Ciencia & Desarrollo, Vol. 16, 21 (2) 65 - 72 (2017)
ISSN 2304 – 8891 Versión impresa
Depósito legal N° 2017-16148

VELOCIDAD DE RESPUESTA DE LOS ALGORITMOS DE BÚSQUEDA DE DATOS CONTENIDOS EN ESTRUCTURAS ESTÁTICAS Y DINÁMICAS

RESPONSE SPEED OF THE ALGORITHMS OF DATA SEARCH CONTAINED IN STATIC AND DYNAMIC STRUCTURES

[1]Edwin Antonio Hinojosa Ramos, [2]Hugo Manuel Barraza Vizcarra

RESUMEN

Los datos almacenados en estructuras de datos dinámicas del tipo Árbol AVL permiten obtener mayor velocidad de respuesta en las operaciones de búsqueda de datos específicos en comparación con las estructuras de datos estáticas del tipo Array unidimensional. Esta velocidad está en función del número de comparaciones efectuadas en el proceso de búsqueda y claramente se verifica que el número de comparaciones efectuada en los arreglos es mucho mayor que las comparaciones efectuadas en el Árbol AVL cuando se realiza el proceso de localización de claves. Pero también se observa que el tiempo que se tarda en insertar las claves en un Array unidimensional es mucho menor que el tiempo requerido para almacenar los datos en un Árbol AVL. Pero, en promedio, la velocidad de respuesta de los algoritmos de búsqueda de datos contenidos en estructuras dinámicas del tipo Árbol AVL es mayor que la velocidad de respuesta cuando el proceso de búsqueda se efectúa en las estructuras estáticas del tipo Array unidimensional.

Palabras clave: Estructura dinámica, estructura estática, velocidad de respuesta.

ABSTRACT

The data stored in dynamic data structures of the AVL Tree type allows to obtain a higher response speed in the specific data search operations, in comparison with the static data structures of the one-dimensional Array type. This speed is a function of the number of comparisons made in the search process and it is clearly verified that the number of similarities achieved in the arrangements is much greater than the comparisons made in the AVL Tree when the key location process is performed. However, it is also observed that the time it takes to insert the keys in a one-dimensional array is much less than the time required to store the data in an AVL Tree. On average, the response speed of the data search algorithms contained in dynamic structures of the AVL Tree type is greater than the response speed when the search process is performed in the static structures of the one-dimensional Array type.

Keywords: Static structure, dynamic structure, response speed.

INTRODUCCIÓN

En todo proceso computacional es necesaria la operación de datos que pueden estar contenidos en memoria interna o en memoria externa. Existen diferentes organizaciones lógicas de los datos a las que denominamos estructuras de datos. Cada una de ellas posee diferentes características que hacen que sean más o menos ventajosas para diferentes procesos en el ordenador. El objetivo fundamental de toda estructura de datos es la organización de los datos en la memoria del ordenador para poder operar luego esos datos en tiempo de ejecución de los programas.

Existen dos grupos bien diferenciados de estructuras de datos. Las estructuras estáticas y las estructuras dinámicas. Las primeras no pueden variar su tamaño y capacidad de almacenamiento en tiempo de ejecución lo que representa un riesgo de desbordamiento cuando la cantidad de datos supera a su capacidad de almacenamiento. En contraste, las estructuras dinámicas pueden modificar su capacidad de almacenamiento de datos en tiempo de ejecución, con lo cual, se consume sólo el espacio de memoria necesario que requiere la aplicación. Con esto se elimina el riesgo de desbordamiento de la estructura. Pero es importante señalar que la complejidad en el diseño de los algoritmos que operan datos contenidos en estructuras dinámicas es alta y el tiempo de desarrollo e implementación de los algoritmos aumenta considerablemente.

En el presente trabajo de investigación se estudia la diferencia de dos estructuras de datos, una estática que es el Array unidimensional y otra estructura dinámica que es el Árbol AVL. Se estudia el tiempo de respuesta para los procesos de búsqueda de datos contenidos en estas estructuras y con ello determinar la velocidad de respuesta de los algoritmos de búsqueda que actúan

[1]Facultad de Ingeniería, Departamento Académico de Ingeniería en Informática y Sistemas. Universidad Nacional Jorge Basadre Grohmann. Tacna-Perú. E-mail: ehr6@hotmail.com
[2]Facultad de Ingeniería, Departamento Académico de Ingeniería en Informática y Sistemas. Universidad Nacional Jorge Basadre Grohmann. Tacna-Perú. E-mail: hmbarrazav@gmail.com

Presentado: 05/12/16 Aprobado: 08/08/17

sobre estas. Para ello, se implementó en el lenguaje C++ un programa que genere en forma automática los datos y los almacene en las dos estructuras y a partir de ello efectuar operaciones de búsqueda cronometrando el tiempo de respuesta de los procesos para evaluar su desempeño

JUSTIFICACIÓN

La presente investigación aporta a las bases teóricas para la elección de la estructura de datos más apropiada en procesos de búsqueda. Ofrece evidencias reales que permitirán elegir la estructura más apropiada para un contexto determinado de datos soportados en un ordenador.

