

Tarea Integradora II: FIBA

Integrantes:

Diana Sofía Olano Montaña (A00369468)

Angélica Corrales Quevedo (A00367954)

Keren López Córdoba (A00368902)

Profesor: Johnatan Garzón

Algoritmos y estructuras de datos, grupo 1

Universidad Icesi

Cali, octubre 31 del 2021

Método de la ingeniería

Contexto de la problemática

La FIBA (Federación Internacional de Baloncesto) busca consolidar en una aplicación los datos de mayor relevancia de cada uno de los profesionales del baloncesto en el mundo, de tal manera que se puedan efectuar diferentes consultas que permitan realizar análisis sobre esta información, se conozcan patrones acerca del desarrollo del deporte, los criterios que toman más fuerza o, en general, hacia dónde se dirige el deporte en la actualidad.

Fase 1: Identificación del problema

Identificación de necesidades y síntomas:

- La solución al problema debe permitir ingresar datos de los jugadores (nombre, edad, equipo y cinco estadísticas), ya sea de manera masiva o manual.
- La solución al problema debe incluir una interfaz gráfica con el fin de que la interacción entre el usuario y el programa sea más amigable.
- Los usuarios de la aplicación requieren eliminar o modificar los datos ingresados.
- Los usuarios de la aplicación requieren realizar búsquedas de jugadores utilizando como criterios de búsqueda cinco categorías estadísticas, ya sean puntos por partido, rebotes por partido, asistencias por partido, robos por partido o bloqueos por partido.
- La solución al problema debe permitir almacenar información de gran tamaño.
- La solución al problema debe permitir el rápido acceso a los datos almacenados.
- La solución del problema debe guardar la información en la memoria secundaria.
- La FIBA no tiene una aplicación que le permita manejar datos de mayor relevancia de cada uno de los profesionales del baloncesto en el mundo.

Definición del Problema:

La FIBA requiere el desarrollo de un programa de software que permita manejar de manera eficiente los datos de mayor relevancia de cada uno de los profesionales del baloncesto en el mundo.

Especificación de requerimientos funcionales

R1. Gestionar un jugador

- Agregar un jugador, con su nombre, edad, equipo y 5 estadísticas (puntos por partido, rebotes por partido, asistencias por partido, robos por partido, bloqueos por partido). Cabe resaltar que no podrá existir un jugador con todas las mismas características o atributos que otro.
- Modificar un jugador. Se podrá modificar su nombre, edad, equipo y sus estadísticas (puntos por partido, rebotes por partido, asistencias por partido, robos por partido, bloqueos por partido).
- Eliminar un jugador.

R2. Realizar consultas de jugadores utilizando como criterios de búsqueda las categorías estadísticas incluidas. Puede realizarse la consulta como una igualdad o una desigualdad (mayores o iguales a ó menores o iguales a). La búsqueda debe ser muy eficiente para los atributos de puntos por partido, asistencias por partido, robos por partido, bloqueos por partido. Además, se debe mostrar el tiempo que se toma en realizar una consulta, y se realizarán búsqueda por dos métodos para los atributos de puntos por partido y asistencias por partido.

R3. Guardar toda la información del programa en archivos con extensión “.ackldo” . Es decir, cada vez que se registre o actualice información, esta se guardará en dichos archivos.

R4. Importar datos de un archivo csv con información de jugadores de baloncesto.

Fase 2: Recopilación de la información necesaria

Definiciones

- **Ejemplo de plataforma web que presenta estadísticas de baloncesto:**

SEGUIMIENTO DE LA SUPER ESTRELLA
Últimos partidos de los mejores jugadores de los Juegos Olímpicos

PLAYER	PTS	REB	AST	STL	BLK
LUKA DONČIĆ SLOVENIA vs AUSTRALIA 93-107 ÚLTIMO PARTIDO, AUG 7, 2021	22	8	7	1	0
KEVIN DURANT FRANCE vs USA 82-87 ÚLTIMO PARTIDO, AUG 7, 2021	29	6	3	0	1
EVAN FOURNIER FRANCE vs USA 82-87 ÚLTIMO PARTIDO, AUG 7, 2021	16	3	2	0	0
RICKY RUBIO SPAIN vs AI 81-95 ÚLTIMO PARTIDO, AI	38	4	2		

Proballers basketball stats

Baloncesto estadísticas de jugadores, equipos, ligas, en todo el mundo. (s. f.).

Proballers. Recuperado de <https://www.proballers.com/es>

- **Rebotes por partido:** Un rebote en baloncesto es el acto de conseguir la posesión del balón después de un lanzamiento de campo o de un tiro libre fallado.

Wikipedia. (2021, 22 julio). *Rebote (baloncesto)*. Recuperado de [https://es.wikipedia.org/wiki/Rebote_\(baloncesto\)](https://es.wikipedia.org/wiki/Rebote_(baloncesto))

- **Asistencias por partido:** Una asistencia es un pase que establece una canasta. Es un movimiento desinteresado, uno de los más importantes en el baloncesto, que puede

llevar a una canasta e incluso ayudar a ganar un juego. Por ejemplo, un armador podría pasar a un pequeño delantero que está cortando la canasta. Ese avance da un paso y golpea una bandeja, y el guardia recibe una asistencia.

TiempoExtraBasket. (s. f.). *Asistencia*. Recuperado de <https://tiempoextrabasket.online/baloncesto/diccionario-de-baloncesto/asistencia/>

- **Robos por partido:** el robo es una "**acción defensiva**" que hace que el oponente pierda la posesión del balón.

Wikipedia. (2021, septiembre 27). *Líderes en robos de la NBA*. Recuperado de https://es.wikipedia.org/wiki/L%C3%ADderes_en_robos_de_la_NBA

- **Árboles binarios de búsqueda balanceados:** Un árbol binario balanceado es un árbol binario en el cual las alturas de los dos subárboles de todo nodo difiere a lo sumo en 1. El balance de un nodo en un árbol binario se define como la altura de su subárbol izquierdo menos la altura de su subárbol derecho. Cada nodo en un árbol binario balanceado tiene balance igual a 1, -1 o 0, dependiendo de si la altura de su subárbol izquierdo es mayor que, menor que o igual a la altura de su subárbol derecho. Universidad Tecnológica de la Mixteca. (2009, diciembre 15). archivos estructuras. Recuperado de <https://www.utm.mx/~jahdezp/archivos%20estructuras/balanceo%20de%20arboles.pdf>
- **Árboles binarios de búsqueda no balanceados:** Es fácil ver que el árbol se vuelve desbalanceado si y sólo si el nodo recién insertado es un descendiente izquierdo de un nodo que tenía de manera previa balance de 1, o si es un hijo derecho descendiente de un nodo que tenía de manera previa balance -1. Universidad Tecnológica de la Mixteca. (2009, diciembre 15). archivos estructuras. Recuperado de <https://www.utm.mx/~jahdezp/archivos%20estructuras/balanceo%20de%20arboles.pdf>
- **Árbol rojo y negro:** Un árbol rojo-negro es un árbol binario de búsqueda en el que cada nodo almacena un bit adicional de información llamado color, el cual puede ser rojo o negro. Sobre este atributo de color se aplican restricciones que resultan en un árbol en el que ningún camino de la raíz a una hoja es más de dos veces más largo que cualquier otro, lo cual significa que el árbol es balanceado.

Cada nodo de un árbol rojo negro contiene la siguiente información: color, clave, hijo izquierdo, hijo derecho y padre. Si un hijo o el padre de un nodo no existe, el apuntador correspondiente contiene el valor NULO, el cual consideraremos como un

nodo cuyo color es negro. En lo sucesivo nos referiremos a los nodos distintos a las hojas (NULO) como nodos internos del árbol y a las hojas y al padre de la raíz como nodos externos.

Todo árbol rojo-negro satisface las siguientes propiedades:

1. Todo nodo es rojo o negro
2. La raíz es negra
3. Toda hoja (NULO) es negra.
4. Si un nodo es rojo, entonces sus dos hijos son negros.
5. Para cada nodo, todas las rutas de un nodo a las hojas (NULOS) contienen el mismo número de nodos negros. El número de nodos negros en esta ruta se conoce como altura-negra del nodo.

Yumpu.com. (s. f.). *Guía Árboles Rojo-Negros*. Recuperado de <https://www.yumpu.com/es/document/view/14904897/guia-arboles-rojo-negros>

- **Algoritmo de búsqueda lineal:** Se comienza en el primer ítem de la lista, simplemente se traslada de un ítem a otro, siguiendo el orden secuencial subyacente hasta que se encuentre lo que se busca o se quede sin ítems. Si se queda sin ítems, hemos descubierto que el ítem que se estaba buscando no estaba presente. La complejidad de tiempo del algoritmo es $O(n)$.

Runestone Academy. (2021). 5.3. *La búsqueda secuencial — Solución de problemas con algoritmos y estructuras de datos*. Recuperado de <https://runestone.academy/runestone/static/pythoned/SortSearch/LaBusquedaSecuencial.html>

- **Complejidad temporal:**

Es el número de operaciones que realiza un algoritmo para completar su tarea (considerando que cada operación dura el mismo tiempo). El algoritmo que realiza la tarea en el menor número de operaciones se considera el más eficiente en términos de complejidad temporal.

Rivera, J. (2021, 5 junio). *Introducción a la complejidad temporal de los algoritmos*. Recuperado de [freeCodeCamp.org.https://www.freecodecamp.org/espanol/news/introduccion-a-la-complejidad-temporal-de-los-algoritmos/](https://www.freecodecamp.org/espanol/news/introduccion-a-la-complejidad-temporal-de-los-algoritmos/)

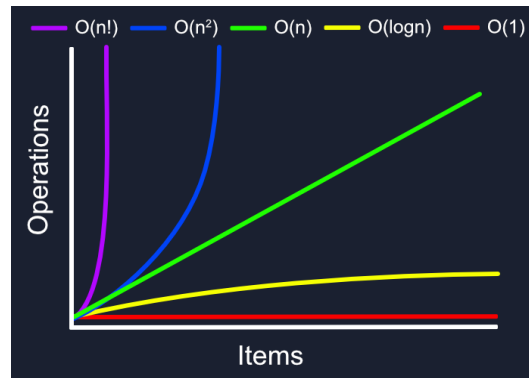


Imagen recuperada de

<https://ichi.pro/es/conocer-la-complejidad-temporal-de-su-algoritmo-164088396689956>

- **Memoria primaria:**

La memoria principal es la memoria principal del sistema informático. Es más rápido acceder a los datos de la memoria principal porque es la memoria interna de la computadora. La memoria principal es volátil, lo que significa que los datos de la memoria principal no existen a menos que se guarden cuando se produce un corte de energía. En esta se encuentran las memorias RAM, ROM y CACHE.

