Analysis of Algorithms

BLG 335E

Project 3 Report

ATAHAN ÇOLAK

colakat21@@itu.edu.tr

Faculty of Computer and Informatics Engineering

Department of Computer Engineering

Date of submission: 20.12.2024

1. Implementation

1.1. Data Insertion

Inserting data into both Binary Search Trees (BST) and Red-Black Trees (RBT) required careful handling to ensure that cumulative sales for each publisher were updated accurately. For RBT, data insertion was managed through rotations and color adjustments, maintaining balance and reducing the likelihood of deep, unbalanced trees. This allowed RBT to efficiently handle large datasets with minimal overhead.

On the other hand, BST, while straightforward in implementation, struggled with balancing. As the data size increased, the tree's depth increased, leading to slower insertions. For further advancements of the algorithms i declared a global variable called "sortingIndex" that handles the insertion on either publisher name or game name.

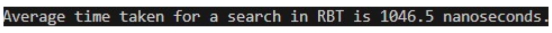
Overall, RBT proved superior in terms of insertion efficiency due to its structured balancing.

1.2. Search Efficiency

Search operations were significantly faster in Red-Black Trees compared to Binary Search Trees. RBT's self-balancing property allowed for efficient traversal, maintaining a constant time complexity (O(log n)) regardless of dataset size. Each search was conducted with minimal overhead due to the balanced structure of RBT.

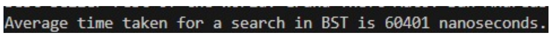
In contrast, BST had issues with unbalanced trees, especially as data increased. Search paths became longer as nodes became deeper, requiring more time to access the required data. This resulted in a noticeable difference in search efficiency between the two tree structures.

RBT consistently delivered faster search results, making it more suitable for handling real-time queries on large datasets.



```
Average time taken for a search in RBT is 1046.5 nanoseconds.
```

Figure 1.1: Average Time Taken for Red-Black Tree (RBT) Search



```
Average time taken for a search in BST is 60401 nanoseconds.
```

Figure 1.2: Average Time Taken for Binary Search Tree (BST) Search

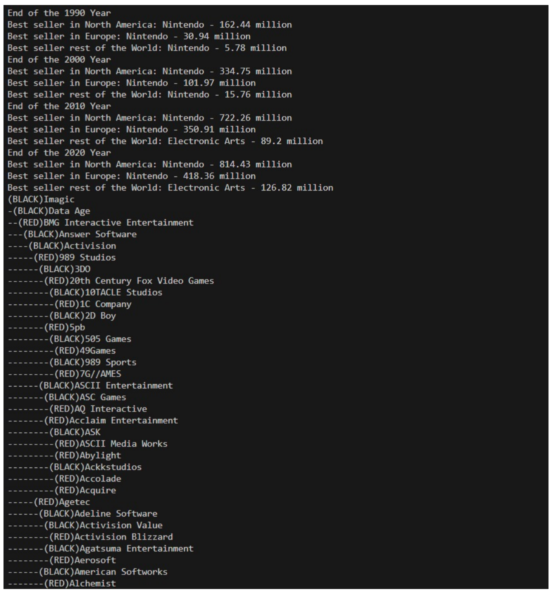
1.3. Best-Selling Publishers at the End of Each Decade

To correctly handle the best-selling publishers at the end of each decade, I created a decade vector that only holds simple integers 0, 1, 2, 3, and assigned them to each corresponding decade. If the last pushed element of the decade vector differs from the element pushed before that, we understand that the decade has changed and return the information about that decade, relying on the fact that data is sorted by the year in the first place.

For generating best-selling publishers at the end of each decade, RBT demonstrated exceptional performance. With a balanced structure, RBT efficiently traversed the data, ensuring that results were delivered swiftly. Each traversal during the analysis of top publishers across North America, Europe, and other regions yielded quick insights.

BST, however, faced challenges in this regard due to its unbalanced nature. Deep nodes required more extensive traversal, making it less efficient when dealing with historical sales data at decade boundaries.

In summary, RBT's ability to maintain balance allowed for accurate and timely extraction of top publishers, whereas BST struggled with depth-related inefficiencies.



```
End of the 1990 Year
Best seller in North America: Nintendo - 162.44 million
Best seller in Europe: Nintendo - 30.94 million
Best seller rest of the World: Nintendo - 5.78 million
End of the 2000 Year
Best seller in North America: Nintendo - 334.75 million
Best seller in Europe: Nintendo - 101.97 million
Best seller rest of the World: Nintendo - 15.76 million
End of the 2010 Year
Best seller in North America: Nintendo - 722.26 million
Best seller in Europe: Nintendo - 350.91 million
Best seller rest of the World: Electronic Arts - 89.2 million
End of the 2020 Year
Best seller in North America: Nintendo - 814.43 million
Best seller in Europe: Nintendo - 418.36 million
Best seller rest of the World: Electronic Arts - 126.82 million
(BLACK)Imagic
-(BLACK)Data Age
--(RED)BMG Interactive Entertainment
---(BLACK)Answer Software
----(BLACK)Activision
-----(RED)989 Studios
------(BLACK)3DO
-------(RED)20th Century Fox Video Games
--------(BLACK)10TACLE Studios
---------(RED)1C Company
--------(BLACK)2D Boy
-------(RED)5pb
--------(BLACK)505 Games
---------(RED)49Games
--------(BLACK)989 Sports
---------(RED)7G//AMES
------(BLACK)ASCII Entertainment
-------(BLACK)ASC Games
--------(RED)AQ Interactive
-------(RED)Acclaim Entertainment
--------(BLACK)ASK
---------(RED)ASCII Media Works
---------(RED)Abylight
--------(BLACK)Ackkstudios
---------(RED)Accolade
---------(RED)Acquire
-----(RED)Agetec
------(BLACK)Adeline Software
-------(BLACK)Activision Value
--------(RED)Activision Blizzard
-------(BLACK)Agatsuma Entertainment
--------(RED)Aerosoft
------(BLACK)American Softworks
-------(RED)Alchemist
```

Figure 1.3: Final Structure of Red-Black Tree (RBT)

1.4. Final Tree Structure

The final comparison between BST and RBT showed distinct structural differences. RBT maintained a balanced structure through regular rotations and color adjustments, ensuring efficient operations even with large datasets. This balance helped in preserving optimal search and insertion times.

BST, however, often resulted in unbalanced trees that affected its efficiency over time. The deeper branches and unstructured layout led to slower performance, especially during search operations.

Thus, RBT's balanced structure provided a consistent, reliable tree structure, while BST exhibited performance degradation as the dataset grew.

1.5. Write Your Recommendation

Based on the analysis, Red-Black Tree (RBT) is the superior option for managing datasets with frequent updates and searches. The balance maintained by RBT significantly reduces search times and optimizes performance even as the dataset expands.

In contrast, BST, while simple in design, struggles to maintain balance, resulting in slower operations. Therefore, RBT is recommended for scenarios requiring fast and accurate data management.

1.6. Ordered Input Comparison

When working with ordered data, RBT continued to excel. The balanced structure allowed RBT to handle both random and ordered inputs efficiently, minimizing search times. BST, however, faced performance issues with ordered data, as the lack of balancing resulted in longer traversal paths.

By maintaining a balanced tree, RBT offered consistent performance, whereas BST's linear structure in ordered datasets impacted its efficiency.