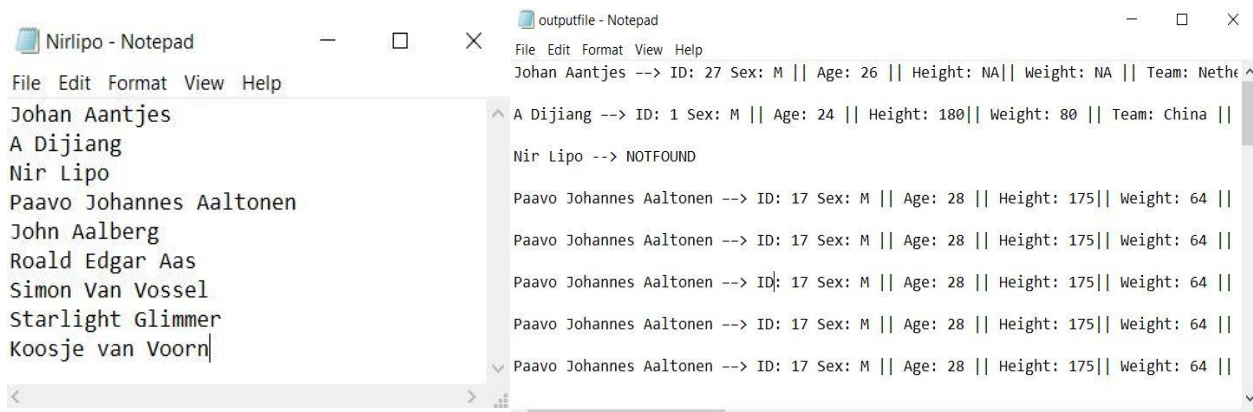


The University of Melbourne
Olympic Dataset Retrieval using a Binary Search Tree
Patrick Tjahjadi (890003)

1. Introduction

The assignment involves the use of a dictionary, an abstract data type that stores key-value pairs. The purpose of the assignment is to create an Information Retrieval system using Binary Search Trees as the underlying data structure. The datasets involved for this project are information about Olympic athletes in the form of a CSV (The datasets can be obtained from the following link: <https://transfer.sh/152BXj/comp20003data.zip>). C is used for the program.

There are two stages in this assignment. Stage 1 involves developing a Binary Search Tree with insertion and search functions. The Binary Search Tree would be able to handle non-unique keys and will output the number of key comparisons required to search a particular athlete. Stage 2 continues by developing a linked list for each node that contain duplicates, so no further key comparisons are required. Hypothetically, this will result in more efficient searching since Stage 2 would save time by not searching further down the tree, unlike Stage 1.



The image shows two Notepad windows. The left window, titled 'Nirlipo - Notepad', contains a list of names: Johan Aantjes, A Dijiang, Nir Lipo, Paavo Johannes Aaltonen, John Aalberg, Roald Edgar Aas, Simon Van Vossel, Starlight Glimmer, and Koosje van Voorn. The right window, titled 'outputfile - Notepad', shows the corresponding output for each name. It lists the same names followed by their attributes: ID, Sex, Age, Height, Weight, and Team. For example, 'Johan Aantjes --> ID: 27 Sex: M || Age: 26 || Height: NA|| Weight: NA || Team: Neth'. Names like 'Nir Lipo' and 'Starlight Glimmer' are marked as 'NOTFOUND'.

```
File Edit Format View Help
Johan Aantjes --> ID: 27 Sex: M || Age: 26 || Height: NA|| Weight: NA || Team: Neth
A Dijiang --> ID: 1 Sex: M || Age: 24 || Height: 180|| Weight: 80 || Team: China ||
Nir Lipo --> NOTFOUND
Paavo Johannes Aaltonen --> ID: 17 Sex: M || Age: 28 || Height: 175|| Weight: 64 ||
Paavo Johannes Aaltonen --> ID: 17 Sex: M || Age: 28 || Height: 175|| Weight: 64 ||
Paavo Johannes Aaltonen --> ID: 17 Sex: M || Age: 28 || Height: 175|| Weight: 64 ||
Paavo Johannes Aaltonen --> ID: 17 Sex: M || Age: 28 || Height: 175|| Weight: 64 ||
Paavo Johannes Aaltonen --> ID: 17 Sex: M || Age: 28 || Height: 175|| Weight: 64 ||
```

Figure 1: Sample input and output used for the experiment.

For this experiment, the inputs are in the form of .txt file that contains names to be searched. The program supports multiple names separated by a new line (Figure 1). The program would then output a file that contains the athlete's information, if found. In particular, the experiment deliberately uses two names that are not found: "Nir Lipo" and "Starlight Glimmer". Multiple tests with various datasets will be conducted to assess and compare the efficiency of the two stages. The results would then be compared against the underlying theories of Big-O.

