The engineering method

Contenido

[Context of the problem 2](#_Toc85313497)

[Identifying the problem 2](#_Toc85313498)

[Identification of needs and symptoms 2](#_Toc85313499)

[Definition of the problem 2](#_Toc85313500)

[Data collection 3](#_Toc85313501)

[Binary search trees 3](#_Toc85313502)

[Secondary memory 3](#_Toc85313503)

[Primary memory 3](#_Toc85313504)

[Red, black tree 3](#_Toc85313505)

[Search for creative solutions 3](#_Toc85313506)

[Transition of the formulation of ideas to preliminary design ideas 4](#_Toc85313507)

[Idea discard 4](#_Toc85313508)

[Alternative 2: 4](#_Toc85313509)

[Alternative 3: 4](#_Toc85313510)

[Alternative 4: 4](#_Toc85313511)

[Alternative 5: 5](#_Toc85313512)

[Alternative 6: 5](#_Toc85313513)

[Evaluation and selection of the best solution 5](#_Toc85313514)

[Criteria 5](#_Toc85313515)

[Evaluation 6](#_Toc85313516)

[Selection 6](#_Toc85313517)

[Preparation of reports and specifications 6](#_Toc85313518)

[Specification of the problem 6](#_Toc85313519)

[Requirements 6](#_Toc85313520)

[Nonfunctional requirements 6](#_Toc85313521)

[Considerations 6](#_Toc85313522)

[Implementation of the design 7](#_Toc85313523)

[Bibliografía 7](#_Toc85313524)

# Context of the problem

Basketball is one of the most important and popular sports around the globe, given that it’s wildly popular and it creates great spectacles. Having said this, recently the International Basketball federation (FIBA) has been overwhelmed with the amount of data that can be considered useful, so they have decided to create an application that can manage the most important data for all professional players in the world so they can be analyzed.

# Identifying the problem

## Identification of needs and symptoms

* The program must be able to handle *at least 200000* players
* The program must manage the following data for each player
  + Name
  + Age
  + Team
  + And 5 other statistics
* The program must be able to search for specific statistics
  + For 4 statistics, the complexity of the search must be O (log n)
  + For the rest of the statistics, linear search is accepted
* Binary search Trees must be used
* The program must have a graphic interface
* The information must be saved in secondary memory

Bonus

* The use of red, black Trees
* The program can edit the data base through the interface

## Definition of the problem

The development of a program for the FIBA. This software must be capable of working with the statics of at least 200000 players and search for specific statistics if needed.

# Data collection

## Binary search trees

It’s a node based abstract data structure. Each node has a parent (except the root of the tree), a right son, who is bigger than the father, and a left son, who is smaller. Each node is itself a subtree. (<https://www.geeksforgeeks.org/binary-search-tree-data-structure/> )

## Secondary memory

Secondary memory refers to most of the nonvolatile memory of a computer, better said, the permanent information saved in places such as hard drive, solid state drive, or USB. This type of memory can hold a lot more of information, and it is saved if the system suddenly lost power, having said this, the CPU can’t access its information as fast as it can access information from primary memory (<https://www.geeksforgeeks.org/secondary-memory/> )

## Primary memory

Primary memory includes RAM, ROM, and Caches. RAM is where the CPU gets most of the information it needs in a fast manner, such as instructions or programs that are currently running. Having said this, the information in a RAM stick is volatile, which means, that when the computer is turned off, all information currently stored here is lost.

ROM, read only memory, is a non-volatile memory that can only be read (hence the name).

Caches: modern computers have 3 levels, starting from the smallest (level 1), to the biggest (level 3). Caches can’t hold much information, but they’re extremely fast, and are the closest to the CPU.

(<https://www.geeksforgeeks.org/secondary-memory/> )

## Red, black tree

Red, black trees follow the same order of a binary search tree, but they have some extra rules: Each node must be red or black, the root and leaves are black, if a node is red its children are black, and all paths from root to a leave has the same number of black nodes.

(<https://www.youtube.com/watch?v=qvZGUFHWChY> )

# Search for creative solutions

* Alternative 1: We used a cloud-based application, so the linear searches can be done faster. Also, we could use the cloud as our secondary storage, so all the statistics won’t take much space.
* Alternative 2: The program receives a .csv file with all the players and players statistics, the organizes 4 of them into binary trees. All searches are done in this tree, if the data wanted is not the one the tree is organized by, we could use a linear search
* Alternative 3: The program receives a .csv file with all the players and players statistics, the organizes 4 of them into lists, then all searches could be done by a binary search.
* Alternative 4: The program receives a .csv file with all the players and players statistics, then it’s organized in a binary tree depending in what the user wants. For example, the user wants to find certain age, so the values are organized in a binary search tree by age, and then the value is searched.
* Alternative 5: The program receives a .csv file with all the players and players statistics. Then we could organize 4 statistics by classes, for example, the players from ages 18 to 25 in a binary tree, then another with the players ages 26 to 30, this way we would have more trees, but each tree would be substantially smaller, making the searches faster
* Alternative 6: The program receives a .csv file with all the players and players statistics. For all 8 statistic an open addressing hash table is created, this way can save memory by using a hash function and know the player each data belongs to by the index where the information is saved.

