left and right children of the popped node to first stack
  - Print contents of second stack



Source: https://www.geeksforgeeks.org/iterative-postorder-traversal/

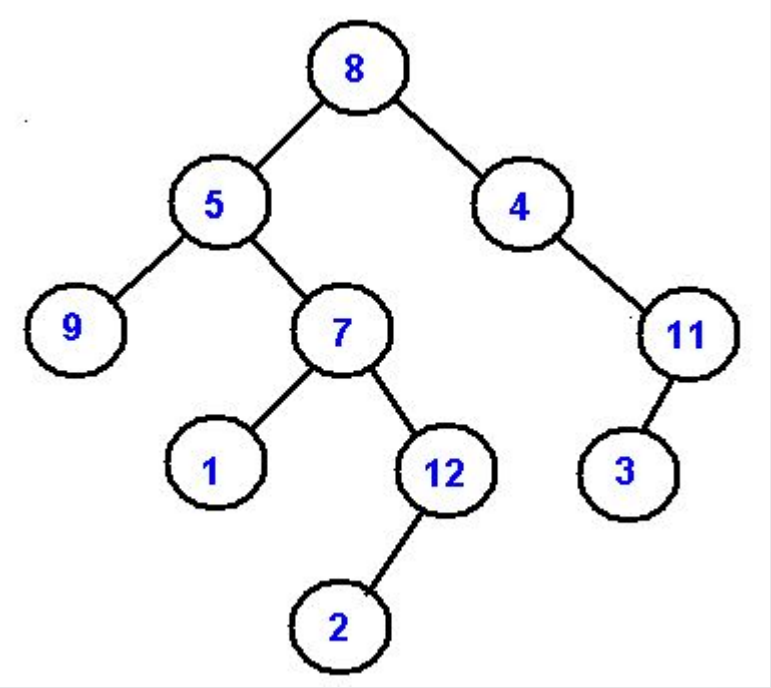
Postorder Traversal : Non-Recursive Implementation

```
void postorder_non(node* root) {
    if (root == NULL)
        return;
    stack<node *> s1, s2;
    s1.push(root);
    node* temp;
    while (!s1.empty()) {
            temp = s1.top();
            s1.pop();
            s2.push(temp);
            if (temp->lchild)
                    s1.push(temp->lchild);
            if (temp->rchild)
                    s1.push(temp->rchild);
        }
        while (!s2.empty()) {
                temp = s2.top();
                s2.pop();
                cout << temp->data;
        }
}
```

Source: https://www.geeksforgeeks.org/iterative-postorder-traversal/

Non-recursive Levelorder Traversal

- Traverse the nodes in the following order
  - Insert the root into queue (FIFO)
  - Repeat till queue is not empty
    - Delete the element from queue and display it
    - Insert its left child into Queue if exist
    - Insert its right child into Queue if exist



Levelorder Traversal: Non-Recursive Implementation

```
void Levelorder_non (node * root)
{
    queue <node *> Q;
    node *temp = root;
     if (temp != NULL)
     {
          Q.push (temp);
           while (!Q.empty ()) {
                temp = Q.front ();
                Q.pop ();
                cout<<temp->data;
                if (temp->lchild)
                     Q.push (temp->lchild);
                if (temp->rchild)
                     Q.push (temp->rchild);
           }
     }
}
```

Binary Tree Construction

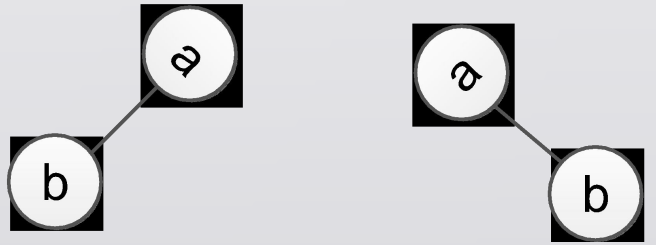
- Let the elements (more than one) in a binary tree are different and one of its traversal sequence is given.
- By using the given traversal sequence, a unique binary tree can not be defined.

Preorder = ab

Inorder = ab

Postorder = ab

Levelorder = ab

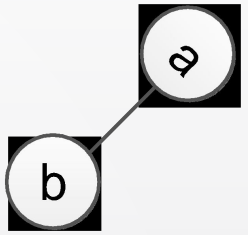


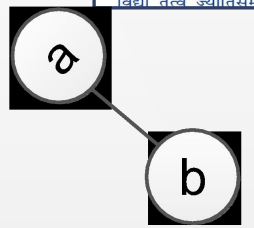
Source: https://www.cise.ufl.edu/~sahni/cop3530/

Binary Tree Construction

- The binary tree can be constructed by using two given traversal sequences but it depends on which two sequences are given.
- Preorder and postorder can not uniquely define a binary tree.
- Preorder and levelorder can not uniquely define a binary tree.
- Postorder and levelorder can not uniquely define a binary tree.

