ALGORITMOS Y ESTRUCTURAS DE DATOS

Basketball Facts

Integrantes

Sean Quintero.
John K. Landazuri.
Bryan A. Guapacha F.
Paola Andrea Osorio H.

Contexto del problema

El baloncesto al ser un deporte que acumula muchos seguidores en el mundo, cada vez necesita irse actualizando; de una canasta de naranjas puesta en lo alto de una escalera; a las cestas en cada cancha de barrio; este deporte ha traído grandes jugadores como Michael Jordan, Kobe Bryant , Lebron, Curry, Giannis, Harden entre otros.

Algo muy clave que se tomó del béisbol fueron las estadísticas; para poderlas definir se necesita recolectar datos, ordenarlos y clasificarlos.

FASE 1

Identificación del problema

Definición del problema.

Se necesita un programa que recoja, ordene y clasifique la información estadística de los jugadores de la NBA.

Identificación de síntomas

- La FIBA ha solicitado la implementación de una herramienta para el manejo de información de gran tamaño que permita ingresar datos, eliminar o modificar datos y realizar consultas de jugadores.
- La solución debe brindar una forma rápida y eficiente de búsqueda.
- La solución debe permitir búsquedas por parámetros estadísticos.
- La solución debe leer datos por interfaz gráfica y por archivos .csv.

FASE 2

Recopilación de la información necesaria y especificación de requerimientos

Especificaciones de requerimientos

- **RF-1.Ingresar datos:** El programa debe permitir ingresar los datos de los jugadores de manera masiva (archivo csv) o a través de una interfaz. Las entradas vienen siendo datos como nombre, edad, equipo y los 5 rubros estadísticos. Esto retorna el jugador que ha sido añadido exitosamente a la aplicación.
- **RF-2.Eliminar datos:** El programa debe permitir eliminar datos de la base construida; como entrada tenemos el dato a eliminar; de manera que se retorna al final el dato eliminado.
- **RF - 3. Modificar Datos:** El programa debe permitir al usuario realizar cambios en los atributos y datos de cada jugador, estas modificaciones se guardan de forma permanente. Como entradas se tienen el jugador al cual se requieren realizar las modificaciones, los datos/atributos a modificar y la nueva información a ingresar.
- **RF-4.Realizar consultas:** El programa debe permitir realizar consultas con la capacidad de recuperar jugadores de acuerdo a la categoría de búsqueda seleccionada y el valor dado para dichas consultas,como entrada se puede buscar con parámetros como nombre, edad, equipo o alguno de los diferentes datos

estadísticos. Al final se retorna, jugador o jugadores que cumplen con el criterio buscado.

- **RF-5. Mostrar tiempo en que se toma realizar una consulta:** El programa debe permitir al usuario visualizar el tiempo que tarda la aplicación en realizar una consulta, teniendo como entrada el criterio de búsqueda que puede ser por nombre, edad, equipo o algunos de los 5 rubros estadísticos, para obtener como resultado el tiempo que tarda la consulta.
- **RF-6. Realizar búsquedas sobre dos criterios estadísticos:** El programa debe permitir realizar búsquedas o consultas basados en dos criterios estadísticos utilizando ABB como estructura para manejo de índices, teniendo como entradas los dos criterios estadísticos sobre los que queremos hacer la búsqueda para obtener como resultado aquellos jugadores que cumplen con los criterios buscados.

Especificaciones limitantes

- Las operaciones de crear, eliminar, modificar, consultar deben ser eficientes.
- Se debe implementar árboles de búsqueda binaria, árboles AVL, árboles rojo y negro como estructuras para los datos.
- Se debe usar interfaz gráfica y carga de datos de archivos a la vez.

Definiciones y ejemplos encontrados

Hemos buscado distinta información sobre software que realiza funciones parecidas:

--FIBA mundial 2019
http://www.fiba.basketball/es/basketballworldcup/2019/playerstats#tab=player_stat
Posee filtros para buscar los diferentes tipos de puntos y las posiciones

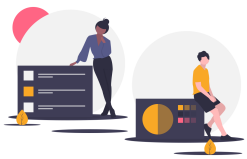
---NBA---
https://www.nba.com/stats/
Sitio web de la NBA donde me permite conocer los 5 mejores dentro de diferentes rubros estadísticos de los jugadores.

---NBA---
https://www.nba.com/players
Sitio web de la NBA donde me permite conocer características de los jugadores, como el nombre, equipo, número, posición, altura, peso, país etc.

