OS: debian-10 (Dockerized, Host: macOS Big Sur), Compiler: g++ (Debian 8.3.0-6) 8.3.0

ASSIGNMENT- 02

Advanced Data Structures and Algorithms

1. INSERT

```
1. insert(value)
2.     rbtree.insert(value)
```

1.1. Average Time Complexity

RBTree insert has O(log n) average time complexity.

1.2. Worst Case Time Complexity

As RBTree's are balanced, worst case time complexity for insert is also O(log n).

2. DELETE

```
1. delete(value)
2.     rbtree.deleteByValue(value)
```

2.1. Average Time Complexity

RBTree delete has O(log n) average time complexity.

2.2. Worst Case Time Complexity

As RBTree's are balanced, worst case time complexity for delete is also O(log n).

3. QUERY

```
1. RBtree::inorderList(node, array, index)
2.     if(node == null)
3.         return index
4.     index = inorderList(node->left, array, index)
5.     array[index++] = node
6.     index = inorderList(node->right, array, index)
7.     return index
8.
9. RBtree::getSortedList()
10.    array = new array[tree.count]
11.    index = 0
12.    inorderList(tree.root, array, index)
13.    return array
14.
15.  query(a, b)
16.    targets = new(rbtree)
17.    sortedList = rbtree.getSortedList()
18.    count = rbtree.count()
19.    if(abs(b-a)<count)
20.        for(i = a to b)
21.            xIndex = 0, yIndex = count-1
22.            while(xIndex < yIndex)
23.                sum = sortedList[xIndex] + sortedList[yIndex]
24.                if(sum ==i)
25.                   targets.insertUnique(i)
26.                   break;
```

```
27.              else if(sum < i)
28.                xIndex++
29.              else
30.                yIndex—
31.    else
32.        for(xIndex = 0 to count-1)
33.           for(yIndex = 0 to count-1)
34.              sum = sortedList[xIndex] + sortedList[yIndex]
35.              if(sum>=a and sum<=b)
36.                 targets.insertUnique(i)
37.    return targets.count()
```

3.1. Time Complexity for different functions

Function	Average Time Complexity	Worst Time Complexity
getSortedList	O(n)	O(n)
insertUnique	O(log n)	O(log n)
count	O(1)	O(1)

T(getSortedList) = T(inorderTraversal) = O(n)

3.2. Average Time Complexity

Condition : b-a < n

T(query) = T(getSortedList) + T(count) + (b-a) * n * T(insertUnique) + T(count)
= O(n) + O(1) + (b-a)*n*O(log(b-a)) + O(1)
= O(n) + (b-a)*n*log(b-a)
= O((b-a)*n*(log(b-a)))

3.3. Worst Case Time Complexity

Condition : b-a>n

T(query) = T(getSortedList) + T(count) + n * n + T(count)
= O(n) + O(1) + O(n * n) + O(1)
= $O(n^2)$

PLAGIARISM STATEMENT

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honour violations by other students if I become aware of it.

Name: Bhavishya Sharma
Roll No: CS20MTECH12006

Signature: