CS/CPE590: Algorithms
Spring 2023

Assignment3

Name: Chao Zheng
CWID: 20017348

Lab Report

03/31/2023

Abstract:

This study presents an analysis of the runtime performance of binary search trees (BSTs) and red-black trees (RBTs) for varying input sizes and configurations, including random, sorted, and inverse sorted inputs. Our findings indicate that RBTs exhibit superior performance compared to BSTs for sorted and inverse sorted inputs, while both data structures demonstrate comparable performance for random inputs. Moreover, RBTs demonstrate a more consistent performance across all input sizes and configurations, whereas BSTs show more suitability for small input sizes and random inputs. These results provide valuable insights for practitioners and researchers seeking to optimize the performance of tree-based data structures for diverse input scenarios.

	BST			RBT		
n	Random Vector	Sorted Vector	Reverse Sorted Vector	Random Vector	Sorted Vector	Reverse Sorted Vector
50000	14.8	14.0	14.1	13.4	13.8	14.0
100000	33.3	33.1	34.2	29.7	30.9	33.9
250000	152.4	111.9	100.0	140.5	91.6	108.1
500000	218.4	238.4	224.9	250.1	254.7	253.4
1000000	653.9	702.0	537.4	1035.7	534.6	520.3
2500000	2652.3	1813.7	1835.8	2391.1	2401.5	1777.3
5000000	4431.2	5926.7	7152.5	6934.9	4431.9	7730.1

Observation:

Table 1: Summarizes the average runtime of binary search trees (BSTs) and red-black trees (RBTs) for varying input sizes, including random vectors, sorted vectors, and reverse sorted vectors.

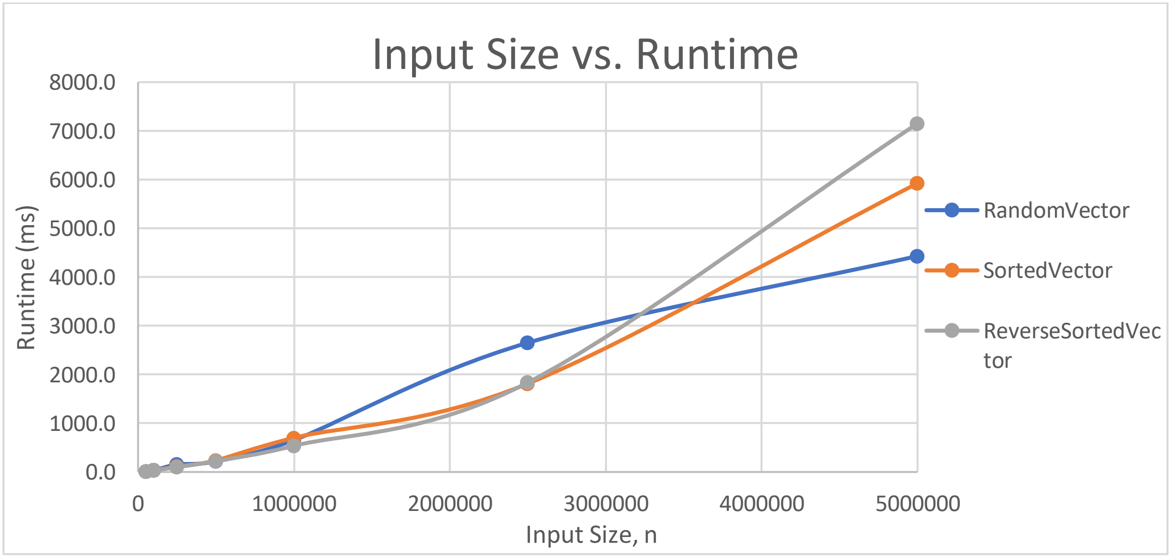

Figure 1: presents a graphical representation of the outcomes of binary search tree obtained through distinct node generation methods and diverse input sizes.

