Analysis of Algorithms

BLG 335E

Project 3 Report

Melike Beşparmak
besparmak22@itu.edu.tr
150220061

Faculty of Computer and Informatics Engineering
Department of Computer Engineering
Date of submission: 20/12/2024

1. Implementation

1.1. Data Insertion

1.1.1. Binary Search Tree

Data insertion in a Binary Search Tree (BST) follows two rules: every element in the left subtree is smaller than the root node, and every element in the right subtree is greater. Initially, the root node is initialized with the first line of the data read. This root node serves as the starting point of the tree and is used for every search, insertion, or deletion operation. For any subsequent insertion, the process begins by comparing the key of the root node and continues searching in the correct subtree until the appropriate position for the new node is found.

Data insertion in a BST takes $O(h)$ time, where h is the height of the tree. The structure of a BST heavily depends on the order of the input data. If the input data is nearly sorted, the tree structure will resemble a linked list or a chain, causing an unbalanced height and inefficiencies. In the worst case, the height is N (the number of nodes), while the expected height is $\log N$. This implies that the expected time complexity to insert a new node is $O(\log N)$, but it can degrade to $O(N)$ in the worst-case scenario.

1.1.2. Red-Black Tree

Red-Black Trees (RBTs) are BSTs with additional properties designed to maintain a balanced structure. These properties are as follows:

- Each node is either red or black.
- The root node is always black.
- The black height (the number of black nodes from a node to its descendant NIL nodes) of every node is the same.
- A red node cannot have a red child.
- Every NIL (leaf) node is black.

These rules ensure that the height of the tree remains $O(\log N)$ during insertion operations, making the structure immune to the order of the input data.

To maintain the balanced state of the RBT, two procedures are used: recoloring and rotation. Each new node is inserted as a red node, and its position is determined in the same way as in a BST. The reason for inserting a new node as red is to preserve the black height property. Since the tree is a valid RBT before insertion, the black height is consistent for all paths. Adding a black node to a subtree would violate this property. However, it is not guaranteed that the tree will remain a valid RBT after inserting a red node, where additional adjustments are necessary.

The `insertFix` function is called after every insertion to correct any violations. If the parent of the new node is black, no violations occur, and no recoloring or rotations are needed. However, if the parent is red, modifications are required. At this point, the color of the uncle node (or aunt node) must be checked. The uncle node is the sibling of the parent node, and NIL values are considered black. If the uncle node is red, a recoloring operation is performed, and the `insertFix` iterates for the grandparent node to re-evaluate the RBT for upper levels. If the uncle node is black, recoloring alone is insufficient, and rotation operations are necessary.

There are two types of rotation operations: left rotation and right rotation. Both adjust the relationships between nodes to restore the balance of the tree. For example, if the new node is a right child, a left rotation is applied to adjust the tree structure.

1.2. Search Efficiency

As explained in detail in the previous section, both of the data structures are Binary Trees and the expected time complexity for the search operation is $O(\log N)$. However, since BSTs are vulnerable due to their dependence on the order of the input data, this complexity can go up to $O(N)$. On the other hand, RBTs are modified BSTs that maintain a balanced structure, ensuring more efficient search operations with a guaranteed time complexity of $O(\log N)$.

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

This section was the same for both parts. I printed the publishers during the creation of the trees from the CSV file. The current and previous decades are compared, and at each new line of data, as well as at the end of each decade, the "find best seller" function is called. This function performs a pre-order traversal of the tree to find the best sellers. Regardless of the traversal order, since each node must be visited, this task takes $O(N)$ time.

1.4. Final Tree Structure

RBTs are more balanced than BSTs due to their additional properties. In contrast, BSTs do not modify the nodes already in the tree, making them a more static data structure.

1.5. Write Your Recommendation

If the input data is not guaranteed to be uniform and there is a risk of it being nearly sorted, the use of RBTs should be preferred. In such cases, the BST may perform poorly. In other scenarios, BSTs can be used since their implementation is straightforward and requires less effort.

1.6. Ordered Input Comparison

The data was sorted by game name, and the results can be seen in Figure 1.3. The BST being less efficient than the RBT was the expected outcome, but since the search is

done randomly each time the results may differ as in Figure 1.1 and 1.2. It is seen that BST search takes more time if the data is ordered when compared to unordered data. Also, the key used in nodes are publishers, not game names. Since the assignment file asks to sort the data according to the game name, it was implemented that way. Theoretically, the data being ordered does not affect the RBT and should be avoided for BST.

```
PS C:\Users\melik\Desktop\algo\hw3_updated\HW3_BLG335E> ./bst
End of the 1990 Year
Best seller in North America: Nintendo - 140.27 million
Best seller in Europe: Nintendo - 23.61 million
Best seller rest of the World: Nintendo - 4.43 million
End of the 2000 Year
Best seller in North America: Nintendo - 319.75 million
Best seller in Europe: Nintendo - 95.55 million
Best seller rest of the World: Nintendo - 13.69 million
End of the 2010 Year
Best seller in North America: Nintendo - 698.7 million
Best seller in Europe: Nintendo - 334.97 million
Best seller rest of the World: Electronic Arts - 79.1 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
Execution time to create BST from CSV: 84316 microseconds
Average execution time to search 50 random publishers: 424 ns
Execution time to create new tree with ordered data: 97777 microseconds
Average execution time to search 50 random publishers with ordered data: 628 ns
```

Figure 1.1: Binary Search Tree

```
PS C:\Users\melik\Desktop\algo\hw3_updated\HW3_BLG335E> ./rbt
End of the 1990 Year
Best seller in North America: Nintendo - 140.27 million
Best seller in Europe: Nintendo - 23.61 million
Best seller rest of the World: Nintendo - 4.43 million
End of the 2000 Year
Best seller in North America: Nintendo - 319.75 million
Best seller in Europe: Nintendo - 95.55 million
Best seller rest of the World: Nintendo - 13.69 million
End of the 2010 Year
Best seller in North America: Nintendo - 698.7 million
Best seller in Europe: Nintendo - 334.97 million
Best seller rest of the World: Electronic Arts - 79.1 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
Execution time to create RBT from CSV: 110477 microseconds
Average execution time to search 50 random publishers: 570 ns
Execution time to create new tree with ordered data: 65372 microseconds
Average execution time to search 50 random publishers with ordered data: 434 ns
```

Figure 1.2: Red-Black Tree

```
(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
------------(BLACK) Aksys Games
```

Figure 1.3: Preorder Traversal