INFORME

1. Contexto Problemático

Una compañía fabricadora de Software, ha solicitado la implementación de una herramienta para el manejo de información de las estadísticas de jugadores de baloncestos, para lo cual se ha generado un banco de datos lo suficientemente representativo para la realización del problema. las estadísticas que se desean manejar son: puntos,rebotes,asistencias,bloqueos,robos,porcentaje de éxito en cada tipo de tiro, etc. Se espera que la herramienta permite navegar por estas estadísticas de una manera rápida y clara.

2. Desarrollo de la solución

Para resolver el caso anterior se eligió el Método de la Ingeniería del libro "Introduction to Engineering" de Paul Wright. Cada uno de los pasos del Método de la Ingeniería se presentan a continuación especificando la descripción y el procedimiento de cada uno de estos pasos.

2.1 Identificación Del Problema

De acuerdo al enunciado se pudieron detectar los siguientes problemas:

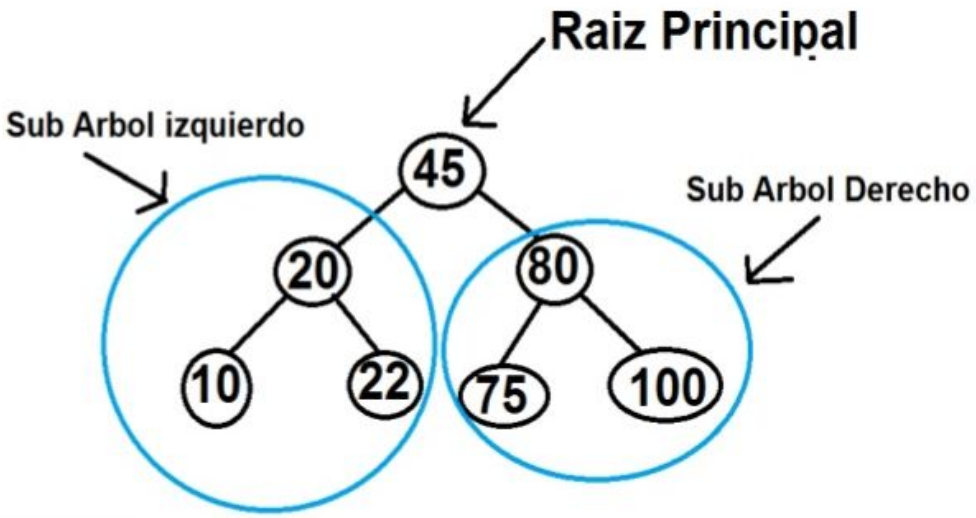
1. Permitir el ingreso de los datos a nuestro programa, ya sea en archivos grandes tipo csv o mediante la interfaz por medio del usuario.
2. Las estadísticas deben de poder eliminarse, modificarse o agregarse según sea el caso que se requiera.
3. Se debe hacer uso de estructuras de datos tipo árbol, para la navegación (consulta) de las estadísticas que se requieran.

Los anteriores tres puntos se pueden resumir en simplemente el problema de manipular los datos estadísticos que se tendrán de la base de datos de los jugadores.

2.2 Recopilación De La Información

Para una adecuada solución del problema propuesto, se requiere la recopilación de información, que nos ayude a decidir cual es la manera más eficiente de llegar a la solución del problema, generando diferentes ideas, y finalmente tomar el camino más conveniente. Para eso debemos esclarecer todo el funcionamiento de los árboles rojinegros, AVL y los binarios de búsqueda.