OBJETIVO

Objetivo general:
El objetivo de la investigación es medir y comparar la velocidad de respuesta de los procesos de búsqueda de datos contenidos en una estructura de datos estática del tipo Array y en otra estructura de datos dinámica del tipo Árbol Binario de Búsqueda Balanceado.

Objetivos específicos:

- Determinar el tiempo que se tarda en localizar un dato contenido en una estructura de datos estática del tipo array y una estructura de datos dinámica del tipo Árbol AVL.
- Determinar el número de comparaciones necesarias para localizar un dato contenido en una estructura de datos estática del tipo Array y una estructura de datos dinámica del tipo Árbol AVL.

BASES TEÓRICAS

Los datos son costosos. Deben ser manejados de tal manera que sean correctos y estén disponibles para producir información (Loomis, 1999).

Los costos por usar una lista ligada (estructura dinámica) en lugar de una representación secuencial (estructuras estáticas) son dos: Primero, el espacio requerido para una representación ligada requiere una cantidad extra para el campo del apuntador. Segundo, con frecuencia la búsqueda de un nodo en una representación ligada será más larga con respecto a la búsqueda en un arreglo que aloje a la lista. Recordemos que podemos calcular la localidad del inicio del n-ésimo elemento en un arreglo. Para encontrar el n-ésimo nodo en una lista ligada (suponiendo que no utilizamos punteros auxiliares), la cadena de los primeros *n-1* nodos debe ser visitada, ya que el *n-ésimo* nodo no puede ser accesado directamente (Loomis, 1999).

Muchos algoritmos requieren una representación apropiada de los datos para lograr ser eficientes. Esta representación junto con las operaciones permitidas se llama estructura de datos. Típicamente estas permiten inserciones arbitrarias. Las estructuras de datos varían en cómo permiten el acceso a miembros del grupo. Algunas permiten tanto accesos como operaciones de borrado arbitrarios. Otras imponen restricciones, tales como permitir el acceso sólo al elemento más recientemente insertado (Weiss, 2004).

Un algoritmo de búsqueda acepta un argumento *a* y trata de encontrar un registro cuya llave es *a*. El algoritmo puede retornar el registro completo o con frecuencia retorna un puntero a ese registro. Es posible que la búsqueda por algún argumento en particular en una tabla no tenga éxito; es decir, que no exista registro en la tabla con ese argumento como llave. En este caso, el algoritmo debe retornar un registro "nulo" o un "puntero nulo". Frecuentemente si la búsqueda no tiene éxito, puede ser mejor adicionar un nuevo registro con el argumento como llave. Un algoritmo que hace esto se llama algoritmo de búsqueda e inserción.

La forma como están organizados pueden ser arreglos de registros, una lista encadenada, un Árbol, o inclusive un grafo. Debido a que diferentes técnicas de búsqueda pueden ser apropiadas para diferentes organizaciones. La tabla generalmente está diseñada para alguna técnica de búsqueda específica. La tabla puede estar contenida en forma completa en memoria, completamente en un almacenamiento auxiliar o puede estar partida entre los dos. En este caso, se requieren diferentes técnicas de búsqueda de acuerdo a las diferentes condiciones que se asuman. Aquellos tipos de búsqueda en las cuales la tabla está completamente contenida en la memoria, son llamados búsquedas internas, mientras que aquellas en las cuales casi toda la tabla es mantenida en almacenamiento auxiliar son denominadas búsquedas externas (Tenenbaum, 2004).

La forma más simple del método de búsqueda es la búsqueda secuencial. Este método es aplicable a una tabla que está organizada ya sea como un arreglo o como una lista encadenada. Asumamos que *k* es un arreglo de *n* llaves y *r* un arreglo de registros tal que *k* (i) es la llave de *r*(i). Asumamos también que *i* es el argumento de búsqueda. El objetivo es asignar la variable (o función identificadora) search() al entero más pequeño *i* tal que *k(i)* = key si este *i* existe, y en caso contrario será igual a cero.

El almacenar una tabla como una lista encadenada tiene la ventaja que el tamaño de la tabla puede ser aumentada dinámicamente de acuerdo a los requerimientos. Asuma que la tabla está organizada como una lista encadenada lineal apuntada por *table* y encadenada por un campo puntero denominado *next* (Tenenbaum, 2004).

Clasificación de las estructuras de datos

Según Cairó (2007) las estructuras de datos se pueden clasificar de dos maneras. De acuerdo a la posibilidad de modificación de las estructuras en tiempo de ejecución y de acuerdo al consumo de memoria en el ordenador, las estructuras de datos se clasifican como se muestra en la tabla 1.

Tabla 1. *Estructuras estáticas y dinámicas*

ESTRUCTURAS ESTÁTICAS	ESTRUCTURAS DINÁMICAS
Arreglos	Listas
Registros	Árboles
Conjuntos	

Fuente: Cairó, O. (2007)

Otra clasificación se efectúa de acuerdo a la forma que enlazan los datos en posiciones de memorias.

