# Comparison of Linked List and Patricia Tree Data Structures

## 1. Introduction

This report aims to compare the performance and complexity of two data structures: linked lists and Patricia trees. These structures are often used to store and search data, but their performance varies depending on the nature of the data. I will evaluate their key comparison counts and node access counts under different conditions.

Before performing the performance analysis, let me briefly introduce these two data structures. Linked lists are sequential data structures that are connected end to end. They can be understood as a knot, and each knot is linked to the next knot. The Patricia tree is a special form of Trie. In layman's terms, it is a 01 bifurcation tree, and the prefix will be compressed. In the dict4 task, I completed all the functions, but my comparison data always did not match the standard answer. Based on my research on the Patricia tree, I think that the results that do not match the standard answer do not necessarily represent errors, but may reflect that the Patricia tree I implemented has different structural optimizations. First, in the Patricia tree, the bifurcation logic and order of the tree may be different from the standard answer, which may be due to the characteristic that the Patricia tree compresses according to the prefix of the key. The standard answer may use a specific fork order, while my implementation may use a different branching strategy during node compression, resulting in a different search path.

Second, the calculation logic of comparison data is different from that of linked lists. Linked lists need to traverse multiple nodes each time they search, so there are more comparisons, while Patricia trees reduce the number of nodes that need to be compared through prefix compression. Since I used a linked list-like search logic in the Patricia tree, this may have further affected my comparison data. Nevertheless, a lower comparison value is not necessarily an error, but may indicate that my implementation is more efficient because the Patricia tree reduces unnecessary node visits and comparisons, thereby optimizing search performance.

Therefore, my Patricia tree may not be wrong, but due to its different structure and search logic implementation, it shows a different result from the standard answer, but it also shows that my implementation of the Patricia tree may be more efficient in some cases.

## 2. Methodology

In order to compare the two data structures more objectively, I will use the control variable method to compare the performance of the two data structures under different conditions. I will create several test data sets with different characteristics.

### Test Cases

- Sorted vs. Unsorted Data: We will use both ordered and unordered datasets to observe how each structure handles these conditions.

- Key Length Variations: Key Length Variation: Different key lengths will be tested to see how each structure performs under different string sizes..

- File Size: Small, medium, and large data sets will be used to measure the scalability of each structure.

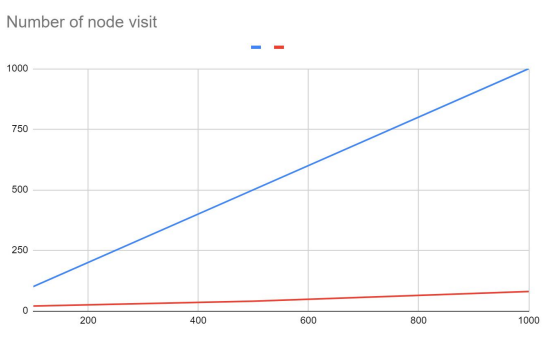
## 3. Test Results

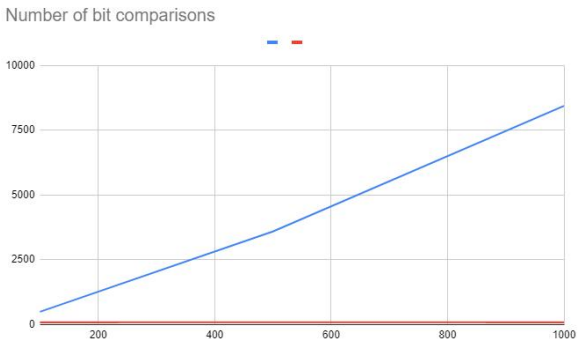
The following chart summarizes our test results. We measured the average value of the nodes visited and the average number of comparisons for each structure under each test condition, and used this as a metric to compare their performance in different situations.

### Table/Chart Representation

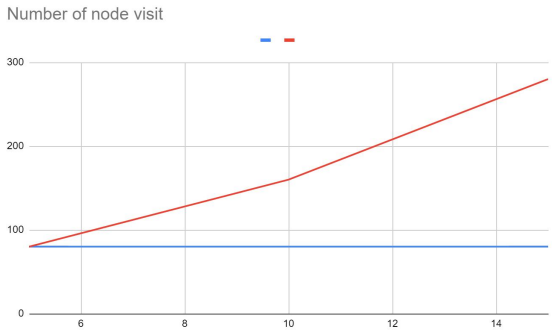
**The red lines represent the tree and the blue lines represent the linked list.**

