# COMP20003: Assignment 1, Program evaluation

Tom Hanson - 913022

## Introduction

To evaluate the effectiveness of the data structures implemented in the 2 programs dict1.exe and dict2.exe several tests were undertaken. These tests consisted of searching through datasets of various sizes and comparing the number of comparisons between the search key and the data. The data was sourced from https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page and broken into random subsets of size 10, 50, 100, 500, 1000,5000,10000 and 50000 records. These datasets were then randomised, sorted and reverse sorted to see how the programs behaved in each case. For each data set, the programs were asked to search for 50 random keys and the average number of comparisons was found.

Both dict1.exe and dict2.exe utilize a Binary Search Tree (BST) data structure sorted by their respective keys which has a theoretical O(logn) search complexity. Duplicate keys are put into a linked list sorted by a secondary sort key which have a theoretical O(n) search complexity.

As per the design requirements from “Assignment 1 Test Script.ipynb”, comparisons when searching along the duplicate linked list in dict2.exe were also included in the results.

# Dict1.exe

The data from experimentation on Dict1.exe can be found in the table below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dict1** |  |  |  |  |  |  |  |  |
| Sorted |  |  |  |  |  |  |  |  |
| Sample size | 10 | 50 | 100 | 500 | 1000 | 5000 | 10000 | 50000 |
| Average Comp | 1 | 1 | 1 | 1 | 1.74 | 2.02 | 2.28 | 12.3 |
| Sorted R |  |  |  |  |  |  |  |  |
| Sample size | 10 | 50 | 100 | 500 | 1000 | 5000 | 10000 | 50000 |
| Average Comp | 10 | 50 | 100 | 500 | 1000 | 4998.68 | 9995.4 | 49931.2 |
| Random |  |  |  |  |  |  |  |  |
| Sample size | 10 | 50 | 100 | 500 | 1000 | 5000 | 10000 | 50000 |
| Average Comp | 3 | 4 | 5 | 6 | 7.74 | 10.18 | 10.8 | 13.84 |

In the sorted case, the number of comparisons is erratic and doesn’t reflect what we expect the worst-case scenario to be O(n). However, looking at the reverse sorted data set we see that the number of comparisons to be very close to the size of the data input, suggesting that a sorted input can in fact lead to a worst case scenario of O(n). These results are likely due to the keys being unintentionally weighted towards the start of the BST, thus, when the sort order is reversed, the corresponding data is entered in last.

In the random case, the BST behaves how we would expect a BST to function in the average case. We find that comparisons are approximately log2.5(n), thus the search complexity in the average case is O(logn).

# Dict2.exe

The data from experimentation on Dict12exe can be found in the table below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dict2** |  |  |  |  |  |  |  |  |
| Sorted |  |  |  |  |  |  |  |  |
| Sample size | 10 | 50 | 100 | 500 | 1000 | 5000 | 10000 | 50000 |
| Average Comp | 4.16 | 7.4 | 7.2 | 14.32 | 22.48 | 92.91837 | 186.12 | 900.48 |
| Sorted R |  |  |  |  |  |  |  |  |
| Sample size | 10 | 50 | 100 | 500 | 1000 | 5000 | 10000 | 50000 |
| Average Comp | 3.125 | 6.2 | 7.64 | 13.14 | 22.4 | 96.42 | 186.24 | 900.7347 |
| Random |  |  |  |  |  |  |  |  |
| Sample size | 10 | 50 | 100 | 500 | 1000 | 5000 | 10000 | 50000 |
| Average Comp | 3.8 | 6.24 | 7.56 | 14.34 | 23.22 | 95.84 | 186.66 | 900.94 |

The results from these 3 tests conclude that Dict2.exe behaves as expected. Most of the data appears to be stored in linked lists and that the BST portion of the data structure is relatively small. We can see this with small sample sizes; most of the data is stored in the BST and thus the average case is closer to O(logn). However, as the sample size increases, most of the data is stored in linked lists and traversing those is a O(n) operation. This is because there is no search function implemented in the linked list component that takes advantage of the sorted nature of the list. From this we can see that there are roughly 900 duplicates at most stored in linked list.

Interestingly, all 3 data sets appeared to have very similar overall search complexity. I believe this is because, this data structure functions as a linked list pointing to linked lists. Thus, when searching through the data, only non-duplicates will be traversed, which appears to be much less than the total input size in the average case. Theoretically, the worst case search complexity is still O(n) if all entries had different keys.