Tabla 2. *Estructuras lineales y no lineales*

ESTRUCTURAS LINEALES	ESTRUCTURAS NO LINEALES
Arreglos	Árboles
Registros	Grafos
Pilas	
Colas	
Listas	

Fuente: Cairó, O. (2007)

Como se puede ver en la tabla 2, se consideran como estructuras de datos no-lineales a los Árboles y a los grafos, sin embargo la estructura tipo Árbol es en sí un tipo especial de grafo dirigido, unidireccional, convexo y sin ciclos (Sisa y Velez, 2002).

Las estructuras lineales, son aquellas que al representarse en el hardware del ordenador, lo hacen situando sus elementos de forma contigua en relación de 1 a 1; esto quiere decir que cada elemento de la estructura sólo puede ir enlazado al elemento siguiente o anterior (Rodríguez *et al.*, 2011).

Las estructuras no-lineales se caracterizan por no existir una relación de sus elementos, es decir, que un elemento puede estar enlazado con uno o más elementos.

Árboles

Los Árboles son las estructuras de datos dinámicas y no-lineales más importantes en la computación. Son dinámicas, puesto que la estructura Árbol puede cambiar en tiempo de ejecución. Y son no-lineales, puesto que a cada elemento del Árbol pueden seguirle uno o varios elementos. Se caracterizan por que la información que almacenan se puede estructurar de forma jerárquica (Garrido y Fernández, 2006)

Algunas definiciones de esta estructura de datos son:

"Intuitivamente el concepto de Árbol implica una estructura en la que los datos se organizan de modo que los elementos de información están relacionados entre sí a través de ramas. Un Árbol consta de un conjunto finito de elementos, denominados «nodos» y un conjunto finito de líneas dirigidas, denominadas «ramas» que conectan los nodos" (Aguilar, 2002).

"La organización de los datos en una estructura no lineal, jerárquica o de niveles, es una opción para representar estructura de datos, las más conocidas y usadas en la computación se denominan Árboles. Su característica principal es que mantienen una relación de uno a muchos (1:n) entre sus elementos" (Martínez y Quiroga, 2002).

"Formalmente se define un Árbol de tipo T como una estructura homogénea que es la concatenación de un elemento de tipo T junto con un número finito de Árboles disjuntos, llamados sub Árboles. Una forma particular de Árbol puede ser la estructura vacía" (Cairó y Guardati, 2007).

De las definiciones anteriores podemos concluir que un Árbol es un tipo de estructura de datos no lineal, jerárquica y homogénea aplicada sobre una conjunto de objetos llamados nodos (uno de los cuales es conocido como raíz). Debido a la estructura jerárquica de los nodos, figura 1, se establece una relación de parentesco entre los nodos, dando lugar a términos como padre, hijo, hermano, antecesor, sucesor, ancestro, etc. Esta estructura jerárquica de los árboles, permite que se puede aplicar operaciones recursivas como la inseción, eliminación y búsqueda de nodos (Peláez y Viso, 2008)

En Ciencias de la Computación, los Árboles se usan exhaustivamente como una estructura eficiente para realizar búsquedas en lista dinámicas grandes y para aplicaciones diversas como los sistemas de inteligencia artificial y los algoritmos de codificación.

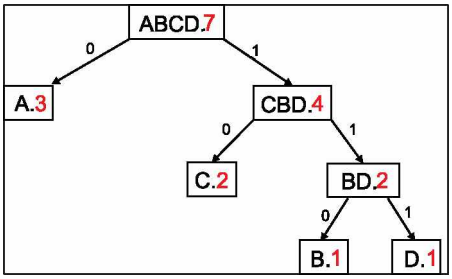

Figura 1. Naturaleza recursiva de un Árbol
Fuente: Elaboración propia

Representaciones de un Árbol

Gráficamente, una estructura Árbol puede representarse de numerosas maneras. En la siguiente figura se muestran cuatro de ellas.

En la figura 2a. se muestra la representación gráfica que es sin lugar a dudas la más utilizada y la que mejor pone de manifiesto las relaciones entre los nodos; en la figura 2b. se representa por medio de diagramas de Venn; en la figura 2c. por medio de la notación indentada; y en la figura 2d. se muestra por representación de paréntesis anidados que ofrece por su parte la ventaja de ser más la representación más compacta.

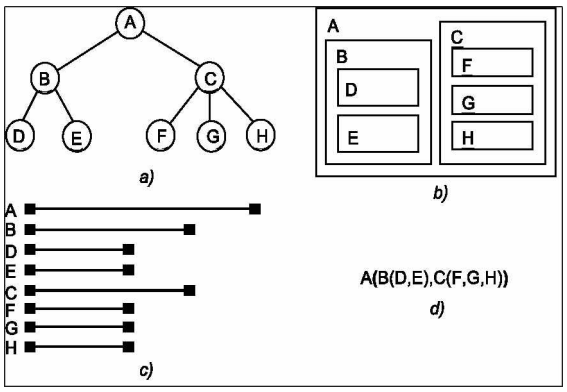

Figura 2. Representación de la estructura tipo Árbol
a) Grafo, b) Diagrama de Venn, c) Notación indentada, d) Anidación de paréntesis
Fuente. Cairó y Guardati (2007)

Terminología usada en Árboles

Los Árboles se basan en el concepto de nodo. Hay otras estructuras de datos que también hacen uso de este concepto como se ilustra en la figura 3. En general, los nodos pueden definirse de la siguiente forma:

Se denomina nodo a cualquier tipo cuyos elementos son registros formados por un campo "Datos" y un número dado de apuntadores o enlaces. (Hernández et al., 2000).

