Lab 12 – 09-12-2024 (One Major Task Only)

For this task, for your help code of AVL and code to measure time in general is available in the drive.

The task involves managing player records by:

Creating player records with specific constraints using a class structure. Generating 10,000 random player records and storing them in a vector. Constructing four AVL trees with different keys:

- Total Runs
- Average
- Number of Fours
- Number of Sixes

Comparing query performance on vector and AVL trees for 1,000 random queries. The steps to accomplish the Task:

1. **Define** the Player Structure: The Player structure/class will contain:

    Static Member: count to assign unique Player Number.

    Data Members:

        playerNumber: Unique identifier for each player.

        matches: Random between 1 and 200.

        average: Random float between 5 and 60.

        totalRuns: Integer value of matches * average.

        notOut: Random between 0 and 50% of matches.

        highestScore: Random greater than or equal to average and less than 200.

        fours: Random between 0 and 50% of totalRuns / 4.

        sixes: Random between 0 and 50% of (totalRuns - runs_by_fours) / 6.

    Constructor: Initializes the data members while adhering to the constraints.

2. **Generate** 10,000 Random Records: Use the Player constructor to populate a std::vector<Player> with 10,000 random player records.

3. **Create** AVL Trees: Implement the AVL tree with nodes storing a key (based on specific fields) and the index of the player in the vector. Create four AVL trees:

    AVL Tree 1: Total Runs as the key.
    AVL Tree 2: Average as the key.
    AVL Tree 3: Number of Fours as the key.
    AVL Tree 4: Number of Sixes as the key.

Insert each player's data into the AVL trees while maintaining the balanced property.

4. **Perform** Query Comparisons: For each of the four keys (Total Runs, Average, Number of Fours, Number of Sixes):

Query on Vector: Perform 1,000 random queries by iterating through the vector to find a matching record.
Query on AVL Tree: Perform 1,000 random queries by searching the AVL tree for the same key.

5. **Measure** Query Times: Use a high-resolution timer to record the time taken for each query set (Vector vs. AVL Tree). Compare the query performance for vector and AVL trees.