Sorry for my english : international student

In class we saw that a Hashtable has expected case time to find of O(1), a BST worst case O(logN) and a MWT worst case O(K), where K is the length of the longest string. We didn't look at the running time of the TST, though the book mentions that its average case time to find is O(logN) (and worst case O(N)). Are your empirical results consistent with these analytical running time expectations? If yes, justify how by making reference to your graphs. If not, explain why not and also explain why you think you did not get the results you expected (also referencing your graphs).

The results makes sense with these analytical running time expectations. First of all, we can see that for the HashTable the average is about 32 000 nano seconds, for the MWT is 40 000 ns and for the BST is 102 000 ns. The complexity of an Hashtable is O(1) it is the lowest average case time to find, indeed we can see that is the lowest with 32 000 ns. The BST has the biggest average case time here to find it is O(log(N)) so it has to be the biggest value and it is with 102 000 ns. The MWT has an average case time to find of O(N) so more than an HashTable but a lot less than a BST ,the results corresponds with 40 000 ns.

Describe how each hash function works, and cite the source where you found this information. *Your description of how the hash function works should be in your own words*.

1. Describe how you verified the correctness of each hash function's implementation. Describe at least 3 test cases you used, what value you expected for each hash function on each test case, and the process you used to verify that the functions gave this desired output.

The hashFunction1 works adding the ASCII values of each character of the word to a variable and at the end dividing it but the size of the tableSize (source : stackflow) . The hashFunction2 it multiplies the seed by the current accumulation of characters at thatr point and keep adding it together and then divide it.

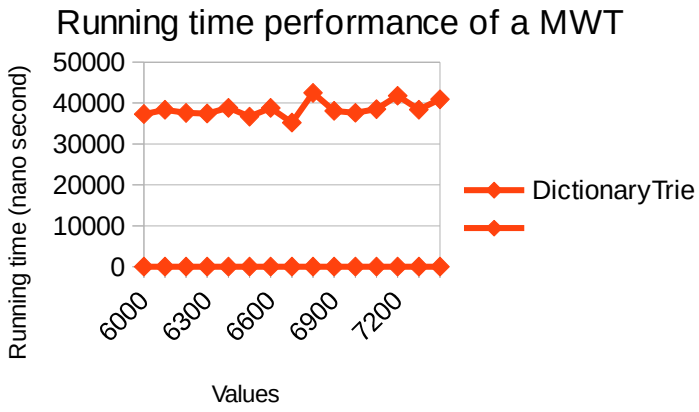
1. hello : 532 hi : 209 then : 431
2. hello : 622 hi : 1729 then : 641

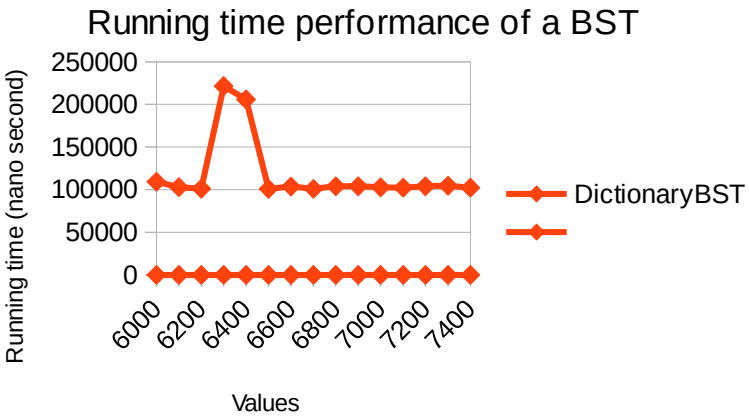
2. Run your benchhash program multiple times with different data and include a table that summarizes the results of several runs of each hash function. Format the output nicely--don't just copy and paste from your program's output.

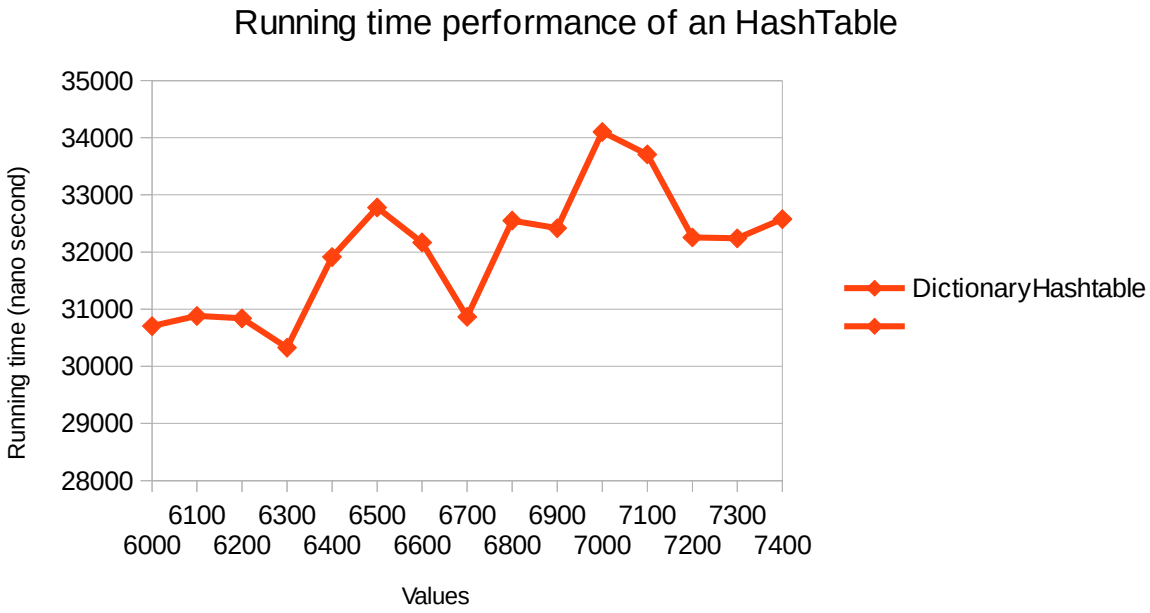
	Hash1	Hash2
1000	1.488	1.241
2000	1.911	1.2485
3000	2.375	1.25733
4000	2.828	1.2515
5000	3.2608	1.2515

3,Comment on which hash function is better, and why, based on your output. Comment on whether this matched your expectation or not.

The second hash function is better because the average number of step stay constant as the number of words increases tand it stay over one step which is very good compared to the other hash function which increases over tuime and approaches average of 3 steps.


Running time performance of a MWT
Running time (nano second)
50000
40000
30000
20000
10000
0
6000
6300
6600
6900
7200
Values
DictionaryTrie


Running time performance of a BST
Running time (nano second)
250000
200000
150000
100000
50000
0
6000
6200
6400
6600
6800
7000
7200
7400
Values
DictionaryBST


Running time performance of an HashTable
Running time (nano second)
35000
34000
33000
32000
31000
30000
29000
28000
6000
6100
6200
6300
6400
6500
6600
6700
6800
6900
7000
7100
7200
7300
7400
Values
DictionaryHashtable