2. Experiment

Firstly, note an assumption that the standard base for $\log(n)$ is 2. Let n be the number of items in the file. We know by theory that the time complexity of a Binary Search Tree is $\Theta(\log(n))$ on average for accessing, searching, inserting and deleting. Moreover, the time complexity is $O(n)$ in the worst case for accessing, searching, inserting and deleting. This would happen if the program reads a dataset with names in sorted order, degrading the Binary Search Tree into a linked list. Finally, the worst case for space complexity of a Binary Search Tree is $O(n)$.

The number of key comparisons from the first experiment shown in Figure 1 from `athlete_events_filtered.csv` is displayed as follows:

```
patpa@DESKTOP-T8FEDVK /c/users/patpa/COMP20003/Assignment 1
$ dict1 athlete_events_filtered.csv outputfile.txt < Nirlipo.txt
Johan Aantjes --> 12
A Dijiang --> 2
Nir Lipo --> 41
Paavo Johannes Aaltonen --> 42
John Aalberg --> 17
Roald Edgar Aas --> 30
Simon Van Vossel --> 73
Starlight Glimmer --> 46
Koosje van Voorn --> 79
```

Figure 2: Key comparisons for Stage 1

```
patpa@DESKTOP-T8FEDVK /c/users/patpa/COMP20003/Assignment 1
$ dict2 athlete_events_filtered.csv outputfile.txt < Nirlipo.txt
Johan Aantjes --> 6
A Dijiang --> 1
Nir Lipo --> 29
Paavo Johannes Aaltonen --> 9
John Aalberg --> 5
Roald Edgar Aas --> 9
Simon Van Vossel --> 33
Starlight Glimmer --> 26
Koosje van Voorn --> 32
```

Figure 3: Key comparisons for Stage 2

As shown from the two figures, the number of key comparisons in Stage 2 is less than Stage 1. This is because Stage 2 involves the implementation of a linked list that check for duplicates, so no extra key comparisons are required. However, in order to assess the feasibility of this experiment, the program is run against other datasets, three randomised dataset from the link provided. Another dataset involves sorted entries to test for worst case performance (**Figure 6**). Using the 9 names provided from Figure 1, the average key comparisons for the searched names are recorded for each dataset in Figure 4 and 5.

Number of Entries	Filtered	Alternative	Randomised	Average Key Comparisons for the 3 Datasets
10	2.78	4.78	2.78	3.45
100	17	8.22	33.78	19.67
1000	23.11	11.89	39.22	24.74
10000	28.89	17.11	47.78	31.26
100000	34.89	22.11	55.67	37.56
271116	38	26.11	59.33	41.15

Figure 4: Key Comparisons for datasets tested in Stage 1.

Number of Entries	Filtered	Alternative	Randomised	Average Key Comparisons for the 3 Datasets
10	2.78	4.78	2.78	3.45
100	7	8.22	33.67	16.3
1000	10	11.78	36.11	19.3
10000	12.78	17	39	22.93
100000	15.22	21	43.33	26.52
271116	16.67	21.89	44.33	27.63

Figure 5: Key Comparisons for datasets tested in Stage 2.

Number of Entries	Stage 1	Stage 2
10	9.11	8.11
100	55.33	29.44
1000	810	533.44
10000	3886.33	2085

Figure 6: Average Key Comparisons for a sorted dataset.

The sorted dataset contains names that are sorted, eventually degrading the binary search tree into a stick, and is acquired from the provided dataset, namely: athlete_events_filtered_alternative2.csv

To demonstrate the trend and correlation of the data, a scatter plot is charted to compare the complexity of the program against the underlying theory. Here, two scatter plots will be charted, one to assess the average case of the program and another to assess the worst case of the program. The number of key comparisons for both stages in the “Randomised” dataset is plotted to assess the average case of the program, whereas the number of key comparisons measured in Figure 6 will be used to assess the worst case of the program.

