Universidad Central "Martha Abreu" de Las Villas
Cuba

Facultad de Matemática, Física y Computación

Análisis de Sistemas Interactivos para la enseñanza del TDA Árboles Binarios de Búsqueda

Tesis presentada en opción al Título Académico de

Master en Computación Aplicada

Autor: Ing. Rafael de la Cruz González
Tutor: Dr. Ramiro A. Pérez Vázquez.

Guadalajara, Jalisco. México
2003

Tabla de contenido

Resumen

Con este proyecto se pretende realizar un análisis crítico y extensivo de los principales sistemas interactivos disponibles en Internet, utilizados para representar de manera gráfica el TDA Árboles Binarios de Búsqueda, y cuyo código fuente esté disponible, de forma que permita diseñar una aplicación capaz de mostrar animaciones con explicaciones claras sobre los pasos que se están llevando a cabo cuando se realiza cualquier tipo de operaciones, todo esto con un enfoque pedagógico, gracias a el cual se garantiza el aprendizaje.

En Internet existen una serie de sitios con animaciones del TDA ABB, para árboles AVL, Rojo/Negro, y Biselado, pero la mayoría no explica el proceso que se muestra de manera gráfica y en ocasiones son tan rápidas que no se percibe el concepto del algoritmo que se está utilizando. Las opciones para los estudiantes se ven reducidas además por la barrera del idioma, y aunque en este tipo de carreras se debe tener la habilidad al menos de leer inglés, no todos han adquirido el conocimiento en esta etapa de la carrera.

Desde el punto de vista de programación se pretende acotar dos problemas:
La programación de los algoritmos para el TDA, sin perder de vista que muchos de estos algoritmos están ya definidos, pero intentando buscar variantes suficientemente claras para que sean entendidos con mayor facilidad por los estudiantes.
La representación gráfica de árboles, tomando como referencia la teoría existente para la representación de grafos.

Abstract

The main objective of this project is to present the outcome of an extensive and critical analysis of some of the main interactive applications posted on the Internet, used to demonstrate graphic animations of the ADT Binary Search Trees, and whose source code is available for perusal, in order to design an application able to demonstrate graphic animations, along with clear and brief explanations about the steps needed to perform operations, all this from a pedagogic perspective, thanks to which the learning will be guaranteed.

In Internet there is a small number of web sites with graphic animations of BST's, whether it is AVL, Red-Black or Splay, but most of these animations do not explain the process shown, and sometimes are so fast that there is not enough time to assimilate the core concept of the algorithm being used. Students' choices are reduced for the barrier of the language, even though they must at least have the ability to read in English, not all of them have acquired the knowledge at this early stages of the career.

Through the paradigm of programming, this project intends to address two issues:
The programming of algorithms to implement the ADT, knowing that even though many of these algorithms have already been implemented, new variations might be searched for students to understand them in more clear ways.
The graphical representation of trees, taking as a reference the existing theory for graphs representation.

Introducción

El principal problema que se quiere abordar con este proyecto es el de la falta de material y/o recursos adecuados que permitan a los estudiantes que estén realizando cursos de Estructuras de Datos, aprender mediante la utilización de herramientas o material didáctico apropiado.

Es común y bien sabido que debido a los vertiginosos y constantes cambios en las tecnologías y tendencias en el área de cómputo, tanto en el sector laboral como en el académico, el profesionista o estudioso de estas materias debe encontrar caminos o medios alternativos que le permitan mantener un nivel de actualización adecuado.

Una práctica bastante común, debido a la gran cantidad de información en el área de cómputo, es el auto-estudio. Mucha gente, tanto en el ámbito laboral, como en las universidades, dedica gran parte de su tiempo a aprender “por su cuenta”. Afortunadamente para muchos de nosotros, existen tanto en el mercado como en Internet, una gran cantidad de recursos que permiten y facilitan este tipo de tareas o prácticas individuales. De tal manera que se ha vuelto habitual comprar un manual o guía de cierto programa o lenguaje de desarrollo nuevo; o bien, buscar en Internet manuales o información de “cómo hacer” algo que no sabemos hacer con los programas o con el conocimiento limitado que tenemos.

En el caso de los estudiantes, en quienes apenas se está fomentando esta conducta de auto-estudio y autoaprendizaje, la existencia de este tipo de recursos no es la excepción. Pero dada la complejidad de los Tipos de Datos Abstractos (TDA), resulta imprescindible contar con una herramienta que realmente ayude a mejorar el proceso de aprendizaje, mediante el uso no sólo de representaciones gráficas, sino mediante un proceso que dé al estudiante, además de la idea conceptual, la información teórica que le permitirá junto con la animación, lograr entender y aprender de una manera más clara, qué son y cómo es que funcionan las diferentes opciones para implementar TDA Árboles Binarios de Búsqueda (ABB).

Tomando en cuenta la gran cantidad de información existente, es realmente limitado el número de aplicaciones o animaciones disponibles en Internet que estén enfocadas particularmente a los ABB. Además de este reducido número de aplicaciones, muy pocas de ellas están enfocadas al proceso de aprendizaje, pareciera como si hubieran sido desarrolladas con el propósito único de demostrar la viabilidad de los algoritmos, o de hacer la representación gráfica sólo porque sí, por el gusto mismo.

Un problema de menor importancia que quedaría también resuelto es el del idioma. De los sitios de Internet que fueron visitados, existen algunos de instituciones educativas en español, pero en

estos sitios sólo se encuentra información relativa a los cursos que ofrecen, y referencias bibliográficas, y ninguno de los sitios revisados cuenta con alguna animación, de hecho las animaciones o aplicaciones encontradas están hechas todas en inglés.

Existen en Internet una gran cantidad de sitios con información referente a estas animaciones o demostraciones gráficas de los algoritmos, pero es sorprendente darse cuenta que son muchos los sitios o páginas que tienen estas ligas, pero todas estas ligas se dirigen hacia las mismas direcciones.

Una vez expuesta la situación actual con respecto a los sistemas mencionados, resulta fácil decir que el objetivo de este proyecto es:

> *Compilar y analizar la información referente a los sistemas existentes, y una vez teniendo esta información, diseñar un sistema en el que se mencionen las ventajas o áreas de mejora de tales aplicaciones, abarcando no sólo el punto de vista de la funcionalidad y facilidad de uso de la aplicación, sino analizando ventajas y mejores prácticas en el desarrollo de la aplicación, que permitirán a los estudiantes saber un poco más acerca de los esfuerzos que se han realizado en esta materia y cómo se han hecho. Básica y principalmente se pretende ayudar a los estudiantes y maestros con información que resulte de utilidad en el proceso de enseñanza del TDA ABB.*

Como objetivos específicos se pueden mencionar:

- Concientizar al lector de la gran ventaja que implica el uso de representaciones gráficas en el proceso de enseñanza en general, y en particular en el caso de estructuras de datos, ya que este tema al ser introducido de manera inicial, resulta un concepto abstracto y en ocasiones difícil de asimilar, que en definitiva podría ser explicado y entendido de mejor manera mediante representaciones gráficas, no sólo para la comprensión de la estructura misma, sino para la demostración de los algoritmos.
- Obtener como resultado final, el análisis de un sistema, que tenga las ventajas de las aplicaciones existentes, pero que al mismo tiempo no caiga en errores que han cometido otros programadores. Dentro del análisis de dicho sistema se pretende lograr que además de contar con una representación gráfica de calidad, guíe al alumno o usuario a través de un tutorial, que le permitan aprender acerca de las operaciones que se estén realizando.
  El sistema en cuestión deberá tener las siguientes características:

- Interactividad, es decir, no debe ser sólo una demostración explicada, sino debe permitir al estudiante o usuario introducir valores y determinar la operación a realizar, considerando además los métodos que apliquen para cada TDA.
- Reutilización de las variantes de los algoritmos existentes, de tal manera que resulten en un producto claro y fácil de entender para los estudiantes.
- Superar la carencia que existe de sistemas de hagan estas demostraciones, pero que estén en español y que expliquen de manera clara y precisa el desarrollo de los algoritmos.

Capítulo I

La abstracción y los tipos de datos abstractos

La abstracción es una herramienta de uso común entre los diseñadores involucrados en gestionar problemas complejos dentro del proceso del diseño de nuevos sistemas. Dentro de este contexto, la abstracción se refiere a la capacidad intelectual de considerar entidades individuales, aislándolas de ejemplares específicos. Este tipo de abstracción es mejor conocida como abstracción de datos, e involucra una descripción abstracta o lógica tanto de los datos como de las operaciones que pueden ser realizadas con esos datos. El usar esta abstracción permite al diseñador concentrarse únicamente en la resolución del problema utilizando estos datos abstractos, sin pensar ni en la presentación o el trato de estos en la memoria de la computadora.

Los tipos de datos abstractos son conocidos como TDA, por sus siglas en español, (en algunos textos se hace referencia a éstos como TAD o tipos abstractos de datos), aunque en algunos textos traducidos en inglés, se respetan las siglas originales y son denominados ADT.

Algunos autores definen el TDA como un modelo matemático de los objetos de datos que constituyen un tipo de datos, así como de las funciones que operan sobre estos objetos, sin olvidar que las operaciones que manipulan los objetos están incluidas en la especificación del TDA.

Podemos pensar en el TDA como un modelo matemático con una serie de operaciones definidas dentro del modelo mismo. Por ejemplo, los conjuntos de números enteros con las operaciones de unión, intersección y diferencia. Las operaciones de un TDA pueden tener como operandos no sólo los casos del TDA que se define, sino también otros tipos de operandos como enteros o casos de otro TDA. Pero se espera que al menos un operando, o el resultado de alguna operación pertenezca al TDA en cuestión.
Los TDA son generalizaciones de los tipos de datos primitivos, como enteros o reales, al igual que los procedimientos son generalizaciones de operaciones primitivas, como sumas o restas. Los TDA encapsulan cierto tipo de datos en el sentido de que es posible localizar la definición del tipo y sus operaciones en una sección de un programa dado.
Para representar los modelos matemáticos de los TDA, se emplean estructuras de datos, que son conjuntos de variables, quizá de tipos distintos, que están conectadas entre sí de diversas formas. El componente básico de estas estructuras de datos es la celda, y a esta última se le puede representar como una caja capaz de almacenar un valor tomado de algún tipo de datos,

ya sea básico o compuesto. Las estructuras de datos se crean dando nombres a agregados de celdas y opcionalmente interpretando los valores de algunas celdas como representantes de conexiones entre celdas, tal como los apuntadores, por ejemplo.
Es importante además mencionar la diferencia que existe entre los tipos de datos y las estructuras de datos, el primer término se refiere a la implementación del modelo matemático especificado por un TDA, mientras que el segundo se refiere a una colección de variables en memoria relacionadas de alguna forma específica.

Especificación de los TDA

Al hablar de la especificación de los TDA se tiene que hacer referencia a dos partes, la primera es una descripción matemática de una colección de objetos, y la segunda es un conjunto de operaciones definidas en ciertos elementos de esta colección de objetos, en un sentido estricto, a esta segunda parte se le agrega un conjunto de axiomas que describen completamente el comportamiento de las operaciones del TDA.

Árboles binarios de búsqueda

Los árboles imponen una estructura jerárquica sobre una colección de objetos, como ejemplos comunes se pueden mencionar los árboles genealógicos y los organigramas, y entre otras cosas, los árboles son útiles para analizar circuitos eléctricos y para representar la estructura de fórmulas matemáticas.

La terminología fundamental del árbol dice que éste es una colección de elementos llamados nodos, uno de los cuales se distingue como raíz, junto con una relación de paternidad, que impone una estructura jerárquica sobre todos los nodos. Los nodos, como elementos de la lista, pueden ser de cualquier tipo. Normalmente se representan por medio de letras, cadenas de caracteres o círculos con un número en el interior.

Un árbol binario se define como un árbol vacío o como un árbol en el que los nodos no pueden tener un número cualquiera de hijos, pueden tener un hijo izquierdo, un hijo derecho, o un hijo izquierdo y un hijo derecho, a diferencia de los árboles ordenados orientados, en los árboles binarios los hijos se tienen que designar como izquierdos o derechos.

Un árbol binario puede ser relleno cuando todos sus nodos tienen dos hijos o bien son una hoja, sin permitir nodos con un hijo único. Puede ser completo cuando todas sus hojas tienen la misma profundidad.

Un árbol binario de búsqueda es un tipo de datos cuyos elementos están organizados como un árbol binario. Los nodos de este árbol están ordenados de tal forma que para cualquier nodo x del árbol, si y es un nodo cualquiera del subárbol izquierdo de x, entonces clave[y] < clave[x], y si y es un nodo cualquiera del subárbol derecho de x, entonces clave[x] < clave[y].1

La definición anterior cumple totalmente con las restricciones dadas a la manera de almacenar los datos en el árbol binario. Estas restricciones dicen que los elementos deben ser almacenados de tal forma que los valores de las claves en el subárbol izquierdo de la raíz sean menores que el valor de la clave de la raíz, y de igual manera, los valores de las claves de todos los nodos del subárbol derecho deben ser mayores que el valor de la clave de la raíz.

Propiedad de los árboles binarios de búsqueda. Esta propiedad se refiere al orden impuesto en los nodos de un árbol binario de búsqueda. Como consecuencia de esta propiedad, los recorridos inorden de los árboles binarios de búsqueda visitan siempre a los nodos en orden.

Operaciones de los TDA conjuntos dinámicos y los árboles binarios de búsqueda

Resulta importante en este momento mencionar el TDA conjuntos dinámicos. En este contexto los conjuntos sirven como base para muchos tipos de datos abstractos, y esto implica que muchas aplicaciones o programas tengan implícita la manipulación de conjuntos de elementos de información.

El concepto de los conjuntos utilizados como tipos de datos es diferente al concepto matemático de los conjuntos, en el que el conjunto es una estructura utilizada para agrupar elementos. De hecho el nombre esta dado porque los conjuntos no son inmutables, ya que constantemente se están incluyendo o excluyendo elementos.

Los conjuntos dinámicos tienen elementos, y cada uno de estos elementos tiene un campo identificador llamado clave, y existe una relación de orden total2 sobre esas claves. Asumiendo que en el conjunto dinámico no existen elementos con la misma clave, podemos decir que las operaciones generales para un conjunto dinámico específico S serían:

[1] Definición dada por Gregory L. Heilmeman, en su libro “Estructuras de datos, algoritmos y programación orientada a objetos. Publicada por McGraw-Hill en 1996.

[2] En la relación de orden total, para cualesquiera tres valores de la clave, clave[a_1], clave[a_m] y clave[a_n]
1. Se cumple exactamente una de las siguientes posibilidades clave[a_1]<clave[a_m], clave[a_1]=clave[a_m] o clave[a_1]>clave[a_m].
2. si clave[a_1]<clave[a_m] y clave[a_m]<clave[a_n], entonces clave[a_1]<clave[a_n].

1. Buscar(S,k). Devuelve el elemento con clave k en S, o el valor nulo.
2. Insertar(S,x). Incluye el elemento x a S. Devuelve el valor booleano verdadero en caso de realizar la operación de manera exitosa.
3. Eliminar(S,k). Excluye el elemento k en S. Devuelve el valor booleano verdadero en caso de realizar la operación de manera exitosa.
4. Mínimo(S). Devuelve el elemento de S con menor valor de clave o el valor nulo si S es vacío.
5. Máximo(S). Devuelve el elemento de S con mayor valor de clave o el valor nulo si S es vacío.
6. Predecesor(S,k). Devuelve el elemento de S que tiene el mayor valor de clave que es menor que k, o el valor nulo si tal elemento no existe.
7. Sucesor(S,k). Devuelve el elemento S que tiene el menor valor de clave que es mayor que k, o el valor nulo si tal elemento no existe.

Los árboles binarios de búsqueda pueden utilizarse para implementar todas las operaciones definidas para los conjuntos dinámicos en un tiempo O(h), donde h es la altura del árbol, en el peor de los casos la altura de h será igual a n, donde n es el número de nodos del árbol, pero en estructuras de datos más sofisticadas se pueden realizar rotaciones para garantizar que la altura de un árbol binario de búsqueda sea O(log n).

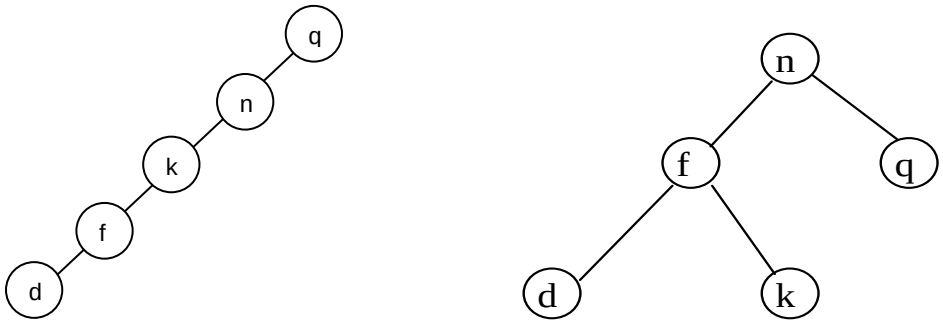

Figura 1 (a) **lo de árboles binarios de búsqueda cor** (b) **no número de claves o nodos p** **rente altura. 4 (a) y 2 (b).**

En la figura 1 se pueden observar ejemplos de árboles binarios de búsqueda con el mismo número de claves o elementos, pero con diferentes alturas. Esta situación de árboles con diferentes alturas es precisamente resuelta mediante la aplicación de las operaciones de rotación, que serán mencionadas posteriormente.

Operaciones de Consulta

Operación buscar

Con esta operación se pretende encontrar un nodo k. Primero se comprueba si tal valor existe en el nodo raíz, de no ser así, lo siguiente es determinar si el nodo podría estar en el subárbol derecho o en el izquierdo. Si el valor del nodo k es menor que el de raíz, la búsqueda se limita al subárbol izquierdo de raíz, de igual manera que si el valor k fuera mayor que el valor de raíz la búsqueda se limitaría al subárbol derecho. Al encontrarse un nodo cuyo valor sea k se devuelve ese nodo, o bien si el nodo no fue encontrado se deberá devolver el valor nulo.

Operaciones máximo y mínimo

Estas son probablemente las operaciones más fáciles de implementar en los árboles binarios de búsqueda, ya que por las mismas propiedades de éstos se garantiza que el nodo mínimo de cualquier árbol será el hijo extremo izquierdo del árbol, y de igual manera el nodo de mayor valor será el hijo extremo derecho. La implementación es idéntica para ambas operaciones, sólo que la operación mínimo busca en el subárbol izquierdo y la operación máximo en el derecho.

Sólo en caso de que el árbol binario de búsqueda sea vacío se devolverá el valor nulo.

Operaciones predecesor y sucesor

Esta operación también resulta ser de fácil implementación en caso de existir el nodo con la clave k. Por ejemplo, con la operación predecesor, una vez localizado el nodo con el valor k, se sabe que todos los nodos ubicados en el subárbol izquierdo de k contienen un valor menor, entonces sólo es cuestión de encontrar el nodo con el valor mayor en el subárbol izquierdo de k, mediante la utilización de la operación máximo previamente mencionada.

En caso de que el nodo k no tenga subárbol izquierdo, gracias a la propiedad de los árboles binarios de búsqueda se garantiza que sí existe un predecesor, y éste puede ser encontrado estableciendo el camino ascendente desde k hacia la raíz hasta que se encuentre un nodo cuyo subárbol izquierdo no contenga a k, éste sería entonces el predecesor de k.
El procedimiento para realizar la operación sucesor se desarrolla de manera casi simétrica.

Inserción y Eliminación

La técnica estándar de estas operaciones puede resultar en árboles binarios de búsqueda no equilibrados, es necesario considerar ciertos procedimientos que se mencionarán posteriormente.

Operación insertar

Esta operación consta de 3 pasos

1. Creación del objeto de datos compuesto conteniendo el dato y su clave.
2. Determinación del punto de inserción en el ABB, después de realizar la búsqueda, si la clave no está en el árbol, la búsqueda regresará un puntero nulo, en el cual deberá ser insertado el nuevo objeto.
3. Asignación de la dirección del nuevo objeto de datos a la variable que contiene el puntero nulo mencionado anteriormente.

Es importante mencionar que si la inserción se está haciendo sobre un árbol vacío, el nuevo nodo se convierte en la raíz del árbol.