Árboles binarios de búsqueda



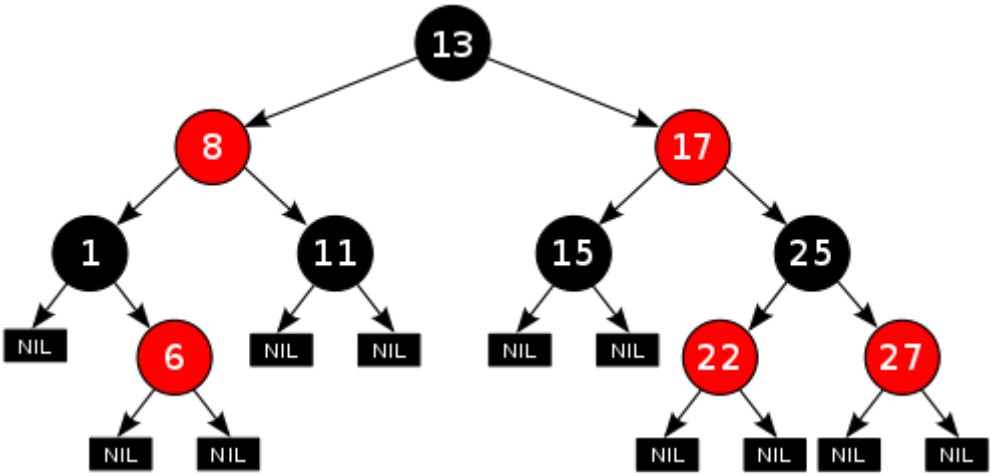
Un árbol binario de búsqueda es un árbol binario en el cual se cumple que para cada nodo x:

- Los nodos del subárbol izquierdo son menores o iguales a x.
- Los nodos del subárbol derecho son mayores o iguales a x.

Las consultas para un árbol de búsqueda binaria son:

- Búsqueda de una llave
- Mínimo
- Máximo
- Sucesor de un nodo
- Predecesor de un nodo

Árboles Rojinegros



Un árbol rojinegro es un árbol de búsqueda binario en el que cada nodo tiene un bit extra para almacenar su color. Algunas de las características de los árboles rojinegros son las siguientes:

- Ciertas condiciones sobre los colores de los nodos garantizan que la profundidad de ninguna hoja sea más del doble que la de ninguna otra.
- El árbol está algo equilibrado
- Cada nodo posee un color (rojo o negro), una clave, que equivale a la clave de búsqueda, dos punteros a los hijos (izquierdo y derecho) y finalmente un puntero al padre. También se debe de tener en cuenta que si los punteros son nil, se deben considerar como hojas y se colocarán de color negro.

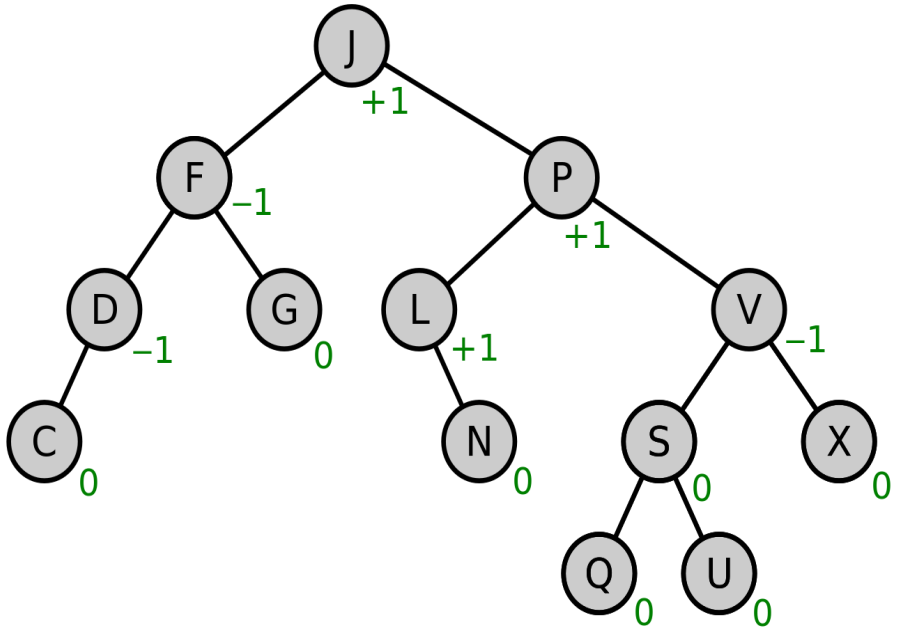
Los árboles rojinegros también deben de cumplir con las siguientes propiedades:

- Todo nodo es negro o rojo.
- La raíz es negra.
- Toda hoja es negra.
- Si un nodo es rojo, entonces sus hijos son negros.
- Cada camino a un nodo a sus hijos descendientes contienen el mismo número de nodos negros.

