Hashing Performance Comparison with Simple BST

Jawad Habib, Roll 10
Year 3, Semester 2

University of Dhaka

1 Introduction

A comparison has been done in the insertion and searching performances of a simple binary search tree (BST) and a hash table with double hashing. The results show a significant reduction in steps, as well as lesser memory usage for insertion and searching in hash tables compared to simple BSTs.

2 Dataset Characteristics

The input sequence consisted of 10000 random number insertions and 3000 random number searches interleaved. 70% of the searches belonged to the insert sequence and 30% of the searches were for non-inserted keys. In the search set, 20% of the keys were duplicates.

3 Hash Table Characteristics

The implemented hash table followed an open-addressing scheme with double hashing. The table size was 120% of the maximum number of inserted keys. The primary hash function was as follows

$$H1(x) = key \ \% \ m1$$
$$where \ m1 = table \ size \ (prime)$$

The secondary hash function was as follows,

$$H2(x) = key \ \% \ m2$$
$$where \ m2 = m1 - 1$$

The probe function was as follows,

$$H(x) = (H1(x) + i * H2(x)) \ \% \ m1$$
$$where \ i = number \ of \ probe \ sequence$$

4 BST Characteristics

The implemented simple binary search tree (BST) was not a self-balancing tree. Duplicate values were allowed in the tree. The tree was implemented using pointers.

5 Performance Comparison

The average number of hops for insertion and deletion was calculated for both the hash table and the BST. The results showed a significant reduction in the average number of hops in the hash table.

The average number of hops across different cases was as follows,

Case	Average Number of Hops
BST Insertion	16.8109
Hash Table Insertion	2.1398
BST Searching	14.1946
Hash Table Searching	2.7716

Here is the data represented graphically,

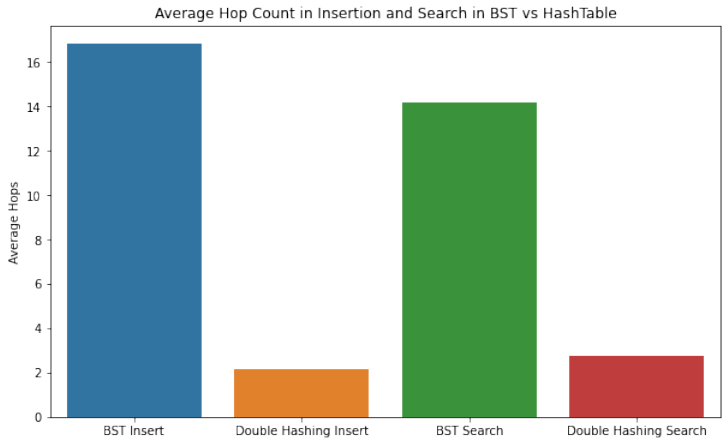

Fig 1. Hash Table vs BST Insertion and Searching

6 Conclusion

From the results, it can be concluded that the double hashing implementation of hash table provided a much better insertion and searching performance over the simple binary search tree. Also, the hash table was more space efficient than the BST. Moreover, hardware lookup cost was lesser in the hash table as the hash table takes up contiguous blocks in memory, whereas the BST nodes were located in separate memory blocks.
Therefore, a hash table with double hashing was a faster and a more space efficient approach for this input sequence of random insertions and deletions.