Operación eliminar

Para esta operación el primer paso es una búsqueda, en caso de ser encontrado el nodo, se seguirán diferentes pasos para eliminarlo, de acuerdo al número de hijos que existan.

Nodo sin hijos. Este es el caso más sencillo, sólo se establece el puntero apropiado del nodo padre al valor nulo.

Nodo con un hijo. Se reasigna el valor del puntero del nodo padre, para que en lugar de apuntar al nodo que está siendo borrado, apunte al hijo.

Nodo con dos hijos. Se reemplaza el nodo que se quiere borrar con su predecesor en recorrido inorden, de esta manera se garantiza el mantenimiento de la propiedad de los ABB.

Árboles binarios de búsqueda aleatorios

Aunque en la práctica no sea así, el hecho de que el conjunto de datos de un ABB sea suministrado de manera aleatoria resulta importante, ya que hará más probable la construcción de un árbol “promedio” en el que los tiempos de búsqueda no son extremos.

Árboles binarios de búsqueda óptimos

En algunas aplicaciones es posible determinar la frecuencia con la que los datos individuales son accedidos, o cuando menos estimar las frecuencias observando la actuación de una muestra de entradas del dominio. Con esta información se puede construir un ABB que minimice el tiempo medio de búsqueda.

Tal estructura es el árbol binario de búsqueda óptima, y se define como aquel que minimiza el tiempo medio de búsqueda, dadas las frecuencias de acceso de cada uno de los datos. La estrategia consiste en situar los elementos más solicitados más frecuentemente cerca de la raíz del árbol, tomando en cuenta que los nodos deben obedecer el orden impuesto por la propiedad de los ABB.

Árboles de búsqueda equilibrados

Todos los árboles binarios de búsqueda equilibrados son casos especiales de los árboles binarios de búsqueda, ya que tienen la misma estructura y obedecen a la misma propiedad de búsqueda. Son considerados casos especiales debido a las restricciones adicionales que tienen y de manera general se puede decir que tienen como propósito restringir la altura de los árboles binarios resultantes. Se puede entonces decir que la imposición de tales restricciones conduce a la construcción de árboles equilibrados.

Una característica común a todas las estructuras de ABB equilibradas es el uso de rotaciones para reorganizar sus árboles, a fin de satisfacer las distintas restricciones que tienen. La característica más importante de las operaciones de rotación es que no destruyen la propiedad de los ABB en los subárboles en los que se ejecutan.

En la figura 2 se muestran claramente las operaciones básicas de rotación utilizadas para implementar los patrones de rotación.

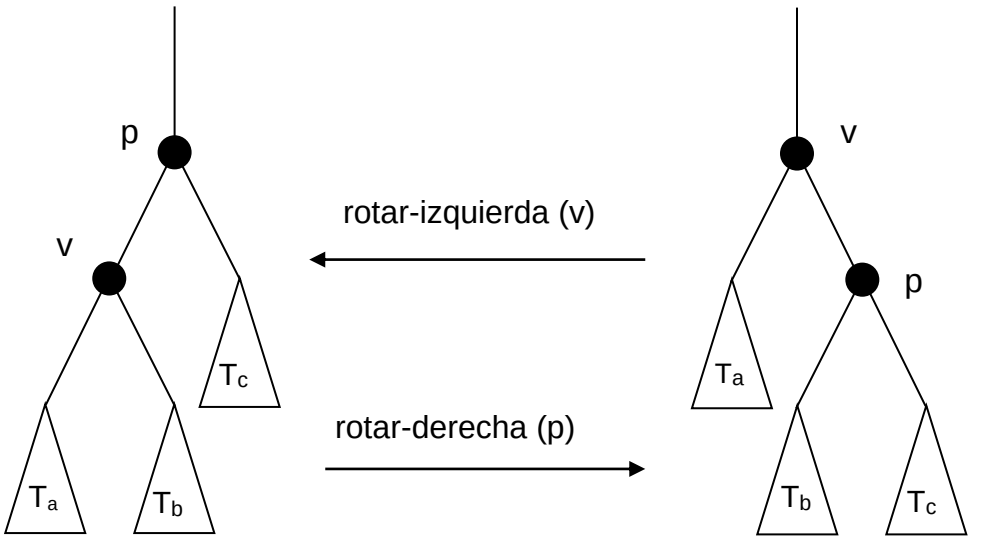

Figura 2. La operación rotar-derecha (p) comienza con el subárbol de la izquierda y produce el subárbol de la derecha, mientras que la operación rotar-izquierda(v) comienza con el subárbol de la derecha y produce el de la izquierda.

Árboles AVL

Un árbol AVL es un árbol binario de búsqueda en el que las alturas de los subárboles izquierdo y derecho de cualquier nodo difieren a lo máximo en 1.

Esta restricción de los árboles AVL con respecto a su altura define también a la llamada propiedad de los árboles AVL. Estos árboles se representan de la misma manera que los árboles binarios de búsqueda, aunque deben almacenar un campo adicional que indique el equilibrio del nodo. Tal equilibrio se determina examinando los subárboles izquierdo y derecho. El equilibrio de un nodo es dado por la altura de su subárbol derecho menos la altura de su subárbol izquierdo, logrando con esto que el equilibrio en un nodo del un árbol AVL sea representado con un –1, que indica que la altura del subárbol derecho de un nodo es uno menos que la altura de su subárbol izquierdo, un 0, que indica que ambos subárboles tienen la misma altura, o bien, +1, que indica que la altura del subárbol derecho de un nodo es uno más que la altura de su subárbol izquierdo.

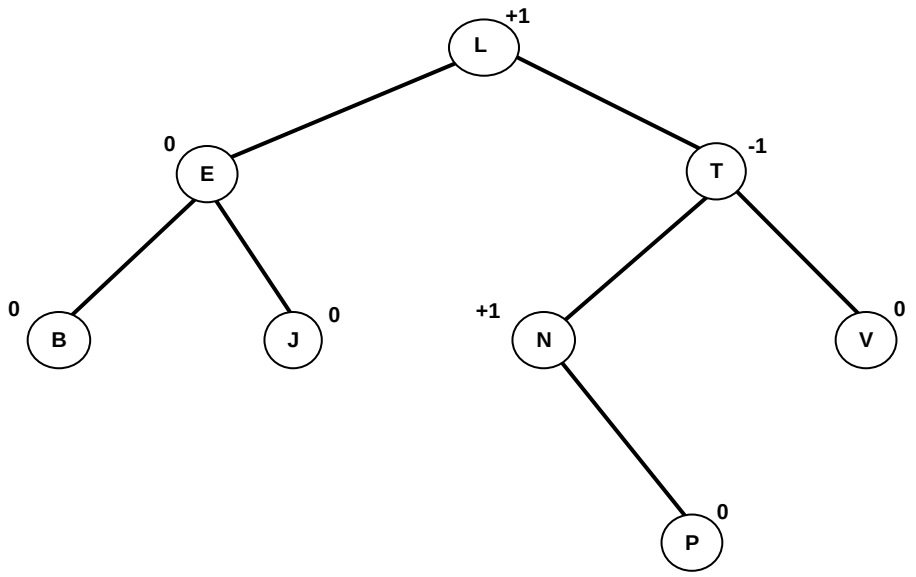

Figura 3. Un árbol AVL de ejemplo con el equilibrio de cada nodo mostrado al lado del nodo.

El árbol AVL es de fondo un árbol binario de búsqueda, y como tal, debe mantener la propiedad de estos árboles, y esto implica que se pueden utilizar los mismos algoritmos desarrollados para los ABB para realizar operaciones de consulta en un AVL. Sin embargo, para los casos de inserción y eliminación la implementación no es tan fácil ni directa, ya que estas operaciones afectan directamente al equilibrio de estas estructuras. En otras palabras, la inserción requiere una operación adicional que garantice el equilibrio.

En la operación insertar primero se recorre un camino desde el nodo raíz y el nuevo elemento se inserta como una hoja. Después se recorre el camino de regreso a la raíz y se van actualizando los equilibrios de cada nodo.

Las operaciones requeridas para equilibrar los árboles AVL son la rotación simple a la izquierda, la rotación simple a la derecha, y las rotaciones dobles ID y DI. Tanto los patrones de rotación simple como los patrones de rotación doble respetan la altura y el equilibrio del resto de los nodos de un árbol.

De esta manera resulta relativamente sencillo balancear los árboles en la etapa de recorrido de regreso del procedimiento de inserción, ya que sólo se tiene que buscar el primer nodo cuyo equilibrio cambie de ± 1 a ± 2. Este nodo es conocido como pivote, y es éste sobre el cual se realiza la rotación, y una vez realizada, la altura del subárbol con raíz en el nodo pivote se

restablece al valor que tenía antes de la inserción, para lograr con esto que todos los nodos del árbol mantengan el equilibrio de 0 o ±1.

En el caso de la operación de eliminación el procedimiento no es muy diferente. Se utiliza de manera inicial el algoritmo de eliminación para árboles binarios de búsqueda para eliminar el nodo deseado. Debido a que este nodo puede ser antecesor o predecesor, se recorre el camino desde el padre del nodo eliminado de regreso hacia la raíz, actualizando los equilibrios en el camino, considerando por supuesto que si se encuentra un nodo cuyo equilibrio pase de ±2, entonces se utilizará un patrón ya sea de rotación simple o doble. Dado que la altura de un subárbol no se conserva después de eliminar un nodo, en algunos casos esta operación puede requerir más de un paso en el proceso de reequilibrio.

Árboles rojo-negro

Un árbol rojo-negro es un árbol binario de búsqueda aumentado en el que todo nodo está coloreado de rojo o de negro, y en el que la disposición de los nodos obedece las siguientes restricciones:

Regla del negro. Toda hoja está coloreada de negro.

Regla del rojo. Si un nodo es rojo, entonces sus dos hijos son nodos negros.

Regla del camino. Todo camino de la raíz a una hoja contiene el mismo número de nodos negros.

Por conveniencia para efectos de análisis, sin que esto implique asignación de memoria para estas estructuras, se asume que se añaden nodos en las posiciones de todo puntero nulo de un árbol binario, tal como se puede observar en la figura que muestra un árbol rojo-negro.

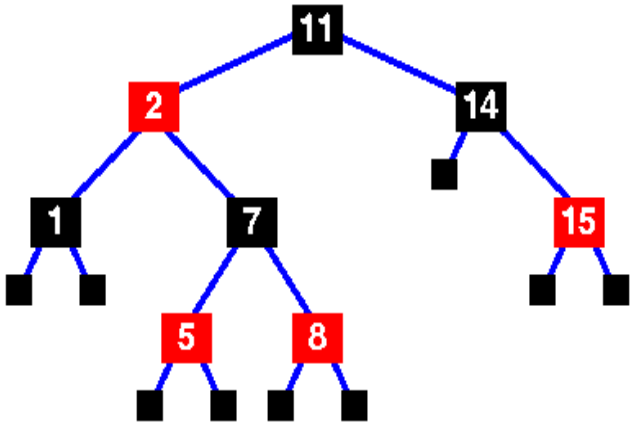

Figura 4. Ejemplo de un árbol rojo-negro, en donde se muestra que se cumplen las restricciones que lo definen. En este ejemplo se agregan nodos en las posiciones de punteros nulos, para efectos de análisis.

En los árboles rojo-negro, todos los caminos de un nodo a cualquier hoja deben contener el mismo número de nodos negros, y por consecuencia, todos los subárboles son a la vez árboles rojo-negro. Un término importante para las operaciones que se realizan en los árboles rojo-negro es la altura negra, ésta se refiere al número de nodos negros desde, pero sin incluir un nodo dado v, hasta un descendiente hoja.

Para esta estructura, al igual que las otras se asume que el árbol rojo-negro se representa de la misma manera que un árbol binario de búsqueda, con la condicionante de almacenar en un campo adicional el indicador de color.

Para los árboles rojo-negro, también se pueden utilizar los algoritmos de búsqueda de los ABB, pero después de operaciones como inserción o eliminación, se realizan los pasos de rotación y recoloración para lograr el equilibrio. En el caso de eliminación por ejemplo, los árboles rojo-negro utilizan de igual manera el algoritmo de eliminación de los ABB para quitar el nodo apropiado, pero dependiendo del color del nodo que se esté eliminando, se elegirá el patrón necesario para hacer que la estructura cumpla con las reglas.

Árboles biselados

Un árbol biselado es un ABB en el que todas las operaciones se implementan utilizando un esquema de reestructuración que es conocido como biselación. En esta estructura se busca trasladar a un nodo dado a la raíz del árbol biselado por medio de una secuencia de rotaciones, trasladando los elementos que son accedidos más frecuentemente hacia la raíz del árbol, para que puedan ser encontrados más rápidamente.

Esta estructura a diferencia de los árboles AVL o rojo-negro, no se preocupa de los accesos individuales, y no almacena ningún tipo de información de equilibrio en sus nodos, logrando con esto que sean más eficientes en espacio y de implementación más simple.

Es importante tener en cuenta la forma en que un nodo es rotado hacia la raíz, para lograr tiempos de ejecución eficientes, la estrategia utilizada es utilizar uno de tres patrones de rotación básicos, que analizaremos suponiendo la biselación en el nodo v, cuyos nodos padre y abuelo son p y g respectivamente:

- *Patrón zig*. Si p es la raíz del árbol y v es un hijo izquierdo, se realiza una operación rotar-derecha(p). En este caso siempre termina la biselacíón porque v llegó a la raíz.

- *Patrón zig-zig*. Si p no es la raíz, y v y p son ambos hijos izquierdos, ser realiza rotar-derecha(g), seguido por rotar-derecha(p).
- *Patrón zig-zag*. Si p no es la raíz, p es un hijo izquierdo y v es un hijo derecho, se realiza rotar-izquierda(p) seguida por rotar-derecha(g).

Para la operación buscar en los árboles biselados se utiliza la misma técnica que se emplea en los ABB. Suponiendo que se busca un nodo con clave k, si se encuentra un nodo v conteniendo un elemento con la clave k, entonces el árbol es biselado en v y se devuelve el elemento k, pero si la búsqueda es fallida, entonces el último nodo examinado antes de alcanzar un puntero nulo es biselado y se devuelve nulo y en este caso el nodo trasladado a la raíz contendrá al sucesor o predecesor en inorden de la clave que estamos buscando.

Para insertar, se comienza buscando el punto de inserción, si se encuentra un puntero nulo se añade el nuevo nodo al árbol. Posteriormente el nuevo nodo es biselado, sin embargo si ya existe en el árbol un elemento con la clave que estamos buscando, se bisela ese nodo.

El proceso para eliminar, es más complicado, ya que involucra dos biselaciones. Primero, se debe encontrar el nodo que se quiere eliminar, y si éste no se encuentra, entonces el último nodo encontrado es biselado. Si suponemos que la clave que se quiere eliminar está en el nodo v, entonces v se bisela a la raíz y es retirado. Esto da como resultado dos árboles biselados separados, para unir estos dos árboles, se reorganiza uno de ellos. En el caso de la reorganización del subárbol izquierdo, se ejecuta la operación Máximo, lo que provoca que el predecesor en inorden de v, denotado por v^- sea trasladado a la raíz, siendo ahora v^- la clave máxima del subárbol izquierdo, no tendrá hijo derecho. Así finalmente la operación se termina haciendo que la raíz del subárbol derecho sea el hijo derecho de v^-. Si se hubiera optado por reorganizar al subárbol derecho, el procedimiento hubiera sido el mismo, sólo que se debería ejecutar la operación Mínimo, antes de añadir el subárbol izquierdo como hijo izquierdo del nodo resultante.

Descripción y evaluación de sistemas dedicados a la enseñanza de este TDA (state of the art)

Las estructuras de datos son una parte fundamental e imprescindible de los programas de estudio de las carreras relacionadas con el área de la computación, tanto en los niveles de pregrado como en los programas de estudio de maestrías y doctorados.

En el Instituto de Estudios superiores de Monterrey, en México, por ejemplo, en las carreras de Ingeniero en Sistemas Computacionales, Ingeniero en Sistemas de Información y Licenciado en Sistemas de Computación Administrativa se incluye de igual manera la materia de "Estructuras de Datos" con la siguiente descripción: *"Al finalizar este curso se espera que el alumno reafirme los conceptos de abstracción de datos y programación orientada a objetos, a través del conocimiento y aplicación de las estructuras de datos fundamentales en el desarrollo de software. Sea capaz de seleccionar la estructura de datos más adecuada en la solución de un problema que la requiera. Sea capaz de diseñar una nueva estructura de datos, de acuerdo a las necesidades de solución de un problema."* Esto demuestra la importancia que tiene ésta área de estudio en las carreras relacionadas con la computación.

En el caso de la Maestría en Ciencias de la Computación de la misma institución el programa presenta un nivel de profundidad más interesante, la materia se llama "Estructuras y Bases y de datos" y en la descripción se incluyen: *"Tipos de datos. Tipos de estructura de datos: pilas (stack), colas, listas encadenadas, etc. Apuntadores. Memoria dinámica. Tablas, árboles, búsqueda binaria, hashing. Ordenamiento de información: diferentes algoritmos y análisis de su eficiencia. Modelos de bases de datos: relacional, semántico y relacional extendido. Integridad. Seguridad. Concurrencia. Respaldo. Bitácora. Recuperación. Transacciones. Organización de archivos. Sistemas operativos. Sincronización y comunicación entre procesos. Memoria virtual. Compiladores."*

Es común sin embargo, encontrar otras universidades que den mayor énfasis a la enseñanza de este tipo de materias, por ejemplo, la Universidad de la Palmas de Gran Canaria, en España, que en sus carreras de Ingeniería en Informática, Informática de Sistemas e Informática de Gestión ofrece tres cursos de estructuras de datos, y en el programa incluyen: *Tipos de datos y estructuras de datos, tipos abstractos de datos, especificación abstracta, interfaz, implementación, encapsulamiento, ocultación de información. Operaciones: generadoras, constructoras, accesoras, modificadoras, asignación y copia, taxonomía de los tipos abstractos de datos y de las estructuras de datos, isomorfismo. Ejemplo de tipos abstractos contenedores: contenedores asociativos (conjuntos y tablas). Ficheros. Reutilización del espacio libre, Índice. Árbol Trie. Listas invertidas. Árboles AVL, Árboles Rojo-Negro, Árboles biselados, Árboles B,*

Árboles B, Árboles B+. Dispersión con tabla de tamaño fijo. Resolución de colisiones por: encadenamiento y direccionamiento abierto, Dispersión extensible y lineal*

Prácticamente la totalidad de las instituciones educativas utilizan recursos disponibles a través de Internet. Estos recursos consisten en información relativa al curso como notas e instrucciones, citas bibliográficas, y en general información relativa al curso y a la misma organización. Es precisamente ésta una de las razones que impulsan este proyecto, es decir, la posibilidad de ofrecer a los estudiantes de la materia de estructuras de datos, información que sea de utilidad, para el futuro desarrollo de aplicaciones interactivas que les den una alternativa dentro del proceso de aprendizaje.

Existen en Internet algunos sitios, con aplicaciones basadas en la teoría de las estructuras de datos, cuyo principal problema es la falta de un enfoque orientado al estudiante. Podría pensarse que algunas de estas aplicaciones fueron desarrolladas con el propósito único de demostrar la viabilidad de algún algoritmo, o bien, el de observar el comportamiento de las representaciones gráficas de estas estructuras de datos. Se pretende dar un enfoque pedagógico a estas aplicaciones, que permitan al estudiante utilizarlas como verdaderas herramientas de aprendizaje.

Caso “AVL-balanced tree applet”

Existe en el sitio http://www.seanet.com/users/arsen/avltree.html una aplicación que resulta por demás interesante, dada la facilidad de uso, y la sencilla explicación de la aplicación que el autor presenta.

La aplicación está desarrollada en Java y está hospedada en un sitio comercial. Desgraciadamente no existe mayor información referente al autor, y aparentemente la cuenta de correo que se publica en caso de buscar comunicación con el autor, no existe más.

La aplicación es sencilla, en apenas un pequeño párrafo describe la ilustración utilizada para mostrar el comportamiento de un árbol AVL balanceado.

Las operaciones permitidas son: inserción, borrado, y búsqueda, todas mediante botones de control.

Todas las operaciones están animadas, y se advierte que en caso realizar una operación sin haber terminado la previa, la animación de la primera será terminada para dar lugar a la animación de la última operación solicitada.