(1)Ebooks online. (2020). *Diferencia entre memoria primaria y memoria secundaria*.

Recuperado de

<https://ebooksonline.es/diferencia-entre-memoria-primaria-y-memoria-secundaria/>

ED team. (2019). *Memoria principal y secundaria*. EDteam. Recuperado de

<https://ed.team/comunidad/memoria-principal-y-secundaria>

- **Memoria Secundaria:**

Todos los dispositivos de almacenamiento secundario capaces de almacenar datos de gran volumen son memoria secundaria. Es más lento que la memoria primaria. Sin embargo, puede ahorrar una cantidad significativa de datos, desde gigabytes hasta terabytes. Esta memoria también se denomina almacenamiento de respaldo o medio de almacenamiento (1).

Fase 3: Búsqueda de soluciones creativas

Técnica empleada: Lluvia de ideas

“Es una técnica de pensamiento creativo utilizada para estimular la producción de un elevado número de ideas, por parte de un grupo, acerca de un problema y de sus soluciones o, en general, sobre un tema que requiere de ideas originales”.

Consultores, A. (2019, 16 septiembre). *Tormenta de Ideas: Creatividad para la Mejora*. Aiteco Consultores. Definición recuperada de <https://www.aiteco.com/tormenta-de-ideas/>

Alternativas propuestas de estructuras de datos para la búsqueda de jugadores eficientemente.

Alternativa 1: Implementar Arraylist.

La Arraylist se utilizaría para guardar a cada uno de los jugadores de baloncesto con su respectiva información. Hay que tener en cuenta que como esta estructura se basa en una búsqueda lineal, esta será de una complejidad temporal $O(n)$.

Alternativa 2: Implementar lista doblemente enlazada.

La lista doblemente enlazada se implementarían para guardar a cada uno de los jugadores de baloncesto con sus respectivos atributos. Se tiene en cuenta que como esta estructura es lineal, su búsqueda tiene una complejidad temporal $O(n)$.

Alternativa 3: Implementar Árboles binarios de búsqueda ABB.

Los árboles binarios de búsqueda estándar se usarían para guardar a cada uno de los jugadores de baloncesto con su respectiva información. Cabe resaltar que la búsqueda en esta estructura tiene una complejidad temporal $O(h)$, donde h es la altura del árbol. Sin embargo, en el peor de los casos, su complejidad temporal llega a ser $O(n)$.

Alternativa 4: Implementar Árboles binarios de búsqueda AVL.

Los árboles binarios de búsqueda balanceados se implementarían para guardar a cada uno de los jugadores de baloncesto con sus respectivos atributos. Es importante mencionar que la búsqueda en esta estructura es de $O(\log n)$.

Alternativa 5: Implementar matrices.

Las matrices se utilizarían para guardar a cada uno de los jugadores en sus espacios disponibles. Se debe tener en cuenta que la búsqueda tiene una complejidad temporal de $O(n)$, y que se debe indicar el tamaño de esta estructura.

Alternativa 6: Implementar Chaining Hash Table.

Este tipo de hash table se implementaría para almacenar a cada uno de los jugadores con sus respectivos atributos. Se debe tener en cuenta que, las claves se podrían repetir, e igualmente la posición dada por la función hash, y varios jugadores irían a la misma posición, por lo que en el peor caso, todos los jugadores se irían al mismo slot, estarían en una lista doblemente enlazada. Así, la búsqueda tiene una complejidad temporal de $O(n)$.

Fase 4: Transición de las ideas a los diseños preliminares.

Las alternativas 2, 5 y 6 las descartamos, ya que el uso de estructuras como lo son la Chaining Hash table, las matrices y las listas doblemente enlazadas no es adecuado, teniendo en cuenta que si lo que queremos es almacenar una gran cantidad de datos y realizar búsquedas eficientes dentro de ellos, su complejidad temporal de $O(n)$ no sería apropiada. Además, para la matriz, se le debería asignar un tamaño, y al querer almacenar millones de datos, esto podría llegar a ser un inconveniente. Para la Chaining Hash Table, la búsqueda por una desigualdad de valores sería un obstáculo.

Si revisamos cuidadosamente las demás alternativas obtenemos lo siguiente:

- **Alternativa 1:** Implementar ArrayList.

Esta alternativa se puede considerar para resolver el problema teniendo en cuenta que una de sus características es que es rápida en cuanto almacenamiento y acceso a los datos en comparación con la lista enlazada. Sin embargo, cuenta con una complejidad temporal en la búsqueda lineal de $O(n)$.

- **Alternativa 3:** Implementar Árboles binarios de búsqueda ABB.

Esta alternativa se adapta a la situación, pero se correría el riesgo de que ocurra el peor de los casos, y se presente una complejidad temporal $O(n)$ para la búsqueda.

- **Alternativa 4:** Implementar Árboles binarios de búsqueda AVL.

Esta alternativa permite adaptarse a la situación presentada. Se destaca que la búsqueda en esta estructura es de $O(\log n)$.

Fase 5: Evaluación y selección de la mejor solución.

Se definen los criterios que permitirán evaluar las alternativas de solución y con base en este resultado elegir la solución que mejor satisface las necesidades del problema planteado. Los criterios que escogimos en este caso son los que enumeramos a continuación. Al lado de cada uno se ha establecido un valor numérico con el objetivo de establecer un peso que indique cuáles de los valores posibles de cada criterio tienen más peso (i.e., son más deseables).

Criterios

- **Criterio A: Complejidad espacial de las estructuras de datos a usar en el peor de los casos.**

[1] $O(n^2)$

[2] $O(n)$
 [3] $O(\log n)$
 [4] $O(1)$

- **Criterio B: Eficiencia en la búsqueda de datos.**

[1] No es muy eficiente
 [2] Es muy eficiente

Evaluación

Alternativa	Criterio A	Criterio B	Total
1. Implementar ArrayList	[2] $O(n)$	[1] No es muy eficiente	3
3. Implementar Árboles binarios de búsqueda ABB.	[2] $O(n)$	[1] No es muy eficiente	3
4. Implementar Árboles binarios de búsqueda AVL.	[3] $O(\log n)$	[2] Es muy eficiente	5

Selección

De acuerdo con la evaluación anterior se debe seleccionar la **Alternativa 4**, ya que obtuvo la mayor puntuación (5) de acuerdo con los criterios definidos.

Implementación del diseño.

Mockups preliminares

¡¡BIENVENIDO!!

Seleccione una opción para continuar

1. Gestionar jugadores

2. Importar jugadores

3. Buscar jugadores

4. Salir

Gestionar jugador

Nombre:

Edad:

Equipo:

Puntos por partido:

Rebotes por partido:

Asistencias por partido:

Robos por partido:

Bloqueos por partido:

Actualizar

Agregar

Eliminar

Tabla de jugadores agregados

Nombre	Edad	Equipo	Puntos pp	Rebotes p.p	Asistencias	Robos	Bloqueos

*Seleccione un jugador de la tabla para actualizarlo o eliminarlo

Atrás

Buscar jugador

Seleccione el criterio de búsqueda a implementar:

☐ Puntos por partido
 ☐ Robos por partido
 ☐ Asistencias por partido
 ☐ Bloqueos por partido
 ☐ Rebotes por partido

Valor del criterio:

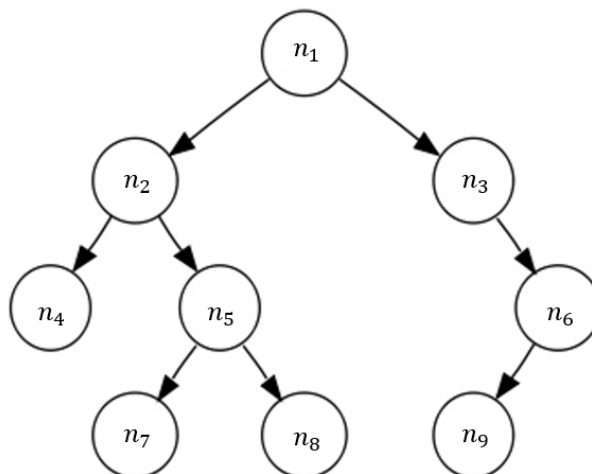
Desea que los resultados encontrados sean...

☐ Iguales
 ☐ Mayores
 ☐ Menores

Nombre	Edad	Equipo	Puntos pp	Rebotes p.p	Asistencias	Robos	Bloqueos

Diseño del TAD para cada estructura de datos requerida

TAD ABB



Donde n_1 es el nodo de la raíz, n_2 es el elemento a la izquierda de esta y n_3 aquel nodo que se encuentra a su derecha.

$key[n_4] < key[n_2] < key[n_7] < key[n_5] < key[n_8] < key[n_1] < key[n_3] < key[n_9] < key[n_6]$

{inv: Sea n_i un nodo del árbol,

Si n_x es un nodo en el subárbol izquierdo de n_i , entonces $key[n_x] < key[n_i]$

Si n_x es un elemento en el subárbol derecho de n_l , entonces $key[n_x] > key[n_l]$

Operaciones principales:

- *Buscar (Analizadora)*: $\text{Árbol } x \text{ key} \rightarrow \text{Nodo}$
- *Insertar (Modificadora)*: $\text{Árbol } x \text{ key } x \text{ valor} \rightarrow \text{Árbol}$
- *Eliminar (Modificadora)*: $\text{Árbol } x \text{ key} \rightarrow \text{booleano}$
- *CrearArbol (Constructora)*: $\rightarrow \text{Árbol}$
- *recorridoInorden (Analizadora)*: $\text{Árbol} \rightarrow \text{Nodos}$

Insertar(*Árbol*,key, valor)

“Inserta un nodo en el árbol binario de búsqueda. Si la key ya existe dentro del árbol, el nuevo nodo a insertar es añadido dentro de la lista de nodos que posee el nodo con la misma key.”

