



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

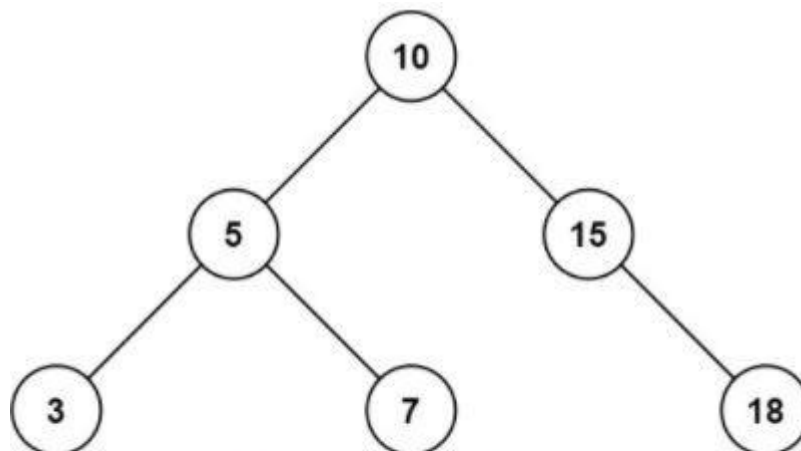
Objective:

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 $[1, 2 * 10^4]$.
- $1 \leq \text{Node.val} \leq 105$
- $1 \leq \text{low} \leq \text{high} \leq 105$
- 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 N 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}")

```

```

1 class TreeNode:
2     def __init__(self, val=0, left=None, right=None):
3         self.val = val
4         self.left = left
5         self.right = right
6
7 # ITERATIVE APPROACH
8
9 class Solution(object):
10     def rangeSumBST(self, root, L, R):
11         def dfs(node):
12             if node:
13                 if L <= node.val <= R:
14                     self.ans += node.val
15                 if L < node.val:
16                     dfs(node.left)
17                 if node.val < R:
18                     dfs(node.right)
19
20         self.ans = 0
21         dfs(root)
22         return self.ans
23
24
25 # RECURSIVE APPROACH
26
27 def rangeSumBST(root, L, R):
28     ans = 0
29     stack = [root]
30     while stack:
31         node = stack.pop()
32         if node:
33             if L <= node.val <= R:
34                 ans += node.val
35             if L < node.val:
36                 stack.append(node.left)
37             if node.val < R:
38                 stack.append(node.right)
39
40     return ans
41
42
43 bst = TreeNode(10)
44 bst.left = TreeNode(5)
45 bst.right = TreeNode(15)
46 bst.left.left = TreeNode(3)
47 bst.left.right = TreeNode(7)
48 bst.right.right = TreeNode(18)
49
50 min = int(input("Enter the Lower value of the range : "))
51 max = int(input("Enter the Higher value of the range : "))
52
53 sol = rangeSumBST(bst, min, max)
54 print(f"The sum of the nodes in the range {min} and {max} is {sol}")

```

Output:

```

bash - "ip-172-31-11-0" x RA1911003010357/bfs.py x RA1911003010357/dfs.py x
Run Command: RA1911003010357/dfs.py Runner: Python 3 CWD ENV

Enter the Lower value of the range : 7
Enter the Higher value of the range : 15
The sum of the nodes in the range 7 and 15 is 32

Process exited with code: 0

```

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)

Exp4(bfs).py x

1 from collections import deque
2
3 # Class for node of the Tree
4 class Node:
5     def __init__(self,v):
6         self.val = v
7         self.left = None
8         self.right = None
9
10 # Function to perform level order
11 # traversal on the Tree and
12 # calculate the required sum
13 def rangeSumBST(root, low, high):
14     sum = 0
15
16     # Base Case
17     if (root == None):
18         return 0
19
20     # Stores the nodes while
21     # performing level order traversal
22     q = deque()
23
24     # Push the root node
25     # into the queue
26     q.append(root)
27
28     # Iterate until queue is empty
29     while (len(q) > 0):
30
31         # Stores the front
32         # node of the queue
33         curr = q.popleft()
34         # q.pop()
35
68.21 Python Spaces: 4

bash - "ip-172-31-0-40" x Immediate (Javascript (br x RA1911003010367/Exp4) x
Run Command: RA1911003010367/Exp4(bfs).py Runner: Python 3 CWD ENV
```

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

File Edit Find View Go Run Tools Window Support Preview Run

Go to Anything (Ctrl-P)

Exp4(bfs).py x

36 # If the value of the node
37 # lies in the given range
38 if (curr.val >= low
39     and curr.val <= high):
40
41     # Add it to sum
42     sum += curr.val
43
44 # If the left child is
45 # not NULL and exceeds low
46 if (curr.left != None
47     and curr.val > low):
48
49     # Insert into queue
50     q.append(curr.left)
51
52 # If the right child is not
53 # NULL and exceeds low
54 if (curr.right != None
55     and curr.val < high):
56
57     # Insert into queue
58     q.append(curr.right)
59
60 # Return the resultant sum
61 return sum
62
63 # Function to insert a new node
64 # into the Binary Search Tree
65 def insert(node, data):
66
67     # Base Case
68     if (node == None):
69         return Node(data)
70
68.21 Python Spaces: 4

bash - "ip-172-31-0-40" x Immediate (Javascript (br x RA1911003010367/Exp4) x
Run Command: RA1911003010367/Exp4(bfs).py Runner: Python 3 CWD ENV
```

Output:

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

32

Process exited with code: 0
```