Experiment no 8

Binary Tree Implementation

OBJECTIVES:

- To learn about the concept of Binary Tree.
- To learn different ways for implementing a Binary Tree.
- To implement binary tree using nodes.

1. BINARY TREE:

Tree represents the nodes connected by edges. It is a non-linear data structure. It has the following properties**:**

- One node is marked as Root node.
- Every node other than the root is associated with one parent node.
- Each node can have an arbiatry number of child nodes.

We create a tree data structure in python by using the concept os node discussed earlier. We designate one node as root node and then add more nodes as child nodes. Below is program to create the root node.

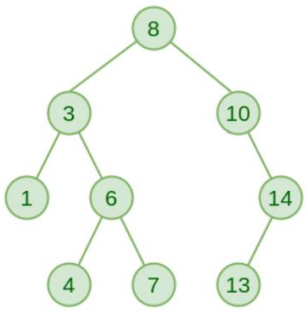


Figure 1: Binary Tree

2. CREATING A NODE:

We just create a Node class and add assign a value to the node. This becomes tree with only a root node.

```
class Node:
   def __init__(self, data):
      self.left = None
      self.right = None
      self.data = data
   def PrintTree(self):
      print(self.data)

root = Node(10)
root.PrintTree()
```

Figure 2: Creating a node

3. INSERTING INTO A TREE:

To insert into a tree we use the same node class created above and add a insert class to it. The insert class compares the value of the node to the parent node and decides to add it as a left node or a right node. Finally the PrintTree class is used to print the tree.

```
class Node:
  def __init__(self, data):
    self.left = None
    self.right = None
    self.data = data

  def insert(self, data):
# Compare the new value with the parent node
    if self.data:
      if data < self.data:
        if self.left is None:
          self.left = Node(data)
        else:
          self.left.insert(data)
      elif data > self.data:
          if self.right is None:
            self.right = Node(data)
          else:
            self.right.insert(data)
    else:
      self.data = data
```

```
# Print the tree
  def PrintTree(self):
    if self.left:
      self.left.PrintTree()
    print( self.data),
    if self.right:
      self.right.PrintTree()

# Use the insert method to add nodes
root = Node(12)
root.insert(6)
root.insert(14)
root.insert(3)
root.PrintTree()
```

Figure 3: Inserting into tree

4. TRAVERSING A TREE:

The tree can be traversed by deciding on a sequence to visit each node. As we can clearly see we can start at a node then visit the left sub-tree first and right sub-tree next. Or we can also visit the right sub-tree first and left sub-tree next. Accordingly, there are different names for these tree traversal methods.

Traversal is a process to visit all the nodes of a tree and may print their values too. Because, all nodes are connected via edges (links) we always start from the root (head) node. That is, we cannot randomly access a node in a tree. There are three ways which we use to traverse a tree.

- In-order Traversal
- Pre-order Traversal
- Post-order Traversal

5. IN-ORDER TRAVERSAL:

In this traversal method, the left subtree is visited first, then the root and later the right sub-tree. We should always remember that every node may represent a subtree itself.
In the below python program, we use the Node class to create place holders for the root node as well as the left and right nodes. Then, we create an insert function to add data to the tree. Finally, the In-order traversal logic is implemented by creating an empty list and adding the left node first followed by the root or parent node.
At last the left node is added to complete the In-order traversal. Please note that this process is repeated for each sub-tree until all the nodes are traversed.

```
class Node:
  def __init__(self, data):
    self.left = None
    self.right = None
    self.data = data
# Insert Node
  def insert(self, data):
    if self.data:
      if data < self.data:
        if self.left is None:
          self.left = Node(data)
        else:
          self.left.insert(data)
      else data > self.data:
        if self.right is None:
          self.right = Node(data)
        else:
          self.right.insert(data)
    else:
      self.data = data
# Print the Tree
  def PrintTree(self):
    if self.left:
      self.left.PrintTree()
    print( self.data),
    if self.right:
      self.right.PrintTree()
```

```
# Inorder traversal
# Left -> Root -> Right
  def inorderTraversal(self, root):
    res = []
    if root:
      res = self.inorderTraversal(root.left)
      res.append(root.data)
      res = res + self.inorderTraversal(root.right)
    return res
root = Node(27)
root.insert(14)
root.insert(35)
root.insert(10)
root.insert(19)
root.insert(31)
root.insert(42)
print(root.inorderTraversal(root))
```