Los datos pueden ser introducidos de forma manual o de aleatoria, ya sea utilizando la opción "Random" o simplemente seleccionando la operación sin parámetros, lo que generará también un valor aleatorio, sólo que este no será desplegado en el campo de entrada.

Se puede lograr que el árbol resultante esté balanceado con sólo escoger la opción "AVL". Además si se selecciona esta opción con un árbol ya generado, éste será rebalanceado.

La aplicación cuenta además con una opción para seleccionar el método que se quiera utilizar para realizar la operación, ya sea inorden, en preorden o en postorden.

Finalmente se incluyeron dos funciones más a la aplicación, "Shape", que contrae o expande el árbol, y "Shake", que según el mismo autor, debe ser utilizada para, como dice literalmente, "keep the bugs out". En los casos que se probaron no se logró generar ningún error, o al menos éste no fue perceptible.

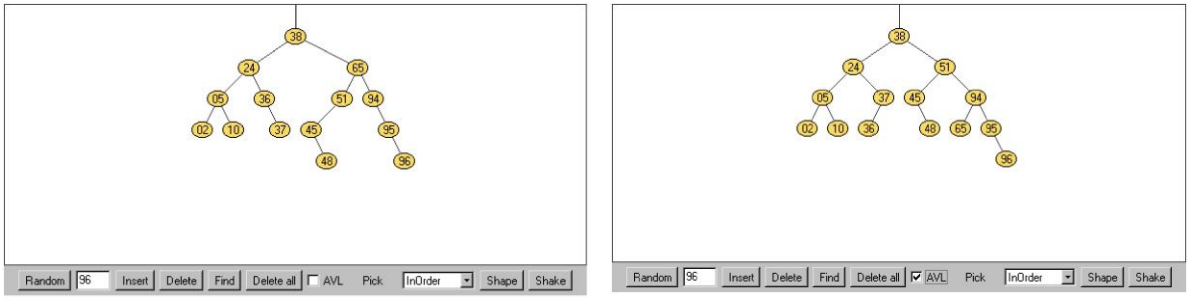

Figura 5. Imágenes de la aplicación AVL-Balanced tree applet. Inicialmente sin la opción AVL, y posteriormente rebalanceado, después de seleccionar el checkbox "AVL"

En el ejemplo se puede observar como se comporta la aplicación con la inserción de 14 elementos introducidos de manera aleatoria. Posteriormente se seleccionó la opción "AVL", lo que generó que el árbol se equilibrara.

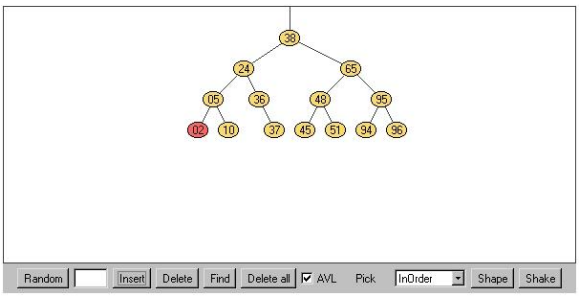

Figura 6. La misma secuencia de números, pero con la opción "AVL" desde el inicio de la inserción.

Sin embargo, la misma secuencia introducida de manera inicial como AVL genera un árbol ligeramente diferente al anterior.

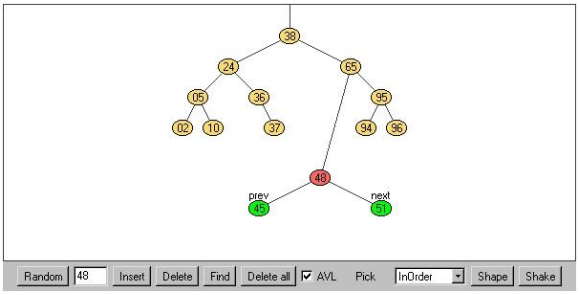

Figura 7. Ejemplo de las operaciones *antecesor y predecesor*.

Finalmente, otras de las operaciones que se pueden realizar en esta aplicación son predecesor y sucesor. Sólo con seleccionar un nodo con el mouse, aparecerán los nodos "prev" y "next" (antecesor y sucesor respectivamente) del nodo seleccionado, si es que existen.

Esta aplicación resulta por demás interesante y útil, dados los objetivos perseguidos, y cuando menos en lo que respecta a la interfaz, resulta agradable trabajar con este sistema. Desgraciadamente el trabajo se ve demeritado por la limitante que tiene la aplicación en cuanto al tamaño fijo de la ventana para desplegar el árbol generado. A pesar de tener un límite de 100 nodos, el espacio resulta insuficiente para desplegar el árbol, o bien, poder observar de manera detallada el proceso de animación que se lleva a cabo durante cada operación.

Es importante mencionar que la presencia de explicaciones detalladas para cada paso realizado en las operaciones cambiaría de manera importante la manera en la que esta aplicación podría

ser utilizada por los estudiantes, actualmente, a pesar de ser una de las aplicaciones más claras y fáciles de usar, no es la mejor herramienta para el proceso de enseñanza.

Varios meses después de haber analizado esta aplicación y dentro del periodo en el que se escribió este documento, se hicieron modificaciones a la aplicación antes mencionada. Se hicieron correcciones y se añadieron funciones que antes no se habían considerado, y que mejoraron de manera sustancial esta aplicación, tanto por las funciones añadidas, como por el mayor beneficio que ahora se puede obtener del uso de la aplicación. Dentro de las modificaciones más importantes se encuentran:

- Animación de balanceos sencillos y dobles después de inserciones
- Opciones para pausar la animación y para hacerla más lenta que permitirán mejor observación de la operación que se está animando
- Inclusión de breves comentarios acumulativos que dan mayor información sobre los pasos que se están llevando a cabo, para cada operación seleccionada
- Se incluyó una opción para desplegar “alturas” de los subárboles, cuando se haya seleccionado modo “AVL”
- Se incluyó la función de árboles rojo-negro, con el que se puede crear un árbol usando estas reglas, o convertir el árbol existente en rojo-negro
- Se agregó la operación “Splaying”, que permite ascender el nodo seleccionado a la raíz del árbol, tanto la operación de “Splay” como la operación de “rotación” son manuales, y eliminan automáticamente la opción de “AVL” o de “Rojo-negro”.
- Además de las correcciones que se hicieron, se eliminó la operación mediante la cual se identificaban el “predecesor y el sucesor” de cualquier nodo seleccionado.

Con todas estas modificaciones, la aplicación realmente se convierte en una herramienta de utilidad, ya que de manera adicional se da información de ligas de interés y se publican algunas partes del código en la dirección: http://www.seanet.com/users/arsen/source.html que pueden resultar de gran utilidad en caso de futuras implementaciones o análisis por parte de estudiantes o docentes.

No se tiene registro acerca de la fecha a partir de la cual esta disponible esta animación en Internet, pero cuando menos la fecha de la última modificación es del 18 de Mayo de 2002.

Caso "AVLVis System, a visualization and tutoring tool for the AVL Data Structure"

Esta es una aplicación escrita por Wiley Fuller y John-Michael Kent, con el apoyo del Dr. Frederic Maire, profesor de la School of Computing Science and Software Engineering, de la Universidad de Tecnología de Queensland, en Brisbane, Queensland, Australia.

La aplicación es denominada AVLVis, al igual que en el caso anterior está desarrollada en JAVA y está publicada en Internet en la página personal que el Dr. Frederic Maire tiene en el sitio de la Universidad de Tecnología de Queensland, Australia, en la siguiente dirección: http://sky.fit.qut.edu.au/~maire/avl/System/AVLVis.html.

La aplicación resulta de gran interés, ya que a pesar de estar limitada solamente a las operaciones de insert y delete, cuenta con tres diferentes modos de operación (Tutor On, Step Mode y Express Mode) que le dan un valor especial. A diferencia del caso del "AVL Balanced Tree Applet" anteriormente analizado y mencionado, esta página cuenta con una liga a otra página en donde se encuentra la información teórica de los árboles AVL, se mencionan los requerimientos de estas estructuras, se describen brevemente las operaciones que pueden realizarse, así como los diagramas de las cuatro posibles rotaciones que pueden ser realizadas en un árbol AVL no balanceado, en los que se muestra tanto el estado previo, como el resultante de la estructura. También cuenta con un pequeño manual de operación de la aplicación en el que se menciona de manera sencilla el uso de cada uno de los botones de la aplicación (*reset, previous y undo*), así como la diferencia de operación entre el "Tutor Mode" y el "Step Mode".

Cuenta con tres funciones que pueden ser seleccionadas en cualquier momento, "reset" elimina los valores y permite comenzar a trabajar con un árbol vacío, "previous", aparentemente debería estar dirigido hacia una ventana con la estructura tal como estaba antes de la última operación realizada, pero desgraciadamente, en la mayoría de las computadoras con MS Internet Explorer esta función no pudo ser ejecutada, salvo en navegadores con Netscape, y "undo" deshace la última operación realizada.

Con el fin de analizar el funcionamiento de esta aplicación, se utilizó la misma secuencia de valores numéricos introducidos en el mismo orden, para comprobar el grafo resultante. En el modo de operación "Tutor On", el sistema determina si se hace o no una pregunta al usuario, lo anterior basado en si el resultado de la pregunta previa hecha al usuario fue correcto. Posteriormente aparece una ventana con la pregunta, y en la parte inferior izquierda aparece una imagen que indica que se está haciendo una pregunta. Otra manera de preguntar utilizada por el sistema, es mediante la línea de texto ubicada en la parte inferior de la sección con la representación gráfica. Una vez dada la respuesta, se cambia la imagen que indicaba que se

estaba haciendo una pregunta, para indicar entonces si la respuesta fue dada de manera correcta o no. Algunas preguntas dan al usuario hasta tres oportunidades para responder de manera correcta, en caso de no ser así, se despliega una [x] en el área del gráfico.

Las preguntas pueden ser, seleccionar el nodo que será padre del nodo a insertar, o si es necesario rebalancear el árbol después de insertar o eliminar, o bien, qué tipo de rotación será necesario efectuar después de haber insertado o eliminado algún nodo determinado, tal como se puede observar en la siguiente figura.

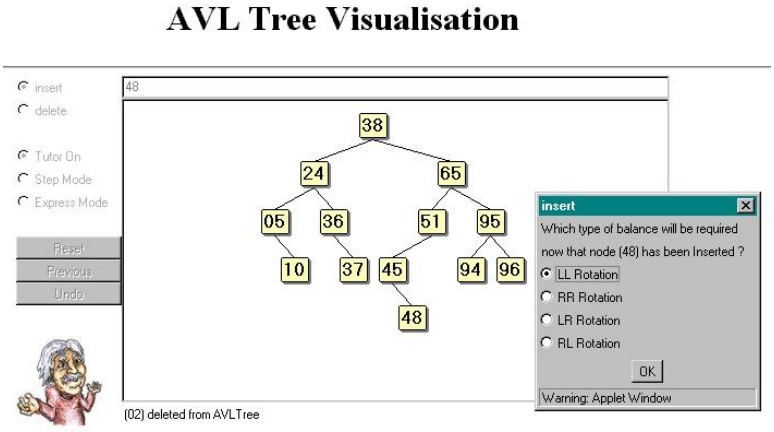

Figura 8. Ejemplo de la aplicación AVL Tree Visualization con el modo de operación "Tutor On".

Finalmente después de haber insertado todos los valores que fueron utilizados en la primera aplicación analizada, se concluye que el resultado es el mismo, tal como se muestra en la figura 9.

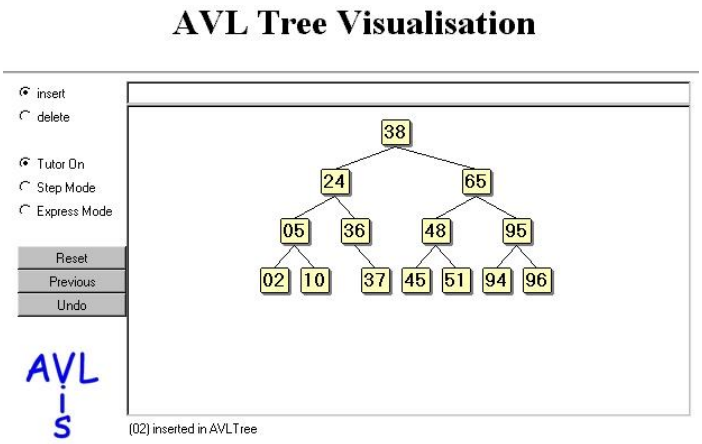

Figura 9. Estado del árbol resultante, utilizando la misma secuencia de números que en otras

El modo “Step Mode” en realidad no difiere mucho del “Tutor on” a excepción de que en el primero no se realizan preguntas, sólo en la parte inferior de la ventana donde se van insertando los nodos, aparece el texto describiendo que operación ha sido realizada. Para continuar con la operación, el usuario sólo tiene que hacer clic en el área del gráfico del árbol AVL.

En el modo “Express Mode”, que se asume no habría preguntas ni pausas debido a las leyendas, no hay diferencia alguna con respecto al “Step Mode”, funciona exactamente igual.

Caso AVL Tree

Esta aplicación, desarrollada por John Kloss está publicada en la dirección http://www.cs.jhu.edu/~goodrich/dsa/trees/avltree.html, página personal del Profesor de la Universidad John Hopkins en Maryland, Michael T. Goodrich, a quien Kloss asiste en proyectos de investigación enfocados a producir clases de estructuras de datos más rápidas y eficientes que las incluidas en las bibliotecas como JDK.

La animación propuesta por Kloss resulta bastante sencilla a simple vista, las operaciones permitidas son inserción, búsqueda y eliminación, la última revisión de la aplicación data de Julio de 1997, y se advierte un error al realizar eliminaciones, ya que éstas no rebalancean el árbol.

La interfaz consta de 3 paneles, en el superior aparecen los botones de las operaciones incluidas en la implementación, insertar, eliminar y buscar. El segundo panel se utiliza para desplegar la

estructura, que puede ser reacomodada con el mouse, ya que en algunos casos, después de ciertas operaciones, el rebalanceo produce un árbol cuyas dimensiones exceden el área de trabajo especificada. Finalmente el tercer panel se utiliza para desplegar un breve mensaje relativo a la operación que recién se efectuó.

Si se consideran como referencia otros elementos existentes en otras aplicaciones, como opción de tutorial, diferentes modos de operación e incluso diferentes tipos de árboles que pueden ser implementados, esta aplicación luce pobre. El contexto general en el que se presenta esta aplicación también es deficiente, ya que no presenta enlaces a otras páginas con información referente a algoritmos de implementación o a otras estructuras de datos, que pueden resultar de mucha utilidad cuando se está trabajando con estas estructuras de datos. Tampoco presenta información clara con respecto a la manera de utilizar la aplicación.
El valor que se encuentra a la aplicación es la disponibilidad del código para análisis posterior.

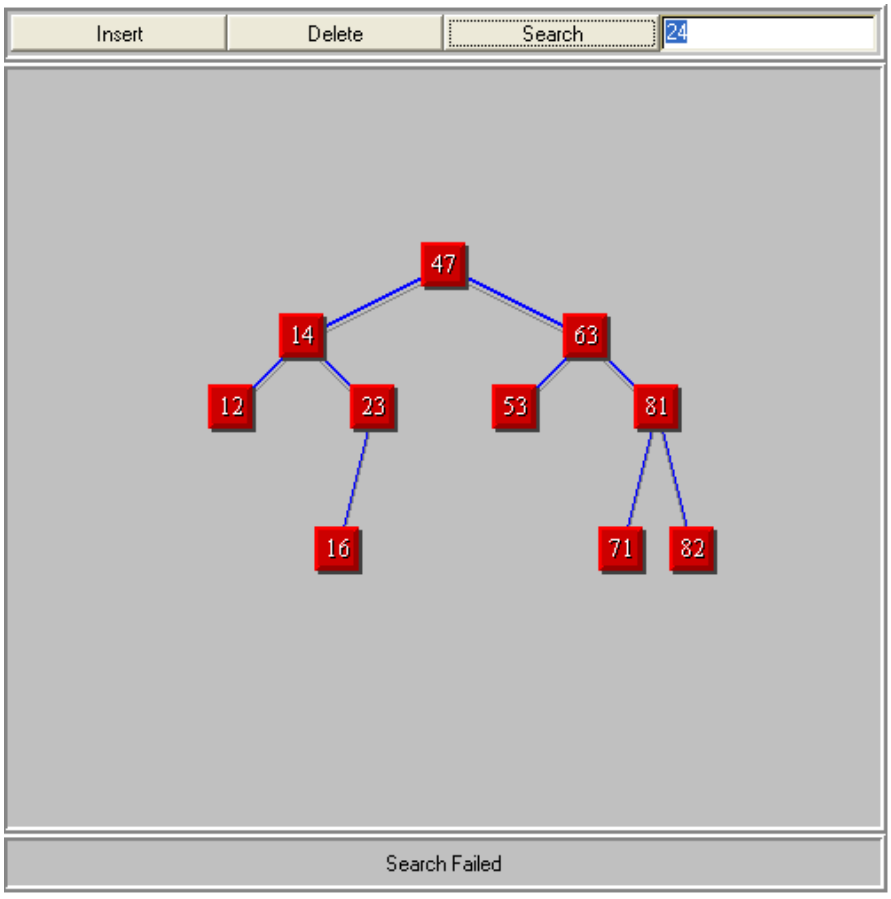

Figura 10. En la figura 10 se puede observar la interfaz de la animación AVL Tree propuesta por John Kloss.

La fecha de última actualización de esta página, que se supone incluye las últimas modificaciones hechas al código fuente de la aplicación, es deL 14 de Julio de 1997.

Caso Red-Black Tree Demonstration

John Franco, un profesor de la Universidad de Cincinnati, en Ohio, tiene en su página personal que está publicada en el sitio de la misma Universidad, diferentes ejemplos de varias aplicaciones, de las cuales, la gran mayoría son relacionadas con el uso de algoritmos. Entre las aplicaciones prácticas más importantes que tiene, se encuentra publicada en Internet la demostración en Java de los árboles rojo-negro en la dirección http://www.ececs.uc.edu/~franco/, bajo la sección de “Class Notes”.

Esta aplicación tiene instrucciones básicas de uso, comienza con un árbol vacío, y tiene la función de “restart” que en cualquier momento generará otro árbol vacío. El botón de “add node” inserta un nodo rojo con el valor introducido en el campo, posteriormente se tendrá que presionar el botón de “next step” para ver cual es siguiente paso en el proceso de inserción. Aunque no existe información de texto de acompañe a la interacción, en donde se explique qué es lo que se está haciendo, cuando menos el usuario puede darse cuenta que la operación no ha concluido, ya que el fondo de la ventana principal está de color verde, y se tiene que presionar el botón de “Next”, para que se concluya la operación actual, una vez hecho esto, el fondo del área de trabajo pierde el color y queda gris claro, indicando que la operación ha concluido. El mecanismo para eliminar nodos es diferente, primero se selecciona el botón “delete node” y después se selecciona el nodo que será eliminado, este nodo se mostrará verde, y mediante el uso del botón “next step” se irán realizando los pasos necesarios para terminar con la operación y balancear el árbol. Finalmente el botón “undo” regresa el árbol a su estado antes de la última operación realizada.

Es importante también mencionar que en esta página el autor proporciona su dirección de correo electrónico para reportar problemas, y de acuerdo a la última vez que esta página fue revisada, la aplicación había sido corregida en Diciembre 8 del 2002, por errores en el proceso de eliminación.

Muy importante e interesante resulta también que el código está disponible, y como menciona el mismo autor, el código está *“feo y sin comentarios”*, y su interés principal para crear la aplicación fue divertirse él mismo y divertir a su clase. Hay además un enlace para una versión de “mantenimiento de la misma aplicación, en esta página la principal diferencia es que en esta versión el árbol comienza no vacío, y que existe además un botón para cambiar el color de los

nodos, que en realidad no tiene mucho sentido, ya que no se puede saber si el cambio fue correcto, y de hecho la aplicación permite hacer cambio incorrectos. Finalmente al final de la página se explican de manera muy breve las propiedades de los árboles rojo-negro.
Siguiendo con la misma secuencia de nodos utilizada para las aplicaciones previamente mencionadas, la estructura resultante es similar.

