**CS 590 Homework 3  
Name – Yash Dinkar Bharambay**

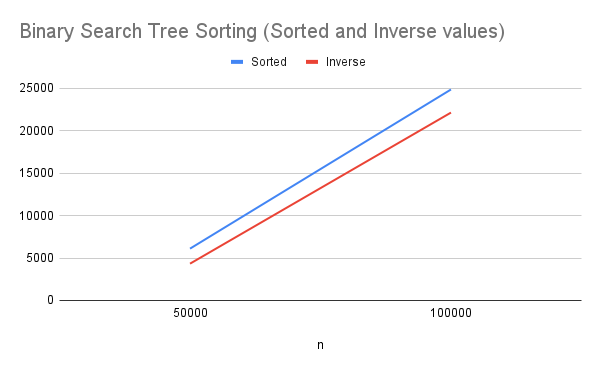
**Abstract –** In this homework we will analyze binary search tree sorting and red black tree sort.

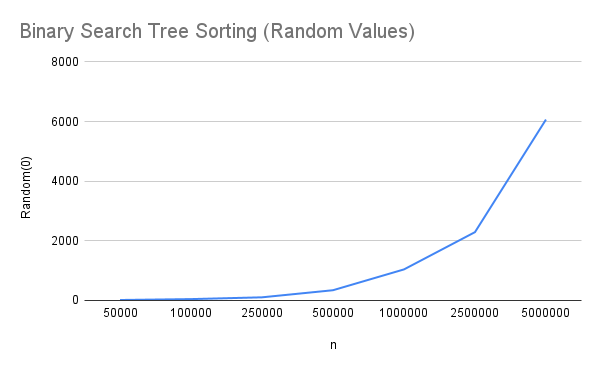
**Tables -**

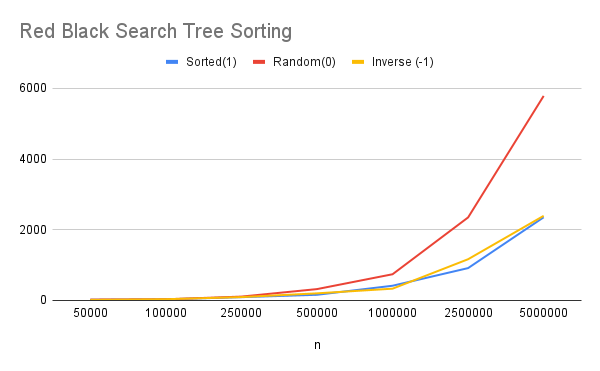
|  |  |  |  |
| --- | --- | --- | --- |
| **Binary Search Tree Sorting** | | | |
|  |  | **Time(ms)** |  |
| **n** | **Sorted(1)** | **Random(0)** | **Inverse (-1)** |
| 50000 | 6105ms | 15 | 4334 |
| 100000 | 24858ms | 45 | 22137 |
| 250000 | - | 105 | - |
| 500000 | - | 343 | - |
| 1000000 | - | 1041 | - |
| 2500000 | - | 2291 | - |
| 5000000 | - | 6067 | - |
|  |  |  |  |
|  |  | **Duplicates** |  |
| **n** | **Sorted(1)** | **Random(0)** | **Inverse sorted(-1)** |
| 50000 | 0 | 2 | 0 |
| 100000 | 0 | 2 | 0 |
| 250000 | - | 14 | - |
| 500000 | - | 60 | - |
| 1000000 | - | 198 | - |
| 2500000 | - | 1478 | - |
| 5000000 | - | 5763 | - |

