1. **Identificación del problema**

**1.1 Definición del problema.**

La Federación Internacional de Baloncesto, mejor conocida como FIBA desea consolidar una aplicación con la que pueda almacenar y acceder de forma rápida a la información de jugadores de basketball a nivel mundial.

**Necesidades y síntomas del problema.**

* La solución al problema debe permitir agregar información de manera masiva a través de archivos con extensiones específicas (txt,csv,etc).
* La solución al problema debe permitir agregar información de manera manual a través de una interfaz gráfica.
* La solución debe permitir realizar búsquedas rápidas y eficientes mediante uno o varios criterios de búsqueda.
* La solución al problema debe permitir visualizar el resultado de una búsqueda a través de una interfaz gráfica.
* La solución al problema debe almacenar y procesar la información en memoria secundaria.

1. **Recopilación de información**

**Punto**

En el [baloncesto](https://es.wikipedia.org/wiki/Baloncesto), un **punto** se utiliza para realizar un seguimiento de la puntuación en un partido. Los puntos pueden ser acumulados por hacer [tiros de campo](https://es.wikipedia.org/wiki/Tiros_de_campo_(baloncesto)) (dos o [tres](https://es.wikipedia.org/wiki/Triple_(baloncesto)) puntos) o [tiros libres](https://es.wikipedia.org/wiki/Tiros_libres) (un punto). Si un jugador anota un tiro de campo dentro de la línea de tres puntos, el jugador anota dos puntos. Si el jugador anota un tiro de campo más allá de la línea de tres puntos, el jugador marca tres puntos. El equipo que ha registrado el mayor número de puntos al final de un partido es declarado ganador de ese partido.

<https://es.wikipedia.org/wiki/Punto_(baloncesto)#:~:text=En%20el%20baloncesto%2C%20un%20punto,el%20jugador%20anota%20dos%20puntos>.

**Rebote**

Un **rebote** en [baloncesto](https://es.wikipedia.org/wiki/Baloncesto) es el acto de conseguir la posesión del balón después de un lanzamiento de campo o de un [tiro libre](https://es.wikipedia.org/wiki/Tiro_libre_(baloncesto)) fallado. Son uno de los objetivos fundamentales de los [jugadores altos](https://es.wikipedia.org/wiki/P%C3%ADvot) de un equipo, aunque cualquier jugador en pista puede intentar conseguirlos.

<https://es.wikipedia.org/wiki/Rebote_(baloncesto)#:~:text=Un%20rebote%20en%20baloncesto%20es,en%20pista%20puede%20intentar%20conseguirlos>.

**Asistencia**

Una asistencia es un pase a un jugador que se encuentra en una posición de ventaja o que le ayuda a conseguir una canasta.

<https://es.wikipedia.org/wiki/Asistencia_(baloncesto)>

**Robo**

Los robos se acreditan al jugador defensivo que causa primero la pérdida de balón, incluso si no termina con la posesión de la pelota viva. Para ganar un robo, el jugador defensivo debe ser el iniciador de la acción que causó la pérdida de balón, no solo el benefactor. Cada vez que un robo se registra por un jugador defensivo, un jugador ofensivo debe ser acreditado con un balón perdido.

<https://es.wikipedia.org/wiki/Robo_(baloncesto)>

**Bloqueo**

En el baloncesto el tapón o el bloqueo es un movimiento a una acción de juego en la cual el jugador defensivo puede realizar para ocasionar que el balón no entre a la canasta

<http://sialdeporte.com/c-basquetbol/bloqueo-en-baloncesto/>

**Faltas personales**

La falta personal es una infracción en el ámbito del [baloncesto](https://es.wikipedia.org/wiki/Baloncesto). En un partido según normas [FIBA](https://es.wikipedia.org/wiki/FIBA) (de 40 minutos de duración, es decir cuatro cuartos de diez minutos) si un jugador acumula cinco faltas personales es expulsado.

[Falta personal - Wikipedia, la enciclopedia libre](https://es.wikipedia.org/wiki/Falta_personal)

1. **Búsqueda de soluciones creativas.**

**Almacenar información de manera masiva**

* Uso de árboles binarios de búsqueda
* Uso de árboles binarios de búsqueda autobalanceados
* Uso de tablas de hash
* Uso de listas enlazadas ordenadas
* Uso de montículos
* Guardar información a través de persistencia de archivos

**Búsquedas eficientes con varios criterios**

* Búsqueda binaria en tablas de hash
* búsqueda binaria a través de árboles binarios de búsqueda
* búsqueda a través de árboles binarios de búsqueda autobalanceados

**Manejo de información desde memoria secundaria**

* Uso de archivos de texto con formatos específicos como CSV
* Uso de archivos seriales binarios
* Uso de carpetas específicas utilizadas con particiones

1. **Transición de las Ideas a los Diseños Preliminares**

**Almacenamiento de información:**

Se descarta el uso de **tablas hash** dado que, en cantidades masivas de información no es eficiente su uso, lo mismo sucede con las **listas enlazadas ordenadas** y los **montículos**.

