***Abstract:***

The purpose of this assignment is to draw a comparison between sorting three different types of arrays i.e., random generated, sorted, and inverse sorted array using Binary Search Tree (BST) and Red Black Tree (RBT) and to evaluate the time complexity of each of these strategies.

***Results:***

|  |  |  |
| --- | --- | --- |
| **N** | **BST** | **RBT** |
| 50,000 | 18ms | 14ms |
| 100,000 | 40ms | 44ms |
| 250,000 | 121ms | 114ms |
| 500,000 | 303ms | 301ms |
| 1,000,000 | 760ms | 733ms |
| 2,500,000 | 2476ms | 2374ms |
| 5,000,000 | 5981ms | 5573ms |

**Table 1: Sort Time Using BST and RST for random generated numbers in random order.**

|  |  |  |
| --- | --- | --- |
| **N** | **BST** | **RBT** |
| 50,000 | 1ms | 0ms |
| 100,000 | 43ms | 0ms |
| 250,000 | 195ms | 2ms |
| 500,000 | 390ms | 3ms |
| 1,000,000 | 688ms | 5ms |
| 2,500,000 | 1142ms | 7ms |
| 5,000,000 | 1765ms | 10ms |

**Table 2: Sort Time Using BST and RST for random generated numbers in sorted order.**

|  |  |  |
| --- | --- | --- |
| **N** | **BST** | **RBT** |
| 50,000 | 1ms | 0ms |
| 100,000 | 53ms | 1ms |
| 250,000 | 185ms | 1ms |
| 500,000 | 368ms | 3ms |
| 1,000,000 | 742ms | 4ms |
| 2,500,000 | 1113ms | 4ms |
| 5,000,000 | 1701ms | 8ms |

**Table 3: Sort Time Using BST and RST for random generated numbers in inverse sorted order.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **Duplicate\_BST** | **Duplicate\_RBT** | **Case1** | **Case2** | **Case3** | **L Rotate** | **R Rotate** |
| 50,000 | 0 | 0 | 914 | 0 | 912 | 0 | 914 |
| 100,000 | 0 | 0 | 3750 | 0 | 3755 | 0 | 3751 |
| 250,000 | 0 | 0 | 7584 | 0 | 7579 | 0 | 7580 |
| 500,000 | 0 | 0 | 15002 | 0 | 15010 | 0 | 15005 |
| 1,000,000 | 0 | 0 | 22013 | 0 | 21995 | 0 | 21993 |
| 2,500,000 | 0 | 0 | 19015 | 0 | 19017 | 0 | 19018 |
| 5,000,000 | 0 | 0 | 32487 | 0 | 32488 | 0 | 32495 |

**Table 4: Average Counter Values for RBT & BST in reverse sorted order**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **Duplicate\_BST** | **Duplicate\_RBT** | **Case1** | **Case2** | **Case3** | **L Rotate** | **R Rotate** |
| 50,000 | 0 | 0 | 924 | 0 | 912 | 944 | 0 |
| 100,000 | 0 | 0 | 3741 | 0 | 3755 | 3745 | 0 |
| 250,000 | 0 | 0 | 7518 | 0 | 7579 | 7580 | 0 |
| 500,000 | 0 | 0 | 14995 | 0 | 14978 | 15005 | 0 |
| 1,000,000 | 0 | 0 | 22013 | 0 | 21995 | 21993 | 0 |
| 2,500,000 | 0 | 0 | 19015 | 0 | 19017 | 19018 | 0 |
| 5,000,000 | 0 | 0 | 29847 | 0 | 31488 | 32495 | 0 |

**Table 5: Average Counter Values for RBT & BST in sorted order**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **N** | **Duplicate\_BST** | **Duplicate\_RBT** | **Case1** | **Case2** | **Case3** | **L Rotate** | **R Rotate** |
| 50,000 | 1 | 1 | 16845 | 9703 | 19388 | 14567 | 14524 |
| 100,000 | 1 | 3 | 45982 | 17549 | 38903 | 29259 | 29153 |
| 250,000 | 21 | 45 | 148795 | 41359 | 78236 | 57214 | 72801 |
| 500,000 | 55 | 13 | 214798 | 87381 | 184658 | 169145 | 146154 |
| 1,000,000 | 198 | 245 | 513040 | 124900 | 387145 | 319584 | 290178 |
| 2,500,000 | 1498 | 1554 | 1256014 | 465926 | 983269 | 721650 | 721171 |
| 5,000,000 | 5140 | 5798 | 2103598 | 966900 | 1939640 | 1652419 | 1455228 |

**Table 6: Average Counter Values for RBT & BST in random order**

***Graphical Visualization:***

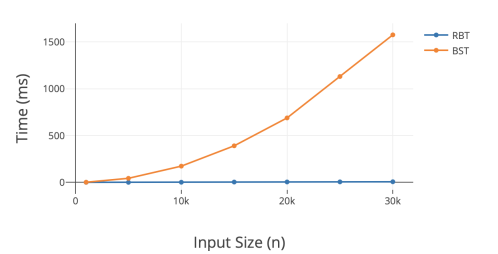


Fig 1: Reverse Sorted Array Time

A graph with a line and numbers

Description automatically generated with medium confidence

Fig 2: Random Sorted Array Time

A graph with a line and a line

Description automatically generated

Fig 3: Pre-Sorted Array Time.

***Evaluation of Results:***

The result shows that there is no significant difference in time taken by BST and RBT algorithms in case of randomly generated as both trees will have a maximum height of lgn, and the in-order tree walk would take O(lgn)

However, incase of sorted and inverse sorted, time complexity would have a significant impact in case of BST as the tree is going to be completely either left or right skewed making the inorder tree walk algorithm execution time to increase upto O(n) which is not true in case of RBT as the RBT is going to balance the tree upon each insertion, which is also evident from Fig 1 & 3 showing little to no impact on the time complexity of sorting the array using RBT.

Table 4-6 shows the average counter values of duplicates in RBT, BST, case 1, case 2 and case 3 for RBT as well as left and right rotation performed by RBT algorithm to balance the tree. It can be observed that there are no instance of case 2 and left rotation for RBT incase of reverse sorted array due to higher probability of case 3 and case 1 to be executed as they involve the left subtree. Same is also true except for the absence of right rotation in case of pre-sorted array.