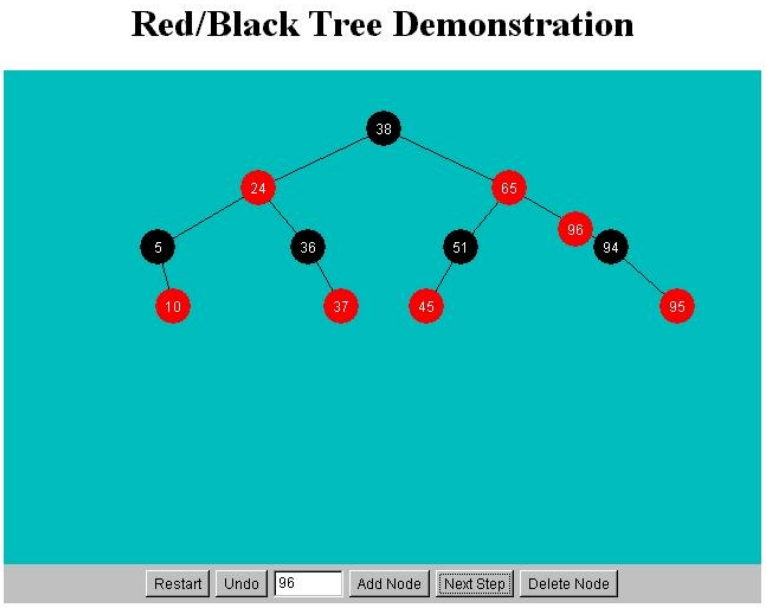

Figura 11. Interfaz de la aplicación "Red/Black Tree Demonstration, durante la ejecución de una operación.

En esta figura se puede observar el proceso de inserción de un nuevo nodo. Se agregó el 96, durante el tiempo en el que se esté ejecutando la operación el fondo de la ventana es verde, que indica que una operación se está llevando a cabo, se puede observar también que el nodo representando al valor 96 se va acercando al nodo adecuado, conforme se va presionando el botón "next step". Una vez que la operación terminó. El fondo de la ventana cambia a gris nuevamente.

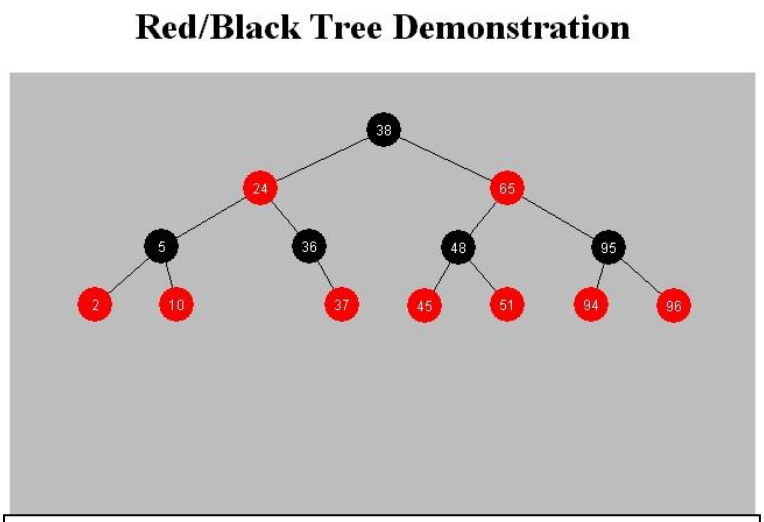

Figura 12. Interfaz de la aplicación "Red/Black Tree Demonstracion, después de concluir una operación.

En esta figura se puede observar el árbol resultante, después de haber insertado todos los nodos. Se puede observar, que los árboles resultantes han sido idénticos en todas las aplicaciones que se han analizado.

Esta aplicación también ha sido actualizada en Diciembre de 2002, según el autor, se corrigieron errores que existían en el proceso de eliminación de nodos, además de otros detalles. La nueva versión, de reciente actualización, cuenta con tres modos de velocidad diferentes, "fast, slow y crawling" que permiten observar los pasos de la animación, conforme se va completando la operación en curso. Se ha adicionado además un área en donde se incluyen pequeños mensajes tales como "deleting" o "adding" que permiten al usuario darse cuenta que la operación no se ha terminado todavía. En la versión anterior, que por cierto, todavía se encuentra disponible en el sitio, el indicador para saber que la operación había concluido era el cambio de color de fondo del área de trabajo.

Adicionalmente para esta versión de la aplicación, se incluye el código fuente de la aplicación en Java, de casi todas las clases utilizadas en el desarrollo de esta aplicación. Existen dos versiones de esta aplicación, cuya diferencia es la versión de Java utilizada para compilarlas, pero cuyo funcionamiento es esencialmente el mismo.

La interfaz de esta nueva versión actualizada es bastante similar a la anterior, tal como se puede ver en la siguiente figura.

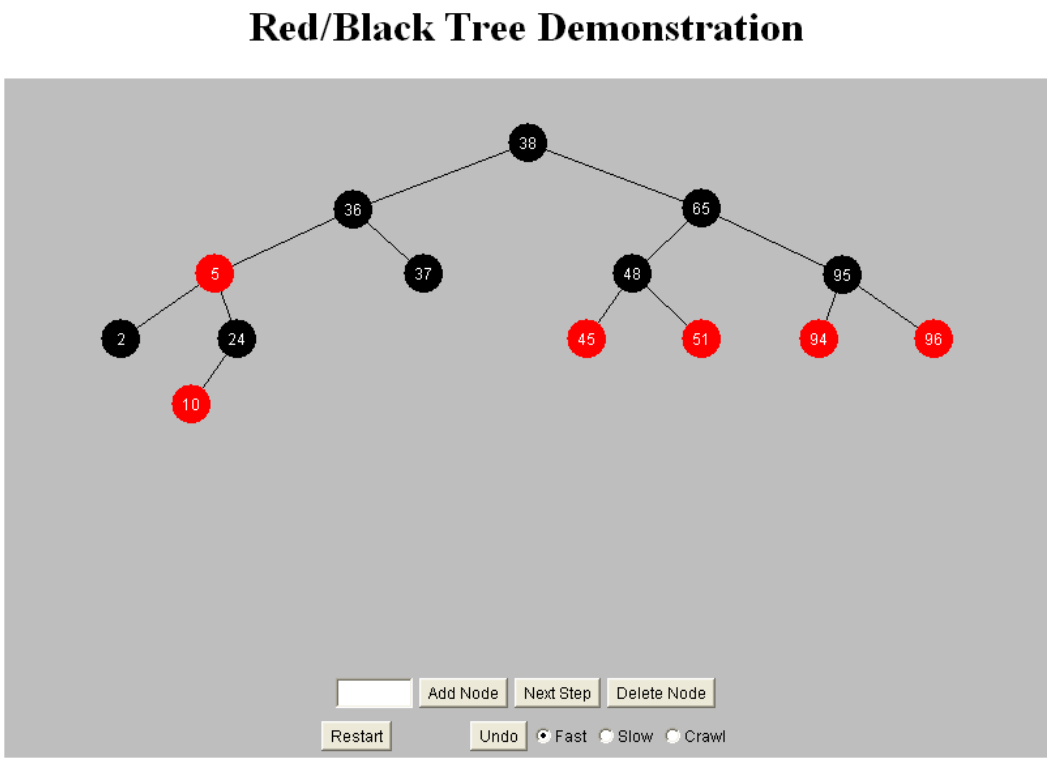

Figura 13. Interfaz de la nueva versión, actualizada en diciembre de 2002, de la aplicación "Red/Black Tree Demonstration.

Caso JDSL Visualizer 1.2.1

Este último caso analizado resulta diferente a los anteriores ya que no se tiene una aplicación en Internet, sino una aplicación que está disponible en Internet para descarga e instalación local, que son condiciones ciertamente diferentes a las que tienen otras aplicaciones que se han analizado. Sin embargo, el contexto en el que se ha desarrollado esta aplicación, los alcances que tiene, y los planes a futuro del grupo que trabaja en el desarrollo, hacen de esta una opción por demás interesante para analizar

Resulta un tanto obvio mencionar que gran parte de los proyectos de investigación actuales se realizan en universidades u organizaciones ligadas a la industria en los Estados Unidos de Norteamérica.

Existe un esquema de colaboración entre la universidad de Brown en Rhode Island, la universidad Johns Hopkins en Baltimore, ambas en los Estados Unidos de Norteamérica, y una compañía llamada AlgoMagic Technologies, dedicada a ofrecer servicios de seguridad a redes

grandes basados en STMS (Secure Transaction Management System), gracias a este esquema de colaboración se logró conformar una organización llamada “The data structures library in java” o JDSL por sus siglas en inglés.

El trabajo de la organización denominada JDLS, que además recibe apoyo gubernamental, es crear una colección de interfaces y clases de java que implementen estructuras de datos y algoritmos fundamentales. Dentro de esos algoritmos se encuentran por ejemplo: secuencias, árboles, colas de prioridad, árboles binarios de búsqueda, tablas hash, algoritmos de búsqueda y ordenación y gráficos entre otros más.

Este organismo considera como principio que, dado que los programas de cómputo organizan la información en estructuras de datos y la procesan con base en algoritmos, entonces proveyendo bibliotecas de componentes con estructuras de datos y algoritmos, se puede habilitar el rápido desarrollo de software para aplicaciones avanzadas.

Resulta interesante analizar los argumentos que los miembros del equipo de desarrollo de JDSL consideran para justificar los beneficios del uso de bibliotecas. Dentro del documento “Overview” publicado en su sitio de Internet, hacen referencia al hecho de que el uso de componentes para algoritmos y estructuras de datos avanzadas es bastante más confiable y rápido que el uso de opciones alternativas, de hecho consideran que reescribir estos componentes implica mayor riesgo para los desarrolladores. Dentro de estos riesgos podemos mencionar:

Inseguridad: ya que los algoritmos complejos y las estructuras de datos son difíciles de implementar y aún más difíciles de probar, ya que los métodos de pruebas del software estándar son frecuentemente inadecuados.

Falta de conocimiento: comúnmente los programadores, al pasar por su etapa de estudiantes, están expuestos sólo a algoritmos y estructuras de datos elementales, por tal razón, los pequeños adelantos y algoritmos de reciente descubrimiento llegan primero a la comunidad de desarrolladores.

Extenso tiempo de desarrollo: los algoritmos avanzados y las estructuras de datos normalmente requieren demasiado tiempo para ser implementados

La teoría de esta organización es que evitando los riesgos antes mencionados, se acelera el proceso de desarrollo de software gracias a la disponibilidad de los componentes de la biblioteca.

El producto final de este esfuerzo conjunto de profesores e investigadores que conforman la asociación, recibe el mismo nombre, JDSL, y está desarrollado en Java precisamente por ser

éste un lenguaje comúnmente utilizado para el desarrollo de aplicaciones de software avanzadas, y en especial para Internet. Existe ya dentro de Java, en la distribución de Sun, Java 2 SDK, Standard Edition, una biblioteca denominada Java Collections o JC que sólo cuenta con implementaciones de las estructuras de datos básicas como mapas y diccionarios.

Otra biblioteca de uso común entre desarrolladores de software con Java es la JGL o Java Generic Library, sin embargo, esta biblioteca no está incluida dentro del paquete de implementación de Java de Sun, sino que es distribuida por una firma comercial llamada "Recursion Software Inc." quienes anuncian su producto como "Una biblioteca para Java, que es una un poderoso extra para el JDK que aumenta la API de Java con más de 50 algoritmos genéricos. JGL está diseñada para trabajar de manera consistente con la API de Java, aligerando el conocimiento del desarrollador mientras provee acceso a un avanzado set de herramientas"3

Una vez hecho el preámbulo con relación al JDSL, se resalta el trabajo de uno de los múltiples grupos que se han formado con los miembros colaboradores. Este grupo formado por cinco investigadores de las dos universidades que integran la organización, trabajó en un proyecto en el que utilizó la biblioteca JDSL desarrollada por la organización misma, y gracias a fondos proporcionados por la oficina de investigación del ejercito de los Estados Unidos (U.S. Army Research Office) y por la Fundación Nacional de Ciencias (National Science Foundation), logró crear un "Visualizador de estructuras de datos para la enseñanza".

Los objetivos que se trazaron en el proyecto son:

- Producir visualizaciones de estructuras de datos logrando la mínima interferencia con su diseño
- Permitir a los usuarios interactuar en tiempo real con todos los métodos de la estructura de datos para verificar su operación
- Desplegar estructuras de datos en un nivel conceptual, en lugar de sólo visualizar el contenido de la memoria

El resultado final fue un sistema, un visualizador que tiene una interfaz con la implementación de una estructura de datos escrita por el usuario, mediante la API de esa estructura. El sistema puede desplegar más de una estructura de manera simultánea, además mantiene un registro extenso de la actividad de las estructuras de datos, ya que registra para cualquier punto en el tiempo, el estado, los métodos invocados y los valores resultantes.

[3] http://www.recursionsw.com/products/jgl/jgl.asp

Otra de las funciones esenciales de este sistema es la “prueba” o tests que realiza al código de los estudiantes, pero en este caso, no se analizará esa función, ya que el énfasis que se pretende resaltar de esta aplicación está relacionado con la visualización de los algoritmos, más que con las funciones de prueba.

Este sistema, con sus dos funciones principales ha sido probado durante los últimos dos años en las universidades de Brown y Johns Hopkins con resultados alentadores. Durante este periodo se han realizado encuestas que han permitido a los investigadores sondear el nivel de aceptación o rechazo de los estudiantes con respecto al uso de este tipo de herramientas. A pesar de la gran variedad de respuestas por parte de los estudiantes, los resultados, son un tanto predecibles, ya que la gran mayoría de los estudiantes que han aceptado y utilizado esta aplicación como herramienta para su proceso de aprendizaje, han entregado en general mejores trabajos, con un número de errores considerablemente menor, en comparación con sus compañeros que no están del todo de acuerdo con la aplicación o su uso.

A pesar de que esta aplicación está todavía considerada como prototipo, y continuamente se siguen realizando modificaciones y correcciones, se puede encontrar más información y el código fuente e instrucciones de instalación para diferentes versiones de sistemas operativos en la siguiente dirección: http://www.cs.brown.edu/cgc/jdsl/explorations/jdslviz/jdslviz.html.

La última versión del visualizador publicada en Internet es la 1.2.1 y la página tiene fecha de última modificación de Enero 4 del 2001.

Capítulo II
Importancia del análisis que se va a desarrollar

Como se ha mencionado anteriormente, es claro que con este trabajo no se pretende crear nuevos algoritmos para implementar árboles binarios de búsqueda, sino analizar el contexto en el que se han utilizado los ya existentes, haciendo especial énfasis en las técnicas de programación utilizadas, así como en la importancia de la existencia de una herramienta como la aquí propuesta, que permita a los estudiantes de carreras relacionadas con el cómputo, en cursos de estructuras de datos, comprender el funcionamiento de los árboles binarios de búsqueda, conocer los algoritmos y visualizar su comportamiento, de una forma sencilla y eficiente.

Como parte fundamental de este capítulo se considera un análisis extenso de las aplicaciones mencionadas anteriormente, desde el punto de vista de la programación, que permita conocer los elementos utilizados en cada caso y determinar las mejores opciones que pueden componer una aplicación adecuada, que además de demostrar la implementación del árbol, cumpla con el objetivo educativo planteado en este documento.

Adicionalmente se toman en cuenta otros factores tales como la interfaz de la aplicación, la facilidad de uso y el entorno, que en este caso es una página de Internet, en donde pequeños elementos como una sección de ayuda o un enlace a una página con instrucciones de uso o un pequeño manual y una página con información adicional que contenga información teórica de los árboles binarios de búsqueda puede lograr una diferencia y notoriedad importantes.

Como marco teórico de referencia dentro de este documento se consideran los resultados o conclusiones de estudios documentados que muestran evidencia empírica que sugiere que los estudiantes aprenden mejor por medio de tareas prácticas, más que de tareas teóricas.[4] En estos estudios se muestra el significativo avance en el proceso de aprendizaje, cuando dentro de las técnicas utilizadas se incluyen herramientas visuales utilizadas en el proceso de aprendizaje.

En este caso tales técnicas educativas no son el objeto de estudio, pero si fundamentan de manera importante las aplicaciones con elementos visuales, las cuales se han venido utilizando

[4] Amir Michail es profesor del departamento de Ciencias Computacionales e Ingeniería de la Universidad de Washington. Amir en su estudio denominado “Teaching Binary Tree Algorithms through Visual Programming” menciona que la idea de usar computadoras para enseñar algoritmos no es nueva, menciona que las animaciones computarizadas se usan para ilustrar muchos algoritmos. Señala sin embargo, que se puede obtener una idea de tan sólo ver una animación de un algoritmo, pero que en la mayoría de los casos, como en cualquier actividad pasiva, importantes detalles pueden ser pasados por alto u omitidos en conjunto.

desde hace varias décadas, cuando las universidades comenzaron a tener acceso a equipo de cómputo.

Aportación

Este análisis de aplicaciones actuales disponibles en Internet que son utilizadas como herramientas en el proceso de aprendizaje de estructuras de datos, en particular de árboles binarios de búsqueda, servirá para conocer los esfuerzos que en esta área se han realizado, y servirá como base para proponer el desarrollo de una aplicación que concentre las ventajas de las implementaciones publicadas, y proporcionará los elementos que permitan desarrollar una aplicación que supere las existentes. En el análisis que se verá posteriormente, se resaltarán los elementos de mejora de las aplicaciones de mayor difusión en Internet, desde el punto de vista funcional, y se concentrarán los elementos que dan mayor utilidad a esas mismas aplicaciones.

Una aportación valiosa que representa este trabajo, es la descripción de la programación, a nivel de clases y métodos utilizados por cada una de ellas, de las diferentes implementaciones que representan los sistemas analizados, tratando de encontrar las ventajas y los puntos de mejora de cada una de ellas.

Derivado del resultado de estudiar el código fuente de las aplicaciones, se concluirá y determinará la forma en que se aprovechan las modernas técnicas de programación, tomando las mejores prácticas detectadas en cada una de las soluciones, e identificando las debilidades, de tal suerte que éstas se puedan solucionar y ofrecer alternativas para evitar dicha problemática en una aplicación que venga a mejorar todo lo encontrado.

Como consecuencia de lo anterior, el producto final será el análisis y diseño de una nueva aplicación, que además contará con la ventaja del idioma, ya que es relativamente poca la información que existe en Internet acerca de este tema, y prácticamente nulo el número de aplicaciones existentes en español.

Enseñanza y aprendizaje con Internet

Es bien sabido que el vertiginoso incremento de la presencia del Internet en muchos ámbitos de la vida moderna se ve reflejado de manera importante en el sector educativo. Si bien es cierto que el uso de esta tecnología puede resultar costoso, sobre todo para países en vías de desarrollo, o con infraestructura de comunicaciones no tan sofisticada, también es cierto que se

están haciendo esfuerzos notables, y que pese a las grandes diferencias entre países, Internet es un medio al alcance de todos, en mayor o menor grado.

El uso de la tecnología está cambiando gradualmente la vida de las personas, desde la manera en la que nos comunicamos, hasta la forma de trabajar e incluso nuestros hábitos de compra. En el caso de la educación, la situación no es muy diferente. El uso de Internet implica nuevas formas de trabajo, y replantea cambios en las técnicas existentes de enseñanza y en los mecanismos tradicionalmente utilizados, tanto por estudiantes como por los profesores mismos. El impacto del uso del Internet es tan fuerte que se debe hacer una seria reflexión sobre los fundamentos teóricos y empíricos que justifican su uso en el ámbito pedagógico.

Existen estudios documentados que dan un enfoque al uso del Internet, desde el punto de vista de su instrumentación para el aprendizaje (Borrás, 1997), en donde se analizan teorías pedagógicas y se fundamenta así el uso del Internet de acuerdo a los rasgos comunes que se encuentran.

No es la intención de este documento abordar tales teorías pedagógicas, ni su instrumentación con respecto al Internet, pero dado el contexto, resulta importante mencionar algunos datos de estas teorías, como fundamento pedagógico de la aplicación resultante del análisis objeto de este documento.