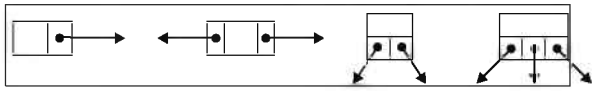

Figura 3. Concepto de nodo
Fuente: Hernández, R. (2001)

Se muestran, los nodos correspondientes a una lista enlazada, un Árbol Binario y un Árbol Ternario, en ese orden.

En el estudio de los Árboles se hace uso de la siguiente terminología, que se ilustra en la figura 6.

Raíz: Es único nodo donde nace la estructura, no tiene predecesor, pero si puede tener sucesores.

Nodos hoja: Nodos que no tienen sucesor, se denominan también nodos terminales.

Nodos interiores: Nodos que no son ni raíz, ni nodos hoja.

Sub-Árboles: Si se aplica la definición recursiva de Árbol, un sub-Árbol es cada uno de los Árboles que salen de un nodo. También se denominan ramas.

Nivel: Los niveles se establecen de la siguiente manera: A la raíz se le asigna el nivel uno, los inmediatos sucesores de un nodo tienen un nivel más que éste, y así sucesivamente, lo que implica que todos los inmediatos sucesores de un nodo tienen el mismo nivel. Al aplicar este concepto a la figura 4, se obtiene la tabla 3.

Tabla 3. *Niveles del Árbol de la figura 2*

NIVEL	NODOS
1	A
2	F H
3	B M C N D
4	E P R

Fuente: Sisa, J. (2002)

Niveles en el Árbol. Se debe tener en cuenta que algunos autores asignan a la raíz el nivel cero.

Altura del Árbol: La altura de un Árbol es el máximo número de niveles de todos los nodos del Árbol.

Grado: Es el número de sub-Árboles que salen de un nodo. Según el grado, los Árboles se denominan binarios (como máximo dos hijos), ternarios (como máximo tres hijos), y así sucesivamente. Los Árboles Binarios son de especial interés ya que cada nodo tiene dos descendientes, denominados sub Árbol izquierdo y sub Árbol derecho.

Todos los términos mostrados anteriormente, se ven reflejados en la figura 4, se puede observar además la relación entre dos nodos x e y cualquiera, a la que llamaremos relación padre-hijo.

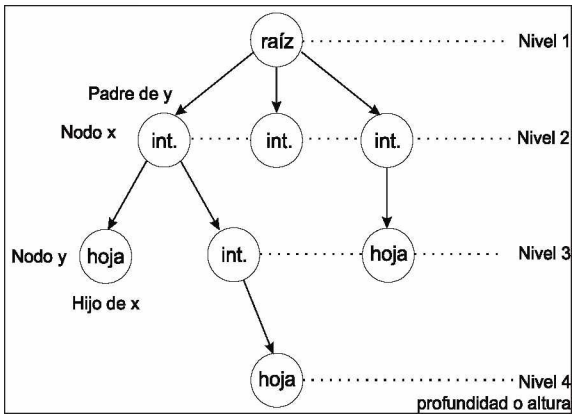

Figura 4. Terminología usada para los Árboles
Fuente: Hernández, R. (2001)

Árboles Binarios

Los Árboles Binarios son Árboles n-arios cuyo número máximo de sub Árboles que sale de cualquier nodo es dos.

Son las estructuras más utilizadas en computación para el tratamiento de los datos en la memoria principal. Los Árboles Binarios tienen múltiples aplicaciones. Se usan para tomar decisiones de dos opciones, para representar un árbol genealógico, para presentar un campeonato clasificatorio de tenis, fútbol, entre otros. En las figuras 5 y 6 se ilustran ejemplos de estas representaciones usando Árboles Binarios.

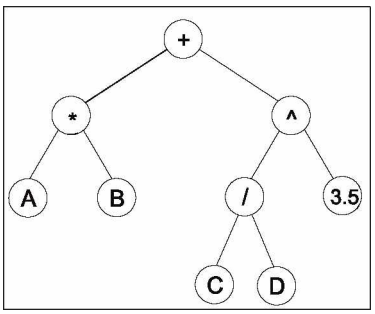

Figura 5. Representación de una expresión algebraica (A * B) +(C / D) ^3,5

Fuente: Elaboración propia

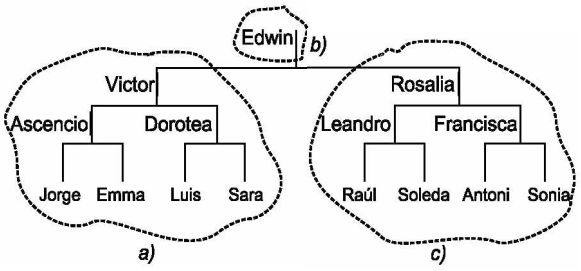

Figura 6. Ejemplo de Árbol genealógico
a) Sub Árbol. b) Raíz. c) Sub Árbol 2

Fuente: Elaboración propia

Definición de la clase Árbol Binario

Un Árbol Binario es un Árbol cuyo número máximo de sub Árboles es 2 y por ende el Árbol es de grado 2. Estos Árboles son de especial interés puesto que representan una de las estructuras de datos más usadas en computación.

