

Practical 1

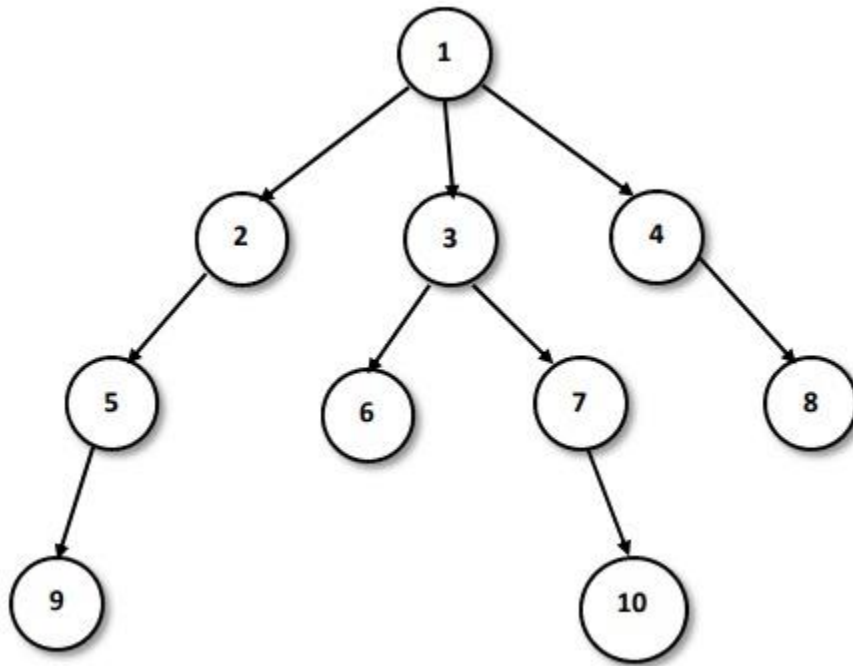
1a) BFS

Aim :Implement BFS treverse on given tree:

Theory :

BFS : Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search when a dead end occurs in any iteration.

Tree :



Code :

```
from IPython.core.display import Image,display
display(Image(url="/images/Practical1tree.jpeg"))
```

```
class TreeNode:
    def __init__(self,data):
        self.data=data

        self.children=[]
        self.parent=None
    def add_child(self,child):
```

```

        child.parent=self
        self.children.append(child)
def get_levels(self):
    level=0
    p=self.parent
    while p:
        level +=1
        p=p.parent
    return level
def print_tree(self, level):
    if self.get_levels() > level:
        return
    spaces = ' ' * self.get_levels() * 3
    prefix = spaces + "|__" if self.parent else ""
    print(prefix + self.data)
    if self.children:
        for child in self.children:
            child.print_tree(level)

```

```

def create_product_tree():

```

```

    root=TreeNode('1')
    node2=TreeNode('2')
    node3=TreeNode('3')
    node4=TreeNode('4')
    node5=TreeNode('5')
    node6=TreeNode('6')
    node7=TreeNode('7')
    node8=TreeNode('8')
    node9=TreeNode('9')
    node10=TreeNode('10')
    node5.add_child(node9)
    node2.add_child(node5)
    root.add_child(node2)
    node7.add_child(node10)
    node3.add_child(node6)
    node3.add_child(node7)
    root.add_child(node3)
    node4.add_child(node8)
    root.add_child(node4)

```

```

return root

```

```

root=create_product_tree()

```

```

display(Image(url="./images/Practical1tree.jpeg"))
root.print_tree(5)

```

The screenshot shows a Jupyter Notebook with the following components:

- EXPLORER:** A sidebar on the left showing a file explorer with folders like 'AI-PRACTICAL', 'datasets', 'docs', 'pdfs', 'words', and 'images'. Files include 'AI_Practical1.pdf', 'Practical1Tree.jpeg', 'tree12.gif', 'Practical1Graph.ipynb', 'Practical1Tree.ipynb', and 'Practical2.ipynb'.
- Code Cell:** Contains the following Python code:


```
display(Image.url="/images/Practical1Tree.jpeg")
root.print_tree(5)
```
- Output:**
 - A binary tree visualization with 10 nodes. The root is 1, which has children 2 and 4. Node 2 has children 5 and 6. Node 5 has child 9. Node 4 has children 7 and 8. Node 7 has child 10.
 - A text-based DFS traversal output:


```
1
  |_2
    |_5
      |_9
    |_6
  |_3
    |_7
      |_10
    |_4
      |_8
```
 - The text "DFS Traverse" is displayed below the output.
- Bottom Bar:** Shows the status bar with "Master*", "Jira: Suryasen", "No active issue", "Bitbucket: Suryasen", "0 0 1", "Jupyter Server: local", "Cell 7 of 12", "Go Live", and "Pretter".

