Data Structures & Algorithms for Problem Solving
Assignment-2

Deadline: 10th October 2021, 11:55 pm

Important Points:

1. Only C/C++ is allowed.
2. Submission Format: <RollNo>_<Question_No>.cpp
   Ex: For question 1, 2020201001_Q1.cpp.
   Copy all the codes in a folder with folder name as your roll number and submit the zip file in moodle.
   Ex: 2020201001_A2.zip
   Note: All those submissions which are not in the specified format or submitted after the deadline will be awarded 0 in assignment.
3. C++ STL is **not allowed** for any of the questions. So #include <bits/stdc++.h> is not allowed.

Any case of plagiarism will lead to a 0 in the 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(logN)
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.

4. For **strings**, you need not implement the **Closest Element** operation.

Parameter to Judge: Time and space complexity.

Format to submit: RollNo_Q1.cpp

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.

Using the unordered_map implemented solve the following question:

Find count of distinct elements in every sub-array of size k

Note: Unordered Map should be generic.

Format to submit: RollNo_Q2.cpp

References:

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

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

Problem 3:

Statement: Implementation of deque.

What is deque?

- Deque is the same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container
- They support insertion and Deletion from both ends in amortized constant time.
- Inserting and erasing in the middle is linear in time.

What is expected as solution?

The C++ standard specifies that a legal (i.e., standard-conforming) implementation of deque must satisfy the following performance requirements:

- deque() - initialize a blank deque. -O(1)
- deque(n,x) - initialize a deque of length n with all values as x. -O(n)
- push_back(x) - append data x at the end. -O(1)
- pop_back() - erase data at the end. -O(1)
- push_front(x) - append data x at the beginning. O(1)
- pop_front() - erase data at the beginning. O(1)
- front() - returns the first element(value) in the deque. O(1)
- back() - returns the last element(value) in the deque. O(1)
- empty() - returns true if deque is empty else returns false. O(1)
- size() - returns the current size of deque. O(1)
- resize(x, d) - changes the size dynamically. If the new size is greater than the current size of the deque, then fill the empty space with the default value d. O(n)
- clear() - remove all elements of deque.O(1)
- D[n] - returns the nth element of the deque. O(1)

Evaluation parameters: Accuracy of operations and performance.

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