Constructivismo

- En la teoría del constructivismo se pueden mencionar los siguientes principios que son:
    a. De la instrucción a la construcción, que menciona que el aprendizaje tiene que ver con el concepto de que el conocimiento no se reemplaza, ni se acumula, sino que se transforma a través del pensamiento activo y original del aprendiz, y mediante la experimentación y resolución de problemas, tomando a los errores como base misma del aprendizaje.
    b. Del refuerzo al interés, que menciona como los estudiantes comprenden mejor cuando están envueltos en tareas y temas que cautivan su atención.
    c. De la obediencia a la autonomía, el profesor debería dejar de exigir sumisión y fomentar libertad responsable, a través de interacciones recíprocas.
    d. De la coerción a la cooperación, en donde las relaciones entre alumnos se consideran vitales, ya que a través de ellas se desarrollan la igualdad, justicia y democracia, y se progresa en el aprendizaje académico.

Se concluye en tales estudios (Borrás, 1997), que Internet presenta rasgos de un entorno de aprendizaje constructivo ya que permite poner en juego los principios mencionados antes. Es un sistema abierto guiado por el interés, iniciado por el aprendiz, e intelectual y conceptualmente provocador. La interacción será atractiva en la medida en que el diseño del entorno sea percibido como soportador del interés

Teoría de la Conversación

La teoría de la conversación, es frecuentemente invocada para fundamentar la validez pedagógica del entorno Internet. Según esta teoría el hecho de aprender es por naturaleza un fenómeno social; la adquisición de nuevo conocimiento es el resultado de la interacción de gente que participa en un diálogo; y aprender es un proceso dialéctico en el que un individuo contrasta su punto de vista personal con el de otro hasta llegar a un acuerdo.

Internet complementa la noción de interacción entre gente que trae diferentes niveles de experiencia en una cultura tecnológica, y es un entorno que presupone una naturaleza social específica y un proceso a través del cual los aprendices crean una zona virtual de "proximal development" (Vygotsky, 1978).

Teoría del Conocimiento Situado

Otra teoría a la que se acude para defender la fiabilidad de Internet como medio de aprendizaje es la del conocimiento situado. En esta teoría se considera que el conocimiento es una relación activa entre un agente y el entorno, y que el aprendizaje ocurre cuando el aprendiz está activamente envuelto en un contexto instruccional complejo y realístico (Young, 1993). La posición más extrema de esta teoría sostiene que no sólo el aprender sino también el pensar es situado y que por lo tanto debería ser considerado desde una perspectiva ecológica. Tal posición se basa en el trabajo de Gibson (1986) que enfatiza que se aprende a través de la percepción y no de la memoria.

El entorno Internet responde a las premisas del conocimiento situado en dos características: realismo y complejidad, ya que por un lado, posibilita intercambios auténticos entre usuarios provenientes de contextos culturales diferentes pero con intereses similares (Brown, Collins y Duguid, 1989); y por otro lado, la naturaleza inestable del entorno Internet constituye un escollo para los no iniciados, que sin embargo, y gracias a su participación periférica continuada, se ven recompensados con una enculturación gradual.

La efectividad pedagógica del Internet

La naturaleza del Internet representa un reto para la gente que trabaja en áreas de control curricular, ya que existen diferentes tipos de aproximación a este medio:

1. Ningún tipo de acceso
2. Acceso restringido limitado
3. Acceso a bases de datos
4. Participación por medio de “newsgroups”
5. Participación a través de comunidades de redes
6. Participación en proyectos independientes o colaborativos

Existen pocos estudios que reflejen resultados acerca del uso de Internet por maestros, pero en el caso de los estudiantes, Internet puede hacerles asumir un papel más activo en el proceso de adquisición de conocimientos. Internet es una invitación abierta a la enseñanza activa donde los estudiantes son a la vez recipientes y generadores de saber (Bruner, 1986; Hannafin, 1992).

Como resultado de diferentes investigaciones de uso de internet por profesores y estudiantes, se ha demostrado la relación causal entre la aplicación de ciertas estrategias de enseñanza, replicadas por ciertas características del diseño del documento de Internet, y los logros de los estudiantes. Entre tales estrategias de enseñanza se incluyen:

1. Provisión de guía efectiva
2. Ajuste de los contenidos a las necesidades de los alumnos
3. Promoción de la práctica intensiva a través de tareas significativas
4. Fomento de colaboraciones de clases
5. Creación conjunta del entorno de aprendizaje por el profesor y los alumnos

Como estrategias del documento de Internet están:

1. Simplicidad de estructura y facilidad de navegación
2. Originalidad de los contenidos
3. Sentido de finalidad de los proyectos requeridos
4. Variedad de oportunidades de colaboración electrónica
5. Colaboración del profesor y los alumnos en el desarrollo del documento WWW

Una aportación interesante del estudio de la Dra. Isabel Borrás, acerca de la enseñanza y aprendizaje con Internet es que es necesario contar con más estudios que demuestren la

eficacia de entornos educativos en donde Internet sea visto no como un mecanismo para enseñar, sino como un mecanismo para aprender.

Si se considera que existe un rezago acerca del conocimiento teórico y práctico sobre la utilización pedagógica de la tecnología, con respecto a la tecnología misma, entonces se entiende como el uso actual del Internet es imitar las prácticas educativas existentes.

Herramientas a utilizar

La herramienta seleccionada para el análisis de la aplicación es Rational Rose 2000 Enterprise Edition, dado el manejo del lenguaje de UML, como herramienta de desarrollo se seleccionó FLASH 5 de Macromedia. Flash es una aplicación orientada principalmente a la creación de películas, clips o videos animados para sitios de Internet. Estos consisten principalmente en gráficos vectorizados, pero pueden contener imágenes de mapa de bits y sonidos. Una característica interesante de Flash es su capacidad para incorporar interactividad que permita entrada de información por parte de los usuarios.

El producto final de Flash puede ser un video o animación que será visto a través de un navegador de Internet, o bien, una aplicación independiente que puede ser ejecutada mediante el Macromedia Flash Player.

La razón de la elección de Flash es principalmente por la flexibilidad que tiene con respecto a su lenguaje script, que permite controlar objetos en las películas de flash, controlar la navegación y los elementos de interacción, tanto en las películas como en las aplicaciones web. Actionscript, el lenguaje de flash, está basado en javascript, pero aunque existen algunas diferencias que evitan que sean totalmente compatibles, es importante mencionar que actionscript es un lenguaje orientado a objetos, por lo que el manejo de clases, objetos, propiedades y métodos es totalmente soportado. Actionscript puede ser utilizado para incluir interactividad en juegos en línea o en programas instructivos.

La aplicación deberá poder ser ejecutada en la mayoría de los navegadores de Internet que sean compatibles con los estándares del mercado, para tal efecto, se considerarán los plug-ins y programas adicionales que sean requeridos para lograr tales fines.

Metodología

Se utilizará la metodología utilizada por el RUP[5] o Rational Unified Process, que es un proceso de ingeniería de software que mejora la productividad de los grupos de trabajo y permite la utilización de lo que se conoce como mejores prácticas, a través de guías y plantillas utilizadas en las actividades de desarrollo de software crítico.

Descripción de RUP

La metodología RUP facilita la elección del conjunto de procesos adecuados según las necesidades del proyecto, tomando como principio que se lograrán resultados mejores y más predecibles si se unifican los procesos comunes que mejoran la comunicación y logran el entendimiento común de todas las tareas y responsabilidades involucradas.

A pesar de estar altamente orientada a la eficiencia en proyectos de equipo, esta metodología ha sido diseñada para ser utilizada en cualquier tipo de aplicación.

Si se expresa en términos de modelado de negocios, el proceso del desarrollo de software es un proceso de negocio, RUP es un proceso de negocios genérico para ingeniería de software orientada a objetos. RUP tiene como función asegurar la producción de software de alta calidad que cumpla con las necesidades de los usuarios finales, considerando el presupuesto y tiempo estimados, de manera adicional. RUP utiliza las mejores prácticas en el desarrollo de software moderno de tal manera que puede ser considerado como un lenguaje de modelado personalizado a la medida de una gran variedad de proyectos y organizaciones.

Para esta metodología, un proceso es un conjunto de pasos parcialmente ordenados, que tienen la intención de alcanzar un objetivo, en ingeniería de software, el objetivo se convierte en un producto de software, o en la mejora de un software existente, en la ingeniería de procesos, el objetivo es desarrollar, o mejorar un proceso. Para la metodología RUP, los procesos se organizan en disciplinas para definir el flujo de trabajo y otros elementos del proceso.

[5] Información obtenida del sitio de la compañía Rational Software Corporation que desarrolla y distribuye la herramienta RUP suite en http://www.rational.com/

Figura 14. Proceso de ingeniería de software propuesto por la metodología RUP

UML, la notación de RUP

La metodología RUP utiliza UML (Unified Modeling Language), que es una notación que puede ser aplicada tanto en ingeniería de sistemas como en ingeniería de negocios. La utilización de una notación estándar tiene entre otras, las siguientes características importantes:

- Sirve como lenguaje para comunicar decisiones que no son obvias o que no pueden ser inferidas por el código mismo.
- Provee semántica lo suficientemente rica para capturar las decisiones estratégicas y tácticas importantes.
- Ofrece una forma lo suficientemente concreta, que puede ser razonada por los humanos y manipulada por las herramientas.

UML representa la convergencia de la mejor práctica de modelado de software en la industria de la tecnología de objetos.
La notación UML es definida como un lenguaje para visualizar, especificar, construir y documentar los artefactos de los sistemas de software, así como para modelado de negocios y otros sistemas no de software. La UML representa una colección de las mejores prácticas de ingeniería que tienen probado éxito en el modelado de sistemas grandes y complejos.

El propósito del modelado previo al desarrollo del sistema de software es de vital importancia, los buenos modelos son esenciales para la buena comunicación entre los equipos de proyectos y para asegurar la firmeza arquitectónica. Conforme se incrementa la complejidad de los sistemas, también se incrementa la importancia del uso de buenas técnicas de modelado. A pesar de ser

tantos los factores que influyen en el éxito de un proyecto, es notorio que el uso de un lenguaje de modelado estándar es sumamente esencial.

Antes de UML, no existía un lenguaje de modelado que fuera claramente líder. Los usuarios tenían que escoger entre diferentes lenguajes de modelado con diferencias menores, ya que la gran mayoría de estos lenguajes compartía un conjunto de conceptos comúnmente aceptados.

UML está basado en los lenguajes OMT, Booch y OOSE, entre otros importantes lenguajes. Cualquier persona con conocimiento de cualquiera de estos lenguajes tendrá muy poca dificultad en el uso de UML, y los beneficios de utilizar un lenguaje basado en el estándar de la industria compensan ampliamente la inversión de tiempo que el aprendizaje requiere.

Entre los beneficios de UML resaltan:

- Reducción significante del costo de capacitación y organización, dado el caso de cambio de proyectos u organizaciones.
- Provee la oportunidad de nueva integración entre herramientas y procesos.
- Permite a los desarrolladores enfocarse en el valor del negocio, ya que proporciona un paradigma que permite lograr este valor.

Finalmente, es importante resaltar que UML sigue en constante análisis y evolución, gracias al trabajo del *Object Management Group*[6], que ha establecido un grupo de trabajo que atiende cualquier revisión necesaria al UML, por mínima que ésta sea.

Descripción General de la Aplicación Propuesta

La aplicación que se propone con base en los resultados del análisis, deberá contar con las siguientes funcionalidades:

En la página de Internet en donde se ejecute la aplicación, deberá existir una sección informativa en donde se haga mención a la parte teórica relacionada con las estructuras de datos y en particular con las abstractas, poniendo especial énfasis en los diferentes tipos de árboles binarios de búsqueda, objeto de estudio de este documento, así como enlaces a sitios de interés. Junto con la información teórica, se deberán considerar imágenes que den una idea clara de las

[6] Mayor información sobre la labor que este organismo realiza con relación al establecimiento y mantenimiento de estándares relacionados con la industria de cómputo puede ser encontrada en el sitio de Internet en la dirección http://www.omg.org/

representaciones de los árboles, y que permitirán entender con mayor facilidad conceptos clave, tales como las rotaciones y otras operaciones.

La aplicación en la página de Internet deberá ser amigable y fácil de utilizar, y sobre todo deberá ser clara y explicativa, dependiendo de la opción que se haya seleccionado para cada tipo de ABB en particular.

Descripción de problemas y oportunidades

Problemas

- La falta de aplicaciones visuales o con animaciones gráficas auxiliares en el aprendizaje de estructuras de datos, en particular de los árboles binarios de búsqueda, orientadas a estudiantes en proceso de aprendizaje de tales estructuras de datos, que además incluyan información detallada, que explique los diferentes pasos que se realizan al ejecutar las operaciones correspondientes a las estructuras de datos.
- Inestabilidad de las aplicaciones realizadas con JAVA. A pesar de la potencia, grandes ventajas de Java, y de la existencia de bibliotecas con estructuras de datos implementadas, en los casos analizados se encuentran problemas de estabilidad en las computadoras en donde se ejecutaron los applets, ya que las animaciones fueron probadas en diferentes versiones de navegadores y con diferentes versiones de sistemas operativos. De hecho en uno de los casos analizados, el autor sugiere descargar la aplicación y compilarla, en caso de incompatibilidad.

  Sin embargo no podemos generalizar y decir que las aplicaciones de Java presentan inestabilidad y errores en todos los casos o en todas las aplicaciones, un factor importante que influye en la estabilidad de la aplicación es la versión de Java en que éstas son compiladas.

Oportunidades

- La existencia de aplicaciones como “Flash”, de Macromedia, que además de contar herramientas para el manejo natural de elementos gráficos auxiliares en el desarrollo de la parte visual, tiene incorporado Actionscript, un lenguaje orientado a objetos, basado en java-script.
- La utilización del player de Flash puede ser visto como inconveniente por la necesidad de contar con él para ejecutar la animación, o bien como ventaja, dado que puede eliminar los problemas de incompatibilidad encontrados en algunas de las aplicaciones.

- La posibilidad de difundir la aplicación, una vez creada, a través de sitios de Internet[7] de instituciones educativas y de otros organismos dedicados a publicar y difundir información y aplicaciones relacionadas con las estructuras de datos y otras materias del área de computación y tecnología.

Descripción de la problemática de la representación de árboles

La representación de los árboles en una aplicación resulta un problema importante a resolver en la medida que los árboles se deban generar de manera aleatoria por el usuario, es decir, mediante interacción con él. Es necesario pensar en el gráfico que se utilizará para representar los nodos, pero el problema radica en que la estructura generada no es de tamaño uniforme porque en dependencia de la cantidad de nodos del árbol, este tendrá que ser de un tamaño tal que permita visualizar correctamente la estructura como un todo, además el contenido de cada nodo. Esta es la razón principal por la cual la mayoría de las aplicaciones limitan el número de nodos que puede tener un árbol.

La aplicación que se sugiere construir no es una aplicación práctica de utilización de árboles binarios de búsqueda, sino es de índole educativa, por lo que también utilizaremos la técnica de restringir el número de nodos a considerar.

Definición de Metas y Objetivos

Meta

Proporcionar la información y fundamentos necesarios para el desarrollo de un sistema agradable, fácil de usar y atractivo, que permita a los estudiantes y profesores contar con una herramienta auxiliar en el proceso de aprendizaje de estructuras de datos, en particular de los árboles binarios de búsqueda, mediante el uso de algoritmos probados y confiables, así como de animaciones que tendrán la función de apoyar en el proceso de aprendizaje.
Demostrar la factibilidad del uso de una herramienta de diseño gráfico y de animaciones como Flash para el desarrollo de una aplicación como la aquí mencionada, que además utilice las mejores prácticas, como programación orientada o objetos y un diseño y modelado realizado con RUP.

[7] Es importante mencionar que al comienzo de este análisis, los sitios de Internet de 2 de las 4 principales aplicaciones evaluadas en este estudio no contaban con todas las opciones de detalle e información con que cuentan ahora, ya que en ambos casos las recientes actualizaciones se han llevado a cabo en los meses previos a la entrega de este documento. Además de la mencionada falta de sitios con información en español.

Objetivos

- Que el sistema incluya una opción de “Tutorial” que paso a paso explique a los usuarios el proceso en el que se encuentra la estructura, y las restricciones que se deben cumplir para cada TDA.
- Demostrar mediante elementos gráficos el comportamiento de los árboles binarios de búsqueda al realizar las principales operaciones (inserción, eliminación y búsqueda).
- Difundir la aplicación a la comunidad universitaria internacional, para su análisis y crítica, a través de Internet.

Criterios de evaluación

Dos de los elementos más importantes a considerar en la evaluación del análisis son los resultados mismos derivados de la revisión de las aplicaciones incluidas en este documento, así como el diseño derivado de eso resultados.

Opciones de solución

De manera inicial se consideró el desarrollo de la aplicación en Delphi, por la potencia del lenguaje y la facilidad y naturaleza para el manejo de elementos en la programación orientada a objetos, y por la facilidad para crear aplicaciones para Internet[8]. Dada la importancia de la parte visual o gráfica de la aplicación, se consideró el uso de Flash 5, de Macromedia, que a pesar de no ser una herramienta meramente de programación como Delphi, o de no tener la potencia e importancia que tiene JAVA, tiene ventajas y elementos importantes que se han mencionado anteriormente que en definitiva pueden ser de utilidad. De tal manera que la decisión final se inclinó sobre el Flash 5.

La gran mayoría de las aplicaciones existentes han sido desarrolladas en Java. Algunas de estas aplicaciones tienen problemas de compatibilidad por la versión de las JVM que han sido utilizadas para compilarlas, a pesar de este pequeño inconveniente, Java resulta por demás una opción interesante, ya que muchos de los algoritmos que podrían ser utilizados en esta demostración están disponibles ya sea en C o en Java, sin embargo, puede de igual manera resultar interesante y retadora la opción de Flash, por el potente manejo de gráficos, y al mismo

[8] Se pretende que la aplicación sea ejecutada a través de un navegador de Internet, que facilitaría la propagación de la misma, mediante su inclusión en los sitios web de las universidades y otros sitios con ligas relacionadas.

tiempo, la facilidad del manejo de su lenguaje propietario “actionscript”, basado en javascript y orientado a objetos, que en teoría, debería permitir el uso de clases de manera transparente.

A continuación se presenta un análisis de las aplicaciones seleccionadas para esta comparación, en donde se pone especial énfasis en la revisión del código fuente:

a) Descripción y Análisis de la Codificación Del Programa ‘Avl Tree Demonstration” por Arsen Gogeshvili.

El sistema fue implementado, como ya se mencionó anteriormente en este documento, en Java, para efectos de este trabajo, el análisis de la programación, se centra en la implementación de los algoritmos necesarios para la operación general de árbol, por lo que no prestamos particular atención a la programación utilizada para las animaciones de cada operación.

El código estudiado está basado en de tres clases principales:

BSTREE

Contiene la estructura general del árbol, y en esta clase se definen los métodos y los algoritmos necesarios para el manejo de los tipos de árbol BST, AVL y SPL (Splay)

Las propiedades de esta clase contienen tanto datos simples, como objetos de la clase BTNode, descrita más adelante. Esto le permite contar con un árbol, virtualmente sin límite de nodos, ya que se pueden ir creando objetos dinámicamente conforme se vayan insertando valores, o bien eliminándolos.