{pre: valor y key no pueden ser nulos y Árbol debe estar previamente inicializado}

{post: se creó un nuevo nodo con la key y el valor dados, este se añadió al árbol, el cual quedó modificado.}

Eliminar(*Árbol*,key)

“Elimina un nodo del árbol binario de búsqueda”

{pre: key tiene que ser diferente de nulo y Árbol debe estar previamente inicializado}

{post: true o false dependiendo de si el nodo relacionado con la key dada fue o no eliminado del árbol.}

Buscar(*Árbol*,key)

“Busca un nodo dentro del árbol a partir de su key y lo retorna”

{pre: key tiene que ser diferente de nulo y Árbol debe estar previamente inicializado}

{post: el nodo a partir de la key dada fue encontrado y retornado}

CrearArbol()

“Crea un nuevo árbol binario de búsqueda vacío”

{pre: TRUE}

{post: árbol vacío creado}

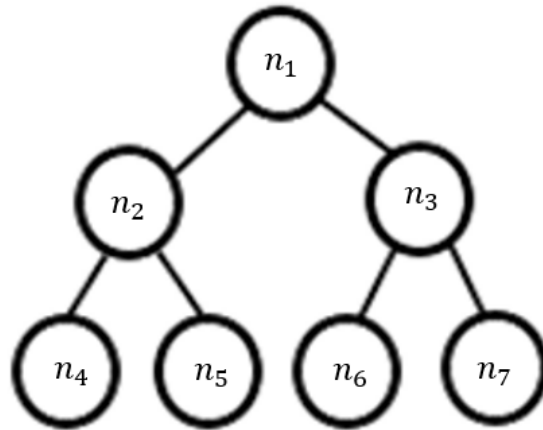
recorridoInorden(*Árbol*)

“Retorna los nodos del árbol en inorden (primero subárbol izquierdo, luego la raíz y por último el subárbol derecho).”

{pre: Árbol debe estar previamente inicializado}

{post: todos los nodos del árbol fueron retornados en inorden}

TAD AVL



Donde n_1 es el elemento de la raíz, n_2 es el elemento a la izquierda de esta y n_3 aquel que se encuentra a su derecha.

$key[n_4] < key[n_2] < key[n_7] < key[n_5] < key[n_6] < key[n_1] < key[n_3] < key[n_9] < key[n_6]$

{inv: Sea n_i un elemento del árbol,

Si n_x es un elemento en el subárbol izquierdo de n_i , entonces $key[n_x] < key[n_i]$

Si n_x es un elemento en el subárbol derecho de n_i , entonces $key[n_x] > key[n_i]$.

Cada nodo tendría un factor de balanceo 0, 1 ó - 1.

Para un nodo v del árbol, el factor de balanceo se calcula como:

$fb(v) = \text{altura del subárbol derecho de } v - \text{altura del subárbol izquierdo de } v$

Operaciones principales:

- Buscar (Analizadora): Árbol x key → Nodo
- Insertar (Modificadora): Árbol x key x valor → booleano
- Eliminar (Modificadora): Árbol x key → booleano
- CrearArbol (Constructora): → Árbol
- Rotar – Izquierda: Árbol x Nodo → Nodo
- Rotar – Derecha: Árbol x Nodo → Nodo
- recorridoInorden (Analizadora): Árbol → Nodos

Insertar(Árbol, key, valor)

“Inserta un nodo en el árbol AVL. Si la key ya existe dentro del árbol, el nuevo nodo a insertar es añadido dentro de la lista de nodos que posee el nodo con la misma key.”

{pre: valor y key no pueden ser nulos y Árbol debe estar previamente inicializado}
{post: se creó un nuevo nodo con la key y el valor dados, este se añadió al árbol, el cual quedó modificado.}

Eliminar(*Árbol*,key)
“Elimina un nodo del árbol binario balanceado”
{pre: key tiene que ser diferente de nulo y Árbol debe estar previamente inicializado}
{post: true o false dependiendo de si el nodo relacionado con la key dada fue o no eliminado del árbol.}

Buscar(*Árbol*,key)
“Busca un nodo dentro del árbol a partir de su key y lo retorna”
{pre: key tiene que ser diferente de nulo y Árbol debe estar previamente inicializado}
{post: el nodo a partir de la key dada fue encontrado y retornado}

CrearArbol()
“Crea un nuevo árbol binario balanceado vacío”
{pre: TRUE}
{post: árbol vacío creado}

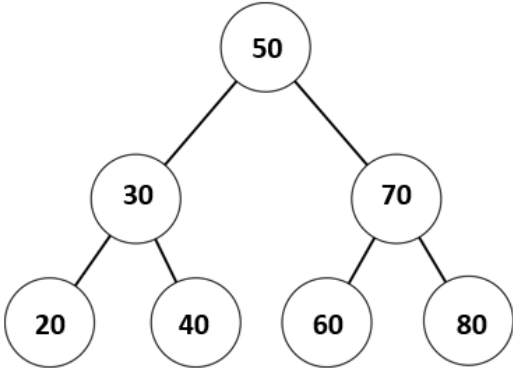
Rotar-Izquierda(*Árbol*, Nodo)
“Rota hacia la izquierda al subárbol del nodo para reestablecer el balance”
{pre: Árbol debe estar previamente inicializado, el nodo debe ser diferente de nulo}
{post: nodo rotado}

Rotar-Derecha(*Árbol*, Nodo)
“Rota hacia la derecha al subárbol del nodo para reestablecer el balance”
{pre: Árbol debe estar previamente inicializado, el nodo debe ser diferente de nulo}
{post: nodo rotado}

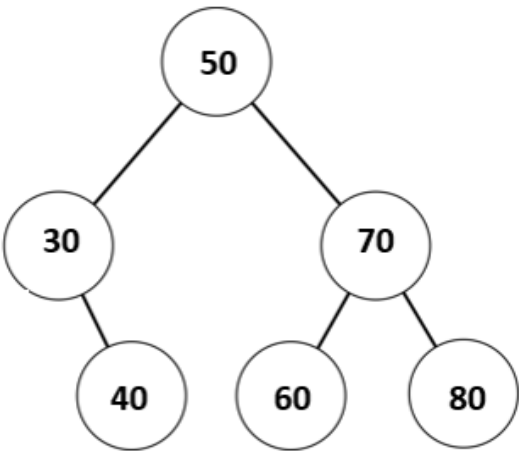
recorridoInorden(*Árbol*, Nodo)
“Retorna los nodos del árbol en inorden (primero subárbol izquierdo, luego la raíz y por último el subárbol derecho).”
{pre: Árbol debe estar previamente inicializado}
{post: todos los nodos del árbol fueron retornados en inorden}

Diseño de pruebas unitarias

Configuración de los escenarios:

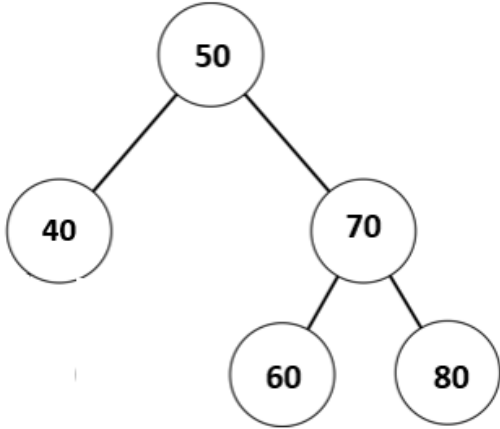
Nombre	Clase	Escenario
setupScenary1	BSTtree	Árbol binario de búsqueda vacío inicializado (sin ningún jugador agregado).
setupScenary2	BSTtree	<p>Árbol binario de búsqueda inicializado con 7 jugadores guardados según sus puntos por partido. En cada uno de sus nodos podemos encontrar jugadores con los siguientes puntos por partido:</p>  <pre>graph TD; 50((50)) --> 30((30)); 50 --> 70((70)); 30 --> 20((20)); 30 --> 40((40)); 70 --> 60((60)); 70 --> 80((80));</pre> <ul style="list-style-type: none">• {key = 50, value = (objeto jugador) player1: { name = “Devin Booker” age = 24 team = “Phoenix Suns” points = 50 bounces = 10 assists = 15 steals = 6 blocks = 8} }• {key = 30, value = (objeto jugador) player2: { name = “Ray Allen” age = 30 team = “Conferencia Este” points = 30 bounces = 6 assists = 12 steals = 10 blocks = 5} }• {key = 70, value = (objeto jugador) player3: {

		<pre>name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 70 bounces = 4 assists = 8 steals = 13 blocks = 9} }</pre> <ul style="list-style-type: none">• {key = 20, value = (objeto jugador) player4: { name = "Charles Barkley" age = 31 team = "Miami Heat" points = 20 bounces = 16 assists = 9 steals = 3 blocks = 7} }• {key = 40, value = (objeto jugador) player5: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 18 blocks = 9} }• {key = 60, value = (objeto jugador) player6: { name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 60 bounces = 13 assists = 8 steals = 2 blocks = 1} }• {key = 80, value = (objeto jugador) player7: { name = "Larry Bird"
--	--	---

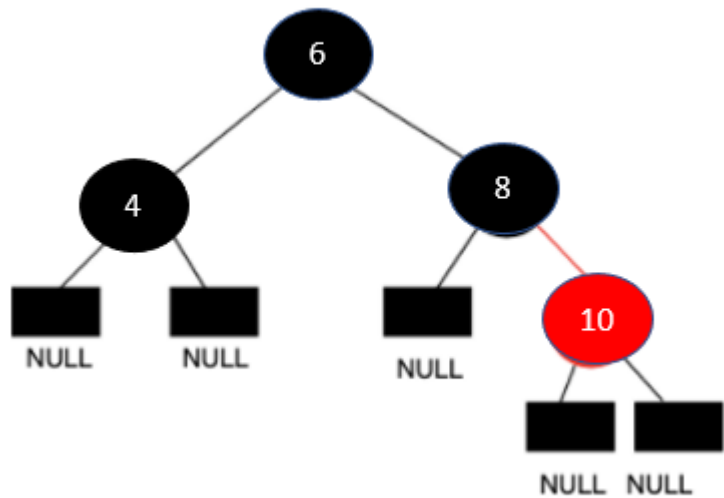
		<pre>age = 28 team = "Chicago Bulls" points = 80 bounces = 2 assists = 19 steals = 17 blocks = 5} }</pre>
setupScenary3	BSTtree	<p>Árbol binario de búsqueda inicializado con 6 jugadores guardados según sus puntos por partido. En cada uno de sus nodos podemos encontrar jugadores con los siguientes puntos por partido:</p>  <pre>graph TD 50((50)) --> 30((30)) 50 --> 70((70)) 30 --> 40((40)) 70 --> 60((60)) 70 --> 80((80))</pre> <ul style="list-style-type: none">• {key = 50, value = (objeto jugador) player1: { name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 50 bounces = 10 assists = 15 steals = 6 blocks = 8} }• {key = 30, value = (objeto jugador) player2: { name = "Ray Allen" age = 30 team = "Conferencia Este" points = 30 bounces = 6 assists = 12 steals = 10 blocks = 5} }