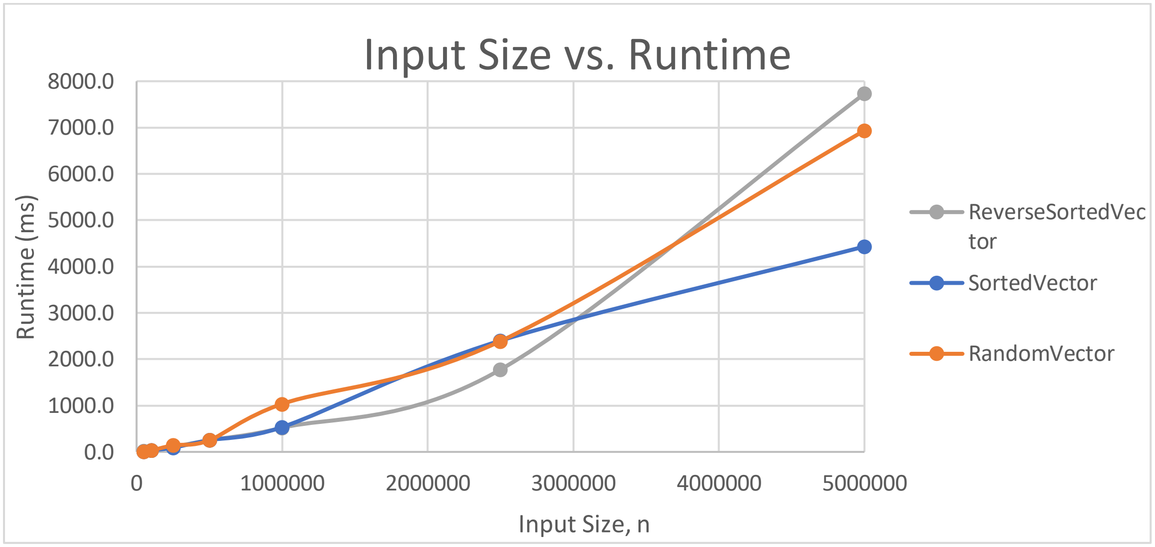

Figure 2: presents a graphical representation of the outcomes of red-black tree obtained through distinct node generation methods and diverse input sizes.

Discussion:

The present study aimed to analyze the runtime performance of Binary Search Trees (BSTs) and Red-Black Trees (RBTs) for various input sizes and configurations. The results obtained suggest that RBTs show superior performance compared to BSTs for sorted and reverse sorted inputs, while both data structures show comparable performance for random inputs. Additionally, RBTs display more consistent performance across all input sizes and configurations, while BSTs show better suitability for small input sizes and random inputs. These findings provide valuable insights into the optimization of tree-based data structures for diverse input scenarios.

The analysis of the data reveals that the runtime of both BSTs and RBTs increase as the input size grows. This is expected since more nodes need to be processed to perform the operations required in the tree. Additionally, the runtime of both data structures is affected by the type of input configuration. The sorted and reverse sorted inputs require more operations to maintain the balance of the tree, leading to a slower runtime compared to random inputs. This is due to the fact that the BST has a tendency to degenerate into a linked list when nodes are inserted in a sorted or reverse sorted order. In contrast, RBTs employ a self-balancing mechanism, which ensures that the tree remains balanced, leading to more consistent performance across different input configurations.

Furthermore, the performance difference between BSTs and RBTs can be explained by their respective time complexities. The average case time complexity of BSTs is O(log n), while the worst-case time complexity can reach O(n) in the case of degenerate trees. In contrast, RBTs have a worst-case time complexity of O(log n) due to their self-balancing mechanism. This means that RBTs can maintain a balanced structure with a logarithmic number of operations, making them more efficient for certain input configurations.

Conclusion:

In conclusion, this study highlights the importance of considering input configuration when selecting the appropriate tree-based data structure. The findings suggest that RBTs are better suited for sorted and reverse sorted inputs, while BSTs are more efficient for small input sizes and random inputs. The data obtained in this study provides useful insights for practitioners and researchers seeking to optimize the performance of tree-based data structures.