SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
(Deemed to be **University** *u/s 3 of UGC Act, 1956)*

18CSC305J – ARTIFICIAL INTELLIGENCE LAB

Exp-4: Implementation of BFS and DFS in an Application – Range Sum of Binary Search Tree

Submitted by-
Name:- Anannya P. Neog
Reg. No. :- RA1911003010367
Course :- Btech
Section :- F1
Branch:- Computer Science Engineering
Sem:- 6th Sem

AI LAB Ex – 4:- Implementation of BFS and DFS in an Application – Range Sum of Binary Search Tree

Team Members:

- ✓ Richa - 357
- ✓ Anannya - 367
- ✓ Pushan - 371
- ✓ Ankit - 372
- ✓ Tanay - 377

Aim:

To implement BFS and DFS in application, e.g.- Range Sum of Binary Search Tree

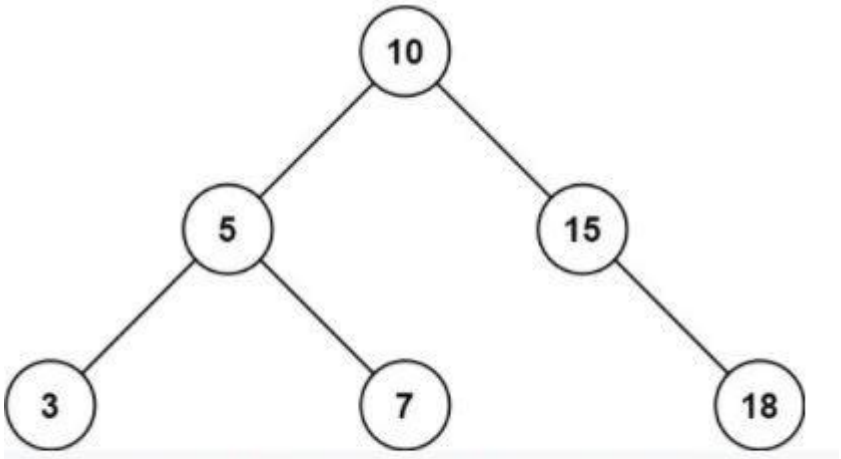
Objective:

Given the root node of a binary search tree, return the sum of values of all nodes with a value in the range [low, high] using depth first and then breadth first search.

Constraints:

- The number of nodes in the tree is in the range [1, 2 * 104].
- 1 <= Node.val <= 105
- 1 <= low <= high <= 105
- All Node.value are unique.

Example:




```
Input: root = [10,5,15,3,7,null,18], low = 7, high = 15
Output: 32
```

Procedure/Algorithm:

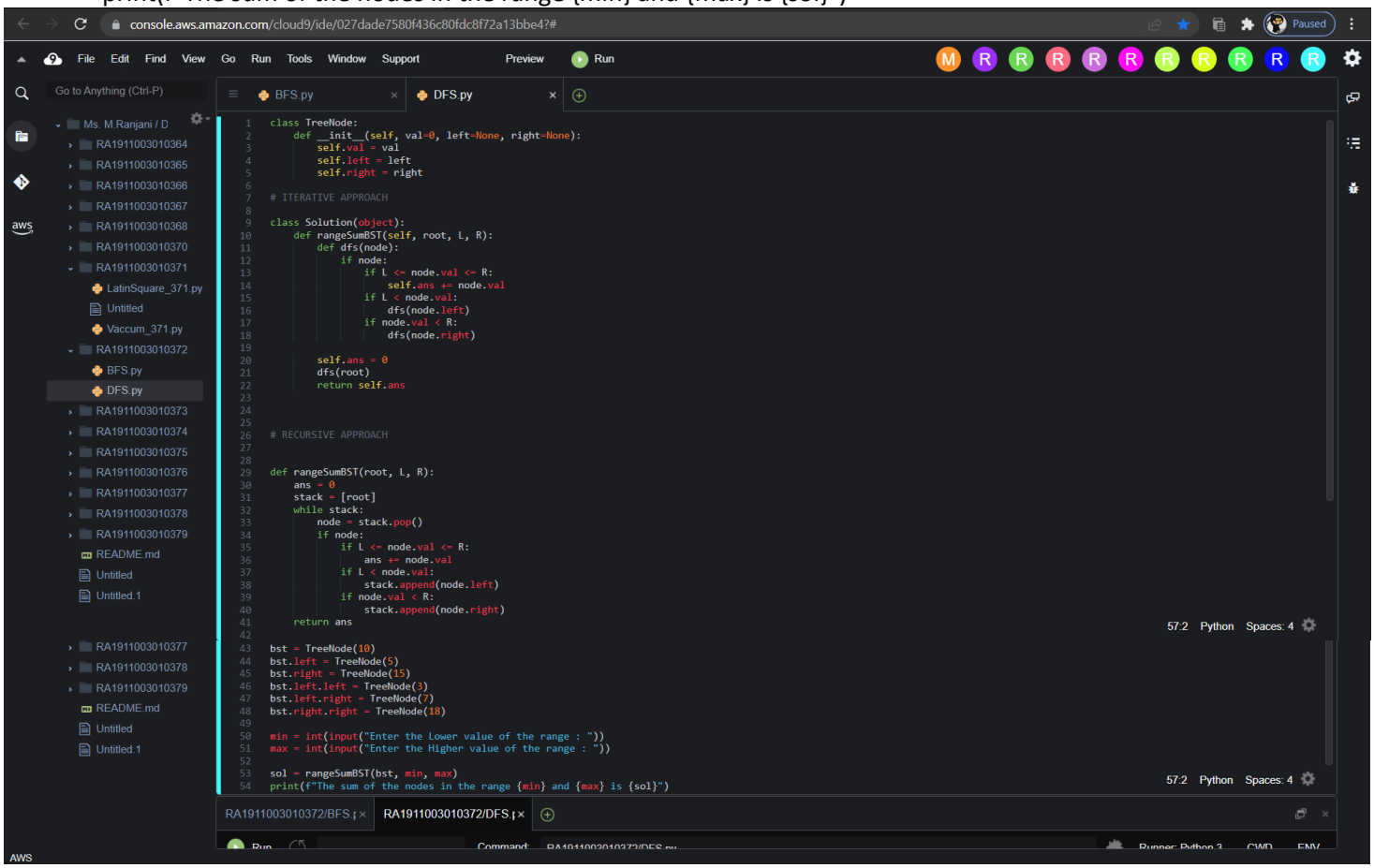
1. We traverse the tree using a depth first search.
2. If node.value falls outside the range [L, R], (for example node.val < L), then we know that only the right branch could have nodes with value inside [L, R].
3. We showcase two implementations - one using a recursive algorithm, and one using an iterative one.
4. Time Complexity: O(N)O(N), where NN is the number of nodes in the tree.
5. Space Complexity: O(N)O(N)
6. For the recursive implementation, the recursion will consume additional space in the function call stack. In the worst case, the tree is of chain shape, and we will reach all the way down to the leaf node.
7. For the iterative implementation, essentially we are doing a BFS (Breadth-First Search) traversal, where the stack will contain no more than two levels of the nodes. The maximal number of nodes in a binary tree is N/2.
8. Therefore, the maximal space needed for the stack would be O(N)O(N).