Enseguida se muestra una descripción de los métodos utilizados

MÉTODO	DESCRIPCIÓN
METODOS PARA ÁRBOLES ABB	
BSTree	Es el constructor de la clase, se utiliza para inicializar la raíz del árbol como nula.
Locate	Regresa el nodo localizado o nulo si no se encuentra, guarda el último nodo en el camino de la búsqueda y el lado del siguiente nodo, utilizado por otros métodos.
Add	Crea una hoja y la anexa al nodo especificado en el lado especificado, el método puede ser utilizado en un árbol vacío.
Remove	Elimina el nodo del árbol, regresa el padre del nodo eliminado.
Destroy	Contiene una única línea que marca ‘node’ como nulo.
Swap	Si el nodo tiene dos hijos, intercambia los nodos antes de eliminar (no aplica para árboles splay)
SwapData	Método auxiliar para swap

Rotate	Rota las ligas de los nodos participantes y regresa el nodo que queda arriba después de la rotación, el 'node' es rotado hacia abajo y hacia el 'side' especificado
Link	Este método es usado por los métodos add(), remove() y rotate(), maneja apuntadores nulos y actualiza el apuntador hacia la raiz cuando es necesario
METODOS PARA ÁRBOLES AVL	
InsertAVL	Inserta el dato en el árbol AVL, regresa el nuevo nodo o nulo si el dato ya existe
DeleteAVL	Elimina el dato del árbol AVL, regresa el padre del nodo eliminado o nulo, si el dato no se encontró. Usa el método auxiliar swap() si es necesario.
RebalanceAVL	Realiza un rebalanceo del árbol después de insertar o eliminar, utilizando rotación, si es necesario.
RotateAVL	Es utilizada por el rebalanceo posterior a insertar o eliminar nodos, realiza una rotación y actualiza los factores de balanceo. Regresa el nodo superior del sub-árbol rotado.
MÉTODOS PARA ÁRBOLES SPLAY	
Find	Intenta encontrar un dato, si es el caso, extiende el ultimo nodo hacia la raiz, si no se encuentra, regresa 'FALSO'
InsertSPL	Expande el último nodo a la raiz independientemente de los resultados de la búsqueda, despliega el árbol e inserta la nueva raiz.
AddSPL	Crea una nueva raiz, despliega el árbol y anexa los subárboles a la nueva raiz
DeleteSPL	Elimina un nodo
Splay	Expansión de abajo hacia arriba, cada rotación de los ancestros sube el nodo, el mantenimiento de los apuntadores no es necesario.

BTNode

Define la estructura de datos de un nodo del árbol, provee una colección de métodos para proteger los datos. Todos los algoritmos de balanceo están contenidos en la clase BSTree. Presenta datos inherentes en forma particular a un nodo, como los apuntadores a su padre, y a sus hijos izquierdo y derecho. Contiene un objeto de la clase BTData, que contiene el valor de la llave o etiqueta del nodo.

Métodos utilizados por esta clase:

MÉTODO	DESCRIPCIÓN
BTNode	Es el constructor de la clase, inicializa todas las variables como nulas.
GetBalance	Obtiene los factores de balance del árbol
SetBalance	Establece los factores de balance del árbol
GetColor	Obtiene el color
SetColor	Establece el color
GetData	Obtiene la etiqueta
SetData	Establece la etiqueta o llave
GetParent	Obtiene el nodo padre

SetParent	Obtiene el nodo hijo
Getchild(int)	Obtiene el hijo, según el lado indicado en el parámetro
GetChild	Obtiene el primer hijo que se tenga.
SetChild	Establece los valores para el hijo
GetSide	Define en que lado del padre se encuentra un nodo dado
PrevInO	Regresa el nodo previo de un árbol leido en In- Orden
NextIn0	Regresa el siguiente nodo, si se lee en In-Orden
FirstInO	Obtiene el primer nodo leido en Inorden
LastInO	Obtiene el ultimo nodo leído en Inorden
CompareTo	Compara los valores de los nodos
HasTwoChildren	Regresa cierto si el nodo tiene dos hijos
IsLeaf	Regresa cierto si el nodo es una hoja

BTData

Esta clase ayuda a encapsular los datos. Provee un método para compararlos.

Los métodos implementados son:

MÉTODO	DESCRIPCIÓN
BTData	Constructor de la clase, inicializa la llave como nula.
BTData(string)	Constructor, se llama si se pasa al inicializar, la llave del primer nodo.
GetKey	Lee la etiqueta del objeto
SetKey	Establece el valor de la etiqueta del objeto
CompareTo	Compara las etiquetas de dos nodos.

Ventajas de la implementación

Desde el punto de vista didáctico, el código fuente es muy entendible y ordenado, así como claramente documentado y comentado. Se muestran de una manera muy explícita los diferentes algoritmos propuestos, para la implementación de las operaciones del árbol. Sin embargo, se detecta que en algunas partes podría ser optimizado.

Desventajas o mejoras detectadas

Aunque se menciona en el código fuente, que los métodos descritos referentes a los diferentes árboles, no se corresponden entre sí, y por lo tanto deben ser implementados en diferentes módulos (o clases), es interesante mencionar que en general, los métodos que coinciden para todos los tipos de árboles podrían ser implementados en el mismo módulo, explotando la propiedad de polimorfismo de la teoría de orientación a objetos, esto nos daría la posibilidad de armar, en una sola solución, y en forma transparente para la interfaz lograda, un sistema en que el estudiante, en forma dinámica pueda seleccionar el tipo de árbol que desea estudiar y el sistema se comporte en la manera correcta según esta especificación, facilitando enormemente la programación, ya que se cargaría la función correspondiente, según los parámetros pasados, con el mismo nombre de función. Por ejemplo la función add(), la función insertAVL() e insertSPL() podrían fusionarse en una función AgregaNodo(), para cualquier tipo de árbol. Sin embargo en la clase BTNode se puede notar que si se utilizó este recurso (polimorfismo), para la función GetChild. Del mismo modo, el constructor de la clase BTData, es polimórfico y obedece a la creación de un objeto sin etiqueta, o bien si se pasa esta como parámetro, lo crea con la misma.

Se considera que la clase BTData no tiene mucho sentido, ya que aunque presenta la ventaja de encapsular y proteger los datos del nodo, es obvio que por cada objeto BTNode, se deberá contar con un objeto BTData, esto es, se trata de una relación de uno a uno. Se considera que este encapsulamiento se puede implementar sin ningún problema en la misma clase BTNode (incluso, los datos de BTData están considerados en los datos de BTNode), simplificando la programación, y eficientando el uso de la memoria, al disminuir el número de objetos creados y manipulados.
Otra forma de optimizar el código, sería utilizar la herencia, y generar una clase genérica de árbol, con los atributos y métodos comunes para cualquier árbol y heredarlos hacia las clases hijas, que contendría los atributos y métodos particulares para cada tipo de árbol.

b) Descripción y Análisis de la Codificación Del Programa 'Avl Animated Tree" por John Kloss

El código de esta implementación, es bastante más extenso que la anterior, ya que en esta se incluye toda la parte gráfica y de animación. Para efectos de este trabajo, esta parte será ignorada, y se describe la parte algorítmica.

Esta implementación está más complicada en el sentido de entendimiento de los algoritmos, y de la lógica de programación utilizada por el autor. En esta solución, existe una serie de clases

interrelacionadas y se ve a la funcionalidad de la aplicación como objetos, por lo que no es sencilla su comprensión. Las clases son públicas, debido a esa interrelación y a que las clases pueden estar contenidas en módulo o unidades diferentes.

Por otro lado, las clases definidas hacen uso del concepto de Interfaz, ofrecida por el lenguaje Java, que consiste en un conjunto de variables estáticas y de funciones que pueden ser implementadas por una (o mas clases). Esto permite reusar el código común a varias clases, sin necesidad de duplicidad. Nota: es importante recordar que este concepto es diferente de 'herencia', ya que los atributos y métodos son usados, mas no heredados, por la clase que los implementa. Las interfaces implementadas por las clases son: Observer, Runnable y Orderable.

A continuación, revisaremos las principales clases relacionadas con el funcionamiento de los árboles, ya que la solución, como se puede apreciar en el siguiente diagrama en el que se muestra una porción de la estructura de clases, es mucho más compleja que otras soluciones encontradas.

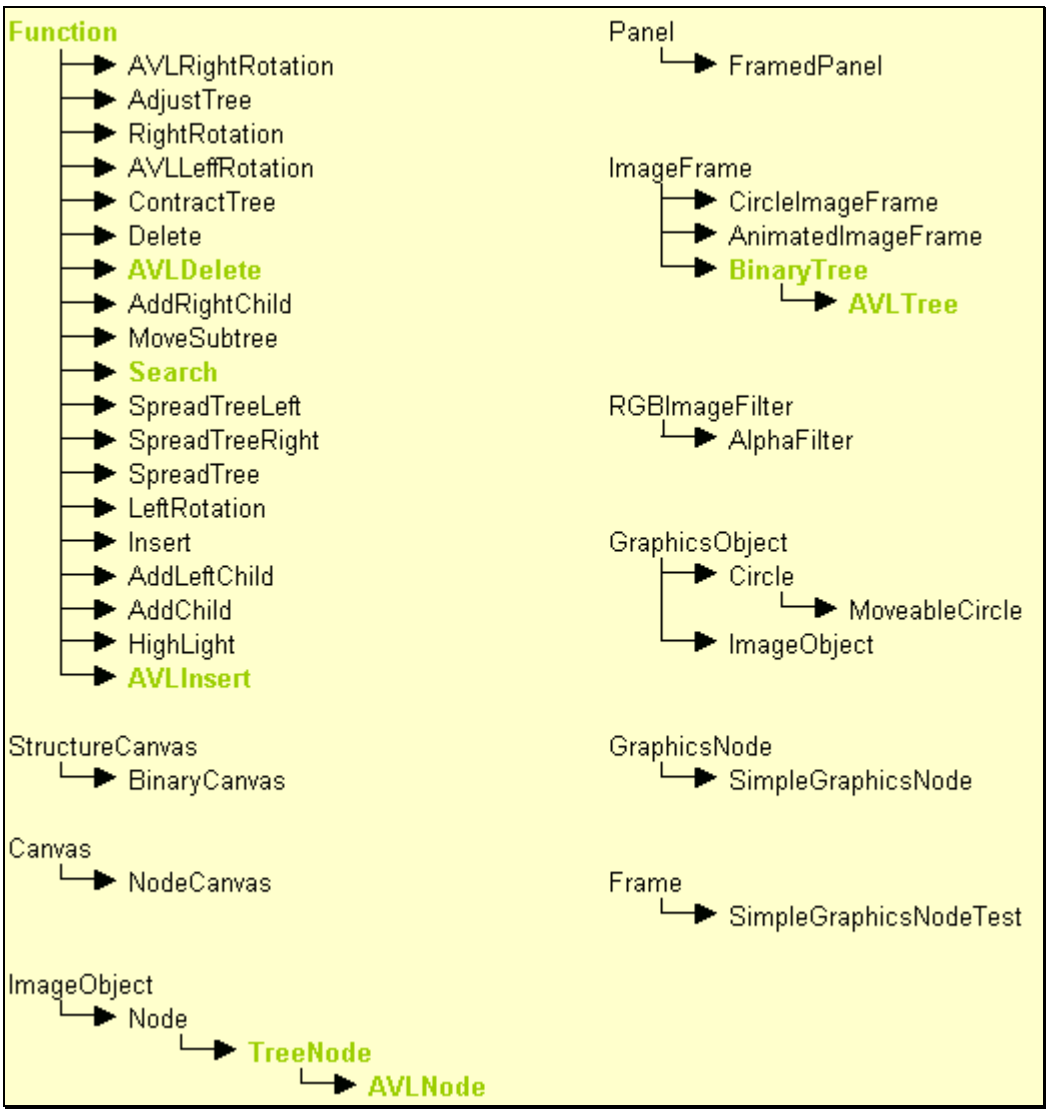

Figura 15. Diagrama de clases utilizado por la aplicación 'Avl Animated Tree"

Las clases marcadas en color verde, se consideran las más relevantes para ser objeto de análisis y comparación con otros programas.

BinaryTree

Esta clase contiene los atributos y métodos necesarios para implementación del árbol (Insertar, eliminar y buscar), hereda a su vez de la clase ImageFrame. Es importante mencionar que se refiere mayormente a las funciones de representación gráfica del árbol, y llama a los algoritmos de operación. Sus métodos son:

MÉTODO	DESCRIPCIÓN
CreateBaseNode	Crea un nodo base, que será complementado con las características de las clases hijas, según el tipo de árbol de que se trate.
CreateHighLight	Realiza el marcado de aquellos nodos involucrados en una operación del árbol.
CreateNewNode	Crea un Nuevo nodo en el árbol , relacionado con la inserción. Dibuja el nodo en la pantalla.
Init	Inicializa el applet, dibuja los botones y crea los frames de operación del mismo.
UpdateLabel	Actualiza la etiqueta del nodo.
Action	Atiende a la acción que se haya generado según el botón utilizado, llamando a los procedimientos padres para cada una de ellas

AVLTree

Esta clase hereda de la clase superior BinaryTree, y encapsula las características propias de este árbol. Los métodos implementados para esta clase son:

MÉTODO	DESCRIPCIÓN
INIT	Llama a la función super.INIT, de la clase BinaryTree
Action	Genera los objetos de tipo funcion pertinentes para llevar a cabo las operaciones como objetos de las clases AVLInsert, Search y Delete

TreeNode

Esta es una clase hija de la superclase Node, de cuyo constructor tiene una copia casi exacta. Maneja un atributo ‘rank’ que es la característica que permite ordenar linealmente todos los nodos del árbol, de tal forma que cada uno de ellos puede ser accesado por este índice en todo momento. Este rango es establecido hasta que la posición final del nodo es determinada.

Se describen los métodos relacionados con las operaciones del árbol (se suprimen las que se refieren a la representación gráfica):

MÉTODO	DESCRIPCIÓN
TreeNode	Constructor raiz
TreeNode	Constructor de copia. Este crea una copia casi exacta de TreNode pasado

	como parámetro, sin embargo las referencias al padre y al hijo no son copiadas.
TreeNode	Constructor nulo. Este construye un TreeNode abstracto, es decir, no tiene representación gráfica. Es utilizado principalmente para las operaciones gráficas que requieren una ubicación en el contexto gráfico, pero que no requieren una representación física.
GetRank	Obtiene el valor de Rank. Si este no ha sido determinado regresa un valor negativo, lo cual significa que ha habido un error
SetRank	Establece el valor de Rank. Este método establece recursivamente los valores de Rank de los hijos también (en caso de que los tenga)
SetLeftChild	Estable la liga al hijo izquierdo (este valor puede ser nulo)
GetLeftChild	Obtiene el hijo izquierdo, puede ser nulo u otro nodo.
SetRightChild	Establece la liga al hijo derecho (este valor puede ser nulo)
GetRightChild	Obtiene el hijo derecho, puede ser nulo u otro nodo.
SetParent	Establece el valor del nodo padre
GetParent	Obtiene le valor del padre del nodo actual.
GetLeastChild	Obtiene el nodo con el valor más pequeño en el sub-arbol actual. Si la raiz del subárbol no tiene hijo izquierdo, entonces, se devuelve el valor de la raiz, por ser el nodo con menor valor.
GetGreatestChild	Obtiene el nodo con el valor mayor del sub-árbol actual. Si la raíz no tiene hijo derecho, regresa el valor de la raíz, por ser el nodo con mayor valor.

AVLNode

Esta clase es hija de TreeNode, y básicamente se definen aquí los datos referentes al balanceo del sub-árbol del cual un nodo es la raíz.

MÉTODO	DESCRIPCIÓN
SetBalanceFactor	Establece el factor para balanceo del árbol.
AdjustBalanceFactor	Ajusta el factor para el balanceo del árbol.
AVLNode	Constructor nulo. Este construye un AVLNode abstracto, es decir, no tiene representación gráfica. Es utilizado principalmente para las operaciones gráficas que requieren una ubicación en el contexto gráfico, pero que no requieren una representación física.
AVLNode	Constructor del AVLNode, establece el factor de balanceo.
AVLNode	Construye la instancia de acuerdo a su origen.

AVLInsert

Esta clase hereda de la superclase Function, y contempla las funciones necesarias para realizar las operaciones relacionadas con el Árbol AVL.

Sus métodos son:

MÉTODO	DESCRIPCIÓN
AVLInsert	Crea el objeto Insert de la clase BinaryTree e inicializa sus valores
Update	Actualiza los valores y posiciones del nodo, después de la inserción
PerformFunction	Ejecuta el objeto de la función de inserción, actualiza la raiz (si es el caso), y ajusta el factor de balance.

AVLDelete

Esta clase se refiere a las operaciones necesarias para efectuar una eliminación de un nodo dentro del árbol. Cumple con los tres casos que se pueden suscitar al realizar una eliminación (dependiendo de si tiene hijo derecho, izquierdo, o ninguno)

Se vale de los siguientes métodos para funcionar:

MÉTODO	DESCRIPCIÓN
AVLDelete	Constructor de la clase, inicializa los valores para la representación gráfica.
Update	Actualiza los valores de los nodos después de la eliminación
PerformFunction	Crea y ejecuta los objetos necesarios para realizar la eliminación.

Search

Esta clase implementa la búsqueda de nodos dentro del árbol, las funciones se crean como instancias de la clase y son ejecutadas por un método de las mismas. Es una clase derivada de la superclase ‘Function’. Es una implementación muy sencilla en sí, ya que sólo cuenta con dos métodos, uno para inicializar

Los métodos programados para esta clase son:

MÉTODO	DESCRIPCIÓN
Update	Inicializa el valor de bloqueado como falso.
PerformFunction	Crea y ejecuta los objetos necesarios para realizar la búsqueda.

Ventajas de la implementación

Esta solución representa, en definitiva, un despliegue de la explotación de la herencia y un modelo de programación orientada a objetos, en su máxima expresión. Si bien no es una aplicación sencilla de entender, una vez establecida la relación entre las 47 clases que lo componen, resulta interesante la lógica de programación de su autor, ya que concibe las operaciones que se pueden realizar en los árboles, como objetos dentro del sistema, que tienen el atributo de poder ser ejecutados.

Otra ventaja importante es la implementación de los recursos ofrecidos por la biblioteca JDK para la implementación de estructuras de datos en general, así como para las operaciones gráficas, lo que hace que su autor aproveche y se valga de esa oportunidad, en lugar de programar estructuras que ya existen.

Desventajas o mejoras detectadas

El programa final incrementaría su valor (sobre todo didáctico), si se comentara a todo lo largo del mismo, cuando menos la versión pública del código fuente está comentado sólo en algunas partes, lo que dificulta la identificación de los objetivos de cada una de las unidades. Otra desventaja es que cada una de las clases está definida en un archivo físico .java diferente. Esto es un requerimiento de Java para poder tener acceso a las clases públicas, sin embargo no favorece la comprensión del código, así como la eficiencia al momento de compilación del applet, al tener que importar unidades en cada una de las clases.

La lógica de programación, si bien se menciona como ventaja, abusa en cierto sentido del concepto de herencia en forma innecesaria, ya que en algunos casos, aunque una clase hereda de otra, en ningún momento utiliza los recursos heredados para su funcionamiento.

Por último, en este programa, se identifica un ‘bug’ que consiste en que el árbol no se rebalancea como debe hacerlo, en el caso de que se produzca la eliminación de un nodo, situación reconocida incluso por su mismo autor.

c) Descripción y análisis de la codificación del programa 'Red Black Tree Demonstration" por Michael Conrad y John Franco

Esta implementación trabaja sobre los algoritmos del árbol binario de búsqueda del tipo "Rojo-Negro", utiliza interfaces que personalizan el tipo de objeto que será almacenado en el árbol e implementaciones de ellas para objetos simples enteros. Todos los objetos deben implementar la interfaz TreeObject quien requiere identidad y valor (por efectos de orden). Para la comparación de objetos, las interfaces RBTree_inorder_class y RBTree_compara_class deben ser implementadas.

La versión estudiada, lee 14 números de una lista de parámetros e instancia los objetos con esos números en el árbol rojo-negro, durante el arranque del applet.

Finalmente, los objetos TockenObjects son utilizados simplemente para facilitar las operaciones basadas en el uso del mouse del applet y no son parte del concepto de árbol Rojo-Negro.

Utiliza en total 15 clases, representadas en el siguiente cuadro, según su jerarquía.

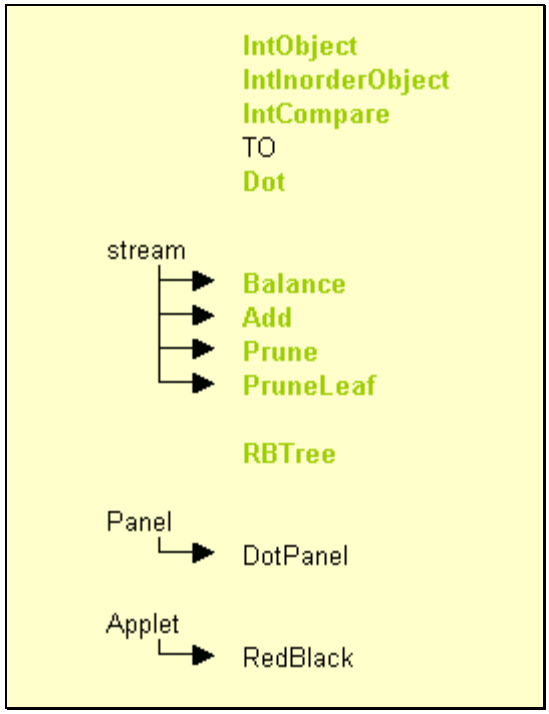

Figura 16. Diagrama de clases utilizado en la aplicación 'Red Black Tree Demonstration"

Las clases marcadas en color verde, se consideran las más relevantes para ser objeto comparación con otros programas.