---fiba LiveStats--
Software estadístico para añadir estadísticas en los partidos de competencias oficiales. Es el oficial de la fiba, como sistema de recolección de datos.
http://www.fibaorganizer.com/

Como definiciones tenemos:

- **Data set:** Conjunto o colección de datos específica, recopilada de diversas fuentes y orientada a aspectos específicos.
- **CRUD: (create, read, update, delete)** Representa las operaciones básicas que debe tener la implementación de una base de datos, las cuales son: crear, leer, actualizar y eliminar la información de la misma.
- **Complejidad algorítmica:** Concepto matemático que describe la eficiencia en recursos informáticos a la hora de realizar una secuencia de operaciones.
- **Árbol binario de búsqueda auto balanceada:** En informática, es una estructura para gestionar datos la cual nos ofrece una complejidad garantizada (log n) en sus principales operaciones: búsqueda, inserción y eliminación.
- **Árbol binario de búsqueda:** es una estructura de datos en informática vista como un árbol ordenado, en el que cada nodo tiene 0, 1 ó 2 hijos (el hijo izquierdo y el derecho.
- **Árbol rojo y negro:** Un árbol rojo-negro es un tipo abstracto de datos. Concretamente, es un árbol binario de búsqueda equilibrada, una estructura de datos utilizada en informática y ciencias de la computación. La estructura original fue creada por Rudolf Bayer en 1972, que le dio el nombre de “árboles-B binarios simétricos”, pero tomó su nombre moderno en un trabajo de Leo J. Guibas y Robert Sedgewick realizado en 1978.”Tomado de wikipedia”



FASE 3

Búsqueda de soluciones creativas

1.Carga de datos:

a. Elaborar un módulo de software que use información a partir de un conjunto de datos de diseño propio.
b. Implementación de colecciones previamente creadas y ofrecidos por plataformas digitales tales como: data.world, indexmundi, entre otros.
c. Cargar datos a través de archivos .csv.
d. Añadir jugadores a través de interfaz

2. Modelamiento de una estructura de datos eficiente:

a. Implementar una estructura Hash Table dinámica para almacenar los datos.
b. Implementar una estructura de árbol de búsqueda binario para el almacenamiento de los datos.
c. Implementar una estructura de árbol de búsqueda binario auto balanceado para el almacenamiento de los datos.

d. Implementar una estructura de árbol de búsqueda rojo y negro para el almacenamiento de los datos.
e. Implementar una estructura de grafos para el almacenamiento de los datos.

3. Búsqueda:

a. Implementación de estructuras como grafos, que brindan un recorrido parcial dada una secuencia de símbolos.
b. Implementación de estructuras como árboles binarios de búsqueda(AVL).
c. Solicitar al usuario el dato a buscar y luego con este dato buscarlo en la base de datos.
d. Búsqueda de datos de jugadores por parámetro.

4. Eliminación de datos:

a. Solicitar al usuario el dato a eliminar y luego removerlo de la base de datos.
b. Eliminar datos por parámetros.
c. Eliminar datos por interfaz.

5. Actualizar datos:

a. Solicitar al usuario el dato modificado y éste reemplazarlo en la base de datos.
b. Reemplazar datos en la interfaz.
c. Cargar archivos con datos actualizados.

FASE 4

Transición de la formulación de ideas a los diseños preliminares

En esta fase vamos a descartar las peores alternativas que no brindan una solución adecuada a los requerimientos.

1.Carga de datos:

a. Elaborar módulo de software que use información a partir de un conjunto de datos diseño propio.
   Esta idea se descarta porque es muy ineficiente ya que, tomaría mucho tiempo recolectar datos de esta manera.

2. Modelamiento de una estructura de datos eficiente:

a. Implementar una estructura Hash Table dinámica para almacenar los datos.
b. Implementar una estructura de grafos para el almacenamiento de los datos.

Se descartan las ideas a y b, debido a la complejidad temporal de la estructura hashTable, que aunque permite almacenar grandes cantidades de datos, la búsqueda de los mismos se realiza en un tiempo de O(n), y para la realización del proyecto se necesita una búsqueda de tipo O(log n).
Las estructura de los grafos no se han tratado en el semestre y debido a esto no podemos realizar la implementación de los mismos.

3. Búsqueda:

a. Implementación de estructuras como grafos, que brindan un recorrido parcial dada una secuencia de símbolos.

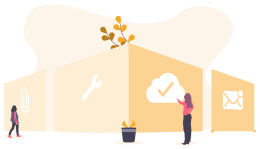
Las estructura de los grafos no se han tratado en el semestre y debido a esto no podemos realizar la implementación de los mismos.

