Project 2 Report: Data Structures and Algorithms

Document Author(s): Louis Warner
Date: 6-26-2015

Overview

This report will discuss the findings of my research on the performance of the Binary Search Tree (BST) implementation of the Dictionary ADT in Project 2. Firstly, a BST was built based off of given input from the file large.txt. It was processed to create a BST containing 513 unique key-value pairs by inserting the first two sets of input values into the tree, and removing the third set of input values from the tree.

After this initial build, three other files were processed after this initial build to determine the number of comparisons needed to find the keys given in these files. The three files were set up with 50000 keys in three different types of key distributions: geometric, uniform, and zipf. The statistics below and the chart on the following page reflect the overall performance of the find algorithm in BST for each of the given sets of data.

Geometric Distribution

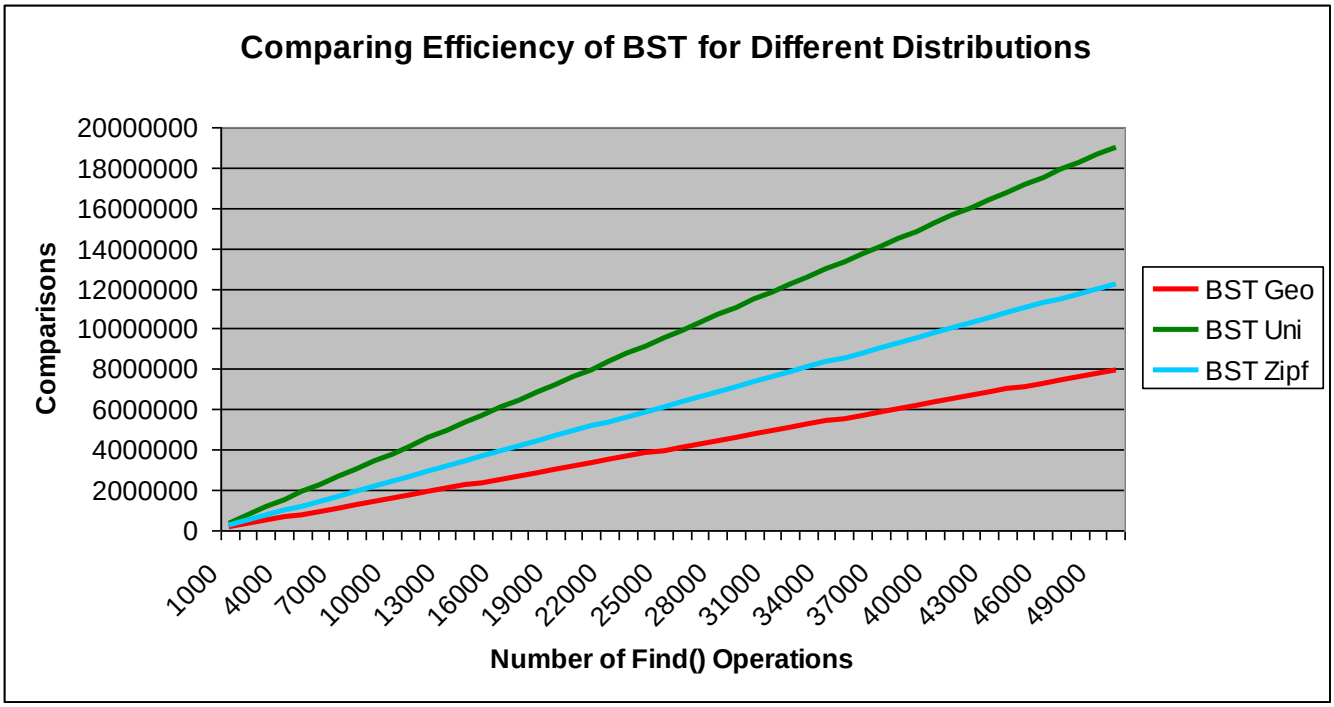
- Total Comparisons: 7,952,420
- Average Comparisons per Find(): 159.0484

Uniform Distribution

- Total Comparisons: 19017563
- Average Comparisons per Find(): 380.35126

Zipf Distribution

- Total Comparisons: 12247794
- Average Comparisons per Find(): 244.95588



Notes on Max, Min, and Averages

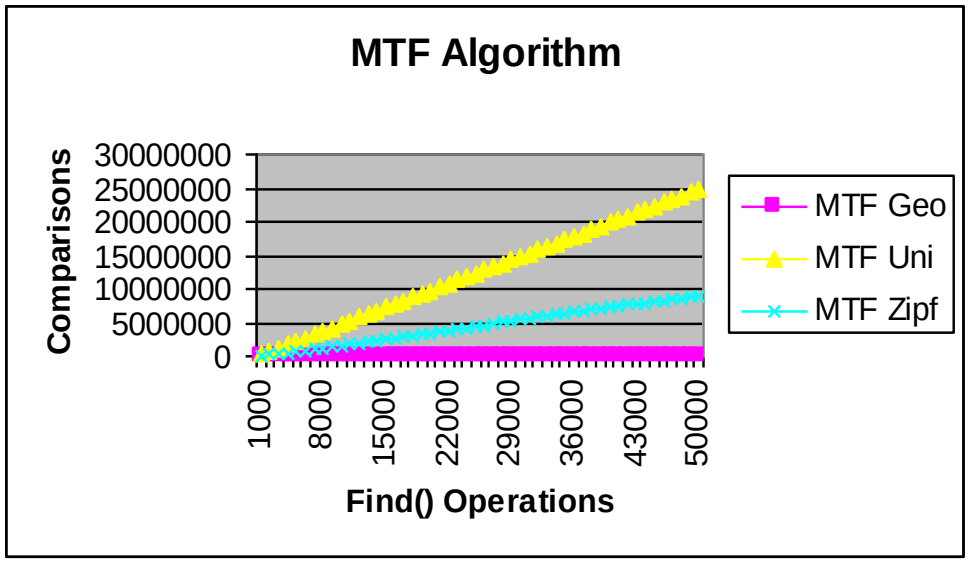
The minimum number of comparisons for the find function is 1. This would indicate that the key being searched for was the root node of the tree.

