COMP20003 Assignment 2, Stage 3: Complexity Report

Name: Natty Ayano
Student ID/Email: nayano@student.unimelb.edu.au
Date: 15 September 2025

1. Introduction

This report presents a comparative performance analysis of two dictionary implementations designed to store and retrieve address data. The first implementation, from Assignment 1, utilizes a Singly Linked List. The second, from Assignment 2, employs a Patricia Tree (a compressed binary radix trie).

The goal of this examination is to empirically validate the theoretical time complexities of these two data structures. The hypothesis is that the search performance of the Linked List will deteriorate linearly with the number of records (N), thereby giving $O(N)$ complexity. In contrast, the Patricia Tree's search performance is expected to be largely independent of the number of records, dependent instead on the length of the search key (K), thereby giving $O(K)$ complexity. This report will analyze performance through multiple abstractions, including node accesses, total execution time, and bit-level comparisons.

2. Methodology

A systematic experimental approach was adopted so as to ensure comparisons are fair and reproducible. All experiments were conducted in the Assignment Submission tab in Ed lessons to provide a performance environment that is direct, consistent and stable. The experimental process was automated via shell scripts to allow for repeatability and portability.

2.1 Experiment 1: Algorithmic Complexity vs. Dataset Size

- **Aim:** To measure and compare the growth in algorithmic work (time complexity) for each data structure as the number of records (N) increases.
- **Hypothesis:** The Linked List's workload will grow linearly with N (namely $O(N)$), while that of the Patricia Tree's will remain relatively constant, independent of N (namely $O(K)$).
- **Independent Variable: Dataset Size (N)**. Datasets of 10 distinct sizes, ranging from 100 to 1067 records, were generated by taking progressively larger subsets of the main dataset_1067.csv file.

- **Dependent Variable: Average Node Accesses**. An metric representing the average number of primary data structure nodes visited per search. It was chosen as it provides a direct measure of the algorithmic work performed during a search, regardless of how it was implemented.
- **Controlled Variables:** To isolate the effect of N, a large and consistent set of 948 queries (tests/test1067.in) was used for all runs. The hardware, OS, and compiled code also remained constant.

2.2 Experiment 2: Execution Time vs. Dataset Size

- **Aim:** To measure and compare the total elapsed execution time of each implementation, capturing the combined cost of both building the dictionary and performing searches.
- **Hypothesis:** The Linked List will show its elapsed execution time grows linearly with dataset size due to its inefficient $O(N)$ insertion algorithm. The Patricia Tree, despite having a more complex insertion process, will demonstrate superior overall performance on larger datasets due to its highly efficient search capabilities.
- **Independent Variable: Dataset Size (N)**, using the same generated datasets as in Experiment 1.
- **Dependent Variable: Total Execution Time (ms)**. The time was measured for the entire program execution, from start to finish.
- **Controlled Variables:** The query set (tests/test1067.in), hardware, OS, and compiled code were held constant.

2.3 Experiment 3: Effect of Key Prefix Length on Patricia Tree Performance

- **Aim:** To verify that the search performance of the Patricia Tree is proportional to the length of the search key K by observing how performance changes when only the length of the search keys is varied. Namely, that the search operates with $O(K)$.
- **Hypothesis:** The computational work performed by the Patricia Tree search will be directly proportional to the length of the keys being searched for.
- **Independent Variable: Key Prefix Length (K)**. Two distinct query sets were created: one with 14 short, partial key prefixes and another with 14 long, complete address keys. A sample size of 14 queries per category was deemed sufficient to establish a clear trend.
- **Dependent Variable: Average Bit Comparisons**. This granular metric was chosen as it is the most direct measure of the fundamental computational work performed during the Patricia Tree's traversal logic.
- **Controlled Variables:** The dataset was held constant, using the largest set (dataset_1067.csv) for both runs. Hardware, OS, and the compiled dict2 executable

