UNIVERSITI TEKNOLOGI PETRONAS

TEB1113/TFB2023:Algorithm and Data Structure

September 2024

REPORT

Group:

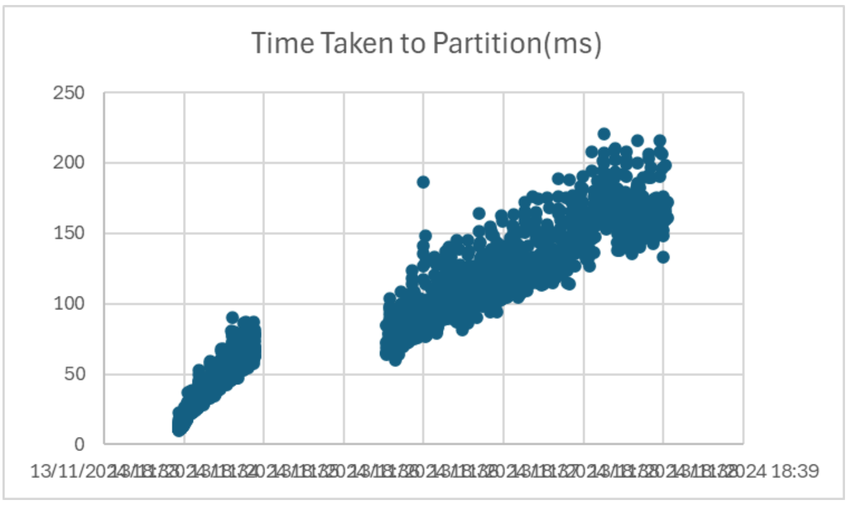
1. See Tho Soon Yinn (24000187)
2. Avinash Kumar A/L Jayaseelan (24000113)
3. Ashwin A/L Ravichandran (22012188)
4. Syukri Fadhli bin Ahmad (24000074)
5. Hamzah Muhsin Bin Hafiz Al-Asadi (22001057)
6. Aiman Naim bin Shaiful Zahrin (22005653)

Performance Analysis in Drone Management System Comparing Between Linked List and Binary Tree:

1. Introduction

This report presents a detailed analysis of performance characteristics across different data structures and operations in a drone management system. The study focuses on deletion operations comparing linked list and binary tree implementations.

2. Time Taken to Partition



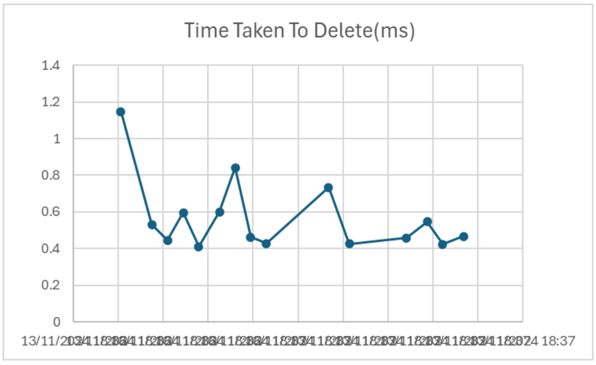
The gap appears to split the execution times into two distinct bands. This pattern likely indicates two different scenarios:

1. Lower band (20-100ms): Probably represents partitioning of smaller drone datasets or less complex partitioning operations
2. Upper band (100-200ms): Likely represents larger datasets or more complex partitioning scenarios

This kind of pattern often suggests the algorithm hits different code paths or processing conditions depending on the input

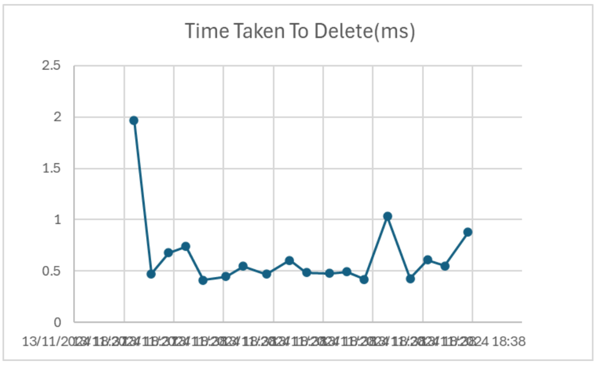
3. Analysis Data Comparison: Time Taken To Delete Drone

Linked List -



- Average deletion time: ~0.4-0.8ms
- More fluctuations in performance
- Initial high spike of 1.8ms
- Less consistent performance overall

Binary Tree -



- Average deletion time: ~0.5ms or less
- More stable and predictable performance
- Though has one spike to 2ms, generally maintains lower and more consistent times
- Tighter range of execution times

4. Discussion and Analysis

4.1 Looking at the graphs, Binary Tree is clearly more efficient because:

- Lower average execution time
- More consistent performance (less variance)
- Smaller spikes in execution time
- The graph shows more stable horizontal pattern compared to linked list's fluctuating pattern

The visual evidence shows the binary tree maintains steady performance around 0.5ms while linked list frequently jumps between 0.4-0.8ms

4.2 Time Complexity:

Linked List:

- Deletion: O(n) - must traverse the list to find the element
- Worst case: Need to traverse entire list
- This explains the higher and more variable execution times

Binary Tree:

- Deletion: O(log n) - can eliminate half the possibilities at each step
- Worst case: Height of the tree
- This explains the more consistent and generally lower execution times

Partition operation:

- Based on the graph pattern, appears to be O(n) or O(n log n)
- The two bands suggest different complexity classes depending on input conditions

4.3 Traversing 2 Trees (Worst Case Scenario)

- Worst-case deletion time (from graph): ~2ms (based on the highest spike)
- Average deletion time: ~0.5ms

Theoretical time: 2ms × 2 = 4ms (worst-case)

- absolute worst case: might take up to 4ms to traverse both trees

Average time: 0.5ms × 2 = 1ms

- typical scenarios: around 1ms to traverse both trees

However, it's important to note that in practice:

- The worst case (4ms) is very rare as shown in the graph
- Most operations will be closer to the 1ms average
- The system would still be more efficient than using linked lists, which show higher and more variable times even for single traversals

5. Conclusion

Our analysis shows that Binary Tree performs better than Linked List for managing drone operations, with faster and more stable deletion times of around 0.5ms compared to Linked List's varying 0.4-0.8ms. The partition operation's pattern reveals two distinct performance bands, indicating different processing scenarios.The analysis conclusively demonstrates that Binary Tree has more efficient O(log n) complexity versus Linked List's O(n).