# Transition of the formulation of ideas to preliminary design ideas

## Idea discard

Alternative 1 is discarded, as we don’t really know how to use cloud-based software just yet, and we like sleeping. Having said this, the storage it will take won’t be that much we would require an external storage solution. The sample data given to use is around 8mb.

## Alternative 2:

* Functions in a way that resembles what has been asked by the FIBA
* 4 statistics will have a linear time complexity, which might slow down the program
* Satisfies the requirement of the use of binary search trees
* The import of data shouldn’t be too difficult
* We would have 4 searches with time complexities of O (log n)
* The 4 binary trees will take some space

## Alternative 3:

* The import of data shouldn’t be too difficult
* We could use binary search to keep a time complexity away from linear
* We would not user Binary search trees
* We could avoid all linear searches
* The implementation would be very easy

## Alternative 4:

* The import of data shouldn’t be too difficult
* We would use a binary search tree
* We would have a time complexity of O (log n)
* Creating a new Binary search tree every time we need to make a search might slow down the program
* We could avoid all linear searches

## Alternative 5:

* The import of data shouldn’t be too difficult
* We would have binary trees
* The search would be faster
* We would have a lot more binary search tree
* It might not work as intended
* We would have searches with a complexity of O (log n)
* We would have to figure out a way to create balanced classes

## Alternative 6:

* The import of data shouldn’t be too difficult
* The time would always be linear
* We would not user Binary search trees
* We could save storage space with the use of hash functions

# Evaluation and selection of the best solution

## Criteria

* Criteria A: Uses binary trees
  + [2] Yes
  + [1] No
* Criteria B: Has a search time complexity of O (log n)
  + [2] Yes
  + [1] No
* Criteria C: Ease of search for data
  + [2] Simple
  + [1] Complex
* Criteria D: Ease of use
  + [2] Easy
  + [1] Requires some explaining
* Criteria E: level of complexity
  + [3] 1
  + [2] 2
  + [1] 3

## Evaluation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Criteria A | Criteria B | Criteria C | Criteria D | Criteria E | Total |
| Alternative 2: 4 binary trees | Yes  2 | Complexity of O (log n)  2 | Simple  2 | Easy  2 | 2 | 10 |
| Alternative 3: No binary trees | No  1 | Complexity of O (log n)  2 | Simple  2 | Easy  2 | 2 | 9 |
| Alternative 4: 1 binary tree that changes depending on the user’s search | Yes  2 | Complexity of O (log n)  2 | Simple  2 | Requires some explaining  1 | 2 | 9 |
| Alternative 5: Binary trees by subcategories | Yes  2 | Complexity of O (log n)  2 | Complex  1 | Requires some explaining  1 | 1 | 7 |
| Alternative 6: Using a hash table | No  1 | Linear time complexity  1 | Simple  2 | Easy  2 | 2 | 8 |

## 

## Selection

Using the evaluation, we can see that the second alternative is the best suited for the problem at hand. Having said this, we could use the binary search proposed in the 3 alternatives to avoid a linear time complexity.

# Preparation of reports and specifications

## Specification of the problem

Problem: Manage the data from basketball players

Input: At least 200,000 basketball players with 8 characteristics each.

Output: What ever the user has searched for in the data base.

## Requirements

* The program must be able to manage at least 200,000 players
* The program must be capable of searching for what the user wants
* The program must be capable of reading csv file
* The program must be able to add or delete players through the user interface
* The program must be able to change a player’s statics through the user interface
* At least 4 statics must have a search time complexity of O (log n)
* Each player must have a name, age, team and 5 statics

## Nonfunctional requirements

* The user interface must be easy to understand
* The program must be organized
* The use of a red, black tree
* The use of serialization for the binary search trees

## Considerations

Things that should be considered

* The searches might cause the user interface to freeze, we could use threads to avoid this
* Serialization might be used so we don’t have to create all 4 trees every time is created
  + Having said this, we would have to add a way for the program to erase these trees and reread the csv file
* The trees must be generic, and we would have to take this into account

The class diagram can be found in the document name “Documentación” inside the folder: doc

# Implementation of the design

List of tasks to implement:

# Bibliografía

# 