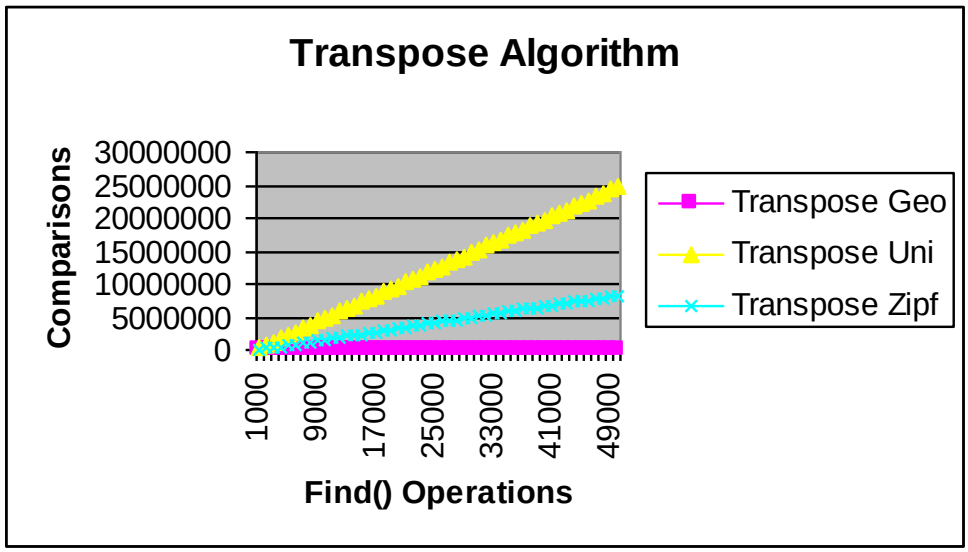
The maximum number of comparisons for the find function in this experiment was 513. 513 comparisons would be performed (one for each key) if the value being searched for was not found in the tree.

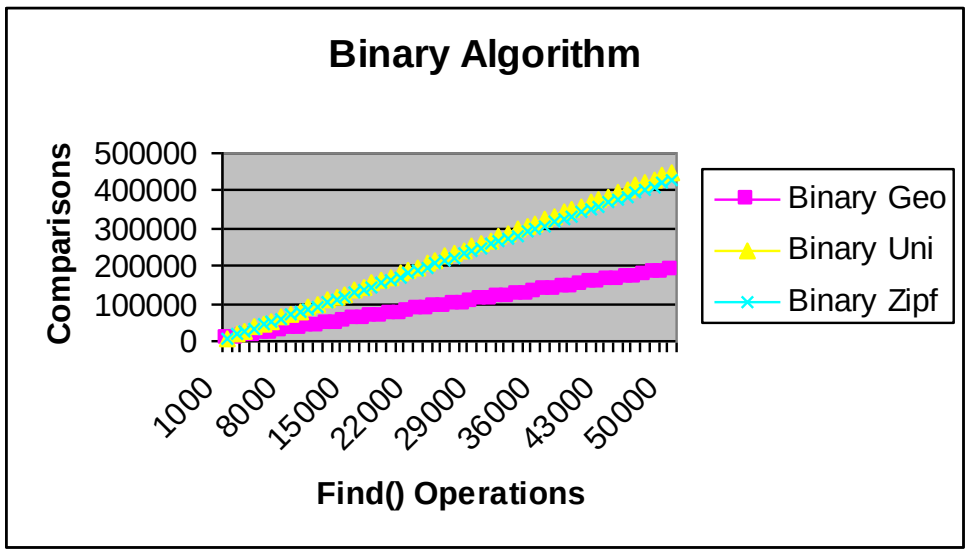
The changes in the average for number of comparisons indicate that the BST performed the most efficiently on the geometric distribution, slightly worse on the zipf distribution, and worst on the uniform distribution.

Comparison to Algorithms from Project 1

The above chart follows the trend for how distributions have affected previously covered algorithms from project 1. The average number of comparisons for each method on each distribution and the charts on the following page also show that for each algorithm, the distributions order from best to worst performance always goes from geometric distribution, to slightly better with zipf distribution, to the worst distribution being uniform.


MTF Algorithm
Comparisons
30000000
25000000
20000000
15000000
10000000
5000000
0
1000
8000
15000
22000
29000
36000
43000
50000
Find() Operations
MTF Geo
MTF Uni
MTF Zipf


Transpose Algorithm
Comparisons
30000000
25000000
20000000
15000000
10000000
5000000
0
1000
9000
17000
25000
33000
41000
49000
Find() Operations
Transpose Geo
Transpose Uni
Transpose Zipf


Binary Algorithm
Comparisons
500000
400000
300000
200000
100000
0
1000
8000
15000
22000
29000
36000
43000
50000
Find() Operations
Binary Geo
Binary Uni
Binary Zipf