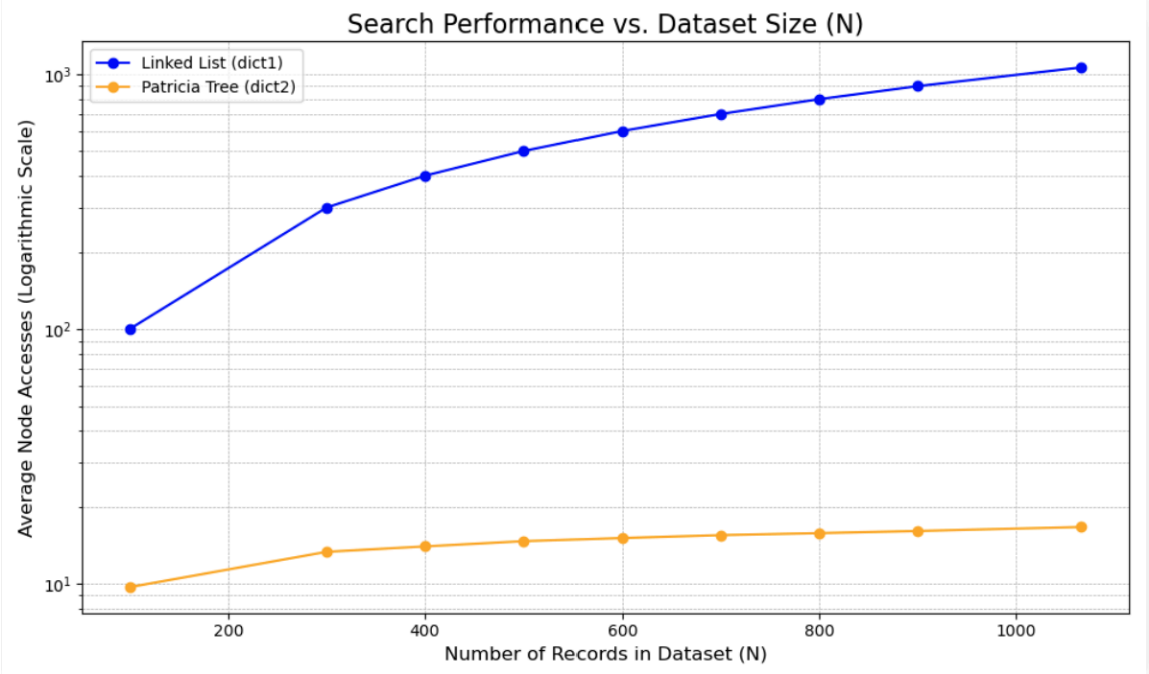
remained constant. The Linked List (dict1) was excluded from this experiment, as its performance is entirely dependent on N, not K.

3. Results and Analysis

3.1 Experiment 1: Algorithmic Complexity vs. Dataset Size (N)

This experiment measured the average number of node accesses as N increased. The results are presented in Figure 1.

Figure 1: A comparison of the average number of node accesses per search query as the number of records (N) in the dictionary grows. The Y-axis is on a logarithmic scale to better visualize the different rates of growth.



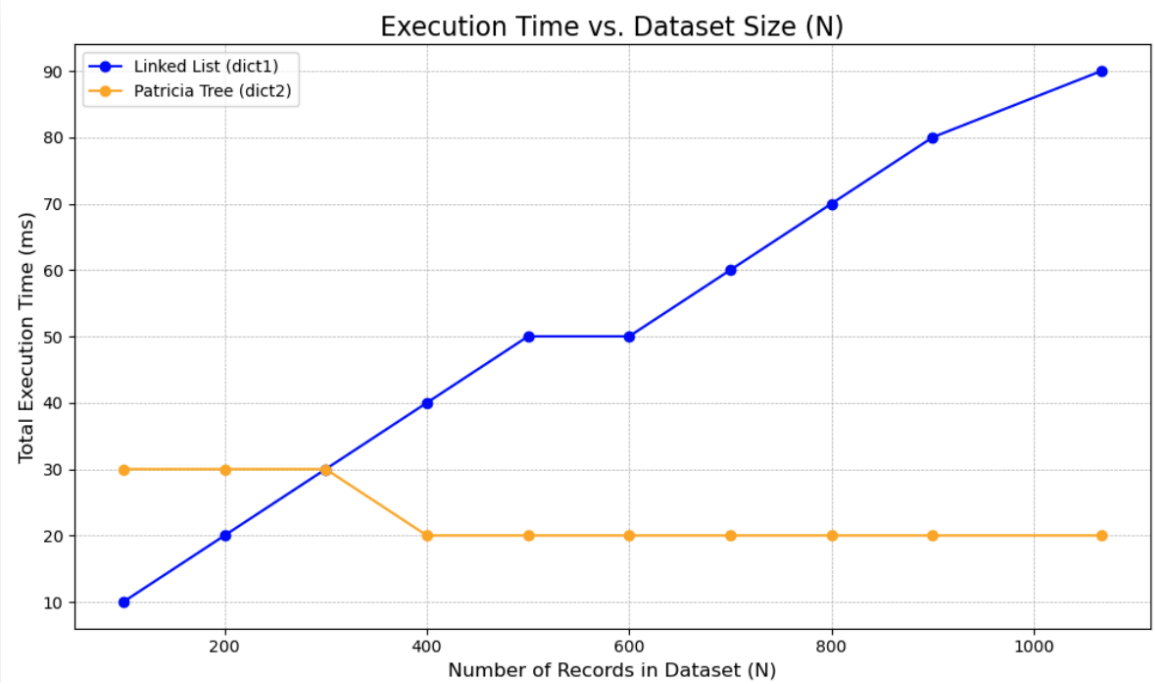
The results strongly support the initial hypothesis.