}

- {key = **70**,
value = (objeto jugador) player3: {
name = "Paul Arizin"
age = 28
team = "Toronto Raptors"
points = 70
bounces = 4
assists = 8
steals = 13
blocks = 9}
}
- {key = **40**,
value = (objeto jugador) player4: {
name = "Rick Barry"
age = 26
team = "Indiana Pacers"
points = 40
bounces = 2
assists = 5
steals = 18
blocks = 9}
}
- {key = **60**,
value = (objeto jugador) player5: {
name = "Dave Bing"
age = 27
team = "Brooklyn Nets"
points = 60
bounces = 13
assists = 8
steals = 2
blocks = 1}
}
- {key = **80**,
value = (objeto jugador) player6: {
name = "Larry Bird"
age = 28
team = "Chicago Bulls"
points = 80
bounces = 2
assists = 19
steals = 17
blocks = 5}
}

<p>setupScenary4</p>	<p>BSTtree</p>	<p>Árbol binario de búsqueda inicializado con 5 jugadores guardados según sus puntos por partido. En cada uno de sus nodos podemos encontrar jugadores con los siguientes puntos por partido:</p>  <pre> graph TD 50((50)) --- 40((40)) 50 --- 70((70)) 70 --- 60((60)) 70 --- 80((80)) </pre> <ul style="list-style-type: none"> • {key = 50, value = (objeto jugador) player1:{ name = “Devin Booker” age = 24 team = “Phoenix Suns” points = 50 bounces = 10 assists = 15 steals = 6 blocks = 8} } • {key = 40, value = (objeto jugador) player2:{ name = “Rick Barry” age = 26 team = “Indiana Pacers” points = 40 bounces = 2 assists = 5 steals = 18 blocks = 9} } • {key = 70, value = (objeto jugador) player3:{ name = “Paul Arizin” age = 28 team = “Toronto Raptors” points = 70 bounces = 4 assists = 8 steals = 13
----------------------	----------------	--

		<pre>blocks = 9} }</pre> <ul style="list-style-type: none"> • {key = 60, value = (objeto jugador) player4: { name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 60 bounces = 13 assists = 8 steals = 2 blocks = 1} } • {key = 80, value = (objeto jugador) player5: { name = "Larry Bird" age = 28 team = "Chicago Bulls" points = 80 bounces = 2 assists = 19 steals = 17 blocks = 5} }
setupScenary1	RedBlackTree	Árbol rojinegro vacío inicializado (sin ningún jugador agregado). Almacena jugadores según sus robos por partido .
setupScenary2	RedBlackTree	Árbol rojinegro inicializado con jugadores guardados según sus robos por partido . En cada uno de sus nodos podemos encontrar jugadores con las siguientes robos por partido:



- {key = 4,
value = (objeto jugador) player1: {
name = "Ray Allen"
age = 30
team = "Conferencia Este"
points = 30
bounces = 6
assists = 9
steals = 4
blocks = 5}
}
- {key = 8,
value = (objeto jugador) player2: {
name = "Paul Arizin"
age = 28
team = "Toronto Raptors"
points = 70
bounces = 4
assists = 20
steals = 8
blocks = 9}
}
- {key = 6,
value = (objeto jugador) player3: {
name = "Devin Booker"
age = 24
team = "Phoenix Suns"

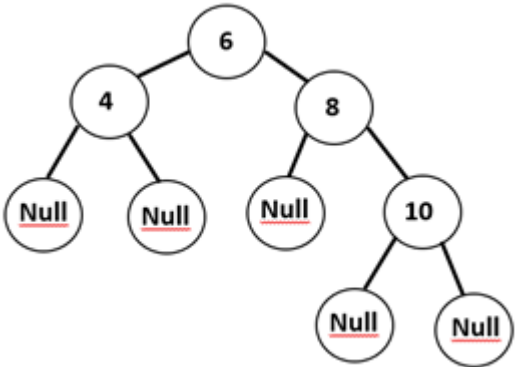
		<div>points = 50 bounces = 10 assists = 17 steals = 6 blocks = 8}</div> <div><ul style="list-style-type: none">{key = 10, value = (objeto jugador) player4: { name = "Charles Barkley" age = 31 team = "Miami Heat" points = 20 bounces = 16 assists = 6 steals = 10 blocks = 7} }</div>
setupScenary3	RedBlackTree	<div>Árbol rojinegro inicializado con jugadores guardados según sus robos por partido. En cada uno de sus nodos podemos encontrar jugadores con las siguientes robos por partido:</div> <div><pre>graph TD; 6((6)) --- 4((4)); 6 --- 8((8)); 4 --- 2((2)); 4 --- NULL1[NULL]; 2 --- NULL2[NULL]; 2 --- NULL3[NULL]; 8 --- NULL4[NULL]; 8 --- NULL5[NULL]; style 6 fill:#000,color:#fff; style 4 fill:#000,color:#fff; style 8 fill:#000,color:#fff; style 2 fill:#f00,color:#fff; linkStyle 0 stroke:#f00; linkStyle 1 stroke:#f00; linkStyle 2 stroke:#000; linkStyle 3 stroke:#000; linkStyle 4 stroke:#000; linkStyle 5 stroke:#000; linkStyle 6 stroke:#000; linkStyle 7 stroke:#000; linkStyle 8 stroke:#000; linkStyle 9 stroke:#000;</pre></div> <div><ul style="list-style-type: none">{key = 8, value = (objeto jugador) player2: { name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 70</div>

```
bounces = 4  
assists = 20  
steals = 8  
blocks = 9}  
}
```

- {key = 4,
value = (objeto jugador) player1: {
name = "Ray Allen"
age = 30
team = "Conferencia Este"
points = 30
bounces = 6
assists = 9
steals = 4
blocks = 5}
}

- {key = 6,
value = (objeto jugador) player3: {
name = "Devin Booker"
age = 24
team = "Phoenix Suns"
points = 50
bounces = 10
assists = 17
steals = 6
blocks = 8}

- {key = 2,
value = (objeto jugador) player4: {
name = "Charles Barkley"
age = 31
team = "Miami Heat"
points = 20
bounces = 16
assists = 6
steals = 2
blocks = 7}
}

setupScenary1	AVLtree	Árbol AVL vacío inicializado (sin ningún jugador agregado). Almacena jugadores según sus bloqueos por partido .
setupScenary2	AVLtree	<p>Árbol AVL inicializado con jugadores guardados según sus bloqueos por partido. En cada uno de sus nodos podemos encontrar jugadores con los siguientes bloqueos por partido por partido:</p>  <pre>graph TD; 6((6)) --- 4((4)); 6 --- 8((8)); 4 --- Null1((Null)); 4 --- Null2((Null)); 8 --- Null3((Null)); 8 --- 10((10)); 10 --- Null4((Null)); 10 --- Null5((Null));</pre> <ul style="list-style-type: none">• {key = 4, value = (objeto jugador) player1: { name = “Ray Allen” age = 30 team = “Conferencia Este” points = 30 bounces = 6 assists = 9 steals = 5 blocks = 4} }• {key = 8, value = (objeto jugador) player2: { name = “Paul Arizin” age = 28 team = “Toronto Raptors” points = 70 bounces = 4 assists = 20 steals = 9

		<div><div><div>blocks = 8}</div><div>}</div></div><div><div>• {key = 6,</div><div>value = (objeto jugador) player3: {</div><div>name = “Devin Booker”</div><div>age = 24</div><div>team = “Phoenix Suns”</div><div>points = 50</div><div>bounces = 10</div><div>assists = 17</div><div>steals = 8</div><div>blocks = 6}</div></div></div> <div><div>• {key = 10,</div><div>value = (objeto jugador) player4: {</div><div>name = “Charles Barkley”</div><div>age = 31</div><div>team = “Miami Heat”</div><div>points = 20</div><div>bounces = 16</div><div>assists = 6</div><div>steals = 7</div><div>blocks = 10}</div><div>}</div></div>
--	--	---

- {key = 8,
value = (objeto jugador) player2: {
name = "Paul Arizin"
age = 28
team = "Toronto Raptors"
points = 70
bounces = 4
assists = 20
steals = 9
blocks = 8}
}

- {key = 4,
value = (objeto jugador) player1: {
name = "Ray Allen"
age = 30
team = "Conferencia Este"
points = 30
bounces = 6
assists = 9
steals = 5
blocks = 4}
}

- {key = 6,
value = (objeto jugador) player3: {
name = "Devin Booker"
age = 24
team = "Phoenix Suns"
points = 50
bounces = 10
assists = 17
steals = 8
blocks = 6}

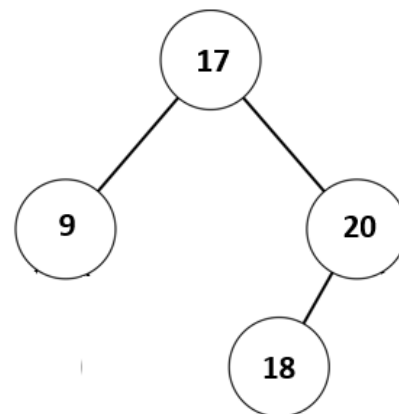
- {key = 2,
value = (objeto jugador) player4: {
name = "Charles Barkley"