Una vez más, aquellas clases utilizadas para el manejo de la representación gráfica del árbol, serán ignoradas y no serán objeto de análisis para efectos del cumplimiento del objetivo de este trabajo. No obstante lo anterior, se menciona en este punto por ser de interés para el resultado final de la funcionalidad de la aplicación, que la clase STREAM contempla los algoritmos necesarios para interrumpir una clase que la implementa, con el fin de explicar paso a paso, los eventos ocurridos con una operación y desplegar los cambios intermedios ocurridos en el árbol.

IntObject

Administra objetos enteros simples. Sus métodos son:

MÉTODO	DESCRIPCIÓN
getValue	Regresa la etiqueta del objeto
GetIdent	Regresa el identificador del objeto

IntInorderObject

Este método compara dos objetos del árbol y determina cuál tiene el menor valor.

MÉTODO	DESCRIPCIÓN
Compare	Compara los valores pasados y regresa el menor.

IntCompare

Es una segunda versión de comparación que compara el número de orden de un objeto contra un valor guardado previamente.

MÉTODO	DESCRIPCIÓN
Evaluate	Compara el valor del objeto y el valor guardado, pasados como parámetro y regresa 1 si el valor es mayor que el objeto y -1 en caso contrario.

Dot

Provee la infraestructura de mantenimiento de objetos de datos interesantes y envío de información. Contiene los siguientes métodos:

MÉTODO	DESCRIPCIÓN
Dot	Constructor de la clase. Inicializa los valores del objeto.
GetPrev	Encuentra el siguiente objeto almacenado de menor orden. Regresa null si no es encontrado. Se utiliza cuando se elimina un nodo, para encontrar cuál nodo deberá tomar su lugar en el árbol.
GetNext	Encuentra el siguiente objeto almacenado mayor. Regresa nulo si no existe. Utilizado cuando se elimina un nodo, para encontrar cuál nodo deberá tomar su lugar en el árbol.
RightRotate	Operaciones de rotación, utilizadas al insertar o eliminar nodos del árbol.
LeftRotate	
LeftSide_LeftRotate	
LeftSide_RightRotate	
RightSide_RightRotate	
RightSide_leftRotate	

Balance

Esta clase se utiliza cuando se agrega un nodo, para efectos de garantizar el balance del árbol después de la inserción.

MÉTODO	DESCRIPCIÓN
Balance	Constructor de la clase. Inicializa los valores del objeto.
Run	Analiza y determina el color de los nodos, después de agregar un nodo, balanceando el resultado

Add

Como su nombre lo indica, se encarga de realizar la operación de adición de un nodo al árbol rojo-negro

MÉTODO	DESCRIPCIÓN
Add	Constructor de la clase. Inicializa los valores del objeto.
Run	Analiza y determina el color de los nodos, después de agregar un nodo, balanceando el resultado

Prune

Utilizada para remover un objeto del árbol rojo-negro. Su implementación consta de los siguientes métodos.

MÉTODO	DESCRIPCIÓN
Prune	Constructor de la clase. Inicializa los valores del objeto.
Run	Analiza y determina el color de los nodos, después de realizar la operación.

PruneLeaf

Se utiliza para realizar “pruning” de nodos que contienen cuando mucho un hijo. Es utilizada por la clase Prune.

MÉTODO	DESCRIPCIÓN
Prune	Constructor de la clase. Inicializa los valores del objeto.
Run	Analiza y determina el color de los nodos, después de realizar la operación.

RBTree

Establece las características generales del árbol rojo-negro, es la clase principal de la aplicación.

MÉTODO	DESCRIPCIÓN
RBTree	Constructor de la clase, inicializa los valores de las instancias.
CopyTree	Copia un subárbol a una nueva posición
SetDots	
ReLevel	Establece el nivel de un nodo, pasando como parámetro los valores del índice y el nivel.
Level	Regresa el nivel de un subárbol.

Ventajas de la implementación

La solución aplica los conceptos de orientación a objetos en forma moderada, sin excesos, como el caso de estudio de Kloss. Los algoritmos son implementados en forma clara, y ordenada, sin presentar recovecos oscuros.

Desventajas o mejoras detectadas

La solución no se encuentra documentada en su totalidad. Para funcionar, requiere que se reciban como parámetro una serie de números, que armarán un árbol inicial siempre igual, si se envían en forma fija y explicita desde la página que llame al applet. Se podría construir una página que preguntara en forma dinámica los nodos iniciales y con ello armar el árbol durante el inicio, sin embargo es una solución externa al applet y requiere otro tipo de programación.
En general las mejoras en esta solución, podrían residir en la funcionalidad de la misma, más que en la programación.

Como se observó en las aplicaciones analizadas, existen diferentes formas de implementar los algoritmos para representar ABB. Una diferencia importante que vale la pena resaltar, es el hecho de que en algunos casos las operaciones de los ABB se implementaron como métodos dentro de la clase, y en otros casos esas mismas operaciones se consideran como clases independientes.

Dado el objetivo de la aplicación resultante como objeto de este análisis, se considera que para fines didácticos resulta de mayor facilidad para el estudiante analizar y entender el código de estas aplicaciones, cuando las operaciones de las clases han sido implementadas como métodos y no como una clase independiente, esto considerando que el análisis de tales aplicaciones puede servir a dos propósitos, primero, conocer acerca de los algoritmos utilizados para implementar los ABB y segundo, analizar las implementaciones desde el punto de vista de programación.

Capítulo III

Una vez realizado el análisis de las aplicaciones, considerando funcionalidad, facilidad de uso, documentación, interfaz y especialmente el código, resulta relativamente fácil proponer una solución que incluya todos los beneficios de las aplicaciones revisadas, y que al mismo tiempo esté libre de los inconvenientes que éstas presentan.
Con respecto al conjunto de aplicaciones revisadas, se puede decir que son las más fáciles de encontrar en Internet, debido a que están incluidas en diferentes sitios y al estar buscando información de árboles binarios es relativamente fácil dar con estas aplicaciones.
Existen otras aplicaciones que inicialmente fueron desechadas por no funcionar en la mayoría de los navegadores y sistemas operativos comunes.

Existe una aplicación que no está considerada en el análisis de código fuente ni en la tabla comparativa que se muestra en esta misma sección, peor que si está incluida en el análisis funcional que se hizo en el capítulo 2. Es el caso del "AVLVis System, a visualization and tutoring tool for the AVL Data Structure", esta aplicación cumple con muchas de las características que se consideran deseables en la comparación, sobre todo con relación al enfoque educativo que podría dársele a este tipo de aplicaciones, desafortunadamente el código no está disponible para análisis, y no es posible determinar si existen elementos que sean claramente susceptibles de corrección o mejora. Esta aplicación tiene dos elementos que la distinguen del resto, tiene una página de ayuda, en donde describe de manera clara cuales son las diferencias entre los diferentes modos de operación, y tiene además una sección de información de árboles AVL, en donde se menciona de manera general información acerca del algoritmo del árbol AVL, los requerimientos de esta estructura, así como las operaciones de inserción, eliminación y rotación, en sus diferentes formas.

Cuadro comparativo entre las soluciones estudiadas y la propuesta en este trabajo

CARACTERÍSTICA	AVL TREE DEMOSTRATION POR ARSEN GOGESHVILI	AVL ANIMATED TREE POR JOHN KLOSS	RED BLACK JAVA TREE POR M.CONRAD	APLICACIÓN PROPUESTA
DE PROGRAMACIÓN				
Lenguaje de programación	Java	Java	Java	Action Script
Orientación a Objetos	□	□	□	□

Utilización de herencia		□	□	□
Utilización de polimorfismo			□	□
Sobrecarga de funciones	Apenas	□	□	□
Comentarios del autor	□	Apenas	Incompletos	□
FUNCIONALES				
Representación gráfica del árbol	□	□	□	□
Interactividad para construir el árbol (estudiante -> applet)	□	□	□	□
Modo de instrucción (applet -> estudiante)				□
Modo autómata (applet –> applet)				□
Modo Inspector (simulación del recorrido del código, durante las operaciones)				□
Operaciones animadas	□	□	□	□
Explicación paso a paso de la operación, conforme ocurre			□	□

Análisis y diseño del sistema propuesto

Considerando los elementos de la tabla de comparación de los sistemas analizados se puede inferir que ninguna de las aplicaciones existentes tiene todos los elementos que se consideran ideales, de acuerdo al objeto de análisis de este documento.

Por tal razón, y tomando como base el trabajo previo, que de ninguna manera es descalificado, podemos decir que una aplicación con las características mencionadas en el cuadro comparativo antes expuesto, sería de gran utilidad para los estudiantes en proceso de aprendizaje, o bien, para aquellos interesados en el desarrollo de una aplicación que cumpla tal función.

Para el análisis se considera que Racional Rose resulta una herramienta por demás útil y adecuada para el modelado, debido al manejo de UML. A continuación se presentan los diagramas propuestos para la aplicación sugerida.

Características de la aplicación

En este documento se han mencionado algunas de las características que deben cumplir los sistemas en Internet, que son consideradas como equivalentes a ciertas estrategias de enseñanza. Tales características no son exclusivas, pero servirán como fundamento de la utilización pedagógica del Internet, y en particular de esta aplicación propuesta.

El hecho de que la aplicación se quiera desarrollar no como una aplicación independiente, sino como elemento de una página de Internet obedece a varios intereses, uno de ellos es la capacidad de distribución que prácticamente se vuelve ilimitable con Internet, y el otro de ellos es el valor pedagógico que pueden tener los sistemas en Internet como se ha referenciado en este documento. En alineación a las mencionadas características, se considera que el sistema propuesto deberá tener las siguientes características (entendiendo por sistema todo el entorno de ejecución, es decir, la página de Internet en que estará hospedado el sistema, y el sistema mismo):

Con relación a la página

- Simple, con una interfaz amigable, intuitiva, fácil de usar, y fácil de navegar
- Se debe considerar que la página en la que esté hospedado el sistema, deberá mantener el sentido de finalidad del proyecto, es decir, no se deberá perder de vista la razón de ser de la página, y principalmente de la aplicación
- Se deberán proveer enlaces a otros sitios de Internet, que presenten información relacionada con el TDA Árboles Binarios de Búsqueda, o bien con estructuras de datos en general.
- Se deberá proveer un manual de usuario de la aplicación con instrucciones básicas de operación, que por sencillo que parezca, servirá como documento de referencia en caso de existir dudas durante la utilización de la aplicación
- La página contará con un enlace a una página estática, en donde se presente información teórica y gráfica de manera jerárquica, de las estructuras de datos y los TDA en general, y en particular del TDA Árboles Binarios de Búsqueda

Con relación a la aplicación

- La aplicación deberá contar con una interfaz amigable, intuitiva y fácil de utilizar

- Deberán estar claramente identificadas el área de despliegue de los árboles representados, el área en donde estarán los botones u opciones de la aplicación, y el área en donde se desplegarán los mensajes después de cada operación realizada
- No es indispensable pero si deseable, que la aplicación cuente con la posibilidad de realizar operaciones adicionales a las básicas, tales como: recorrido (en in-orden, pre-orden y post-orden), identificación de máximo y mínimo, así como de nodo antecesor y predecesor, y "deshacer", que regresará el árbol al estado anterior a la operación seleccionada, todo lo anterior, dada la importante aportación que puede representar al proceso de aprendizaje
- El sistema deberá contar con una opciones para activar un modo de operación o de animación, que permitirá al estudiante determinar si desea observar la animación paso a paso o de manera normal
- El sistema deberá permitir al usuario cambiar de un tipo de árbol a otro en cualquier momento, y actualizará el árbol de acuerdo a las reglas que apliquen para el nuevo tipo de árbol seleccionado
- El sistema deberá contar con las siguientes opciones
    - Interactividad con el estudiante, es decir, este podrá construir el árbol con base en datos ingresados por él
    - Modo de instrucción, el sistema interactuará con el estudiante, presentando preguntas acerca de los pasos requeridos para completar una operación dada
    - Modo autómata, el sistema realizará una demostración del árbol seleccionado, sin considerar datos ingresados por el usario
    - Modo inspector, el sistema simulará el recorrido del código (similar a procesos de debug), que permitirán al usuario conocer de manera más detallada qué se está haciendo para completar la operación seleccionada

Con relación a la programación

- El sistema deberá utilizar las técnicas de programación orientada a objetos, poniendo especial énfasis en la utilización de herencia, polimorfismo y sobrecarga de funciones
- Dado el carácter didáctico de la aplicación, ésta deberá estar totalmente documentada, para dar al estudiante no sólo una idea clara del algoritmo utilizado, sino un ejemplo de un sistema claramente documentado por su autor

Diagrama de casos de uso

Para comenzar con el modelado, se inicia creando diagramas de casos de uso. Estos diagramas consisten en actores, casos de uso y las relaciones existentes entre ambos. De manera inicial se

identifican los actores, como las “cosas” o “entes” que interactúan con el sistema que se pretende desarrollar, en este caso en particular, se identifica solamente a el estudiante o usuario como actor del sistema, ya que no existen otros agentes que intervengan con el sistema.

Posteriormente se identifican los casos de uso para cada actor. Un caso de uso es la pieza de funcionalidad que provee el sistema, y se identifican viendo de qué manera los actores interactúan con el sistema.

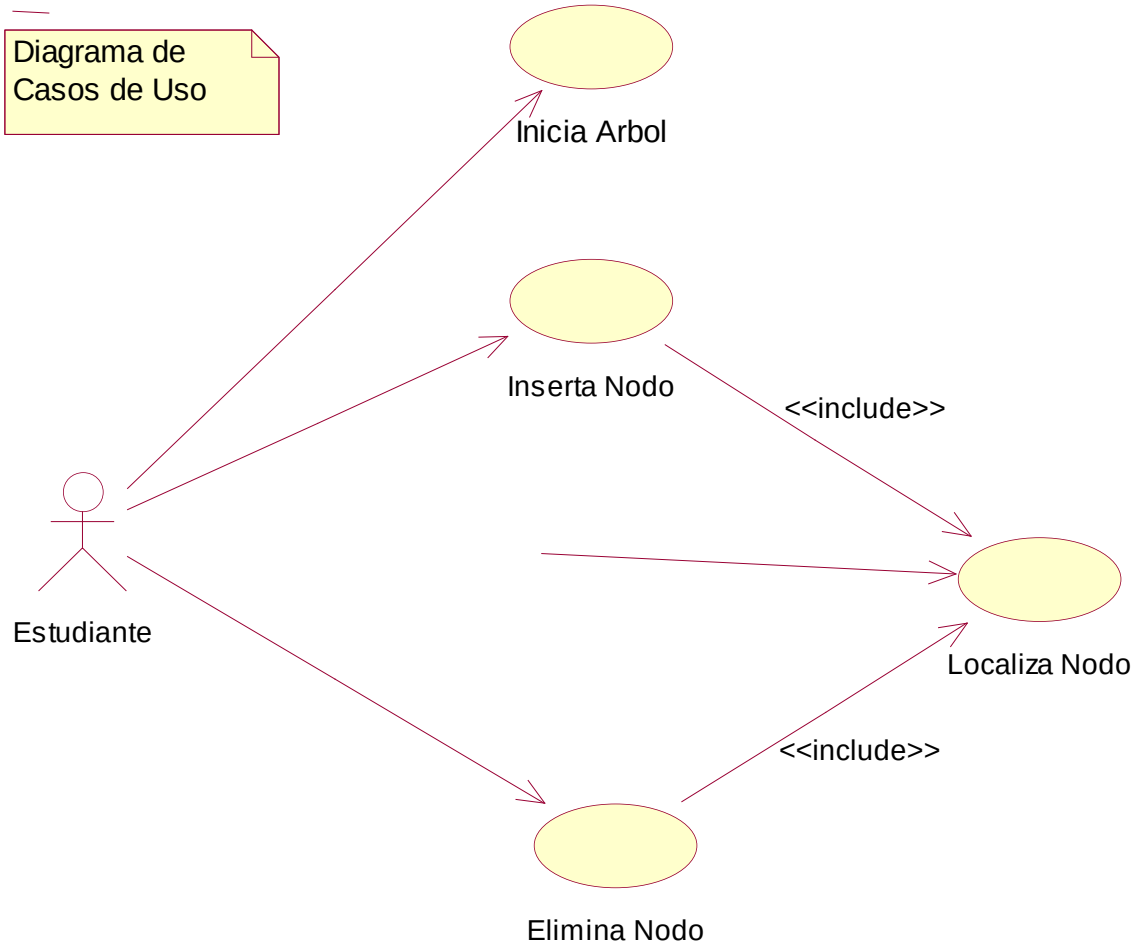

Figura 17. Diagrama de Casos de Uso

Para el caso particular del diseño del sistema que se propone, se considera solamente la intervención del estudiante o usuario como actor, al no haber más agentes involucrados en el proceso, durante la ejecución de la aplicación.

Los casos de uso que se identifican son “Inicia Arbol”, “Inserta Nodo” y “Elimina Nodo”, estos dos últimos tienen una relación “uses” con el caso de uso “Localiza Nodo”, ya que existe una porción de comportamiento similar. De igual manera se establece una relación entre el actor estudiante y el caso de uso localiza, ya que este último representa una operación que el mismo actor puede realizar.

Diagrama de clases

El diagrama de clases se genera tomando como referencia los objetos del diagrama de secuencia, ya que estos objetos pueden ser agrupados en clases. Tanto el proceso de diagramación de escenarios, como la búsqueda de clases continua hasta que se alcanza la “ley de disminución de retorno”, que es cuando ya no se encuentran más clases o cuando aparentemente se está escribiendo la misma información una y otra vez.

En el diagrama de clases se especifican las relaciones que implican comunicación entre los objetos, esta información se obtiene analizando los diagramas de secuencia para ver qué objetos se comunican entre si.

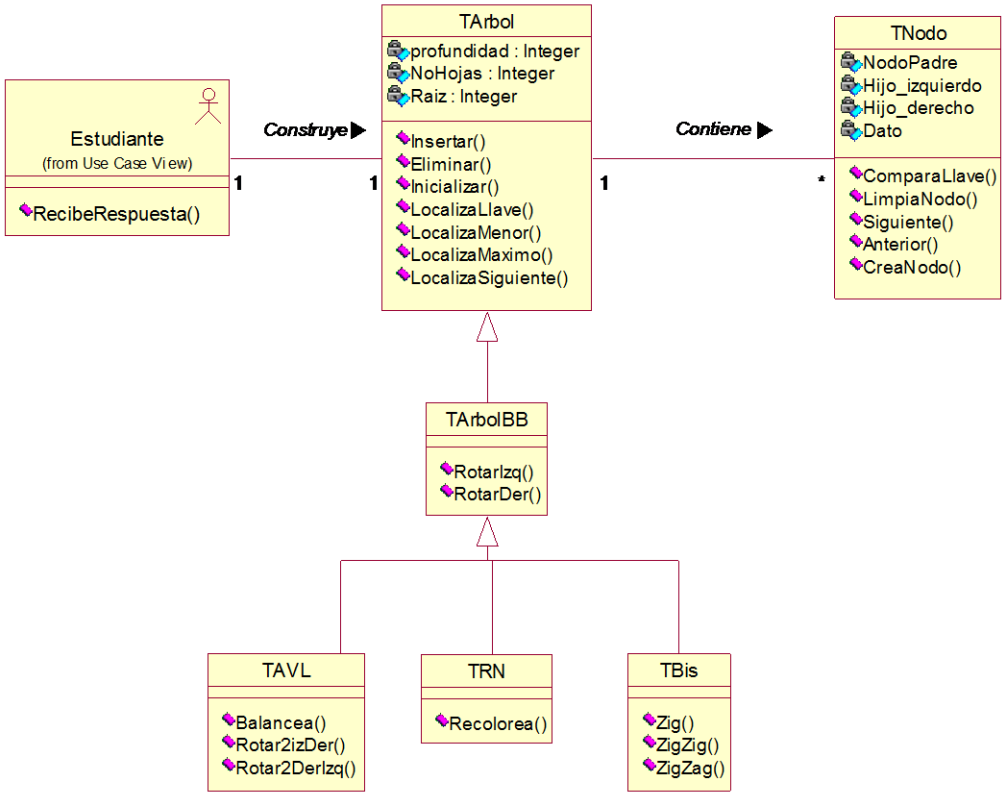

Figura 18. Diagrama de Clases

En el caso de nuestra aplicación propuesta las dos clases que intervienen en el sistema son Tarbol y las clases TAVL, TRN y TBis, que pueden ser consideradas como subclases hijas de la superclase TArbol. TNodo, por su parte, es una clase relacionada con TArbol, que se instancia por cada nodo del árbol.