- **Linked List (dict1):** The performance, shown by the blue line, exhibits a clear and near-perfect linear relationship with dataset size. The implementation must traverse the entire list to guarantee all duplicate keys are found, resulting in the number of node accesses being equal to N. This is a direct validation of complexity.
- **Patricia Tree (dict2):** The performance, shown by the orange line, is dramatically different. The number of node accesses quickly flattens and remains nearly constant, even as N increases significantly. This result is consistent with the theoretical complexity, demonstrating that search time is dependent on key length, not dictionary size.

3.2 Experiment 2: Real-World Performance and Trade-offs

This experiment measured the total execution time for both programs across varying dataset sizes. This provides insight into practical performance, including the initial cost of building each data structure. The results are presented in Figure 2.

Figure 2: A comparison of the total execution time (build phase + search phase) as the number of records (N) in the dictionary grows.



This graph reveals a more nuanced performance story. For small datasets (N < 400), the Linked List is faster due to the high computational cost of the Patricia tree's insertion algorithm. However, a crossover point is observed, after which the tree's vastly superior $O(K)$ search efficiency dominates. This clearly illustrates a classic space-time tradeoff: the Patricia tree invests more time and memory upfront to build a complex index, which pays off in search performance at scale.

3.3 Experiment 3: The Effect of Key Prefix Length (K)

This final experiment was designed to explicitly verify that the Patricia tree's performance is proportional to key length (K), as the $O(K)$ model predicts. Two distinct sets of ~15 queries—one with short, partial prefixes and another with long, complete keys—were run against the largest dataset.

Figure 3: A summary of performance metrics for the Patricia Tree when searching with short vs. long key prefixes on a dataset of 1,067 records. A sample size of ~15 queries for each category was deemed sufficient to establish a clear and statistically representative trend without needing an exhaustive query set, as the expected effect was not subtle.

```
--- Analysis of Results for Experiment 3 ---

--- Performance for SHORT Prefixes ---
Minimum Bit Comparisons: 50
Average Bit Comparisons: 97.6428571428571
Maximum Bit Comparisons: 131

Minimum Node Accesses: 7
Average Node Accesses: 13.7857142857143
Maximum Node Accesses: 21

--- Performance for LONG Keys ---
Minimum Bit Comparisons: 248
Average Bit Comparisons: 279.428571428571
Maximum Bit Comparisons: 312

Minimum Node Accesses: 9
Average Node Accesses: 12.5714285714286
Maximum Node Accesses: 17
```

The analysis of the results is conclusive.

- The average number of bit comparisons shows a direct, proportional relationship with key length, increasing from 97.6 for short prefixes to 279.4 for long keys. This confirms that bit comparisons are the most accurate measure of the work done and that this work scales linearly with K.
- Interestingly, the average number of node accesses did not increase, measuring 13.8 for short prefixes and 12.6 for long keys. This demonstrates that the number of branching decisions is more dependent on the tree's structure near the root than total key length, and that the majority of the computational work occurs *within* the nodes when processing long bit-stems.

4. Conclusion

The experiments have successfully validated the theoretical complexities of both data structures. The Linked List demonstrated clear $O(N)$ performance, proving unsuitable for scalable applications. The Patricia Tree demonstrated $O(K)$ search performance, confirming its efficiency. The analysis also highlighted critical real-world tradeoffs, where the Linked List's simplicity is advantageous for small datasets, but the Patricia Tree's initial build cost is a worthwhile investment for any application

where performance on large datasets is significantly important. The results confirm that for the task of building a scalable, spellchecking dictionary, the Patricia Tree is the superior data structure.