4.Eliminación de datos:

No eliminamos alguna porque todas son útiles.

5. Actualizar datos

No eliminamos alguna porque todas son útiles.



FASE 5

Evaluación y selección de la mejor solución

1.Criterios carga de datos:

a. Permita hacer uso de un conjunto de datos previamente definidos.
b. Permita incluir parámetros previamente establecidos para su manejo correspondientes.
c. Permita una rapidez de tiempo en su creación, o que ya esté creada.

2. Criterios modelamiento de una estructura eficiente

a. Permita modelar los datos de manera eficiente.
b. Permita almacenar una gran cantidad de datos.
c. Lo que conocemos sobre el tema.

3. Búsqueda de datos

a. Que ofrezca una complejidad de espacio temporal eficiente.
b. Lo que conocemos sobre el tema.

4. Eliminación de datos

a. Llevar a cabo la eliminación simple de un dato dado.

5. Actualizar datos

a. Llevar a cabo la modificación simple de un dato dado.

6.Criterios carga de datos

	Criterio a	Criterio b	Criterio c	Total
Alternativa a	5	5	0	10
Alternativa b	5	5	5	15
Alternativa c	5	5	4	14
Alternativa d	5	5	3	13

Elegimos las alternativas b, c y d de acuerdo a su puntaje.

7. Criterios modelamiento de una estructura eficiente

	Criterio a	Criterio b	Criterio c	Total
Alternativa a	5	3	5	13
Alternativa b	5	3	5	13
Alternativa c	5	5	5	15
Alternativa d	5	5	4	14
Alternativa e	5	5	0	10

Elegimos las alternativas c, d y e por sus puntajes

8. Búsqueda de datos

	Criterio a	Criterio b	Total
Alternativa a	5	0	5
Alternativa b	5	5	10
Alternativa c	5	5	10
Alternativa d	5	5	10

Elegimos las alternativas b, c y d

9. Eliminación de datos

	Criterio a	Total
Alternativa a	5	5
Alternativa b	5	5
Alternativa c	5	5

Elegimos todas las opciones.

10.Actualizar datos

	Criterio a	Total
Alternativa a	5	5
Alternativa b	5	5
Alternativa c	5	5

Todas las opciones las elegimos.

TestAVLBSTree

Stages configuration:

Name	Class	Stage
emptySetup	TestAVLBSTree	Un objeto de la clase AVLBSTree vacía.
nonEmptySetup	TestAVLBSTree	Un objeto de la clase AVLBSTree con los nodos 5, 8, 17, 25 y 32 ya agregados.

Test cases design:

Objetivo de la prueba: Comprobar que la clase añade correctamente un nuevo nodo a un árbol AVL vacío.				
Class	Method	Setup	Input	Result
AVLBSTree	add	emptySetup	1, 1	Se ha añadido correctamente un nuevo nodo con la clave y el valor dados.
AVLBSTree	add	emptySetup	2, 2	Se ha añadido correctamente un nuevo nodo con la clave y el valor dados.
AVLBSTree	add	emptySetup	3, 3	Se ha añadido correctamente un nuevo nodo con la clave y el valor dados.

Objetivo de la prueba: Comprobar que la clase añade correctamente un nuevo nodo a un árbol AVL no vacío.				
Class	Method	Setup	Input	Result
AVLBSTree	add	nonEmptySetup	12, 12	Se ha añadido correctamente un nuevo nodo con la clave y el valor dados.
AVLBSTree	add	nonEmptySetup	40, 40	Se ha añadido correctamente un nuevo nodo con la clave y el valor dados.
AVLBSTree	add	nonEmptySetup	20, 20	Se ha añadido correctamente un nuevo nodo con la clave y el valor dados.

Objetivo de la prueba: Comprobar que la clase borra correctamente un nodo con una clave determinada de un árbol AVL vacío.				
Class	Method	Setup	Input	Result
AVLBSTree	delete	emptySetup	1	El nodo con la clave dada ha sido eliminado.
AVLBSTree	delete	emptySetup	38	El nodo con la clave dada ha sido eliminado.
AVLBSTree	delete	emptySetup	25	El nodo con la clave dada ha sido eliminado.

Objetivo de la prueba: Comprobar que la clase borra correctamente un nodo con una clave determinada de un árbol AVL no vacío.				
Class	Method	Setup	Input	Result
AVLBSTree	delete	nonEmptySetup	8	El nodo con la clave dada ha sido eliminado.
AVLBSTree	delete	nonEmptySetup	5	El nodo con la clave dada ha sido eliminado.
AVLBSTree	delete	nonEmptySetup	32	El nodo con la clave dada ha sido eliminado.