Codes:

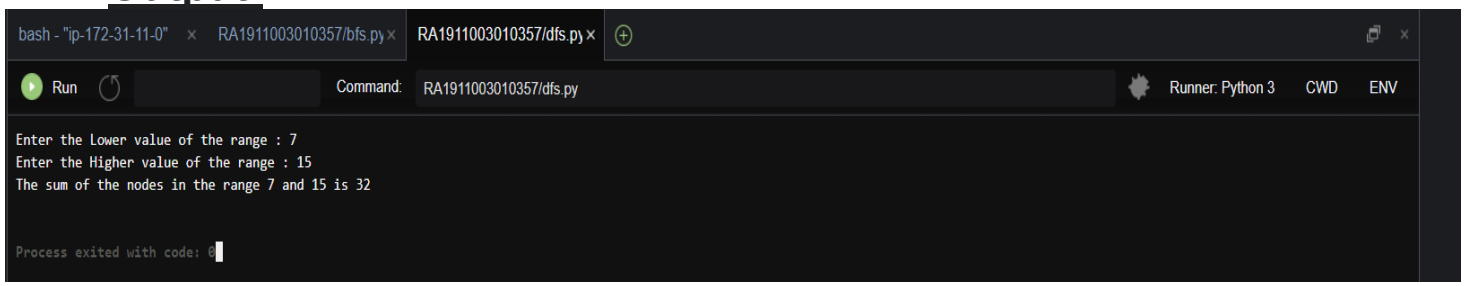
DFS:

```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

# ITERATIVE APPROACH

class Solution(object):
    def rangeSumBST(self, root, L, R):
        def dfs(node):
            if node:
                if L <= node.val <= R:
                    self.ans += node.val
                if L < node.val:
                    dfs(node.left)
                if node.val < R:
                    dfs(node.right)

        self.ans = 0
        dfs(root)
        return self.ans

# RECURSIVE APPROACH

def rangeSumBST(root, L, R):
    ans = 0
    stack = [root]
    while stack:
        node = stack.pop()
        if node:
```

```
            if L <= node.val <= R:
                ans += node.val
            if L < node.val:
                stack.append(node.left)
            if node.val < R:
                stack.append(node.right)
        return ans

bst = TreeNode(10)
bst.left = TreeNode(5)
bst.right = TreeNode(15)
bst.left.left = TreeNode(3)
bst.left.right = TreeNode(7)
bst.right.right = TreeNode(18)

min = int(input("Enter the Lower value of the range : "))
max = int(input("Enter the Higher value of the range : "))

sol = rangeSumBST(bst, min, max)
print(f"The sum of the nodes in the range {min} and {max} is {sol}")
```



Output:



BFS:

```
from collections import deque

# Class for node of the Tree
class Node:
    def __init__(self,v):
        self.val = v
        self.left = None
        self.right = None

# Function to perform level order
# traversal on the Tree and
# calculate the required sum
def rangeSumBST(root, low, high):
    sum = 0

    # Base Case
    if (root == None):
        return 0

    # Stores the nodes while
    # performing level order traversal
    q = deque()

    # Push the root node
    # into the queue
    q.append(root)

    # Iterate until queue is empty
    while (len(q) > 0):

        # Stores the front
        # node of the queue
        curr = q.popleft()
        # q.pop()

        # If the value of the node
        # lies in the given range
        if (curr.val >= low
            and curr.val <= high):

            # Add it to sum
            sum += curr.val

        # If the left child is
        # not NULL and exceeds low
        if (curr.left != None
        and curr.val > low):

            # Insert into queue
            q.append(curr.left)

        # If the right child is not
        # NULL and exceeds low
        if (curr.right != None
            and curr.val < high):

            # Insert into queue
            q.append(curr.right)
```

```
    # Return the resultant sum
    return sum

# Function to insert a new node
# into the Binary Search Tree
def insert(node, data):

    # Base Case
    if (node == None):
        return Node(data)

    # If the data is less than the
    # value of the current node
    if (data <= node.val):

        # Recur for left subtree
        node.left = insert(node.left, data)
    # Otherwise
    else:
        # Recur for the right subtree
        node.right = insert(node.right, data)

    # Return the node
    return node

# Driver Code
if __name__ == '__main__':
    # /* Let us create following BST
    #      10
    #     / \
    #    5   15
    #   / \    \
    #  3   7    18  */
    root = None
    root = insert(root, 10)
    root = insert(root, 5)
    root = insert(root, 15)
    root = insert(root, 3)
    root = insert(root, 7)
    root = insert(root, 18)

    L, R = 7, 15
    print(rangeSumBST(root, L, R))
```

https://console.aws.amazon.com/cloud9/ide/027dade7580f436c80fdc8f72a13bbe4?#

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

Ms. M.Ranjani / D

RA1911003010364
RA1911003010365
RA1911003010366
RA1911003010367
Exp2(Vacuum Clean
Exp3(Latin Square F
Exp4(bfs).py
Exp4(dfs).py
RA1911003010368
RA1911003010370
RA1911003010371
RA1911003010372
BFS.py
DFS.py
RA1911003010373
RA1911003010374
RA1911003010375
RA1911003010376
RA1911003010377
RA1911003010378
RA1911003010379
README.md
Untitled
Untitled.1

Exp4(bfs).py

```python
from collections import deque

# Class for node of the Tree
class Node:
    def __init__(self,v):
        self.val = v
        self.left = None
        self.right = None

# Function to perform level order
# traversal on the Tree and
# calculate the required sum
def rangeSumBST(root, low, high):
    sum = 0

    # Base Case
    if (root == None):
        return 0

    # Stores the nodes while
    # performing level order traversal
    q = deque()

    # Push the root node
    # into the queue
    q.append(root)

    # Iterate until queue is empty
    while (len(q) > 0):

        # Stores the front
        # node of the queue
        curr = q.popleft()
        # q.pop()
```

68:21 Python Spaces: 4

bash - "ip-172-31-0-40" Immediate (Javascript (br RA1911003010367/Exp4\

Run Command: RA1911003010367/Exp4\(bfs\).py Runner: Python 3 CWD ENV

AWS

https://console.aws.amazon.com/cloud9/ide/027dade7580f436c80fdc8f72a13bbe4?#

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

Ms. M.Ranjani / D

RA1911003010364
RA1911003010365
RA1911003010366
RA1911003010367
Exp2(Vacuum Clean
Exp3(Latin Square F
Exp4(bfs).py
Exp4(dfs).py
RA1911003010368
RA1911003010370
RA1911003010371
RA1911003010372
BFS.py
DFS.py
RA1911003010373
RA1911003010374
RA1911003010375
RA1911003010376
RA1911003010377
RA1911003010378
RA1911003010379
README.md
Untitled
Untitled.1

Exp4(bfs).py

```python
        # If the value of the node
        # lies in the given range
        if (curr.val >= low
            and curr.val <= high):

            # Add it to sum
            sum += curr.val

        # If the left child is
        # not NULL and exceeds low
        if (curr.left != None
        and curr.val > low):

            # Insert into queue
            q.append(curr.left)

        # If the right child is not
        # NULL and exceeds low
        if (curr.right != None
            and curr.val < high):

            # Insert into queue
            q.append(curr.right)

    # Return the resultant sum
    return sum

# Function to insert a new node
# into the Binary Search Tree
def insert(node, data):

    # Base Case
    if (node == None):
        return Node(data)
```

68:21 Python Spaces: 4

bash - "ip-172-31-0-40" Immediate (Javascript (br RA1911003010367/Exp4\

Run Command: RA1911003010367/Exp4\(bfs\).py Runner: Python 3 CWD ENV

AWS

Output:

RA1911003010372/BFS.p RA1911003010372/DFS.p

Run Command: RA1911003010372/BFS.py Runner: Python 3 CWD ENV

```
32

Process exited with code: 0
```