Las siguientes operaciones se pueden implementar en los árboles rojinegros:

- Búsqueda
- Mínimo
- Máximo
- Sucesor
- Predecesor
- Inserción
- Eliminación
- Rotación a la izquierda
- Rotación a la derecha

Árboles AVL



Un árbol AVL es un árbol binario de búsqueda balanceado en altura, su nombre proviene de las iniciales de sus creadores, Adelson- Velskii y Landis y es conocido como el primer ABB balanceado publicado en la historia.

Definición del factor de balanceo

El factor de balanceo de un nodo v de árbol binario se define como la altura del subárbol derecho del nodo v - la altura del subárbol izquierdo del nodo v.

Se debe de tener en cuenta que cada nodo de un árbol AVL debe tener un factor de balanceo. Y el árbol se encontrará balanceado en altura su cada nodo perteneciente a dicho árbol tiene un factor de balanceo de -1, 0 ó 1.

Las propiedades que debe cumplir un ABB para ser considerado un árbol AVL son las siguientes:

- Los subárboles de cada nodo difieren en altura 1 como máximo.
- Cada subárbol es un árbol AVL.

Las operaciones que se pueden realizar en un árbol AVL son:

- Inserción
- Eliminación

Para realizar las operaciones de inserción y eliminación, se trabajan como un

árbol ABB y seguidamente rebalancear aquellos nodos que se encuentran desbalanceados, para lograr balancear los nodos que se encuentran desbalanceados se hace uso de las rotaciones en árboles AVL, con el objetivo de arreglar los factores de balanceo. Las rotaciones que se pueden realizar son:

- Rotación a la izquierda
- Rotación a la derecha

Estas rotaciones se utilizarán de acuerdo a los distintos escenarios planteados, en los que tenemos un subárbol cuya raíz está desbalanceada en una unidad, esto quiere decir, que posea un factor de balanceo de 2 ó -2.

Casos de balanceo

Se van a tener exactamente 6 casos, si el factor de balanceo del nodo desbalanceado es 2, se van a utilizar los casos A, B ó C, si por el contrario, el factor de balanceo es -2, se van a utilizar los casos D, E ó F

CASO A: Este caso se cumple cuando el factor de balanceo del hijo derecho del nodo desbalanceado es igual a 1.Consiste en rotar hacia la izquierda, sobre el nodo que se encuentra desbalanceado y calcular nuevamente el factor de balanceo de cada nodo.

CASO B: Este caso se cumple cuando el factor de balanceo del hijo derecho del nodo desbalanceado es igual a 0. Al igual que el caso A, consiste en rotar hacia la izquierda, sobre el nodo que se encuentra desbalanceado y reestablecer el balance.

CASO C: Este caso se cumple cuando el factor de balanceo del hijo derecho del nodo desbalanceado es igual a -1. Para este caso, se debe de rotar a la derecha sobre dicho nodo que tiene factor de balanceo -1 y seguidamente se rota hacia la izquierda sobre el nodo desbalanceado y se reestablece el balance.

CASO D: Este caso se cumple cuando el factor de balanceo del hijo izquierdo del nodo desbalanceado es igual a -1. Para este caso, se aplica la rotación hacia la derecha sobre dicho nodo que se encuentra desbalanceado y se reestablece el balance.

CASO E: Este caso se cumple cuando el factor de balanceo del hijo izquierdo del nodo desbalanceado es igual a 0. Para realizar este caso, se hace lo mismo que se hace en el CASO D.

