Project Report Submitted By,

Tanvi Patil(tsp130130)

Pradnya Mohite(pam130430)

- ## Abstract

This report includes performance evaluation results of Binary Search Tree (BST), AVL Tree and Red Black Tree (RB). Performance is evaluated based on different inputs given.

- ## Problem statement

This problem has two parts (Part a and Part b) mentioned below:

Part a:

Assume that keys and values are long integers. The dictionary ADT is defined on a set of (key,value) pairs, where the keys are totally ordered. The following operations are defined:

a. **Insert(k,v):**
   insert a new entry with key k, value v. If a key with key k already exists, its value is replaced by v. Insert returns 1 if key k is new, and 0 if it already existed in the dictionary.
b. **Find(k):**
   return the value associated with key k. If there is no element with key k, it returns 0.
c. **Min():**
   return the current smallest key
d. **Max():**
   return the current largest key
e. **Delete(k):**
   remove element with key k. Returns value of deleted element (0 if such a key does not exist).
f. **Size():**
   return the number of elements currently stored.

Empirically evaluate the performance of 3 or more data structures on the above operations (e.g., BST, AVL, RB). Sample input files will be provided with millions of operations. Compare the running times on each input file.

Input specification:

Programs read input from stdin and print output to stdout. Keys and values are long integers. Initially the dictionary is empty. The input contains a sequence of lines. Each line contains one operation followed by parameters needed for that operation, separated by spaces.

Output specification:

The output is a number, which is the sum of the return values of the operations (mod 997), followed by the time taken, in milliseconds.

- **Test results** –

Following table includes values of Time taken in miliseconds

Input Sequence	**BST**	**AVL**	**Red Black Tree**
5k.txt	171	143	136
c1.txt	77	31	15
c2.txt	59	53	51
k1.txt	101	73	91
k2.txt	63	58	49
l1.txt	9	11	10
l2.txt	10	8	11
m1.txt	2826	2507	2405
m2.txt	2547	2422	2496

- **Discussion of results**

  If we observe above results, BST is taking more time than rest 2 in most cases. AVL and Red Black Tree has comparable performances. Among these 2 ,Red black gives edge performance (6 out of 9 input sequences) than AVL.

- **Conclusion**

  Red black tree gives better performance than AVL and BST.

- **References**

avl delete : http://www.dreamincode.net/forums/topic/214510-working-example-of-avl-tree-remove-method/

avl insert : http://www.sanfoundry.com/java-program-implement-avl-tree/

bst delete : http://www.sanfoundry.com/java-program-implement-binary-search-tree/

red black tree : http://algs4.cs.princeton.edu/33balanced/RedBlackBST.java.html