**Engineering Method**

1. **Identification of the problem**

The International Basketball Federation (FIBA) has realized the importance of data in order to predict different outcomes in the matches. In addition to this, the tremendous amount of information that can be collected about each player makes this a powerful tool. Because of this, they need a software to save the information about the players and make it easy to access, all of this to get to know patterns about the development of this sport, the stronger criteria and generally, where basketball is heading nowadays.

**Problem:** enter a great amount of data and be able to make quick consultations according to the criteria selected, the value of comparison and an operator for making the comparison (<, >, =).

1. **Information collection**

**Binary Search Tree (BTS):** A binary search tree is a rooted binary tree, whose internal nodes each store a key (and optionally, an associated value), and each has two distinguished sub-trees, commonly denoted *left* and *right*. The tree additionally satisfies the binary search property: the key in each node is greater than or equal to any key stored in the left sub-tree, and less than or equal to any key stored in the right subtree.

**BST Concepts**

**Root node:** it’s the top node in the hierarchy

**Child node:** it has exactly one Parent node

**Parent node:** it has at most two child nodes

**Sibling nodes:** they share the same Parent node

**Leaf node:** it has no child nodes

Every node in the BST is a Subtree of the BST rooted at that node

**In-order traversal:** it’s a way to traverse a BST in which the result is the nodes in descending order. This traversal visits the left child first, then its father and finally the right child.

Source: https://www.cs.swarthmore.edu/~newhall/unixhelp/Java\_bst.pdf

**Self-Balancing Binary Search Tree (AVL):** An AVL tree is a binary search tree that is height balanced: for each node x, the heights of the left and right subtrees of x differ by at most 1. To implement an AVL tree, we maintain an extra attribute in each node: x:h is the height of node x. As for any other binary search tree T, we assume that T:root points to the root node.

Source: Introduction to Algorithms (2009), Cormen

**AVL Rotations**

**Left Rotation:** If a tree becomes unbalanced, when a node is inserted into the right subtree of the right subtree, then we perform a single left rotation.

**Right Rotation:** AVL tree may become unbalanced, if a node is inserted in the left subtree of the left subtree. The tree then needs a right rotation.

**Left-Right Rotation:** A left-right rotation is a combination of left rotation followed by right rotation.

**Right- Left Rotation:** The second type of double rotation is Right-Left Rotation. It is a combination of right rotation followed by left rotation.

Source: https://www.tutorialspoint.com/data\_structures\_algorithms/avl\_tree\_algorithm.htm

**Red-Black Tree:** A red-black tree is a kind of self-balancing binary search tree where each node has an extra bit, and that bit is often interpreted as the colour (red or black). These colours are used to ensure that the tree remains balanced during insertions and deletions. Although the balance of the tree is not perfect, it is good enough to reduce the searching time and maintain it around O(log n) time, where n is the total number of elements in the tree.

#### **Rules That Every Red-Black Tree Follows:**

* 1. Every node has a colour either red or black.
  2. The root of the tree is always black.
  3. There are no two adjacent red nodes (A red node cannot have a red parent or red child).
  4. Every path from a node (including root) to any of its descendants NULL nodes has the same number of black nodes.

Source: https://www.geeksforgeeks.org/red-black-tree-set-1-introduction-2/

1. **Search for creative solutions**

**Alternative 1**

The data is obtained from a .csv file, and through the Buffered Reader each line is read, in order to create objects of a class Player. These objects are stored in an Arraylist in order to perform a binary search, due to the different criteria to classify the players, the arraylist must be sorted according to the selected criterion at the moment of consulting.

**Alternative 2**

The data is obtained from a .csv file, and through the Buffered Reader each line is read, in order to create objects of a class Player. All of these players are stored in a double linked list, in order to perform different operations on it. By this way, it’s not necessary to sort the elements when consulting or performing another operation.

**Alternative 3**

The data is obtained from a .csv file, and through the Buffered Reader each line is read, in order to create objects of a class Player. The objects are mapped in a Hashtable, the hash function is defined according to the available data. When consulting the hashtable, it is required to perform the hash calculus to know the position in the map and access to the element quickly.

**Alternative 4**

The data is obtained from a .csv file, and through the Buffered Reader each line is read, in order to create objects of a class Player. These objects are stored in different binary search trees, one for each criterion, then a search is performed in the desired BST and the elements that meet the criterion are accessed.

**Alternative 5**

The data is obtained from a .csv file, and through the Buffered Reader each line is read, in order to create objects of a class Player. The players are stored with its index and the key (criterion) in balanced search trees and red-black trees and all of them are stored in an arraylist. The search is performed in the mentioned structures, and when the players that meet the criterion are found, they are accessed quickly with the index in an arraylist with all the players.

**Alternative 6**

The data is obtained from a .csv file, and through the Buffered Reader each line is read, in order to create objects of a class Player. The players are stored with its index and the key (criterion) in balanced search trees and red-black trees and all of them are stored in an arraylist. The search is performed in the mentioned structures, and when the players that meet the criterion are found, they are accessed quickly with the index in an arraylist with all the players.

1. **Moving from ideas to preliminary designs:**

First, it’s necessary to discard alternatives 1 and 2 since they don’t solve the problem in a feasible way. In the case of the alternative 1, even though the binary search is a good option, when more than one player meets the criterion, logical errors may occur and the necessity of sorting with every criterion isn’t effective.

For the alternative 2, the problem of sorting the elements isn’t present but the consult in a double linked list is O(n) in the worst case, taking into account the amount of data going to be managed, it must be discarded.

1. **Evaluation and selection of the best solution**

**Criterias:**

* Solution precision: The solution is viable and meets the effective development of all the requirements of the problem

1. Imprecise
2. Almost Accurate
3. Accurate

* Efficiency: Regardless of the complexity, the solution complies with a perfect effectiveness lacking any logical error and presents fast response

1. No Effective
2. Low effectiveness
3. Medium effectiveness
4. Effective

* Usability: The solution can be used because it meets all the proposed objectives

1. Not usable at all
2. Complex
3. Usable

* Accessible: The solution presents easy access to the data generated

1. Not accessible
2. Moderately accessible
3. Accessible

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Effectiveness | Usability | Accessible |
| Alternative 3 | 2 | 3 | 2 | 1 |
| Alternative 4 | 3 | 3 | 2 | 2 |
| Alternative 5 | 3 | 4 | 2 | 3 |
| Alternative 6 |  |  |  |  |

**Alternative 3:** 8 points

**Alternative 4:** 10 points

**Alternative 5:** 12 points

According to the previous evaluation, Alternative 5 should be selected, since it obtained the highest score according to defined criteria. It’s the best way to accomplish the goals for the project in relation to time complexity and effectiveness.