CASO F: Este caso se ejecuta cuando el factor de balanceo del hijo izquierdo del nodo desbalanceado es igual a 1. En este caso se debe realizar dos rotaciones, primero se rota hacia la izquierda sobre dicho nodo mencionando anteriormente, el cual tiene un factor de balanceo igual a 1, y finalmente se rota hacia la derecha sobre el nodo que se encuentra desbalanceado.

Tomado de:

- https://labs.xjtudlc.com/labs/wldmt/reading%20list/books/Algorithms%20and%20optimization/Introduction%20to%20Algorithms.pdf
- http://webdiis.unizar.es/asignaturas/TAP/material/1.3.rojinegros.pdf
- http://interactivepython.org/runestone/static/pythoned/Trees/ImplementacionDeUnArbolAVL.html

2.3 Búsqueda Soluciones Creativas

1. Para la solución del problema, podremos crear diferentes Arreglos en donde almacenaremos los datos por categorías, es decir, un arreglo para asistencias, un arreglo para rebotes, etc.
2. Para la solución del problema, podremos crear diferentes listas (ArrayList en lenguaje Java) en donde almacenaremos los datos por categorías, es decir, una lista para asistencias, una lista para rebotes, etc.
3. Los árboles binarios de búsqueda son estructuras eficaces que nos ayudan a acceder a los datos que tenemos almacenados, por lo tanto una estrategia sería hacer uso de estos tipos de árboles para encontrar los datos según los cuatro criterios pedidos.
4. Las estructuras estudiadas de tipo árboles, son avl, rojos-negros y abb, por lo cual, se plantea que el problema sea resuelto haciendo uso DE LAS TRES estructuras, según se requiera.

2.4 Transición De Las Ideas A Los Diseños Preliminares

- **Alternativa 1:** Al hacer uso de arreglos, tendremos un problema y es el tamaño, pues al agregar, modificar o eliminar un elemento no haremos uso de un tamaño fijo. lo cual nos podría traer dificultades al momento de manejar los datos.
- **Alternativa 2:** Con las ArrayList el problema del tamaño se soluciona, pues esta estructura se actualiza su tamaño a medida que se agregan o eliminan, lo cual podría considerarse como una alternativa viable. Su desventaja ocurre al momento de comparar con los árboles, pues en este caso es más rápido (eficaz) trabajar con árboles, ya que el acceder a un dato dentro de él es mucho más rápido que buscar un dato en ArrayList.
- **Alternativa 3 y 4**: Durante nuestra preparación en el curso, hemos aprendido a usar los tres diferentes tipos de datos, arboles abb, árboles avl, y árboles rojos-negros. y dado que tenemos cierto desempeño en estas estructuras es mas preferible usar las tres estructuras, en vez de usar solo una, ya que entendemos que para diferentes casos será mejor usar una estructura que otra. Por este motivo decidimos que usaremos la ALTERNATIVA 4.

2.5 Evaluación Y Selección De La Mejor Solución

De acuerdo a las observaciones que le hicimos a cada alternativa encontramos al final la aternativa que cumplira con el objetivo de manera óptima.

- Hacer uso de los tres tipos de árboles para la solución del problema que nos proponen.

2.6 Preparación De Informes Y Especificaciones

Especificación del Problema (en términos de entrada/salida)

- Problema: teniendo en cuenta la base de datos que tendremos en nuestro programa, se debe manipular los datos que estarán almacenados: acceder, modificar y eliminar los datos.
- Entradas: los datos que se desean consultar, eliminar o modificar.
- Salida: la base de datos actualizada según los datos

para la solución del problema se hizo uso de árboles avl para :

- acceder a los datos USG
- acceder a los datos AST
- el resto de las estadísticas faltantes

para la solución del problema se hizo uso de árboles roji-negros para:

- acceder a los datos Defense
- accedera los datos Offense

para la solución del problema se hizo uso de árboles binarios de búsqueda para:

- acceder a los datos deaño
- accedera los datos de equipo

2.7 Implementación Del Diseño

- Los requerimientos funcionales serán entregados en un archivo adicional.
- El código del programa será entregado en lenguaje Java, implementado en Java eclipse.