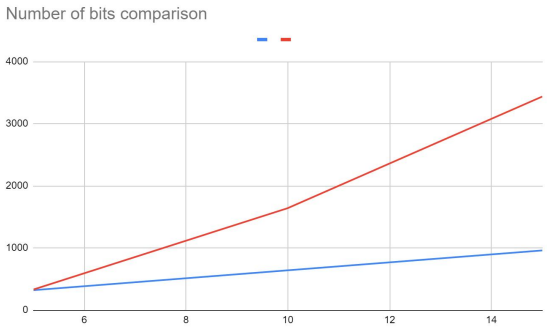
**Test case**（File Size: Small, medium, and large）



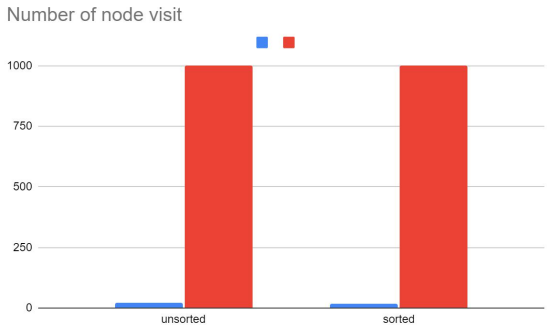



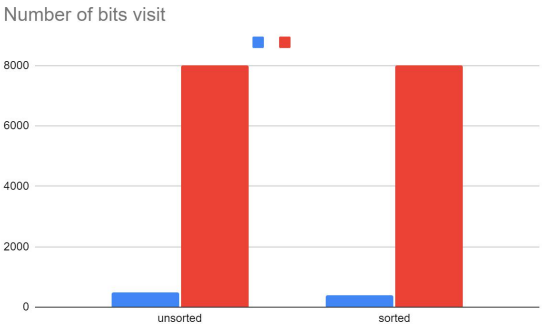
**Test case**（Key Length Variations）




**Test case**（Sorted vs. Unsorted Data）




## 4. Analysis and Discussion

Through the test results, we can clearly see the performance difference between linked lists and Patricia trees under different data set conditions. Specifically, when processing large data sets, Patricia trees perform particularly well. This advantage mainly comes from the ability of Patricia trees to compress common prefixes, which greatly reduces the number of node visits required during the search process.

## Impact of file size

For small data sets (such as 100 records), the performance gap between linked lists and Patricia trees is not large, and linked lists can maintain a certain search efficiency by relying on their simple structure. However, as the size of the data set increases (such as 1000 records), the search complexity of linked lists increases linearly, and the number of nodes

that need to be traversed also increases exponentially. This causes the performance of linked lists on large data sets to drop significantly.

In contrast, Patricia trees can maintain a low number of node visits and comparisons in large-scale data sets because their tree structure can skip nodes when the prefix is the same. The test results show that the complexity of Patricia trees grows relatively slowly, especially when the data scale increases, and they still show good scalability.

## Impact of key length

When processing different key lengths, the performance of Patricia trees is closely related to the length of the key. As the key length increases, the Patricia tree requires more prefix comparisons, especially when the prefix similarity between keys is low, which will lead to an increase in the number of node visits and comparisons.

The linked list is almost unaffected by the key length. Although the number of bit comparisons increases with the length of the key, the number of nodes visited remains unchanged. The performance of the linked list has little to do with the length of the key, and the search efficiency is mainly affected by the size of the data set

## Impact of data sorting

The test results of sorted and unsorted data show that linked lists are insensitive to data order. Regardless of whether the data is ordered, the linked list needs to traverse each node one by one, so there is almost no difference in performance. However, the Patricia tree shows a slight advantage under ordered data, because ordered data can make the tree structure more compact and reduce unnecessary node visits and comparisons.

## Expected and actual findings

By comparing the expected theoretical time complexity with the actual test results, we found that the search performance of the linked list is consistent with expectations, with a time complexity of O(n). In the test, as the size of the data set increases, the number of node accesses and comparisons of the linked list increases linearly, showing an obvious performance bottleneck. The actual performance of the Patricia tree is also in line with expectations, with a complexity close to O(log n). Especially when dealing with large-scale, sparse data sets, the Patricia tree significantly reduces the number of node accesses by compressing prefixes, maintaining a high search efficiency.

## 5. Conclusion

In summary, the linked list performs well on small data sets, but as the data size increases, its performance drops rapidly, meeting the O(n) complexity expectation. In contrast, the Patricia tree shows better scalability on large-scale data sets, with a complexity close to O(log n). Actual tests have proved that the Patricia tree can effectively reduce the number of comparisons and node accesses, and is a better choice for dealing with large-scale, complex data sets.