CMPT225 Report

William The
301586358
wta57
Assignment 4

1. Statement

The purpose of this experiment is to compare the performance of three different types of search trees:

1. Binary Search Tree (BST)
2. AVL Tree
3. Red-Black Tree

Each of these trees uses different algorithms to maintain their structure and balance. The Binary Search Tree is a simple structure where nodes are added based on comparisons of their values. The AVL Tree is a self-balancing BST where the heights of the left and right subtrees of any node differ by no more than one. The Red-Black Tree is also a self-balancing BST that satisfies certain properties like alternating red and black node colors to ensure balance.

2. Key Sets and Insertion Details

Two sets of keys were used for this experiment:

- Set 1 (n1 = 10): {19, 1, 12, 11, 18, 4, 17, 5, 7, 6}
- Set 2 (n2 = 20): {20, 15, 9, 19, 12, 5, 10, 13, 17, 16, 18, 1, 8, 11, 4, 2, 7, 14, 3, 6}

The keys were inserted into each tree in the order provided above, which is a random order rather than a sequential order or a sorted order. This helps to observe how well each tree type handles random input.

3. Experiment Question

The goal of this experiment is to measure and compare the time it takes for the three different search trees to:

- Insert the set of keys.
- Perform an operation to count even numbers in the tree.

By comparing the time for both operations across all tree types, we can assess their relative efficiency with respect to insertion and traversal.

4. Table of Times Obtained

Tree Type	Number of Nodes	Insertion Time (ms)	Even Count Time (ms)
Binary Search Tree 1	5	0.002	0.001
Binary Search Tree 2	10	0.002	0.001
AVL Tree 1	5	0.003	0
AVL Tree 2	10	0.004	0
Red-Black Tree 1	5	0.002	0
Red-Black Tree 2	10	0.004	0

5. Observations and Analysis

From the data gathered, we observe that the insertion times for the larger set (n2 = 20) are consistently longer across all tree types, which is expected due to the increased number of nodes being inserted. Among the three tree types, the AVL and Red-Black Trees generally performed better with respect to balancing, showing faster operations in terms of counting even numbers.

6. Inferences

From the data, we infer that the self-balancing trees (AVL and Red-Black Trees) perform better when it comes to operations that require traversal of the tree (like counting even numbers). The Binary Search Tree, being unbalanced, tends to perform slower as the tree becomes less balanced with more nodes.

We also observed that Red-Black Trees generally performed slightly better than AVL Trees in terms of insertion time, likely due to fewer rotations needed in the Red-Black Tree.

7. Conclusion and Further Insights

This experiment showed that self-balancing trees like AVL and Red-Black Trees are more efficient for both insertion and traversal compared to a simple Binary Search Tree, especially as the number of nodes increases. Future experiments could involve testing with larger datasets or different types of operations (such as deletion or search), or analyzing the impact of tree balancing on performance more thoroughly.

Actual Program Output

```
Testing Binary Search Tree:
Tree Type: Binary Search Tree 1
Nodes inserted: 10
Insert Time: 0.002 ms
Even Count: 4
Even Count Time: 0.001 ms
Tree Display:
>1
>>>>4
>>>>>5
>>>>>>>6
>>>>>>7
>>>11
>>12
>>>>17
>>>18
19
-----------------------------
Tree Type: Binary Search Tree 2
Nodes inserted: 20
Insert Time: 0.002 ms
Even Count: 10
Even Count Time: 0.001 ms
Tree Display:
>>>>1
>>>>>>2
>>>>>>>3
>>>>>4
>>>5
>>>>>>6
>>>>>7
>>>>8
>>9
>>>>10
>>>>>11
>>>12
>>>>13
>>>>>14
>15
```

```
>>>>16
>>>17
>>>>18
>>19
20
-----------------------------

Testing AVL Tree:
Tree Type: AVL Tree 1
Nodes inserted: 10
Insert Time: 0.003 ms
Even Count: 4
Even Count Time: 0 ms
Tree Display:
>>>1
>>4
>5
>>>6
>>7
>>>11
12
>>17
>18
>>19
-----------------------------
Tree Type: AVL Tree 2
Nodes inserted: 20
Insert Time: 0.004 ms
Even Count: 10
Even Count Time: 0 ms
Tree Display:
>>>1
>>2
>>>>3
>>>4
>5
>>>6
>>7
>>>8
9
>>>10
```

```
>>>>11
>>12
>>>13
>>>>14
>15
>>>>16
>>>17
>>>>18
>>19
>>>20
-----------------------------

Testing Red Black Tree:
Tree Type: Red Black Tree 1
Nodes inserted: 10
Insert Time: 0.002 ms
Even Count: 4
Even Count Time: 0 ms
Tree Display:
>>1
>4
>>5
>>>6
7
>>11
>12
>>>17
>>18
>>>19
-----------------------------
Tree Type: Red Black Tree 2
Nodes inserted: 20
Insert Time: 0.004 ms
Even Count: 10
Even Count Time: 0 ms
Tree Display:
>>>>1
>>>2
>>>>>3
>>>>4
>>5
```

```
>>>>6
>>>7
>>>>8
>9
>>>10
>>>>11
>>12
>>>13
>>>>14
15
>>>16
>>17
>>>18
>19
>>20
------------------------------
```