La **persistencia de archivos**, al igual que los **árboles binarios de búsqueda** y **árboles binarios de búsqueda autobalanceados** permiten guardar cantidades masivas de información, siendo la persistencia capaz de mantener la información aún después de la finalización del programa.

**Búsquedas eficientes con varios criterios**

Aunque las **tablas hash** brinden una baja complejidad temporal en su búsqueda en varios casos, eso no sucede cuando se trabaja con mucha información, por lo que no es eficiente para el problema planteado y se descarta esta opción.

El uso de **árboles binarios de búsqueda** (abb) nos permite dado su criterio de orden aumentar la eficiencia al momento de operar con cantidades masivas de información.

Al igual que con los ABB, los **árboles binarios de búsqueda autobalanceados** nos brindan la capacidad de operar eficientemente con mucha información, pero teniendo como única diferencia la capacidad de auto balancearse manteniendo una baja complejidad temporal para cualquier caso.

**Manejo de información desde memoria secundaria**

El uso de **carpetas con particiones** nos permite guardar información de forma ordenada, pero su complejidad para acceder a un dato particular hace que su uso no sea eficiente para resolver el problema.

Con los **archivos seriales** se nos permite solucionar el problema de forma eficiente manteniendo la información en memoria secundaria y cargándola al momento de ejecutar el problema.

El uso de **archivos de texto con formatos como .csv**, nos permite manejar la información en la memoria secundaria de forma efectiva, cargándola al inicio del programa o bien cargar partes especificas a medida que el programa lo requiera.

1. **Evaluación y Selección de la mejor Solución**

Para evaluar las soluciones planteadas anteriormente, se plantearán diversos criterios para cada problema propuesto hallando así la mejor solución en cada contexto.

Criterios:

* Almacenamiento de información:
  + Criterio A. Estabilidad de eficiencia para cualquier caso
    - [3] Alta
    - [2] Media
    - [1] Baja
  + Criterio B. Eficiencia.
    - [2] Alta
    - [1] Media
* Búsquedas eficientes con varios criterios
  + Criterio A. Capacidad para mantener la misma complejidad
    - [2] Capaz
    - [1] Incapaz
* Manejo de información desde memoria secundaria
  + Criterio A. Eficiencia
    - [3] Alta
    - [2] Media
    - [1] Baja

**Almacenamiento de información:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Criterio A** | **Criterio B** | **Total** |
| **Persistencia de Archivos** | **1** | **1** | **2** |
| **Arboles binarios de búsqueda** | **2** | **2** | **4** |
| **Arboles binarios de búsqueda autobalanceados** | **3** | **2** | **5** |

**Búsquedas eficientes con varios criterios:**

|  |  |  |
| --- | --- | --- |
|  | **Criterio A** | **Total** |
| **Arboles binarios de búsqueda** | **1** | **1** |
| **Arboles binarios de búsqueda autobalanceados** | **2** | **2** |

**Manejo de información desde memoria secundaria:**

|  |  |  |
| --- | --- | --- |
|  | **Criterio A** | **Total** |
| **Archivos seriales** | **2** | **2** |
| **Archivos de texto con formatos específicos como .csv** | **3** | **3** |

**Selección:**

Debido a la evaluación anterior se deben usar los **arboles binarios de búsqueda autobalanceados** para almacenar información, así como para hacer búsquedas eficientes por medio de varios criterios. Por otro lado, se deben usar los **archivos de texto como los .csv** para manejar la información desde memoria secundaria evitando cargarla y haciendo eficiente así la solución del problema.

1. **Preparación de informes y Especificaciones**

La especificación del problema se encuentra en el siguiente archivo: [..\Solucion del problema\solucion del problema.docx](../Solucion%20del%20problema/solucion%20del%20problema.docx)

El TAD del árbol binario de búsqueda autobalanceado se encuentra en el siguiente documento: [..\ADT-design\Binary Search Tree auto balanced.docx](../ADT-design/Binary%20Search%20Tree%20auto%20balanced.docx)

El TAD del árbol binario de búsqueda se encuentra en el siguiente documento: [..\ADT-design\Binary search tree.docx](../ADT-design/Binary%20search%20tree.docx)

El diagrama de clases para la implementación de la solución se encuentra en el siguiente documento:

El diagrama de clases de las estructuras de datos se encuentra en el siguiente documento: [..\Class Diagram Images\BST-Diagram.pdf](../Class%20Diagram%20Images/BST-Diagram.pdf)

El diagrama de clases para los casos de prueba se encuentra en el siguiente documento: [Falta](file:///C:\Users\dahp-\OneDrive\Documents\GitHub\aed-tarea-integradora-2\docs\TAD-desing\Binary%20Search%20Tree%20auto%20valanced.docx)

1. **Implementación del Diseño**

La implementación del diseño fue hecha en el lenguaje de programación Java y se encuentra en el siguiente repositorio: <https://github.com/JuanFernandoMartinez/aed-tarea-integradora-2>