Assignment 03
Data Structures CS218

Note: The submission deadline is 12th June 2020. In case of plagiarism in any question, straight zero marks will be assigned for the complete assignment.

Total Points: 30

Question 01:

Quicksort algorithm is used to sort the data elements. The time complexity of this algorithm depends upon the selection of pivot element. Henceforth, there are different techniques of choosing pivot. Your task is to implement quick sort algorithm using the following techniques for pivot selection:

a) Select the middle element as pivot element
b) Select the median as pivot element
c) Select any random number as pivot element
d) Select the first element as pivot element

To do:

- Use a random function to generate 100 random data elements
- Select the pivot element as mentioned above.
    - Apply quick sort algorithm if the data to be sorted is greater than 15. Else apply insertion sort.
- Use the time library to estimate the run time of quicksort for each technique of pivot selection.
- Analyze your results and conclude that which of the pivot selection technique is a good choice to avoid worst case.

Question 02:

AVL trees are famous for its height balancing nature. The prerequisite of an AVL tree is a binary search tree (BST). Your task is to create a BST of the following data elements.

78, 61, 57, 55, 27, 24, 21, 18, 15, 13, 10, 7, 5, 3, 1

Once the BST is created, do the following tasks:

- Compute the balance factor of each node.
- Starting from the bottom, balance each violated node (if the balance factor is greater than 1, perform rotations).
- Show step by step. (No marks will be assigned for direct conversion)
- Show the final tree in which each node must be balanced.

- Perform in order, post order, and pre order traversal
- Give some suitable applications of AVL tree
- Only dry run is required for question 02. No implementation is needed.

Question 03:

Hash tables are used for efficient searching. This efficiency highly depends upon the choice of hash function. The best hash functions are those in which there are less or no collisions. On the other hand, we also consider the simplicity of hash function so that it's overhead won't downgrade the performance of searching. Henceforth, there is always a chance of collision. Several collision resolution techniques are available in the literature. Your task is to perform separate chaining for collision resolution.

To do:

- Insert the following data elements using the given hash function:

  10, 15, 20, 25, 30, 35, 43, 45, 90, 87, 65, 98, 76, 70, 32, 43, 56, 97

  Hash function: Key mod size_of_table
- If the load factor of any index is greater than 3, double the size of hash table. Reinsert all the elements using modified hash function.
- Show the dry run as well implementation of question 03

Good Luck ☺