		<pre> age = 31 team = "Miami Heat" points = 20 bounces = 16 assists = 6 steals = 7 blocks = 2 } </pre>
setupScenary1	Fiba	<p>Objeto de la clase controladora Fiba inicializado.</p> <p>Árbol ABB de puntos por partido inicializado.</p> <p>Árbol ABB de asistencias inicializado.</p> <p>Lista de jugadores organizados por rebotes inicializada.</p>
setupScenary2	Fiba	<p>Objeto de la clase controladora Fiba inicializado.</p> <p>Lista de rebotes y árboles ABB de puntos por partido y de asistencias inicializados y con los siguientes jugadores guardados:</p> <ul style="list-style-type: none"> <p>(objeto jugador) player1: {</p> <pre> name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8} </pre> <p>(objeto jugador) player2: {</p> <pre> name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5} </pre> <p>(objeto jugador) player3: {</p> <pre> name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 20 bounces = 4 assists = 18 steals = 13 </pre>

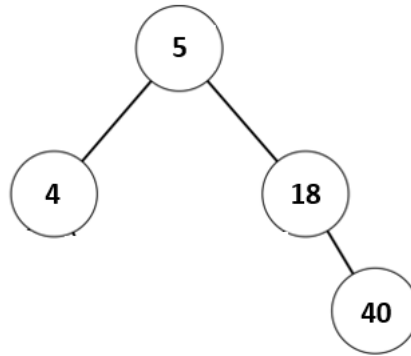
```
blocks = 9}
}
```

- (objeto jugador) player4: {
name = "Charles Barkley"
age = 31
team = "Miami Heat"
points = 9
bounces = 12
assists = 18
steals = 3
blocks = 7}
}
- (objeto jugador) player5: {
name = "Dave Bing"
age = 27
team = "Brooklyn Nets"
points = 18
bounces = 15
assists = 40
steals = 2
blocks = 1}
}

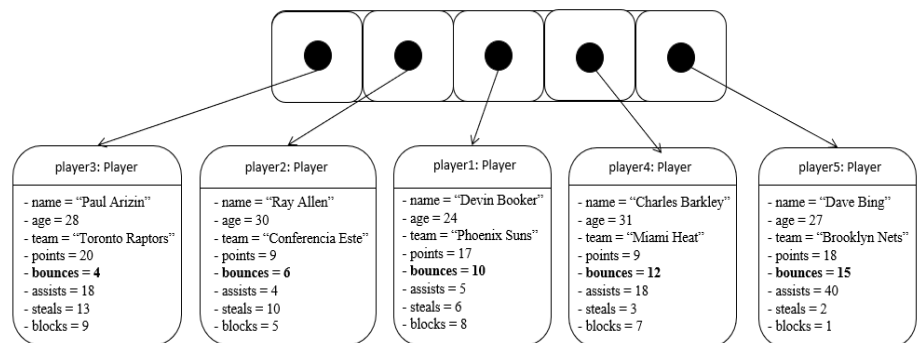
Representación del árbol de puntos por partido:



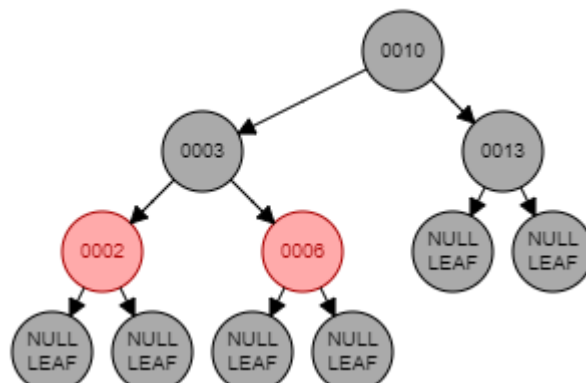
Representación del árbol de asistencias:



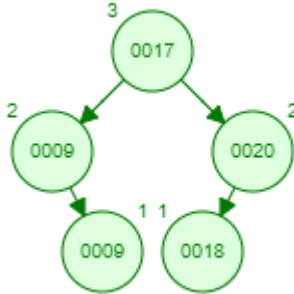
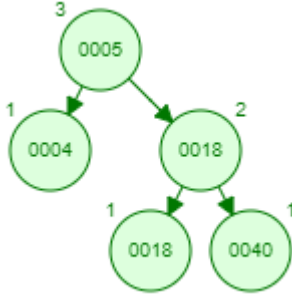
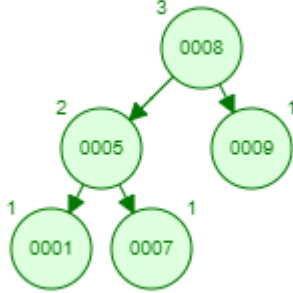
Representación de la lista de rebotes:



Representación del árbol de robos:



Representación del árbol puntos por partido (AVL):

		 <p>Representación del árbol de asistencia (AVL):</p>  <p>Representación del árbol de bloqueos (AVL)</p> 
--	--	---

Diseño de los casos de prueba

Objetivo de la Prueba: Validar que se agrega correctamente un nuevo jugador al árbol binario de búsqueda.				
Clase	Método	Escenario	Valores de Entrada	Resultado

BSTtree	insertNode	setupScenary1	key = 50 , value = player1:{ name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 50 bounces = 10 assists = 15 steals = 6 blocks = 8}	El jugador ha sido agregado al árbol binario de búsqueda exitosamente según sus puntos por partido. Ahora, el árbol tiene su raíz ocupada con dicho jugador.
BSTtree	insertNode	setupScenary2	key = 10 , value = player8:{ name = "Chris Bosh" age = 24 team = "Atlanta Hawks" points = 10 bounces = 10 assists = 15 steals = 6 blocks = 8}	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la izquierda del jugador con 20 puntos por partido.
BSTtree	insertNode	setupScenary2	key = 90 , value = player8:{ name = "Carl Braun" age = 23 team = "Orlando Magic" points = 90 bounces = 10 assists = 15 steals = 6 blocks = 8}	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la derecha del jugador con 80 puntos por partido.

Objetivo de la Prueba: Validar que se busca y retorna correctamente un jugador del árbol binario de búsqueda.				
Clase	Método	Escenario	Valores de Entrada	Resultado
BSTtree	searchNode	setupScenary1	key = 50	El jugador no ha sido encontrado en el árbol binario de búsqueda de puntos por partido. El nodo retornado es nulo, ya que el árbol está vacío.

BSTtree	searchNode	setupScenary2	key = 10	El jugador no ha sido encontrado en el árbol binario de búsqueda de puntos por partido, por lo tanto, el nodo retornado es nulo.
BSTtree	searchNode	setupScenary2	key = 60	<p>El jugador ha sido encontrado en el árbol binario de búsqueda de puntos por partido. El nodo retornado contiene los siguientes atributos:</p> <pre> {key = 60, value = (objeto jugador){ name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 60 bounces = 13 assists = 8 steals = 2 blocks = 1} }</pre>

Objetivo de la Prueba: Validar que se elimina correctamente un jugador en el árbol binario de búsqueda..				
Clase	Método	Escenario	Valores de Entrada	Resultado
BSTtree	deleteNode	setupScenary1	key= 50 .	false El árbol se encuentra vacío, ya que solamente se ha inicializado, por lo tanto no se puede eliminar ningún nodo de él.
BSTtree	deleteNode	setupScenary2	key= 10	false El árbol no contiene ningún nodo que coincida con la key dada, por lo que no se puede eliminar un nodo que no existe dentro del árbol.
BSTtree	deleteNode	setupScenary2	key= 20	true Según el escenario, el nodo con la key dada es una hoja dentro del árbol, por lo que entra en el primer caso que resuelve este método. Asimismo, el nodo con key 30 quien era el padre del nodo con la key dada, ahora tiene su hijo derecho nulo.
BSTtree	deleteNode	setupScenary3	key= 30	true Según el escenario, el nodo con la key

				dada es un nodo que tiene un hijo dentro del árbol, por lo que entra en el segundo caso que resuelve este método. Asimismo, la raíz quien era el padre del nodo con la key dada, ahora tiene su hijo izquierdo lleno con el nodo de key 40.
BSTtree	deleteNode	setupScenary4	key= 50	true Según el escenario, el nodo con la key dada es un nodo que tiene dos hijos dentro del árbol, por lo que entra en el último y tercer caso que resuelve este método. Cabe resaltar que el nodo a eliminar es la raíz, además, quien ahora pasa a ser su sucesor es el nodo con la key 60, el cual va a contener como hijo izquierdo al nodo con key 40 y como hijo derecho al nodo con key 70.

Objetivo de la Prueba: Validar que se retornan correctamente los elementos de un árbol binario de búsqueda en recorrido inorder.				
Clase	Método	Escenario	Valores de Entrada	Resultado
BSTtree	inorderTraversal	setupScenary1		Una lista de nodos vacía, ya que el árbol no tiene elementos añadidos.
BSTtree	inorderTraversal	setupScenary2		<p>Una lista de nodos en el siguiente orden:</p> <ul style="list-style-type: none"> • {key = 20, value = (objeto jugador) player4: { name = "Charles Barkley" age = 31 team = "Miami Heat" points = 20 bounces = 16 assists = 9 steals = 3 blocks = 7} } • {key = 30, value = (objeto jugador)

				<pre> player2: { name = "Ray Allen" age = 30 team = "Conferencia Este" points = 30 bounces = 6 assists = 12 steals = 10 blocks = 5} } </pre> <ul style="list-style-type: none"> <pre> {key = 40, value = (objeto jugador) player5: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 18 blocks = 9} } </pre> <pre> {key = 50, value = (objeto jugador) player1: { name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 50 bounces = 10 assists = 15 steals = 6 blocks = 8} } </pre> <pre> {key = 60, value = (objeto jugador) player6: { name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 60 bounces = 13 assists = 8 steals = 2 blocks = 1} } </pre>
--	--	--	--	---

				<ul style="list-style-type: none"> • {key = 70, value = (objeto jugador) player3: { name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 70 bounces = 4 assists = 8 steals = 13 blocks = 9} } • {key = 80, value = (objeto jugador) player7: { name = "Larry Bird" age = 28 team = "Chicago Bulls" points = 80 bounces = 2 assists = 19 steals = 17 blocks = 5} }
--	--	--	--	--

Objetivo de la Prueba: Validar que se agrega correctamente un nuevo jugador al árbol rojinegro organizado según los robos por partido de cada jugador.				
Clase	Método	Escenario	Valores de Entrada	Resultado
RedBlackTree	insert	setupScenary1	{key = 4 , value = (objeto jugador) player: { name = "Ray Allen" age = 30 team = "Conferencia Este" points = 30 bounces = 6 assists = 9 steals = 4 blocks = 5} }	El jugador ha sido agregado al árbol rojinegro exitosamente. Ahora, el árbol tiene 1 sólo nodo, en este caso su raíz ocupada con dicho jugador, de color negro.