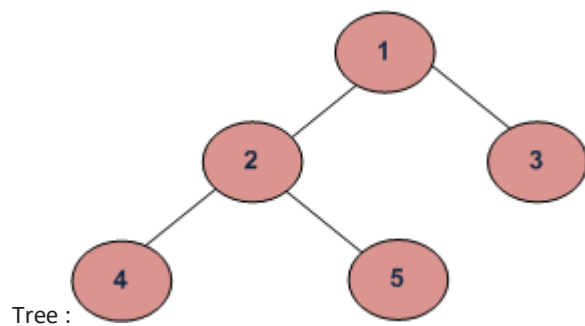
Output :

Code Link : <https://github.com/tonudon86/AI-practicals/blob/master/Practical1Tree.ipynb>

1b)DFS

Aim :Implement DFS on Binary Tree :

Thoery : DFS : Depth First Search (DFS) algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search when a dead end occurs in any iteration.



Code :

```
display(Image(url="./images/tree12.gif"))
```

class Node:

```
    def __init__(self, key):  
        self.left = None  
        self.right = None  
        self.val = key
```

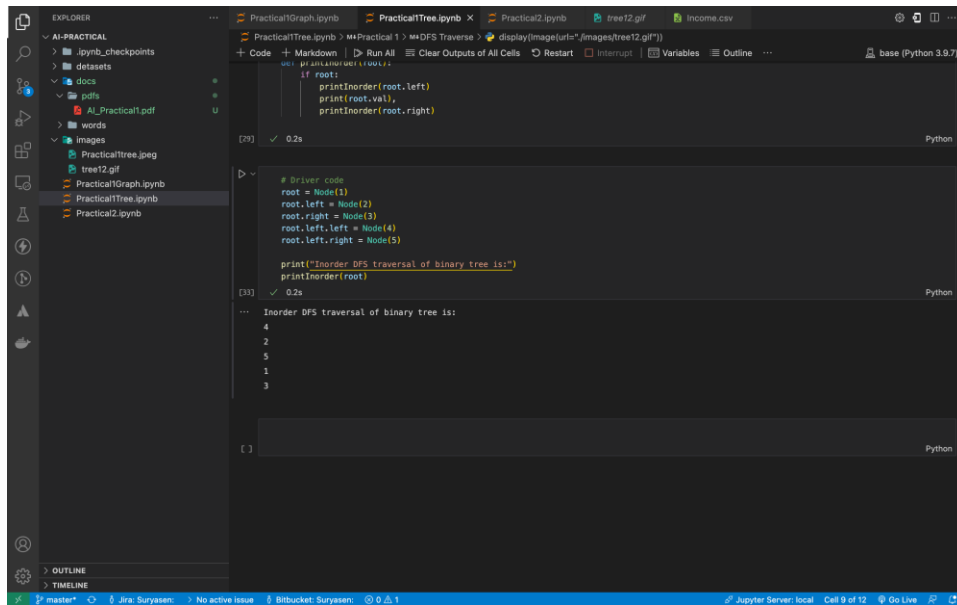
```
def printInorder(root):  
    if root:  
        printInorder(root.left)  
        print(root.val),  
        printInorder(root.right)
```

Driver code

```
root = Node(1)  
root.left = Node(2)  
root.right = Node(3)  
root.left.left = Node(4)  
root.left.right = Node(5)
```

```
print("Inorder DFS traversal of binary tree is:")  
printInorder(root)
```

Output :



The screenshot shows a Jupyter Notebook with the following content:

```
def printInorder(root):  
    if root:  
        printInorder(root.left)  
        print(root.val)  
        printInorder(root.right)
```

[29] ✓ 0.2s Python

```
# Driver code  
root = Node(1)  
root.left = Node(2)  
root.right = Node(3)  
root.left.left = Node(4)  
root.left.right = Node(5)  
  
print("Inorder DFS traversal of binary tree is:")  
printInorder(root)
```

[33] ✓ 0.2s Python

... Inorder DFS traversal of binary tree is:
4
2
5
1
3

[] Python

Code link : <https://github.com/tonudon86/AI-practicals/blob/master/Practical1Tree.ipynb>