**BST Report – Assignment 3**

**Binary Search tree**

Binary Search Tree is specially used for data storage purpose. Root, Child, leaf, subtree are few of the key elements of the tree structure.

Basic functions may include:

* Insert
* Search
* Delete
* Pre order Traversal
* Post Order Traversal
* In order Traversal

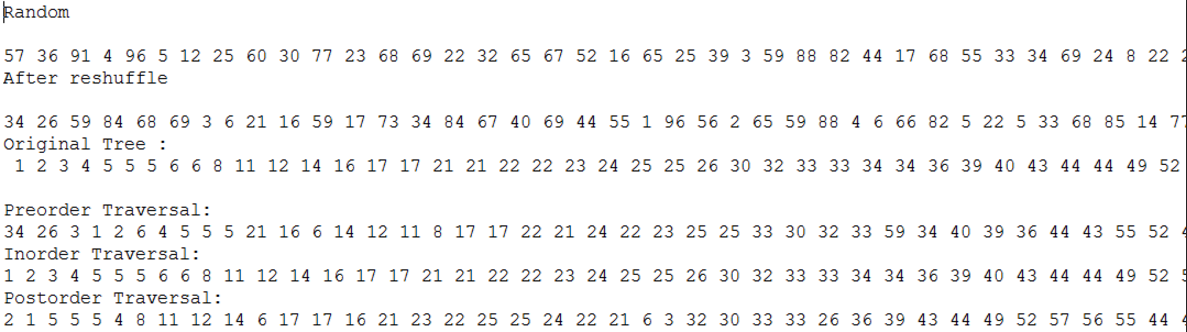
1. Search has an average of O(log n) and Worst case of O(n).
2. Insert has an average of O(log n) and worst case of O(n).
3. Delete has an average of O(log n) and worst case of O(n).

Functions used in the program:

1. Find/Search
2. Delete
3. Knuth Shuffle algorithm
4. Random generator
5. Pre Order Traversal
6. Post Order Traversal
7. In order Traversal
8. Height Calculations

The snapshot for an output run is shown below which follows the steps:

* Creates Random generation with a limit n<100
* Implements Knuth Shuffle Algorithm
* Prints the BST
* Prints the pre order traversal
* Print the in order traversal
* Prints the post order traversal
* Calculates the height.



**Readings:**

Knuth Reshuffle Algorithm:

Also called as Fisher – Yates shuffle which is an algorithm for random generation of a set with limits.

It exhaustively determines the next element until all elements are visited. Observed most of the application for shuffling card deck.

In Order Traversal:

In In-order traversal, the left subtree is visited, followed by the root and then the right.

Pre Order Traversal:

In Pre order traversal, root node is considered first followed by the left subtree and then right subtree.

Post Order Traversal:

In post order traversal, left subtree is considered first, followed by the right subtree and then root.

**Height Conclusions and Observations:**

* A mixed sequence of insertion and deletion, the height of the tree approaches square root of n which is much faster than log n while it grows.
* Search: In search algorithm, farthest left root is considered from the root, the time is proportional to the height. So, average case is O(log n) and O(n) is the worst.
* Insertion: Average case is O(log *n*) , but O(*n*) time in the worst case.
* Traversal: Traversal requires O(n) time, since it must visit every node.
* The height of a complete binary tree of size N is logN
* So, the height of BST has to be logarithmic.
* The average case cost after N inserts is N and worst case cost is 1.39 lgN
* An approximated logarithmic graph of multiple runs -10,50,100,500 vs height. 