"Un Árbol Binario es un conjunto finito de elementos que puede estar vacío o contener un elemento denominado la raíz del Árbol y otros elementos divididos en dos subconjuntos separados, cada uno de los cuales es en sí un Árbol Binario. Estos dos subconjuntos son denominados sub Árbol izquierdo y subárbol derecho del árbol original. Cada elemento de un Árbol Binario se denomina un nodo del Árbol" (Tanenbaum y Augenstein, 1993).

Un árbol binario puede representarse en memoria por medio de dos formas (Drozdek, 2007):

- Por medio de arreglos
- Por medio enlaces tipo puntero

Árboles Binarios de Búsqueda (ABB)

En un arreglo (estructura lineal), la manera más eficiente de realizar una búsqueda es con el algoritmo de búsqueda binaria aplicado en una tabla de memoria estática (Bisbal, 2009). Sin embargo, esta estructura presenta la desventaja de no ser eficiente para la inserción y la eliminación de elementos. Por otro lado, una lista enlazada ordenada (estructura lineal) tiene mejor comportamiento en las inserciones y bajas de sus elementos, pero no en el algoritmo de búsqueda binaria.

Ante esta disyuntiva, contar con estructuras no lineales como los Árboles, representa una opción para conjuntar las características positivas de estas estructuras lineales. La propuesta inmediata es el Árbol Binario de Búsqueda.

El Árbol Binario de Búsqueda es una estructura sobre la cual se pueden realizar eficientemente las operaciones de búsqueda, ya que las operaciones de inserción y eliminación se aseguran de que el Árbol Binario este optimizado para la este fin (Cairó y Guardati, 2007).

Para lograr esta eficiencia, es necesario que se cumpla una condición importante que se resume en la siguiente cita:

"Los Árboles Binarios de Búsqueda son Árboles Binarios en los cuales se cumple que para cualquier registro X perteneciente al Árbol, todos los datos de los nodos a la izquierda de X son menores que el dato de X y todos los datos a la derecha de X son mayores que el dato de X". (Flores, 2005).

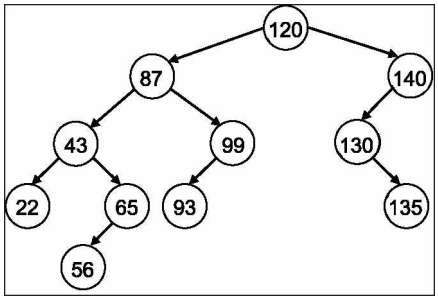

Figura 7. Árbol Binario de Búsqueda
Fuente: Cairo, O. (2007)

En la figura 7 se presenta un Árbol Binario de Búsqueda. Si se realiza un recorrido inorden sobre un Árbol de Búsqueda obtendremos una clasificación de los nodos en forma ascendente. Los diferentes recorridos en el Árbol de la figura 11 producen el siguiente resultado:

Preorden: 120 – 87 – 43 – 22 – 65 – 56 – 99 – 93 – 140 -130 – 135
Inorden: 22 – 43 – 56 – 65 – 87 – 93– 99 – 120 – 130 – 135 – 140
Postorden: 22 – 56 – 65 – 43 – 93 – 99 – 87 – 135 – 130 – 140 – 120

Árboles AVL

Un Árbol AVL, originalmente llamado Árbol admisible, "admisible tree" (Krishnamoorthy, 2008) o "árbol Fibonacci" (Puntambekar, 2009) es un Árbol Binario de Búsqueda auto-balanceado que trata de mantenerse lo más balanceado posible mientras se realizan operaciones de inserción y eliminación. Fue propuesto por los matemáticos Adelson *et al.* (1962) en el artículo "An Algorithm for the organization of information" (Un algoritmo para la organización de la información).

Formalmente se define un Árbol AVL como un Árbol Binario de Búsqueda en el cual se debe cumplir la siguiente condición: "Para todo nodo del Árbol, la

altura del sub Árbol derecho e izquierdo no debe diferir en más de una unidad" (Caicedo *et al.*, 2010).

La condición anterior es la que define al término «factor de equilibrio».

$$FE = HRD - HRI$$

Por lo tanto, el factor de equilibro de todos los nodos de un Árbol AVL puede ser -1, 0 o 1. (Gómez De Silva y Ania, 2008).

