Data Structures & Algorithms for Problem Solving

Assignment 2

Deadline: 8th October 2020 , 11.55 pm

IMPORTANT POINTS

1. Languages Allowed : C/C++

2. **C++ STL are not allowed in any of the questions.**

Any case of plagiarism will lead to 0 in assignment or "F" in the course.

Problem 1: AVL Tree

AIM :To have end to end knowledge of the balanced binary search tree and how it can be used to solve a wide range of problems efficiently.

TASK: Implement AVL Tree with Following Operations.

Operations to implement:

	Operations	**Complexity**
1	Insertion	O(log N)
2	Deletion	O(log N)
3	Search	O(log N)
4	Count occurrences of element	O(log N)
5	lower_bound	O(log N)
6	upper_bound	O(log N)
7	Closest Element to some value	O(log N)
8	K-th largest element	O(log N)
9	Count the number of elements in the tree whose values fall into a given range.	O(log N)

IMPORTANT POINTS:

1. Implement it with **class** or **struct**. It should be **generic**.

2. **Duplicates** are allowed. (We know that AVL tree doesn't have duplicates but in this task you have to handle it.)

3. For **strings,** you can simply compare them but for **Class data type,** you have to pass the comparator object so that you can compare two objects.

Parameter to Judge: Time and space complexity.

References: https://en.wikipedia.org/wiki/AVL_tree

Problem 2: Hashing

Task: Implement an Unordered Map.

Aim: To learn how Hashing works and importance of Hash Functions. Also look how Universal Hashing is implemented.

Parameter to Judge: Time and space complexity.

Hashing should be efficient and appropriate reasons must be given on choice of hash function.

Functions to implement:

1. **insert(key, value)** – insert key value pair.
2. **erase(key)** – erase if key is present otherwise do nothing.
3. **find(key)** – returns true or false.
4. **map[key]** – returns the value mapped to key.

Note: Unordered Map should be generic.

References:

https://en.wikipedia.org/wiki/Hash_function

https://en.wikipedia.org/wiki/Universal_hashing

Problem 3: Ordered Map

Task : Implement an Ordered Map which supports the given Operations

Operations:

1. **insert(key, value)** – Insert the key value pair. If the key is already present, update it's current value. [O(log n)]

2. **erase(key)** – Erase the given key from the map if it is present. [O(log n)]

3. **find(key)** – returns true if key was found else return 0. [O(logn)]

4. **map_obj[key]** (subscript operator) – Access the element with the given key(if it is present in the map). Also, you should be able to modify the value at this key using this operator. If the value is not present, then insert this key with it's corresponding assigned value. [O(log n)]

5. **size()** – returns the number of keys present in the Map. [O(1)]

6. **clear()** – removes all the elements from the Map. [O(n)]

Note:

I. You should implement this Ordered Map in a class or struct.
II. Duplicates are not allowed.
III. All the operations should meet their expected time complexity.
IV. You can add constructors and destructors as required.

Parameters to Judge : Correctness, Space and Time Complexity

Note : For all the questions, accuracy will be tested on the basis of test cases passed which will be provided during evaluation.