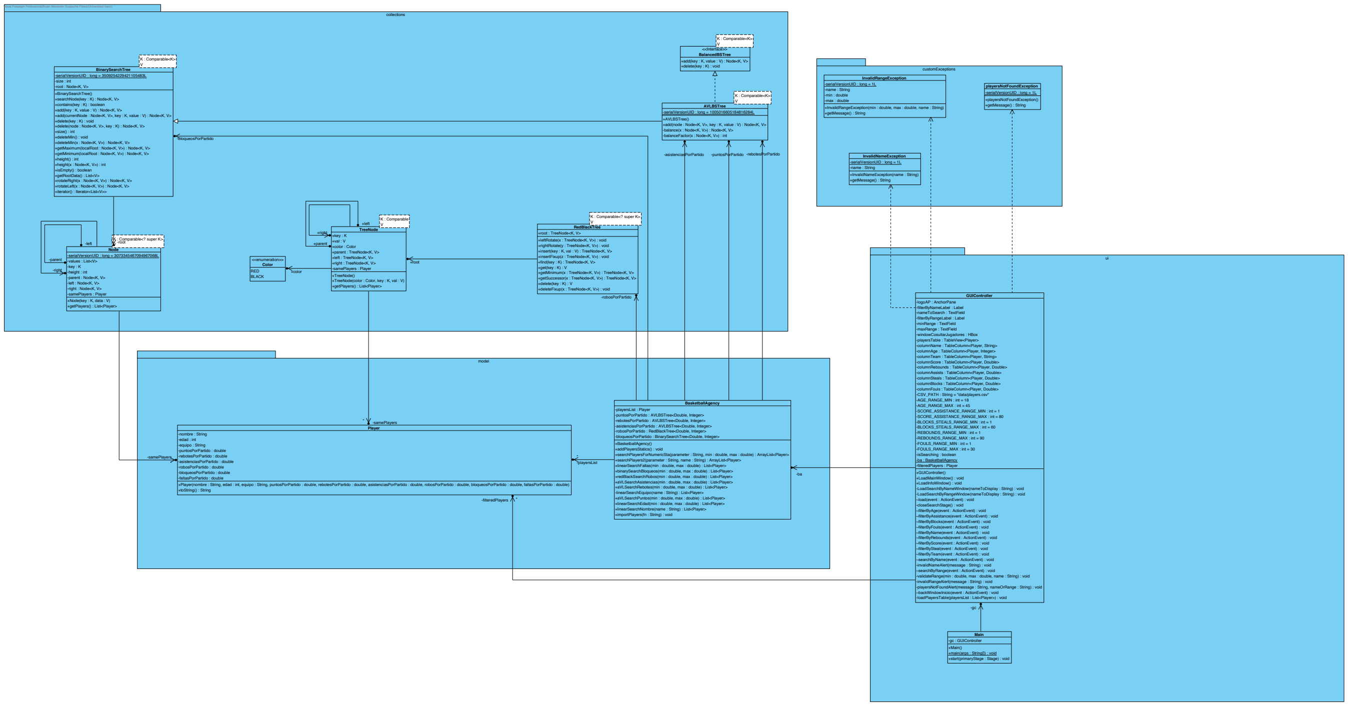
TestBinarySearchTree
Stages configuration:

Name	Class	Stage
nonEmptySetup	TestBinarySearchTree	Un objeto de la clase BinarySearchTree con los nodos 8, 15, 17, 20, 23, 28 y 32 ya agregados.

Test cases design:

Objetivo de la prueba: Prueba la correcta aplicación de rotateLeft a un nodo dado.				
Class	Method	Setup	Input	Result
BinarySearchTree	rotateLeft	nonEmptySetup	BST.root.left	El nodo dado es rotado a la izquierda.
BinarySearchTree	rotateLeft	nonEmptySetup	BST.root	El nodo dado es rotado a la izquierda.

Objetivo de la prueba: Prueba que la clase aplique correctamente la rotación a la derecha de un nodo de el BST.				
Class	Method	Setup	Input	Result
BinarySearchTree	rotateRight	nonEmptySetup	BST.root.right	El nodo dado es rotado a la derecha.
BinarySearchTree	rotateRight	nonEmptySetup	BST.root	El nodo dado es rotado a la derecha.


collections
BinarySearchTree
Node
TreeNode
Color
RedBlackTree
BalancedBSTree
AVLBSTree
model
Player
BasketballAgency
customExceptions
InvalidRangeException
InvalidNameException
playersNotFoundException
ui
GUIController
Main