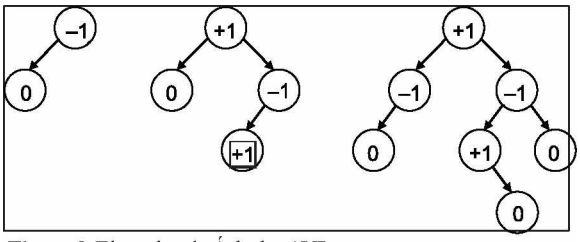

Figura 8. Ejemplos de Árboles AVL
Fuente: Cairó, O. (2007)

En la figura 8, Cairó hace una representación gráfica de Árboles AVL. Dentro de cada nodo se observa su factor de equilibrio (FE). Los Árboles AVL facilitan el manejo del balanceo pues sólo se tiene en cuenta la diferencia de alturas de los sub Árboles; es decir que el balanceo de Árboles puede realizarse localmente si sólo afecta a una porción del Árbol cuando se requieren cambios después que un elemento se inserta o elimina en un Árbol.

Reestructuración de un Árbol AVL

El proceso de inserción en un Árbol balanceado es sencillo pero con algunos detalles un poco complicados. Primero debe seguirse el camino de búsqueda del Árbol, hasta localizar el lugar donde hay que insertar el elemento. Luego se calcula su FE, que obviamente será 0, y regresamos por el camino de búsqueda calculando el FE de los distintos nodos. Si en alguno de los nodos se viola el criterio de equilibrio, entonces debe reestructurarse el Árbol.

El proceso termina al llegar a la raíz del Árbol, o cuando se realiza la restructuración del mismo, en cuyo caso no es necesario determinar el FE de los nodos restantes.

Reestructurar el Árbol significa rotar los nodos del mismo. En la figura 9, se observa que la rotación puede ser simple o compuesta. El primer caso involucra dos nodos y el segundo caso afecta a 3. Si la rotación es simple puede realizarse por las ramas derechas (DD) o por las ramas izquierdas (II). Si la rotación es compuesta puede realizarse por las ramas derecha e izquierda (DI) o por las ramas izquierda y derecha (ID).

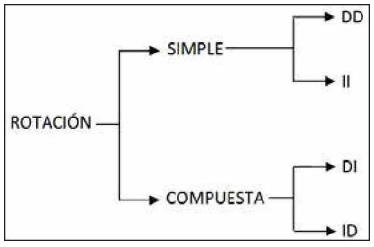

Figura 9. Tipos de rotaciones en un Árbol AVL
Fuente: Cairó, O. (2007)

Las rutinas de balanceo se pueden resumir en movimientos de apuntadores, y se ilustran en las figuras 10, 11, 12 y 13.

La nomenclatura que se usa en las imágenes para indicar la forma de balanceo es la siguiente: «raíz» es un apuntador que apunta a la raíz del sub Árbol por balancear, en todos los casos «raiz1» es un apuntador adicional que indica la dirección del nodo que está desbalanceado.

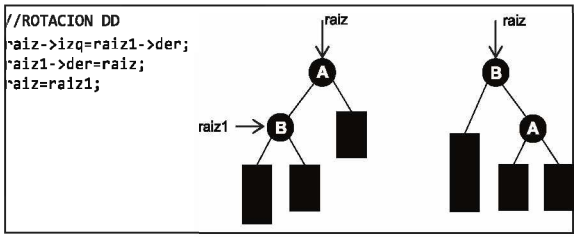

Figura 10. Rotación simple a la derecha (DD)
Fuente: Sisa, J. (2002)

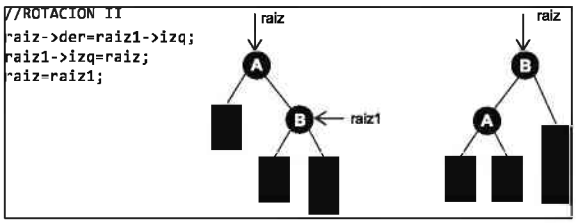

Figura 11. Rotación simple a la izquierda (II)
Fuente: Sisa, J. (2002)

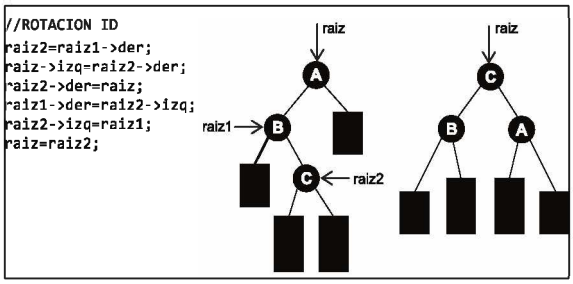

Figura 12. Rotación compuesta Izquierda Derecha (ID)
Fuente: Sisa, J. (2002)

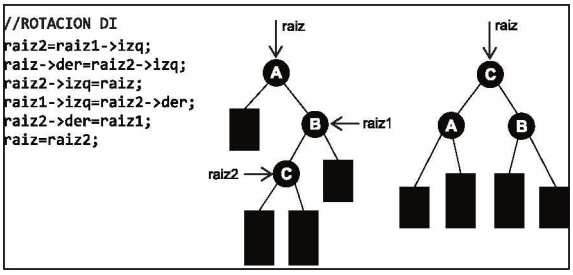

Figura13. Rotación compuesta Derecha Izquierda (DI)
Fuente: Sisa, J. (2002)

MARCO METODOLÓGICO

Caracterización o tipo del diseño de investigación

El diseño de la presente investigación es No experimental – transversal de nivel descriptivo.