RedBlackTree	insert	setupScenary2	<pre>{key = 9 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 9 blocks = 9} }</pre>	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la derecha del jugador con 6 robos por partido, y tiene color negro.
RedBlackTree	insert	setupScenary2	<pre>{key = 11 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 11 blocks = 9} }</pre>	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la derecha del jugador con 10 robos por partido, y tiene color rojo.
RedBlackTree	insert	setupScenary2	<pre>{key = 5 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 5 blocks = 9} }</pre>	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la derecha del jugador con 4 robos por partido, y tiene color rojo.

RedBlackTree	insert	setupScenary3	<pre>{key = 1 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 1 blocks = 9} }</pre>	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la izquierda del jugador con 2 robos por partido, y tiene color rojo.
RedBlackTree	insert	setupScenary3	<pre>{key = 3 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals =3 blocks = 9} }</pre>	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la izquierda del jugador con 6 robos por partido, y tiene color negro.
RedBlackTree	insert	setupScenary3	<pre>{key = 7 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 7 blocks = 9} }</pre>	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la izquierda del jugador con 8 robos por partido, y tiene color rojo.

Objetivo de la Prueba: Validar que se busca correctamente un jugador en el árbol rojinegro según los robos por partido de cada jugador.				
Clase	Método	Escenario	Valores de Entrada	Resultado
RedBlackTree	search	setupScenary1	key = 7	El jugador no ha sido encontrado en el árbol. El nodo retornado es nulo, ya que el árbol está vacío.
RedBlackTree	search	setupScenary2	key = 10	El jugador ha sido encontrado en el árbol. El nodo retornado contiene los siguientes atributos: <ul style="list-style-type: none"> {key = 10, value = (objeto jugador) player4: { name = "Charles Barkley" age = 31 team = "Miami Heat" points = 20 bounces = 16 assists = 6 steals = 10 blocks = 7} }
RedBlackTree	search	setupScenary3	key= 2	El jugador ha sido encontrado en el árbol. El nodo retornado contiene los siguientes atributos: <ul style="list-style-type: none"> {key = 2, value = (objeto jugador) player4: { name = "Charles Barkley" age = 31 team = "Miami Heat" points = 20 bounces = 16 assists = 6 steals = 2 blocks = 7} }

Objetivo de la Prueba: Validar que se elimina correctamente un jugador en el árbol rojinegro organizado según los robos por partido de cada jugador.				
Clase	Método	Escenario	Valores de Entrada	Resultado

RedBlackTree	delete	setupScenary2	key= 6	El jugador con 6 robos por partido se ha eliminado del árbol.
RedBlackTree	delete	setupScenary2	key= 8	El jugador con 6 robos por partido se ha eliminado del árbol.
RedBlackTree	delete	setupScenary2	key= 4	El jugador con 4 robos por partido se ha eliminado del árbol.
RedBlackTree	delete	setupScenary2	key= 10	El jugador con 10 robos por partido se ha eliminado del árbol.
RedBlackTree	delete	setupScenary3	key= 6	El jugador con 6 robos por partido se ha eliminado del árbol.
RedBlackTree	delete	setupScenary3	key= 4	El jugador con 4 robos por partido se ha eliminado del árbol.
RedBlackTree	delete	setupScenary3	key= 8	El jugador con 8 robos por partido se ha eliminado del árbol.
RedBlackTree	delete	setupScenary3	key= 2	El jugador con 2 robos por partido se ha eliminado del árbol.

Objetivo de la Prueba: Validar que se retornan correctamente los elementos de un árbol rojinegro en recorrido inorder.				
Clase	Método	Escenario	Valores de Entrada	Resultado
RedBlackTree	inorderTraversal	setupScenary1		Una lista de nodos vacía, ya que el árbol no tiene elementos añadidos.
RedBlackTree	inorderTraversal	setupScenary2		Una lista de nodos de los elementos del árbol, en el siguiente orden de acuerdo a los robos por partido de los jugadores en el árbol: 4, 6, 8, 10

RedBlack Tree	inorderTraversal	setupScenary3		Una lista de nodos de los elementos del árbol, en el siguiente orden de acuerdo a los robos por partido de los jugadores en el árbol: 2, 4, 6, 8
---------------	------------------	---------------	--	--

Objetivo de la Prueba: Validar que se agrega correctamente un nuevo jugador al árbol AVL organizado según los bloqueos por partido de cada jugador.				
Clase	Método	Escenario	Valores de Entrada	Resultado
AVLTree	insert	setupScenary1	{key = 4, value = (objeto jugador) player: { name = "Ray Allen" age = 30 team = "Conferencia Este" points = 30 bounces = 6 assists = 9 steals = 5 blocks = 4 } }	El jugador ha sido agregado al árbol AVL exitosamente. Ahora, el árbol tiene 1 sólo nodo, en este caso su raíz ocupada con dicho jugador.
AVLTree	insert	setupScenary2	{ key = 9 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 9 blocks = 9 } }	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la derecha del jugador con 6 bloqueos por partido.

AVLTree	insert	setupScenary2	{key = 11 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 9 blocks = 11} }	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la derecha del jugador con 10 bloqueos por partido.
AVLTree	insert	setupScenary2	{key = 5 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 9 blocks =5} }	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la derecha del jugador con 4 bloqueos por partido.
AVLTree	insert	setupScenary3	{key = 1 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 9 blocks = 1} }	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la izquierda del jugador con 2 bloqueos por partido.

AVLTree	insert	setupScenary3	{key = 3 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 9 blocks = 3} }	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la izquierda del jugador con 6 bloqueos por partido.
AVLTree	insert	setupScenary3	{key = 7 value = (objeto jugador) player: { name = "Rick Barry" age = 26 team = "Indiana Pacers" points = 40 bounces = 2 assists = 5 steals = 9 blocks = 7} }	El jugador ha sido agregado al árbol exitosamente. Ahora, este se encuentra ubicado a la izquierda del jugador con 8 bloqueos por partido.

Objetivo de la Prueba: Validar que se busca correctamente un jugador en el árbol AVL según los bloqueos por partido de cada jugador.				
Clase	Método	Escenario	Valores de Entrada	Resultado
AVLTree	search	setupScenary1	key = 7	El jugador no ha sido encontrado en el árbol. El nodo retornado es nulo, ya que el árbol está vacío.
AVLTree	search	setupScenary2	key = 10	El jugador ha sido encontrado en el árbol. El nodo retornado contiene los siguientes atributos: <ul style="list-style-type: none"> {key = 10, value = (objeto jugador) player4: { name = "Charles Barkley" age = 31 team = "Miami Heat"

				<pre> points = 20 bounces = 16 assists = 6 steals = 7 blocks = 10 } </pre>
AVLTree	search	setupScenary3	key= 2	<p>El jugador ha sido encontrado en el árbol. El nodo retornado contiene los siguientes atributos:</p> <ul style="list-style-type: none"> • {key = 2, value = (objeto jugador) player4: { name = "Charles Barkley" age = 31 team = "Miami Heat" points = 20 bounces = 16 assists = 6 steals = 7 blocks = 2 }

Objetivo de la Prueba: Validar que se elimina correctamente un jugador en el árbol AVL organizado según los robos por partido de cada jugador.				
Clase	Método	Escenario	Valores de Entrada	Resultado
AVLTree	delete	setupScenary2	key= 6	El jugador con 6 bloqueos por partido se ha eliminado del árbol.
AVLTree	delete	setupScenary2	key= 8	El jugador con 6 bloqueos por partido se ha eliminado del árbol.
AVLTree	delete	setupScenary2	key= 4	El jugador con 4 bloqueos por partido se ha eliminado del árbol.

AVLTree	delete	setupScenary2	key= 10	El jugador con 10 bloqueos por partido se ha eliminado del árbol.
AVLTree	delete	setupScenary3	key= 6	El jugador con 6 bloqueos por partido se ha eliminado del árbol.
AVLTree	delete	setupScenary3	key= 4	El jugador con 4 bloqueos por partido se ha eliminado del árbol.
AVLTree	delete	setupScenary3	key= 8	El jugador con 8 bloqueos por partido se ha eliminado del árbol.
AVLTree	delete	setupScenary3	key= 2	El jugador con 2 bloqueos por partido se ha eliminado del árbol.

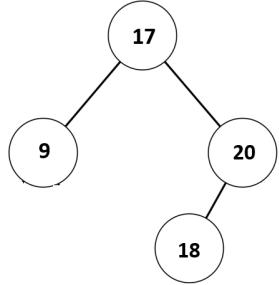
Objetivo de la Prueba: Validar que se retornan correctamente los elementos de un árbol AVL en recorrido inorden.				
Clase	Método	Escenario	Valores de Entrada	Resultado
AVLTree	inorderTraversal	setupScenary1		Una lista de nodos vacía, ya que el árbol no tiene elementos añadidos.
AVLTree	inorderTraversal	setupScenary2		Una lista de nodos de los elementos del árbol, en el siguiente orden de acuerdo a los bloqueos por partido de los jugadores en el árbol: 4, 6, 8, 10
AVLTree	inorderTraversal	setupScenary3		Una lista de nodos de los elementos del árbol, en el siguiente orden de acuerdo a los bloqueos por partido de los jugadores en el árbol: 2, 4, 6, 8

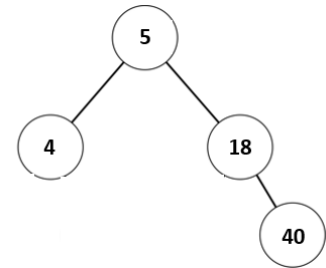
Objetivo de la Prueba: Validar que se agrega correctamente un nuevo jugador al programa.				
Clase	Método	Escenario	Valores de Entrada	Resultado