|  |  |  |  |
| --- | --- | --- | --- |
| **Red Black Search Tree Sorting** | | | |
|  |  | **Time(ms)** |  |
| **n** | **Sorted(1)** | **Random(0)** | **Inverse (-1)** |
| 50000 | 18 | 24 | 15 |
| 100000 | 31 | 33 | 35 |
| 250000 | 93 | 109 | 95 |
| 500000 | 160 | 321 | 201 |
| 1000000 | 416 | 741 | 333 |
| 2500000 | 915 | 2348 | 1166 |
| 5000000 | 2352 | 5784 | 2393 |
|  |  |  |  |
|  |  |  |  |
|  |  | **Parameter** |  |
| **n** | **Sorted(1)** | **Random(0)** | **Inverse sorted(-1)** |
| 50000 | Case 1: 25592 Case 2: 9796 Case 3: 19515 Left Rotate: 14611 Right Rotate: 14700 Duplicates: 0 | Case 1: 25592 Case 2: 9796 Case 3: 19515 Left Rotate: 14611 Right Rotate: 14700 Duplicates: 0 | Case 1: 49966 Case 2: 0 Case 3: 49971 Left Rotate: 0 Right Rotate: 49971 Duplicates: 0 |
| 100000 | Case 1: 99964 Case 2: 0 Case 3: 99969 Left Rotate: 99969 Right Rotate: 0 | Case 1: 51374 Case 2: 19436 Case 3: 38763 Left Rotate: 29082 Right Rotate: 29117 | Case 1: 99964 Case 2: 0 Case 3: 99969 Left Rotate: 0 Right Rotate: 99969 |
| 250000 | Case 1: 249961 Case 2: 0 Case 3: 249967 Left Rotate: 249967 Right Rotate: 0 Duplicates: 0 | Case 1: 128268 Case 2: 48704 Case 3: 97083 Left Rotate: 72771 Right Rotate: 73016 Duplicates: 5 | Case 1: 249961 Case 2: 0 Case 3: 249967 Left Rotate: 0 Right Rotate: 249967 Duplicates: 0 |
| 500000 | Case 1: 499959 Case 2: 0 Case 3: 499965 Left Rotate: 499965 Right Rotate: 0 Duplicates: 0 | Case 1: 256553 Case 2: 96975 Case 3: 194567 Left Rotate: 146001 Right Rotate: 145541 Duplicates: 54 | Case 1: 499959 Case 2: 0 Case 3: 499965 Left Rotate: 0 Right Rotate: 499965 Duplicates: 0 |
| 1000000 | Case 1: 999957 Case 2: 0 Case 3: 999963 Left Rotate: 999963 Right Rotate: 0 Duplicates: 0 | Case 1: 513168 Case 2: 193882 Case 3: 388167 Left Rotate: 291181 Right Rotate: 290868 Duplicates: 238 | Case 1: 999957 Case 2: 0 Case 3: 999963 Left Rotate: 0 Right Rotate: 999963 Duplicates: 0 |
| 2500000 | Case 1: 2499952 Case 2: 0 Case 3: 2499960 Left Rotate: 2499960 Right Rotate: 0 Duplicates: 0 | Case 1: 1282576 Case 2: 485224 Case 3: 970557 Left Rotate: 728154 Right Rotate: 727627 Duplicates: 1430 | Case 1: 2499952 Case 2: 0 Case 3: 2499960 Left Rotate: 0 Right Rotate: 2499960 Duplicates: 0 |
| 5000000 | Case 1: 4999950 Case 2: 0 Case 3: 4999958 Left Rotate: 4999958 Right Rotate: 0 Duplicates: 0 | Case 1: 2564559 Case 2: 970055 Case 3: 1940507 Left Rotate: 1455442 Right Rotate: 1455120 Duplicates: 5956 | Case 1: 4999950 Case 2: 0 Case 3: 4999958 Left Rotate: 0 Right Rotate: 4999958 Duplicates: 0 |

**GRAPHS –**

****

****

****

**Discussion –**

* Both the graphs and the tabular data show that when dealing with the worst-case scenarios, Binary Search Tree takes a very long time. However, for random, the time required is much less because the tree height is O(logN), which is far smaller than worst-case scenarios and is O(N) for inorder tree traversal.
* For Red-black trees, as we can see in the graphs and tabular data, the time required for worst-case scenarios is far shorter than the time required for BST. This is due to the fact that the traversal in the worst case scenario of RBT is still O(logN), and as the data grows, the time complexity of O(logN) makes traversal quick.
* The time required for Random in the case of RBT is almost similar to that of BST. This is expected because the traversal for

both the situations is O(logN). Also, we see that the duplicates for BST and RBT when Sorted and Inverse Sorted are almost all 0, while there are a few for Random. This is a behavior that is expected.

* In RBT, the left and right rotations for random are similar in terms of numbers, whereas for sorted and inverse sorted, there are only right rotations and no left rotations. Cases 1 and 3 are present for both sorted and inverse sorted, while case 2 remains zero.

**Conclusion –**

Hence the analysis of practical coding is in accordance with our theoretical knowledge that we gained in our class lectures.