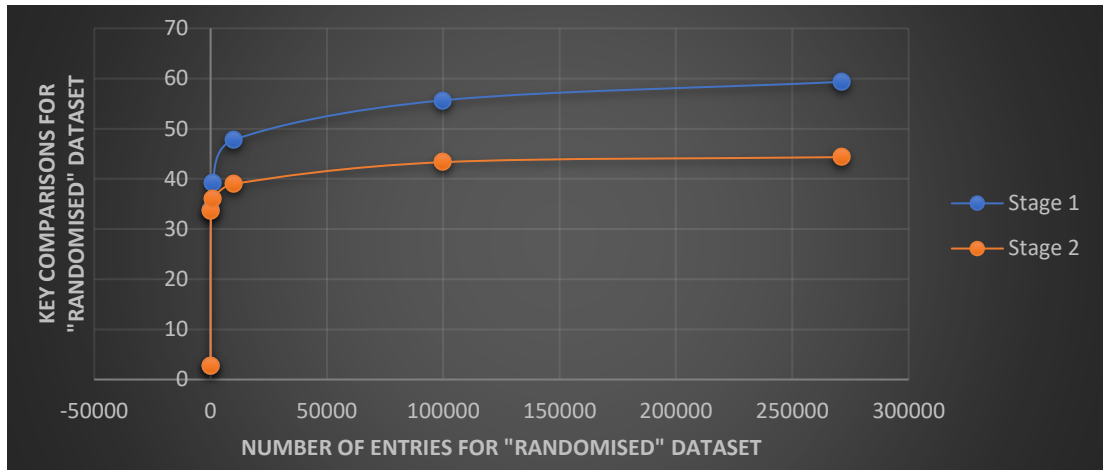


Figure 7: Scatter plot for the “Randomised” Dataset

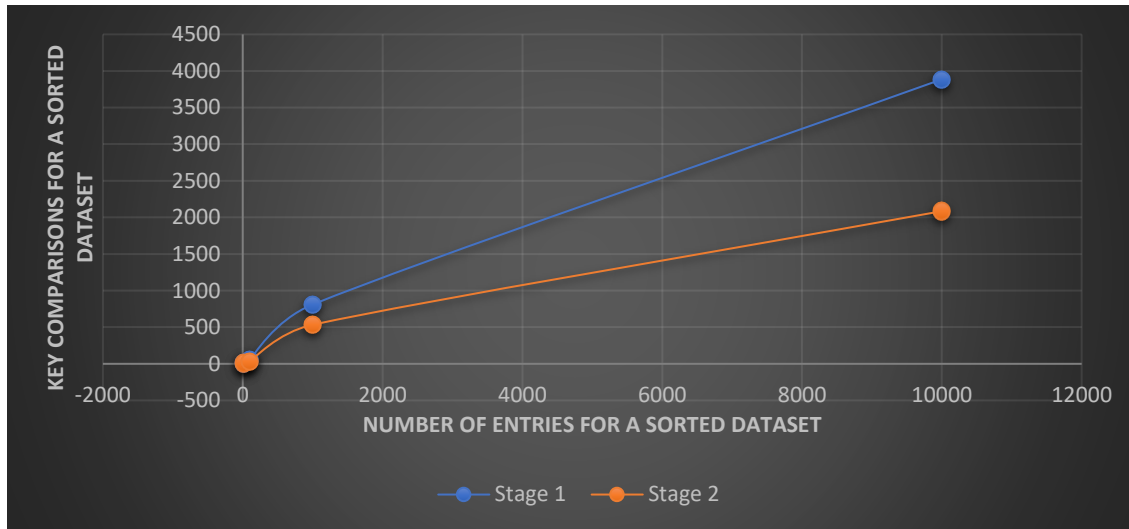


Figure 8: Scatter plot for a Sorted Dataset.

The graphs in Figure 7 largely resembles that of a $\log(n)$ function. Any constant before $\log(n)$ can be disregarded since it is a small value. This illustrates that the program behaves as expected according to the Big-O theory of Binary Search Trees, with an average case of $\Theta(\log(n))$.

However, the graphs in Figure 8 resembles a linear function. This is expected from theory as the worst-case time complexity of a Binary Search Tree is $O(n)$. Again, any constant before n can be disregarded as we are concerned when n becomes large, in which the constant would be considered a small value.

Another test was conducted with a dataset having the same name for every entry. This results in Stage 1 outputting the number of key comparisons equal to the number of entries and Stage 2 outputting 1 key comparison regardless of the number of entries.

3. Discussion

Overall, the results of the experiment went as expected according to their underlying Big-O theories. Moreover, experiments with larger datasets suggests that Stage 2 becomes more and more effective in comparison against Stage 1 since fewer key comparisons are required. This suggests that a marginally faster search through the Binary Search Tree can be done using Stage 2.

Note that this experiment was done using a recursive function

As one must anticipate the worst-case in terms of algorithmic complexity, both Stage 1 and Stage 2 degrades to $O(n)$, as compared to its average $\Theta(\log(n))$. However, it is seen that in both cases that Stage 2 has an advantage in the number of key comparisons required thanks to its linked list implementation to search for data with similar keys.

In conclusion, the program has worked as expected, and the implementation of the Binary Search Tree within the program matches against the expected theories. Although the experiment may have slight uncertainty and error in terms of key comparisons, Figure 7 and 8 approximately reflects the expected theories within a Binary Search Tree. Overall, the experiment can be considered a success.