Fiba	addPlayer	setupScenary1	name = "Joe Harris", age = 27, team = "Phoenix Suns", points = 9, bounces = 10, assists = 18, steals = 6, blocks = 8	true El jugador ha sido agregado al programa. Ahora, el árbol de puntos por partido tiene su raíz ocupada, la cual contiene la key que es igual a 9 y su value es el jugador creado con los atributos dados. Asimismo, el árbol de asistencias tiene su raíz ocupada, la cual contiene la key que es igual a 18 y su value es el jugador creado con los atributos dados. La lista de jugadores por rebote tiene 1 elemento en su primera casilla, el cual es el jugador creado a partir de los atributos dados. El árbol de robos tiene su raíz ocupada, al igual que el de bloqueos.
Fiba	addPlayer	setupScenary2	name = "Joe Harris", age = 27, team = "Phoenix Suns", points = 9, bounces = 10, assists = 18, steals = 6, blocks = 8	true El jugador ha sido agregado al programa exitosamente. Ahora, dentro del árbol de puntos por partido el nuevo jugador fue insertado en la posición 1 de la lista de nodos con la misma key, del jugador con las sgtes características: <pre> {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5} } </pre> Asimismo, dentro del árbol de asistencias el nuevo jugador fue insertado en la posición 1 de la lista de nodos con la misma key, del jugador con las sgtes características: <pre> {name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 20 bounces = 4 </pre>

				assists = 18 steals = 13 blocks = 9} } <p>La lista de jugadores por rebote tiene ahora 6 elementos y el nuevo jugador quedó guardado en la casilla #3.</p> <p>En el árbol de robos, el nuevo jugador fue insertado en la primera posición de la lista de nodos con la misma key, del jugador con las sgtes características: {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8}</p>
Fiba	addPlayer	setupScenary2	name = "Joe Harris", age = 27, team = "Phoenix Suns", points = -9, bounces = 10, assists = 18, steals = 6, blocks = 8	false <p>El jugador no fue agregado al programa, ya que el valor ingresado para los puntos por partido es negativo.</p>
Fiba	addPlayer	setupScenary2	name = "Joe Harris", age = 27, team = "Phoenix Suns", points = 9, bounces = diez, assists = 18, steals = 6, blocks = 8	false <p>El jugador no fue agregado al programa, ya que el valor ingresado para los rebotes no es un número.</p>

Objetivo de la Prueba: Validar que se elimina correctamente un jugador del programa.

Clase	Método	Escenario	Valores de Entrada	Resultado
Fiba	deletePlayer	setupScenary2	(objeto Player) player: { name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7	<p>El jugador escogido ha sido eliminado exitosamente del programa.</p> <p>Cada una de las estructuras queda de la siguiente manera:</p> <ul style="list-style-type: none"> ABB puntos por partido: Ahora, la lista de nodos con la misma key que tiene el jugador con estas características se encuentra vacía... <pre>{name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5}</pre>  ABB asistencias: Ahora, la lista de nodos con la misma key que tiene el jugador con estas características se encuentra vacía... <pre>{name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 20 bounces = 4 assists = 18 steals = 13 blocks = 9}</pre>

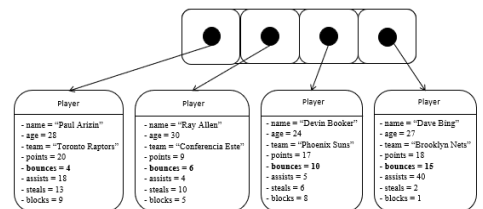


- Lista de jugadores por rebotes:

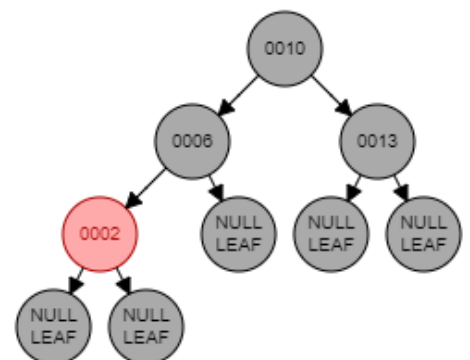
Ahora su tamaño es 4 y la casilla #3 que estaba ocupada por el elemento escogido para borrar contiene el siguiente jugador...

```


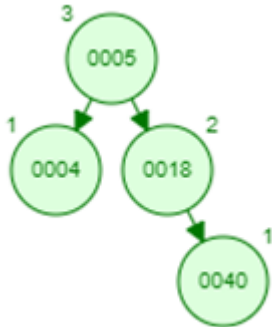
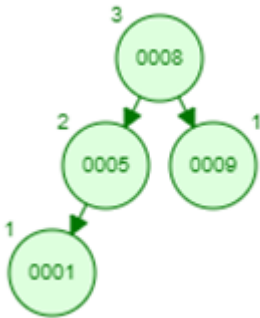
{
  name = "Dave Bing"
  age = 27
  team = "Brooklyn Nets"
  points = 18
  bounces = 15
  assists = 40
  steals = 2
  blocks = 1
}
  
```



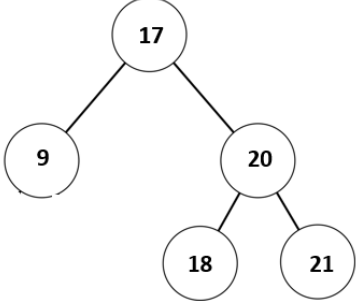
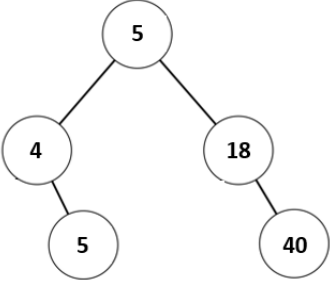
- Árbol de robos:



Representación del árbol puntos por partido (AVL):

				 <p>Representación del árbol de asistencia (AVL):</p>  <p>Representación del árbol de bloqueos (AVL):</p> 
--	--	--	--	--

Objetivo de la Prueba: Validar que se actualiza correctamente un jugador del programa.				
Clase	Método	Escenario	Valores de Entrada	Resultado

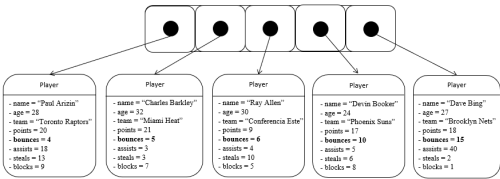
Fiba	updatePlayer	setupScenary2	<p>(objeto Player)</p> <p>player: {name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7}</p> <p>name = "Charles Barkley" age = 32 team = "Miami Heat" points = 21 bounces = 5 assists = 5 steals = 7 blocks = 4</p>	<p>true</p> <p>Cada una de las estructuras quedó de la siguiente manera:</p> <ul style="list-style-type: none">• ABB de puntos por partido: <pre>graph TD; 17((17)) --> 9((9)); 17 --> 20((20)); 20 --> 18((18)); 20 --> 21((21));</pre> <p>Ahora, la lista de nodos con la misma key del jugador con estas características está vacía...</p> <p>{name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5} }</p> <p>Además, el nodo que contiene al jugador con veinte rebotes ahora tiene su hijo derecho con el jugador que se actualizó.</p> <ul style="list-style-type: none">• ABB de asistencias: <pre>graph TD; 5((5)) --> 4((4)); 5 --> 18((18)); 4 --> 5_2((5)); 18 --> 40((40));</pre> <p>Ahora, la lista de nodos con la misma key del jugador con</p>
------	--------------	---------------	---	--

estas características está vacía...

```
{name = "Paul Arizin"  
age = 28  
team = "Toronto Raptors"  
points = 20  
bounces = 4  
assists = 18  
steals = 13  
blocks = 9}
```

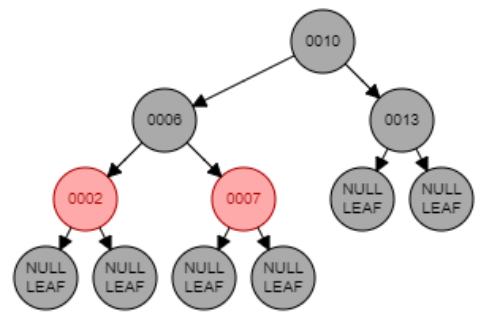
Además, el nodo que contiene al jugador con cinco rebotes (la raíz), ahora tiene en su lista de nodos con la misma key a 1 elemento, en este caso el jugador que se actualizó.

- Lista de jugadores por rebote:

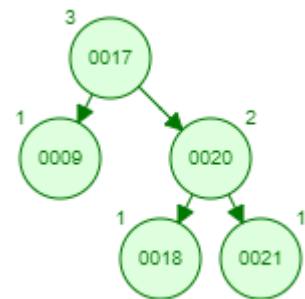


Las casillas de la lista quedaron organizadas de la siguiente manera:

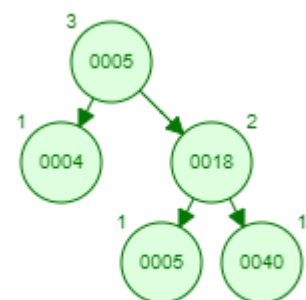
- #0: contiene al jugador con 4 rebotes.
- #1: contiene al jugador con 5 rebotes.
- #2: contiene al jugador con 6 rebotes.
- #3: contiene al jugador con 10 rebotes.
- #4: contiene al jugador con 15 rebotes.
- Árbol de robos:



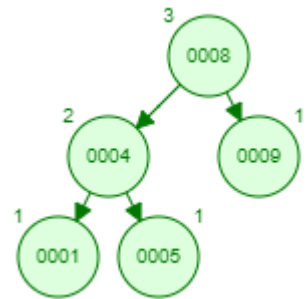
Representación del árbol puntos por partido (AVL):



Representación del árbol de asistencia (AVL):



Representación del árbol de bloqueos (AVL):



Fiba	updatePlayer	setupScenary2	(objeto Player) player: {name = “Charles Barkley” age = 31 team = “Miami Heat” points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7} name = “Charles Barkley” age = 32 team = “Miami Heat” points = -5 bounces = 5 assists = 3 steals = 3 blocks = 7	false El jugador no fue actualizado, ya que el valor ingresado para los puntos por partido es negativo.
Fiba	updatePlayer	setupScenary2	(objeto Player) player: {name = “Charles Barkley” age = 31 team = “Miami Heat” points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7} name = “Charles Barkley” age = 32	false El jugador no fue actualizado, ya que el valor ingresado para los rebotes no es un número.

			team = "Miami Heat" points = 21 bounces = 5 assists = nueve steals = 3 blocks = 7	
--	--	--	---	--

