

Especificación de requerimientos funcionales:

El programa debe estar en capacidad de leer una base de datos de muchos jugadores de baloncesto, para poder realizar consultas eficientes sobre las estadísticas de los jugadores. El programa debe permitir realizar la consulta de cualquier atributo de los jugadores (puntos, rebotes, asistencias, robos o bloqueos por partido) que satisfaga el criterio (igualdad o desigualdad).

Requerimientos no funcionales:

La base de datos se debe leer de memoria secundaria, ya que pueden ser demasiados datos. Para garantizar rapidez de consultas, se deben utilizar estructuras de datos y algoritmos de ordenamiento eficientes. En este caso, los arboles binarios balanceados, permiten eficiencia $O(\log n)$ para el agregar y buscar.

DISEÑO DEL CASO DE PRUEBA

Etapas:

MODEL:

Nombre	Clase	Escenario
setUpStage1	FIBA	Vacío
setUpStage2	FIBA	Clase controladora sin ningún player

COLLECTIONS:

Nombre	Clase	Escenario
setUpStage1	BST	Vacío
setUpStage2	BST	Árbol binario de búsqueda vacío
setUpStage3	AVL	Vacío
setUpStage4	AVL	Árbol AVL vacío

DISEÑO DE CASOS DE PRUEBA DE MODEL

Objetivo de la prueba: Agregar jugadores de forma remota.				
Clase	Método	Escenario	Entrada	Salida
FIBA	addPlayerToAllTrees()	setupStage1	Ninguno	No se agrega nada.
FIBA	addPlayerToAllTrees ()	setupStage2	Player = ("Sam", 20, "USS",1,2,3,4,5)	Agrega a un jugador.

Objetivo de la prueba: Buscar los valores menores a un valor en específico				
Clase	Método	Escenario	Entrada	Salida
FIBA	searchLess()	setupStage1	Tree = 0 Condition = 0	No hay ningún valor
FIBA	searchLess()	setupStage2	Tree = 1 Condition = 2	Encuentra los valores menores e iguales al valor buscado.

Objetivo de la prueba: Buscar los valores mayores o iguales a un valor en específico.				
Clase	Método	Escenario	Entrada	Salida
FIBA	searchMore()	setupStage1	Tree = 0 Condition = 0	No hay ningún valor
FIBA	searchMore()	setupStage2	Tree = 1 Condition = 2	Encuentra los valores mayores e iguales al valor buscado

Objetivo de la prueba: Buscar los valores iguales al valor especificado.				
Clase	Método	Escenario	Entrada	Salida
FIBA	searchEquals	setupStage1	Tree = 0 Condition = 0	No encuentra el valor.
FIBA	searchEquals()	setupStage2	Tree = 1 Condition = 0	Encuentra los valores iguales al valor especificado.

DISEÑO DE CASOS DE PRUEBA DE COLLECTIONS

Objetivo de la prueba: Agregar un valor al árbol binario de búsqueda.				
Clase	Método	Escenario	Entrada	Salida
BST	add()	setupStage 1	Ninguno	No inserta ningún valor

Objetivo de la prueba: Agregar un valor al árbol binario de búsqueda.				
Clase	Método	Escenario	Entrada	Salida
BST	add ()	setupStage 2	Value = 3	Agrega el valor asignado a la raíz

Objetivo de la prueba: Buscar un valor especificado en el árbol binario de búsqueda				
Clase	Método	Escenario	Entrada	Salida
BST	search()	setupStage 1	Value = 3	No encuentra el valor

Objetivo de la prueba: Buscar un valor especificado en el árbol binario de búsqueda				
Clase	Método	Escenario	Entrada	Salida
BST	search()	setupStage 2	Value = 3	Encuentra el valor solicitado

Objetivo de la prueba: Obtener los valores menores o iguales a un valor especificado

Clase	Método	Escenario	Entrada	Salida
BST	getLessThan()	setupStage 2	T = 10	Retorna una lista con los valores menores al valor solicitado

Objetivo de la prueba: Obtener los valores mayores al valor especificado				
Clase	Método	Escenario	Entrada	Salida
BST	GetGreaterThanOr()	setupStage 2	T =10	Obtiene los valores mayores al valor especificado

--	--	--	--	--

Objetivo de la prueba: Obtener los valores mayores al valor especificado				
Clase	Método	Escenario	Entrada	Salida
BST	GetGreaterThan()	setupStage 2	T = 10	Obtiene los valores mayores al valor especificado

Objetivo de la prueba: Recorrer en inorden la derecha de un nodo				
Clase	Método	Escenario	Entrada	Salida
BST	inOrderRight()	setupStage 2	Node = root	Una lista de los nodos a la derecha de la raíz en inorden

Objetivo de la prueba: Recorrer el lado izquierdo del nodo en inorden				
Clase	Método	Escenario	Entrada	Salida
BST	inOrderLeft()	setupStage 2	Node = root	Una lista de los nodos a la izquierda de la raíz en inorden

Objetivo de la prueba: Obtener la altura del árbol o subárbol correspondiente al nodo				
Clase	Método	Escenario	Entrada	Salida
BST	getHeight()	setupStage 2	Node = tree	Obtiene la altura del árbol principal

Objetivo de la prueba: Obtener el factor de balanceo correspondiente al nodo dado				
Clase	Método	Escenario	Entrada	Salida
AVL	balanceFactor()	setupStage 2	Node = tree	Obtiene el factor de balanceo de todo el arbol

Objetivo de la prueba: Rotar hacia la izquierda el nodo elegido				
Clase	Método	Escenario	Entrada	Salida
BST	leftRotate()	setupStage 2	Node = hijo izquierdo del hijo izquierdo de la raiz	Se aplica correctamente la rotación izquierda

Objetivo de la prueba: Rotar hacia la derecha el nodo elegido				
Clase	Método	Escenario	Entrada	Salida
BST	rightRotate()	setupStage 2	Node = hijo izquierdo del hijo izquierdo de la raiz	Se rota correctamente el nodo hacia la derecha

Objetivo de la prueba: Obtener el factor de balanceo de un nodo

Clase	Método	Escenario	Entrada	Salida
BST	balanceFactor()	setupStage 2	Node = hijo izquierdo del hijo izquierdo de la raiz	Obtiene el factor de balanceo correspondiente al subArbol del Hijo izquiero del hijo izquiero de la raiz