Población y muestra

La población de datos contenidos en registros estará conformada por un total de 1 583 códigos de postulantes según el formato generado por el sistema de inscripción de los postulantes que se preparan en el Centro Preuniversitario de la Universidad Nacional Jorge Basadre Grohmann, Tacna. Se toma esta cantidad de registros basado en el número total de postulantes registrados en el proceso CEPU invierno 2015. De esta población se escogerá una muestra que se calcula de la siguiente manera ya que se trata de una población finita:

$$n=\frac{N\times Z^2\times p\times q}{e^2\times(N-1)+Z^2\times p\times q}$$

Los datos a considerar son los siguientes:
N: Tamaño de la población 1, 583
p: 50% Probabilidad a favor
q: (1- p) = 50% Probabilidad en contra
Z: Nivel de confianza de 95% el valor correspondiente es 1,96
e: 10% = 0,1

$$n=\frac{1583\times 1{,}6^2\times 0{,}5\times 0{,}5}{0{,}1^2\times(1583-1)+1{,}96^2\times 0{,}5\times 0{,}5}=90{,}60$$

Entonces, el tamaño de la muestra que se debe considerar es de 91 registros.

Tratamiento de datos

Para el tratamiento de los datos se utilizó la estadística descriptiva y las pruebas de contrastación de hipótesis. Se utilizó como herramienta el software Minitab para el procesamiento de los datos.
Se desarrolló una aplicación computacional para la generación automática de registros que contengan un campo de datos denominado código, el cual posee la misma estructura que se le asigna como código de un postulante a la Universidad Nacional Jorge Basadre Grohmann (UNJBG).
Este campo de código está formado por seis caracteres: los dos primeros caracteres están referidos al código de la Escuela Académico profesional a la que postula y los otros cuatro caracteres se forman por el correlativo de las fichas de matrícula del registro de postulantes de la UNJBG.
A continuación, se muestra la información que se utiliza para generar la codificación automática del campo código de los registros.

Limitaciones

En la presente investigación no se presentaron mayores limitaciones o inconvenientes ya que el análisis, diseño e implementación de los algoritmos de búsqueda en datos contenidos en las estructuras de datos sometidos a prueba no requieren de grandes equipos ni grandes recursos materiales. Fundamentalmente la investigación está centrada en la abstracción y capacidad de análisis del investigador.

RESULTADOS

Análisis de resultados

Muestra 1: Para la primera muestra se observa que el tiempo promedio de búsqueda de un registro almacenado en un Árbol AVL (7,028ms) es menor que el tiempo promedio que tarda en localizarlo en un Array lineal (12,917ms).
Muestra 2: Para la segunda muestra se advierte que el tiempo promedio de búsqueda de un registro almacenado en un Árbol AVL (6,4435ms) es menor que el tiempo promedio que tarda en localizarlo en un Array lineal (11,912ms).
Muestra 3: Para la tercera muestra se repara que el tiempo promedio de búsqueda de un registro almacenado en un Árbol AVL (6,322ms) es menor que el tiempo promedio que tarda en localizarlo en un Array lineal (11,870ms).

Respecto a la Desviación Estándar de los datos podemos observar:

Muestra 1: En promedio, los datos se alejan 1,028 unidades respecto a la media del tiempo de búsqueda en Árboles AVL. Así mismo, en promedio, los datos se alejan 5,064 unidades respecto a la media del tiempo de búsqueda en Arrays lineales.
Muestra 2: En promedio, los datos se alejan 0,9467 unidades respecto a la media del tiempo de búsqueda en Árboles AVL. Así mismo, en promedio, los datos se alejan 0,055 unidades respecto a la media del tiempo de búsqueda en Arrays lineales.
Muestra 3: En promedio, los datos se alejan 1,152 unidades respecto a la media del tiempo de búsqueda en Árboles AVL. Así mismo, en promedio, los datos se alejan 4,729 unidades respecto a la media del tiempo de búsqueda en Arrays lineales.

Del estadígrafo Coeficiente de Variación, se observa que los datos se encuentran más dispersos en los Arrays con respecto a los Árboles AVL. Esto se explica por la secuencia lineal en que se encuentran los datos en los Arrays, estos se almacenan de acuerdo al orden de llegada a la estructura. Sin embargo, en los Árboles AVL por no ser lineal su estructura se puede descartar grandes volúmenes de claves en una sola comparación.

Tiempos de demora para insertar las claves en las estructuras:

Respecto al tiempo de demora en insertar las claves en los registros se tiene:

Estadísticos descriptivos: t_ins_AVL, t_ins_array

Variable	Conteo total	Media	Desv.Est.	Varianza	CoefVar	Mediana
t_ins_AVL	15	16 728	1 809	3 272 960	10,82	17 135
t_ins_array	15	4656	799	638 436	17,16	4486

En promedio, el tiempo de inserción de 1 583 claves en Árboles AVL es de 16,728 milisegundos y para

insertarlos en el Array es de 4 656 milisegundos. Claramente, es más rápido el proceso de inserción de datos en los Arrays aunque esto no implica mayor velocidad de respuesta en los procesos de búsqueda tal como se muestra en el análisis anterior.