Diagrama de secuencia

El diagrama de secuencia es la representación gráfica de la funcionalidad del caso de uso, y muestra el camino del flujo de sus eventos. Los diagramas de secuencia contienen objetos y mensajes entre los objetos, que muestran su comportamiento.

Estos diagramas son iniciados por los actores e incluye, objetos y los mensajes existentes entre estos dos últimos. Para dar mayor claridad al diagrama, es de utilidad nombrar a los actores y objetos.

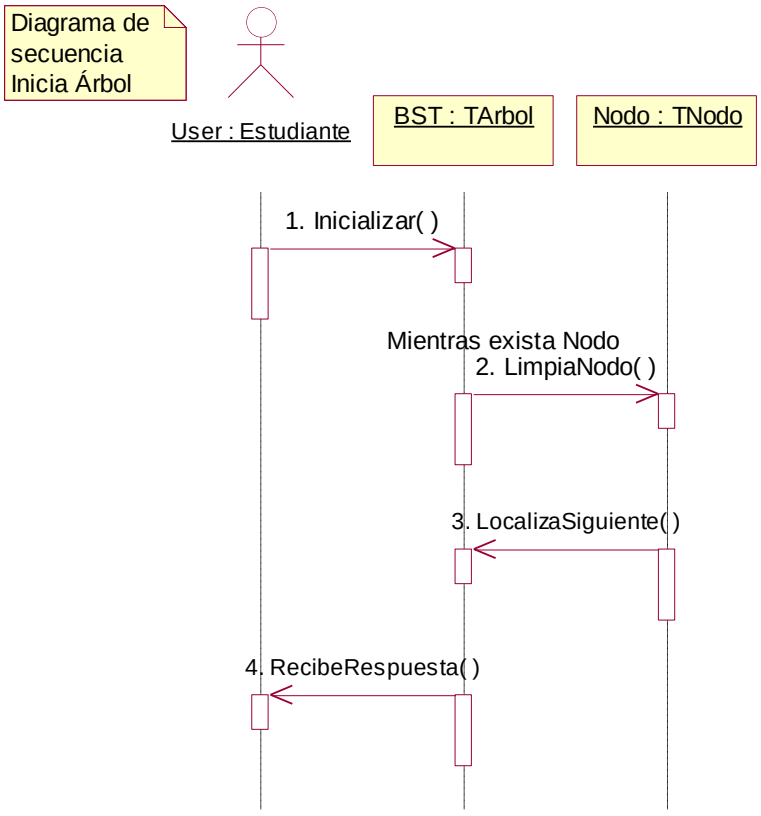

Figura 19. Diagrama de Secuencia Inicia Arbol

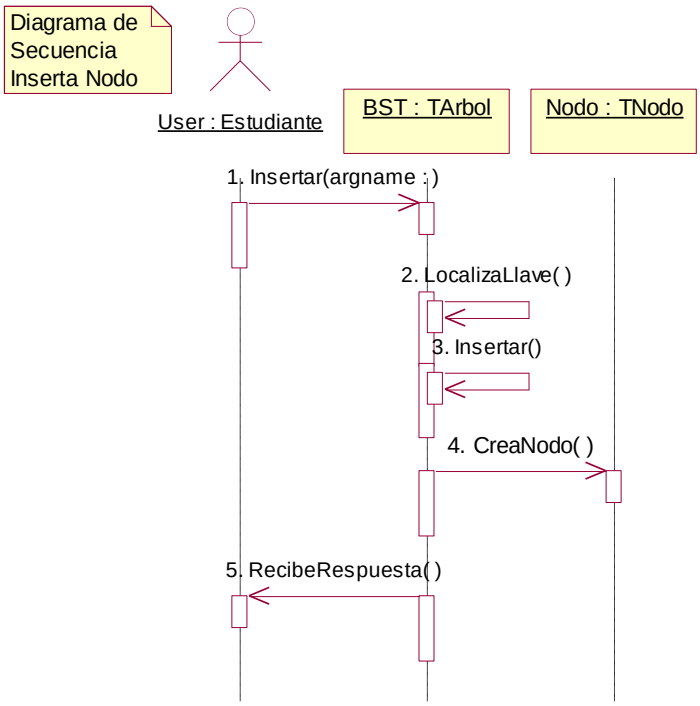

Figura 20. Diagrama de Secuencia Inserta Nodo

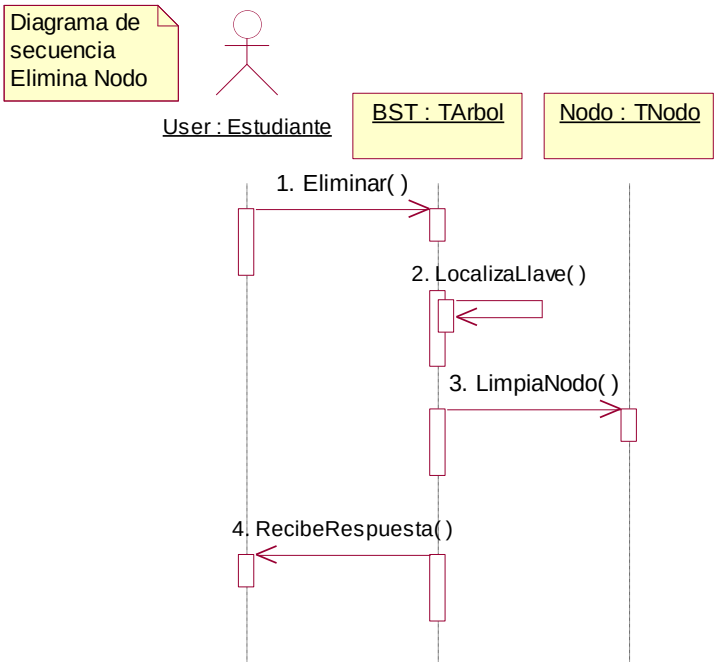

Figura 21. Diagrama de Secuencia Elimina Nodo

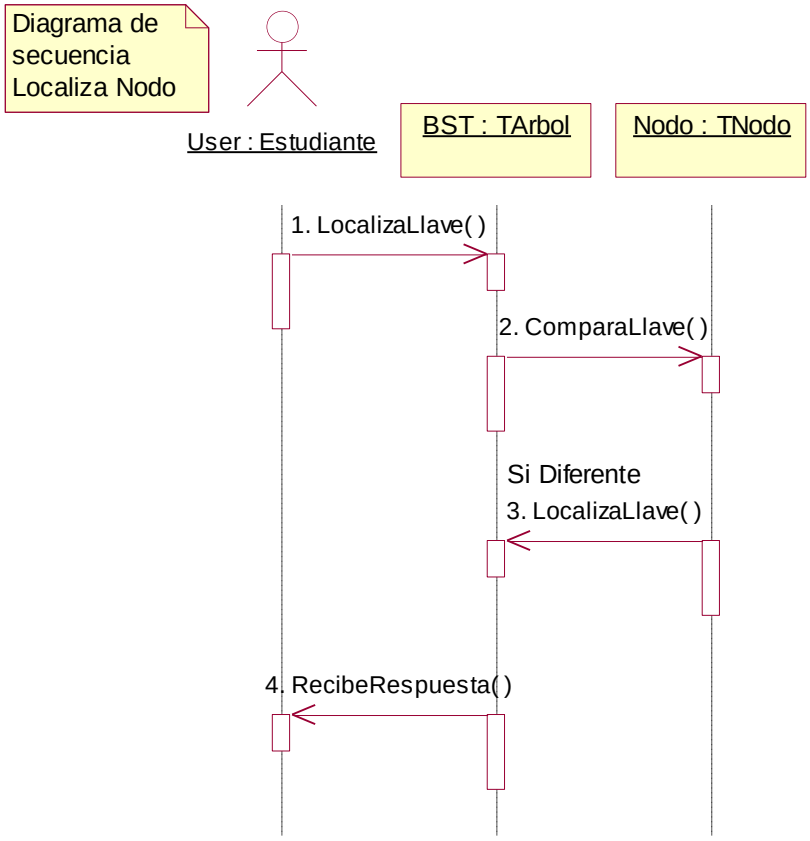

Figura 22. Diagrama de Secuencia Localiza Nodo

Se presentan los diagramas de secuencia de los cuatro casos de uso detectados para la aplicación, la inicialización del árbol, así como las tres operaciones básicas de los árboles binarios de búsqueda: inserción, eliminación y búsqueda.

Diagrama de componentes

Los diagramas de componentes dan una vista física del modelo y muestran las organizaciones y dependencias entre los componentes de software.

Estos diagramas también muestran el comportamiento externamente visible de los componentes, al desplegar sus interfaces.

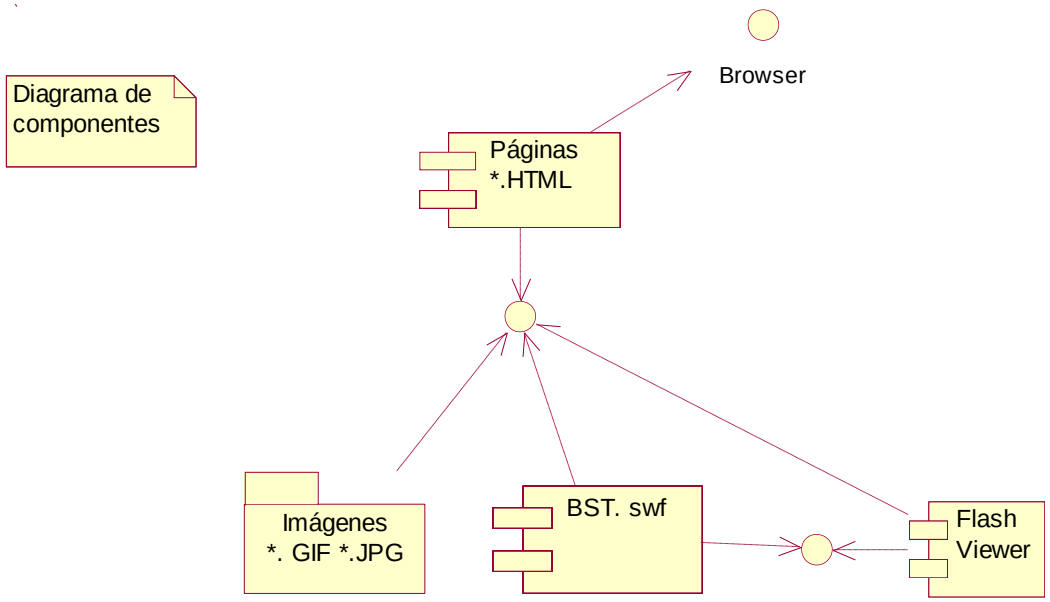

Figura 23. Diagrama de Componentes

En el diagrama de componentes propuesto se muestra cómo la aplicación sugerida deberá estar hospedada en una página de Internet, que como se mencionó además, tendrá ligas a otras páginas, así como a otros sitios con información de interés.

Diagrama de despliegue

El diagrama de despliegue muestra las relaciones físicas entre los componentes de software y de hardware en el sistema.

Cada nodo representa alguna clase de equipo de cómputo, en la mayoría de los casos se trata de hardware.
Los componentes representan módulos físicos de código.
Finalmente las dependencias entre los componentes muestran la comunicación entre ellos, y la dirección de una dependencia dada indica el conocimiento en la comunicación.

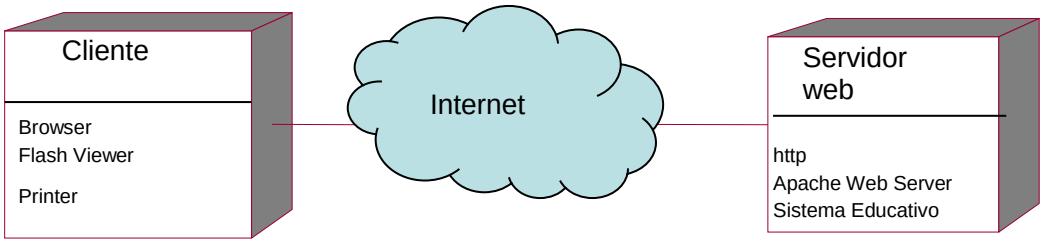

Figura 24. Diagrama de Despliegue.

Prototipo de Interfaz

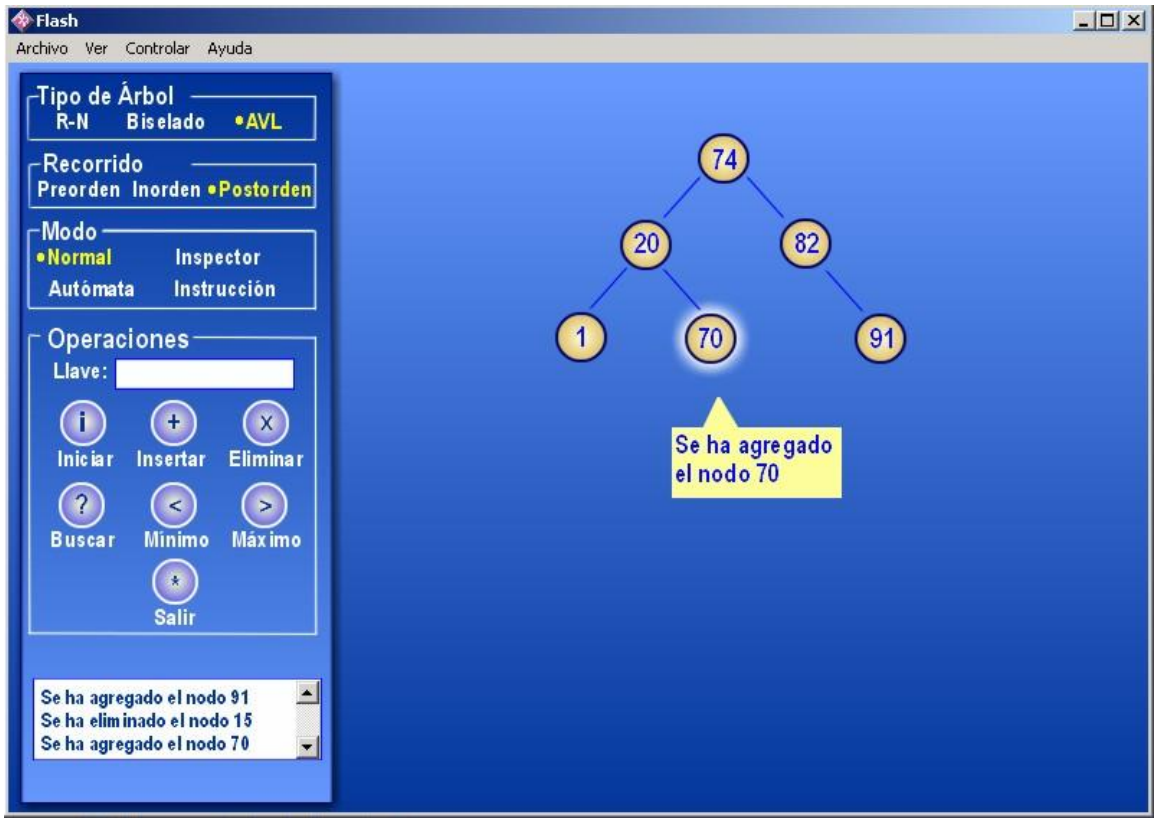

Figura 25. Prototipo de la interfaz del sistema propuesto.

En el prototipo de interfaz presentado se incluye la opción para seleccionar los diferentes tipos de árboles binarios de búsqueda que pueden ser representados por la animación gráfica. Se incluyen además botones para inicializar el árbol, así como para realizar las operaciones más comunes en los árboles, inserción, eliminación y búsqueda, y otras operaciones adicionales como recorrido, y localización de nodos máximo y mínimo.

Para el área de mensajes se contempla que se puedan grabar para consultas de operaciones posteriores. Una operación que no se consideró y que es recomendable, es la operación de deshacer, que regresará al árbol al estado original en el que estaba antes de realizar la operación.

Conclusiones

- Con el análisis de las aplicaciones, tanto en su interfaz, facilidad de uso, como programación, se cumple el objetivo de compilar y presentar la información resultante que permitirá rescatar los elementos para concluir el diseño aquí presentado.

- En este diseño, se consideran las áreas de mejora que fueron determinadas en el caso de cada aplicación, es decir, se utilizan las mejores prácticas de las técnicas de programación y diseño actuales.

- Se considera además que la información presentada es representativa de los elementos que pueden ser considerados al desarrollar una aplicación, cuando se tiene como objetivo primordial que ésta sea utilizada en el proceso de enseñanza, tanto por personal docente como por estudiantes.

Recomendaciones

- Se recomienda el desarrollo de la aplicación aquí propuesta y diseñada, por considerar que contará con elementos que serán de utilidad para el aprendizaje de la representación e implementación de los Árboles Binarios de Búsqueda.

- Se considera que las clases propuestas son susceptibles de cambios o mejoras, que podrían verse reflejados en beneficios relativamente fáciles de obtener, tales como la representación de cualquier tipo de árboles binarios, una vez realizados los ajustes correspondientes a los métodos y atributos de las clases propuestas.

Bibliografía

1. Aho, Alfred V.; Hopcroft, John E.; Ullman, Jeffrey “Data Structures and Algorithms”, Addison-Wesley Iberoamericana S.A. (1988).
2. Albizuri Romero, Miren Begoña. “Estructuras de Datos e Introducción a Bases de Datos”, Editorial Limusa, S.A. de C.V. (1989).
3. Baker, Boilen, Goodrich, Tamassia, y Stibel, “Testers and Visualizers for Teaching Data Structures”.
4. Borrás, Isabel (1997). “Enseñanza y aprendizaje con la Internet: Una aproximación crítica.” Pixel-Bit. Revista de Medios y Educación, 9, Junio, 5-13
5. Folk, Michael; Zoellick, J. Bill. “Estructuras de Archivos, Un conjunto de herramientas conceptuales”. Addison-Wesley Iberoamericana S.A. (1992).
6. Fowler, Martin; Scout, Kendall. “UML gota a gota”, Addison Wesley Longman de México, S.A. de C.V. (1999).
7. Fuller, Wiley; Kent, John-Michael. “AVL Tree Visualisation” en http://sky.fit.qut.edu.au/~maire/avl/System/AVLVis.html . 1999
8. Gogeshvili, Arsen. “AVL Tree applet” en http://www.seanet.com/users/arsen/avltree.html. 2002
9. Heileman, Gregory L. Estructuras de Datos, Algoritmos, y Programación Orientada a Objetos, McGraw Hill/Interamericana de España, S.A.U. (1998).
10. Internet. “AVL Tree demo” en http://alife.mic.dundee.ac.uk/courses/ac202/trees/avldemo/avltree.html
11. Internet. “Red-Black Tree demonstration” en http://www.ececs.uc.edu/~franco/C321/html/RedBlack/redblack.html
12. Kloss, John. “AVL Tree” en http://www.cs.jhu.edu/~goodrich/dsa/trees/avltree.html . 1997
13. Michael, Amir. “Teaching Bynary search Tree algorithms trough programming, proof and animation” en http://opsis.sourceforge.net/. 1999
14. National Institute of Standards and Technology. “Dictionary of Algorithms and Data Structures” en http://www.nist.gov/dads/
15. Plank, James S. “Red-Black Tree Code Home Page” en http://www.cs.utk.edu/~plank/plank/rbtree/rbtree.html. University of Tennesse.
16. Rational Software Corporation. en http://www.rational.com/
17. Tamassia, Roberto; Cohen,Robert y otros. “The data structures Library in Java” en http://www.jdsl.org . 2001