Birla Institute of Technology & Science, Pilani

2nd Semester 2016-17 - CS F211 – Data Structures and Algorithms

Lab 9 – 30th March 2017

Topics – AVL Trees

Problem 1

Implement an AVL Tree with the following operations it supports -

- *Find*: Finds whether a given key *k* is present a given AVL Tree
- *Add*: Inserts a given key *k* into the given AVL Tree. This operation needs you to appropriately implement the rotate operations which happen during the insertion phase. For doing this, you must keep an additional field at each and every node which can contain 0, -1 or 1 to indicate height balance of the node.
- *Construct*: Calls insert operation repeatedly for a given list of elements one after the other to construct a fresh AVL Tree.
- *Measure*: Measure the time spend during rotations using the time function.

You can use the following table to design your functions. You can also refer to the sample input and output case provided.

Key	Function	Input Format	Description
0	*readData*	0 *N* k_1 k_2 k_3 ... k_N	Reads the next *N* lines containing *N* keys and stores them into an array of size *N* called *Arr*.
1	*add*	1 *k*	Inserts the given key *k* into the AVL Tree.
2	*construct*	2	Initializes an empty AVL tree and inserts all the elements of *Arr* into it, in the same order as they were inserted into *Arr*.
3	*find*	3 *k*	Finds whether a given key *k* exists in the AVL tree. Prints 1 if found, 0 otherwise.
4	*measure*	4	Measures the total time taken in rotations during the construction and prints it.

Sample input and output - 1

Sample Input	Sample Output

0 4
40
23
35
43
2
3 33
3 35
4
-1
	0
	1
	<total time>