**Breadth First Search**

**Program-:**

class Node:

def \_\_init\_\_(self, value):

self.left = None

self.right = None

self.value = value

def BFS(root, search\_value):

if root is None:

return

queue = []

queue.append(root)

while queue:

node = queue.pop(0)

print(node.value, end=' ')

if node.value == search\_value:

print("\nElement found!")

return

if node.left is not None:

queue.append(node.left)

if node.right is not None:

queue.append(node.right)

if \_\_name\_\_ == '\_\_main\_\_':

root = Node(1)

root.left = Node(2)

root.right = Node(3)

root.left.left = Node(4)

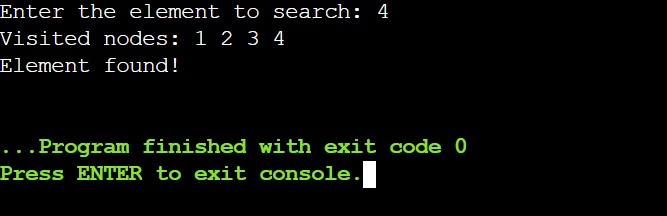
root.left.right = Node(5)

search\_value = int(input("Enter the element to search: "))

print("Visited nodes: ", end='')

BFS(root, search\_value)

**Output -:**

****

**Depth First Search**

**Program-:**

class TreeNode:

def \_\_init\_\_(self, val):

self.val = val

self.left = None

self.right = None

def dfs(root, goal, visited=[]):

if root is None:

return False

visited.append(root.val)

if root.val == goal:

print(visited)

return True

if dfs(root.left, goal, visited):

return True

if dfs(root.right, goal, visited):

return True

return False

# Example usage

root = TreeNode(1)

root.left = TreeNode(2)

root.right = TreeNode(3)

root.left.left = TreeNode(4)

root.left.right = TreeNode(5)

goal = int(input("Enter the node to search for: "))

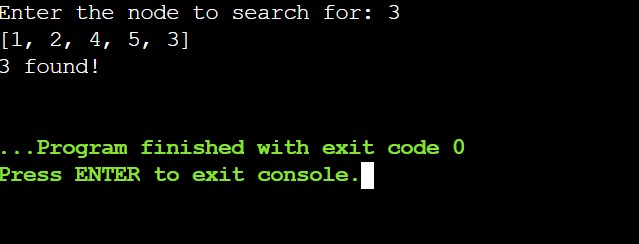
if dfs(root, goal):

print(f"{goal} found!")

else:

print(f"{goal} not found!")

**output -:**

****