Objetivo de la Prueba: Validar que se buscan y retornan correctamente los jugadores almacenados por sus rebotes de acuerdo con el valor y criterio dado.				
Clase	Método	Escenario	Valores de Entrada	Resultado
Fiba	searchPlayersLinearly	setupScenary1	value = "20" comparison = "LESS"	Lista de jugadores vacía, ya que esta estructura no contiene ningún jugador y solo ha sido inicializada.
Fiba	searchPlayersLinearly	setupScenary2	value = "6" comparison = "EQUAL"	Lista de jugadores con 1 elemento, en este caso el jugador con las siguientes características: posición #0: {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5}
Fiba	searchPlayersLinearly	setupScenary2	value = "6" comparison = "GREATER"	Lista de jugadores con 3 elementos, en este caso los jugadores con las siguientes características: posición #0: {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8} posición #1: {name = "Charles Barkley" age = 31 team = "Miami Heat"

				<p> points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7} </p> <p> posición #2: {name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 18 bounces = 15 assists = 40 steals = 2 blocks = 1} </p>
Fiba	searchPlayersLinearly	setupScenary2	value = "12" comparison = "LESS"	<p>Lista de jugadores con 3 elementos, en este caso los jugadores con las siguientes características:</p> <p> posición #0: {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8} </p> <p> posición #1: {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5} </p> <p> posición #2: {name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 20 bounces = 4 assists = 18 steals = 13 blocks = 9} </p>
Fiba	searchPla	setupScenary2	value = "6"	Lista de jugadores con 4 elementos, en

	yersLin aerly		comparison = "EQUALGREATER"	<p>este caso los jugadores con las siguientes características:</p> <p>posición #0: {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5}</p> <p>posición #1: {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8}</p> <p>posición #2: {name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7}</p> <p>posición #3: {name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 18 bounces = 15 assists = 40 steals = 2 blocks = 1}</p>
Fiba	searchPla yersLin aerly	setupScenary2	value = "12" comparison = "EQUALLESS"	<p>Lista de jugadores con 4 elementos, en este caso los jugadores con las siguientes características:</p> <p>posición #0: {name = "Charles Barkley" age = 31</p>

				<p>team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7}</p> <p>posición #1: {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8}</p> <p>posición #2: {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5}</p> <p>posición #3: {name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 20 bounces = 4 assists = 18 steals = 13 blocks = 9}</p>
Fiba	searchPlayersLinearly	setupScenary2	value = "-12" comparison = "EQUALLESS"	Lista de jugadores vacía, ya que el valor ingresado para realizar la búsqueda es negativo.
Fiba	searchPlayersLinearly	setupScenary2	value = "doce" comparison = "EQUALLESS"	Lista de jugadores vacía, ya que el valor ingresado para realizar la búsqueda no es un número.

Objetivo de la Prueba: Validar que se buscan y retornan correctamente los jugadores almacenados en un ABB de acuerdo con el valor y criterio dado.

Clase	Método	Escenario	Valores de Entrada	Resultado
Fiba	searchPlayersABB	setupScenary1	value = "12" comparison = "POINTSLESS"	Lista de jugadores vacía, ya que esta estructura no contiene ningún jugador y solo ha sido inicializada.
Fiba	searchPlayersABB	setupScenary2	value = "9" comparison = "POINTSEQUAL"	<p>Lista de jugadores con 2 elementos, en este caso los jugadores con las siguientes características:</p> <p>posición #0: {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5}</p> <p>posición #1: {name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7}</p>
Fiba	searchPlayersABB	setupScenary2	value = "17" comparison = "POINTSGREATER"	<p>Lista de jugadores con 2 elementos, en este caso los jugadores con las siguientes características:</p> <p>posición #1: {name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 20 bounces = 4 assists = 18 steals = 13 blocks = 9}</p> <p>posición #0: {name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 18 bounces = 15 assists = 40 steals = 2</p>

				blocks = 1}
Fiba	searchPlayersABB	setupScenary2	value = "17" comparison = "POINTSLESS"	<p>Lista de jugadores con 2 elementos, en este caso los jugadores con las siguientes características:</p> <p>posición #0: {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5}</p> <p>posición #1: {name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7}</p>
Fiba	searchPlayersABB	setupScenary2	value = "17" comparison = "POINTSEQUALGREATER"	<p>Lista de jugadores con 3 elementos, en este caso los jugadores con las siguientes características:</p> <p>posición #0: {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8}</p> <p>posición #2: {name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 20 bounces = 4 assists = 18 steals = 13 blocks = 9}</p> <p>posición #1: {name = "Dave Bing"</p>

				age = 27 team = "Brooklyn Nets" points = 18 bounces = 15 assists = 40 steals = 2 blocks = 1}
Fiba	searchPlayersABB	setupScenary2	value = "17" comparison = "POINTSEQUALLESS"	Lista de jugadores con 3 elementos, en este caso los jugadores con las siguientes características: posición #2: {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8} posición #0: {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5} posición #1: {name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7}
Fiba	searchPlayersABB	setupScenary2	value = "-17" comparison = "POINTSEQUALLESS"	Lista de jugadores vacía, ya que el valor ingresado para realizar la búsqueda es negativo.
Fiba	searchPlayersABB	setupScenary2	value = "diecisiete" comparison = "POINTSEQUALLESS"	Lista de jugadores vacía, ya que el valor ingresado para realizar la búsqueda no es un número.

			SS”	
--	--	--	-----	--

Objetivo de la Prueba: Validar que se buscan y retornan correctamente los jugadores almacenados en un RedBlackTree (por robos) de acuerdo con el valor y criterio dado.				
Clase	Método	Escenario	Valores de Entrada	Resultado
Fiba	searchPlayersRedBlackTree	setupScenary1	value = “10” comparison = “STEALSLESS”	Lista de jugadores vacía, ya que esta estructura no contiene ningún jugador y solo ha sido inicializada.
Fiba	searchPlayersRedBlackTree	setupScenary2	value = “3” comparison = “STEALSEQUAL”	Lista de jugadores con 1 elemento, en este caso el jugador con las siguientes características: {name = “Charles Barkley” age = 31 team = “Miami Heat” points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7}
Fiba	searchPlayersRedBlackTree	setupScenary2	value = “3” comparison = “STEALSGREATER”	Lista de jugadores con 3 elementos, en este caso los jugadores con las siguientes características: {name = “Ray Allen” age = 30 team = “Conferencia Este” points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5} {name = “Paul Arizin” age = 28 team = “Toronto Raptors” points = 20 bounces = 4 assists = 18 steals = 13 blocks = 9}

				<pre> {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8} </pre>
Fiba	searchPlayersRedBlackTree	setupScenary2	<pre> value = "10" comparison = "STEALSLESS" </pre>	<p>Lista de jugadores con 3 elementos, en este caso los jugadores con las siguientes características:</p> <pre> {name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 18 bounces = 15 assists = 40 steals = 2 blocks = 1} {name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7} {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8} </pre>
Fiba	searchPlayersRedBlackTree	setupScenary2	<pre> value = "3" comparison = "STEALSEQUALGREATER" </pre>	<p>Lista de jugadores con 4 elementos, en este caso los jugadores con las siguientes características:</p>

				<pre> {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5} {name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 20 bounces = 4 assists = 18 steals = 13 blocks = 9} {name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7} {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8} </pre>
Fiba	searchPlayersRedBlackTree	setupScenary2	value = "10" comparison = "STEALSEQUALLESS"	<p>Lista de jugadores con 4 elementos, en este caso los jugadores con las siguientes características:</p> <pre> {name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 18 bounces = 15 assists = 40 steals = 2 </pre>

				<pre> blocks = 1} {name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7} {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8} {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5} </pre>
--	--	--	--	---

Objetivo de la Prueba: Validar que se buscan y retornan correctamente los jugadores almacenados en un AVLtree (por puntos, asistencias y bloqueos) de acuerdo con el valor y criterio dado.				
Clase	Método	Escenario	Valores de Entrada	Resultado
Fiba	searchPlayersAVLTree	setupScenary1	value = "13" comparison = "POINTSLESS"	Lista de jugadores vacía, ya que esta estructura no contiene ningún jugador y solo ha sido inicializada.
Fiba	searchPlayersAVLTree	setupScenary2	value = "20" comparison = "POINTSEQUAL"	Lista de jugadores con 1 elemento, en este caso el jugador con las siguientes características: <pre> {name = "Paul Arizin" age = 28 team = "Miami Heat" points = 20 </pre>

				bounces = 4 assists = 18 steals = 13 blocks = 9}
Fiba	searchPlayersAVLT ree	setupScenary2	value = "5" comparison = "ASSISTS GREATER"	<p>Lista de jugadores con 3 elementos, en este caso los jugadores con las siguientes características:</p> <p>{name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 20 bounces = 4 assists = 18 steals = 13 blocks = 9}</p> <p>{name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7}</p> <p>{name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 18 bounces = 15 assists = 40 steals = 2 blocks = 1}</p>
Fiba	searchPlayersAVLT ree	setupScenary2	value = "18" comparison = "ASSISTS LESS"	<p>Lista de jugadores con 2 elementos, en este caso los jugadores con las siguientes características:</p> <p>{name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5}</p>

				<pre> {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8} </pre>
Fiba	searchPlayersAVLTree	setupScenary2	<pre> value = "5" comparison = "BLOCKSEQUALGREATER" </pre>	<p>Lista de jugadores con 4 elementos, en este caso los jugadores con las siguientes características:</p> <pre> {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8} {name = "Paul Arizin" age = 28 team = "Toronto Raptors" points = 20 bounces = 4 assists = 18 steals = 13 blocks = 9} {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5} {name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 </pre>

				assists = 18 steals = 3 blocks = 7}
Fiba	searchPlayersAVLT ree	setupScenary2	value = "10" comparison = "BLOCKSEQUALLESS"	Lista de jugadores con 4 elementos, en este caso los jugadores con las siguientes características: {name = "Dave Bing" age = 27 team = "Brooklyn Nets" points = 18 bounces = 15 assists = 40 steals = 2 blocks = 1} {name = "Ray Allen" age = 30 team = "Conferencia Este" points = 9 bounces = 6 assists = 4 steals = 10 blocks = 5} {name = "Charles Barkley" age = 31 team = "Miami Heat" points = 9 bounces = 12 assists = 18 steals = 3 blocks = 7} {name = "Devin Booker" age = 24 team = "Phoenix Suns" points = 17 bounces = 10 assists = 5 steals = 6 blocks = 8}

Diagrama de clases