Figure 4: In-Order Traversal

OUTPUT:

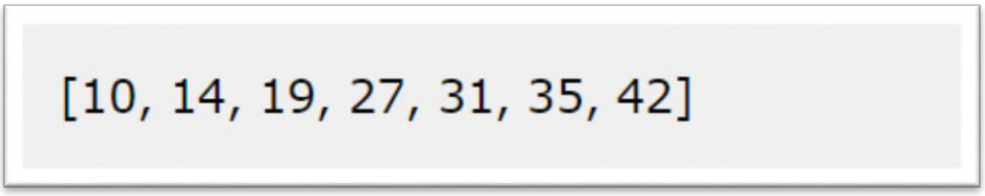


Figure 5: Output of in-order traversal

6. ACTIVITIES:

A1) Write a code for binary tree implementation using node class for a single node.

A2) Write a code for pre-order traversal and the binary tree of your choice.

A3) Write a code for post-order traversal and the binary tree of your choice.

IMPORTANT: All the activities` code should be attached to the manual before summary section.

Lab Exercise and Summary

Summary should cover Introduction, Procedure, Data Analysis and Evaluation.

Student's Signature:_____ **Date:**_____

LABORATORY SKILLS ASSESSMENT (Psychomotor)

Total Marks:100

Criteria (Max Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S< 70%	Level 3 70% ≤ S< 90%	Level 4 90%≤ S ≤100%	Score (S)
Procedural Awareness (30)	Selects inappropriate skills and/or strategies Required by the task.	Selects and applies appropriate skills and/or strategies required by the task with major errors.	Selects and applies the appropriate strategies and/or skills specific to the task without significant errors.	Selects and applies appropriate strategies and/or skills specific to the task without any error.	
Practical Implementation (30)	Makes major critical errors in applying procedural knowledge related to python Binary Tree Implementation	Makes numerous critical errors in applying procedural knowledge related to python Binary Tree Implementation	Makes some non-critical errors in applying procedural knowledge related to python Binary Tree Implementation.	Applies the procedural knowledge in optimized ways related to python Lists, Binary Tree Implementation	
Use of Tool/Equipment (30)	Uses tools, equipment and materials with limited competence.	Uses tools, equipment and materials with some competence.	Uses tools, equipment and materials with considerable competence.	Uses tools, equipment and materials with a high degree of competence.	
Safety (10)	Requires constant reminders to follow safety procedures.	Requires some reminders to follow safety procedures.	Follows safety procedures with only minimal reminders.	Routinely follows safety procedures.	
				Marks Obtained	

Instructor Name:_____ **Sign:** _____

LABORATORY SKILLS ASSESSMENT (Affective)

Total Marks: 40

Criteria (Max. Marks)	Level 1 0% ≤ S < 50%	Level 2 50% ≤ S < 70%	Level 3 70% ≤ S < 90%	Level 4 90% ≤ S ≤ 100%	Score
Introduction (5)	Very little background information provided or information is incorrect	Introduction is brief with some minor mistakes	Introduction is nearly complete, missing some minor points	Introduction complete and well-written; provides all necessary background principles for the experiment	
Procedure (5)	Many stages of the procedure are not entered on the lab report.	Many stages of the procedure are entered on the lab report.	The procedure could be more efficiently designed but most stages of the procedure are entered on the lab report.	The procedure is well designed and all stages of the procedure are entered on the lab report.	
Data Record (10)	Data is brief and missing significant pieces of information.	Data provides some significant information and has few critical mistakes.	Data is almost complete but has some minor mistakes.	Data is complete and relevant. Tables with units are provided. Graphs are labeled. All questions are answered correctly.	
Data Analysis (10)	Data is presented in very unclear manner.	Data is presented in ways that are not clear enough.	Data is presented in ways that can be understood and interpreted.	Data is presented in ways that best facilitate understanding and interpretation.	
Report Quality (10)	Report contains many errors.	Report is somewhat organized with some spelling or grammatical errors.	Report is well organized and cohesive but contains some grammatical errors.	Report is well organized and cohesive and contains no grammatical errors. Presentation seems polished.	
				Marks Obtained	

LABORATORY SKILLS ASSESSMENT (Cognitive)

Total Marks: 10

(If any) **Marks Obtained**	

Instructor's Signature:_____ **Date:**_____