Preorder = ab





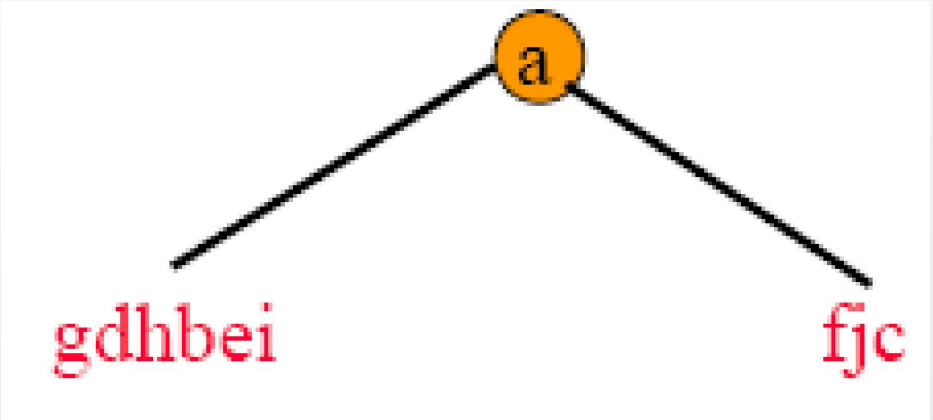
Postorder = ba

Levelorder = ab

Source: https://www.cise.ufl.edu/~sahni/cop3530/

Binary Tree Construction

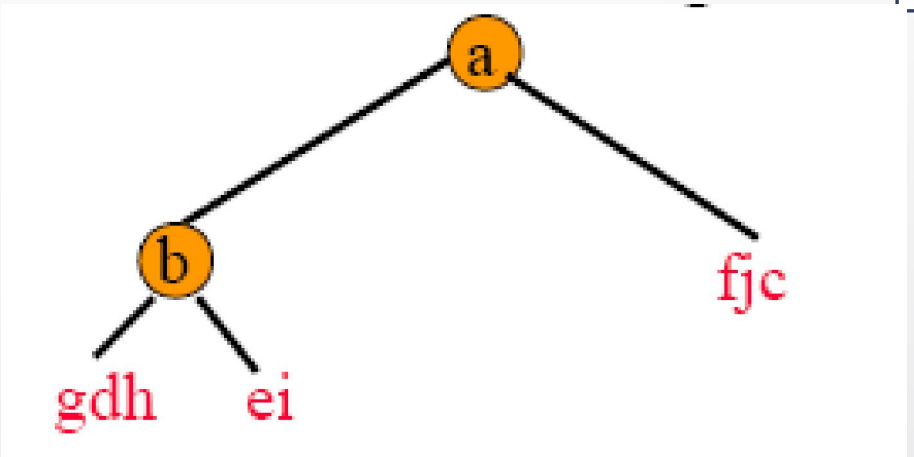
- inorder = g d h b e i a f j c
- preorder = a b d g h e i c f j
- The preorder sequence is traversed from left to right using the inorder to separate left and right subtrees.
- Here in this example, a becomes the root of the tree; gdhbei becomes the left subtree elements; fjc are in the right subtree.

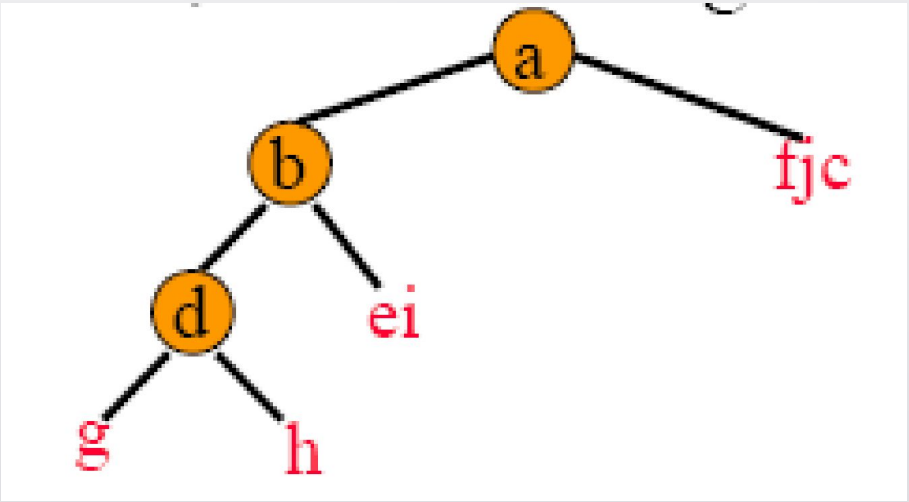


Source: https://www.cise.ufl.edu/~sahni/cop3530/

Binary Tree Construction

- inorder = g d h b e i a f j c
- preorder = a b d g h e i c f j
- b will be the next root
- d will be the next root and so on





Source: https://www.cise.ufl.edu/~sahni/cop3530/

References

- *Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2009) [1990]*
- *Introduction to Algorithms (3rd ed.). MIT Press and McGraw-Hill. ISBN 0-262-03384-4. 1320 pp.*
- Adam Drozdek, Data Structures and Algorithms in C++ (2nd Edition), 2001
- https://www.geeksforgeeks.org/iterative-postorder-traversal/
- https://www.tutorialspoint.com/cplusplus-program-to-perform-inorder-non-recursive-traversal-of-a-given-binary-tree
- https://www.tutorialspoint.com/cplusplus-program-to-perform-preorder-non-recursive-traversal-of-a-given-binary-tree