DISCUSIÓN DE RESULTADOS

Para la obtención de resultados se aplicó el procesamiento estadístico de los datos mediante la comparación de las medias de los tiempos de respuesta de los procesos de búsqueda de datos contenidos en un array y en un árbol AVL (para las tres muestras). Se ha generado las 3 muestras, cada una de 91 ítems que corresponde al muestreo que se hace de una población de 1 583 registros de postulantes a la UNJBG. Se genera el CÓDIGO del postulante que consta de 6 caracteres. Estos registros se almacenan en estructuras de datos estáticas tipo Array o Arreglo unidimensional y en estructuras de datos dinámicas del tipo Árbol AVL.

En la siguiente figura se muestra los datos generados y que se almacenan también en un archivo de texto para el posterior procesamiento y análisis estadístico.
Para validar la no redundancia de datos se utiliza la moda como estadístico y se obtuvo los resultados que confirman que no existen códigos repetidos.

Resumen de las pruebas de no repetición de códigos utilizando el software Minitab v17.

Estadísticos descriptivos: m1

Variable	Modo	N para moda
m1	*	0

Estadísticos descriptivos: m2

Variable	Modo	N para moda
m^2	*	0

Estadísticos descriptivos: m3

Variable	Modo	N para moda
m^3	*	0

A partir de estos datos generados que son en total 1 583 registros, se toman 3 muestras de tamaño 91 cada una.

CONCLUSIONES

Para los datos sometidos a prueba en esta investigación se concluye que la velocidad de respuesta para localizar un dato es menor cuando el dato se almacena en una estructura del tipo Árbol AVL debido a que se requiere menos comparaciones para acceder a este elemento previamente almacenado en memoria.

El tiempo que se tarda en localizar un dato contenido en una estructura de datos dinámico no lineal del tipo Árbol AVL es menor que el tiempo que se tarda en localizar el mismo dato contenido en una estructura estática del tipo Array lineal: 7,028 milisegundos y 12,917 milisegundos, respectivamente. Pero el tiempo que se requiere para insertar los datos en un Árbol AVL es mayor que el tiempo que se tarda en insertar los mismos datos en un Array lineal: 16,728 milisegundos y 4,656 milisegundos, respectivamente.

El número de comparaciones necesarias para localizar un dato contenido en un Árbol AVL es mucho menor que el número de comparaciones necesarias para localizar el mismo dato en un Arreglo lineal. En promedio 9,859 comparaciones y 783,4 comparaciones respectivamente.

REFERENCIAS BIBLIOGRÁFICAS

BisbaL, J. (2009). *Manual de Algorítmica. Recursividad, complejidad y diseño de algoritmos*. España: Editorial UOC.

Cairó, O. y GuardatI, S. (2007). *Estructura de Datos*. México: McGraw-Hill.

Caicedo, A., Wagner, G. y Méndez, R. (2010). *Introducción a la Teoría de Grafos*. Colombia: Ediciones Elizcom.

Drozdek, A. (2013). *Data Structures and Algorithms in C++*. USA: Cengage Learning.

Flores, R. (2005). *Algoritmos, estructuras de datos y programación orientada a objetos*. Bogotá, Colombia: Ecoe ediciones.

Garrido, A. y Fernández, J. (2006). *Abstracción y Estructuras de Datos en C++*. Madrid, España: Delta Publicaciones.

Gómez de Silva, G., y Ania, I. (2008). *Introducción a la Computación*. Mexico DF: Cengage Learning.

Hernández, R., Lázaro, J., Dormido, R. y Ros, S. (2000). *Estructura de datos y algoritmos*. Madrid: Prentice Hall.

Joyanes, L., Sanchez, L. y Zahonero, I. (2002). *Estructura de Datos en C++*. España: McGraw-Hill.

Krishnamoorthy, R. (2008). *Data Structures Using C and C++*. Prentice Hall.

Loomis, M. (1999). *Estructura de Datos y Organización de Archivos*. México: Prentice Hall Hispanoamericana.

Martínez, R. y Quiroga, E. (2002). *Estructura de datos: referencia práctica con orientación a objetos*. Cengage Learning.

Peláez, C. y Viso, E. (2008). *Introducción a las Ciencias de la Computación*. México: Las prensas de Ciencias – UNAM

Puntambekar, A. (2010). *Data Structures*. USA: Technical Publications Pune

Rodríguez, M., González, P. y Gomez, M. (2011). *Estructuras de Datos. Un enfoque moderno*. Madrid, España: Editorial Complutense.

Sisa, J. y Velez (2002). *Estructuras de datos y algoritmos*. Mexico: Prentice Hall.

Tenenbaum A. y Augenstein, M. (2004). *Estructura de Datos en C*. México: Prentice Hall Hispanoamericana.

Weiss, M. (2004). *Estructuras de Datos en Java*. Mexico: Pearson Educación.