
	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLE-001	<b>Página:</b> 1

## INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	ESTRUCTURA DE ALGORITMOS				
TÍTULO DE LA PRÁCTICA:	ÁRBOLES DE BÚSQUEDA				
NÚMERO DE PRÁCTICA:	05	AÑO LECTIVO:	2DO	NRO. SEMESTRE:	III
INTEGRANTE (s):				NOTA:	
JOSELIN SHARON CONDORI CATUNTA					
DOCENTE(s): Edson Luque Mamani					

DESARROLLO
<h1>Informe del Árbol AVL en Java</h1> <hr/> <h2>1. Objetivo del programa</h2> <p>Crear un árbol binario balanceado (AVL) que permita:</p> <ul style="list-style-type: none"> <li>- Insertar valores (en este caso, letras convertidas a ASCII).</li> <li>- Buscar elementos.</li> <li>- Obtener el valor mínimo y máximo del árbol.</li> <li>- Mostrar el padre y los hijos de un nodo dado.</li> </ul> <h2>2. Fundamentos teóricos</h2> <p>Un Árbol AVL es un tipo especial de árbol binario de búsqueda que se mantiene balanceado automáticamente mediante rotaciones después de cada inserción.</p> <p>Características:</p> <ul style="list-style-type: none"> <li>- La diferencia de altura entre subárboles izquierdo y derecho no debe ser mayor a 1.</li> <li>- Utiliza rotaciones simples y dobles para mantenerse balanceado.</li> </ul> <h2>3. Estructura del programa</h2>

### 3.1 Clase NodoAVL

Define la estructura de cada nodo del árbol:

- valor: almacena el número (código ASCII de una letra).
- altura: se actualiza para mantener el balance del árbol.
- izquierdo y derecho: apuntan a los hijos.

### 3.2 Clase ArbolAVL

Contiene los métodos principales:

- insert(): inserta un nodo y realiza rotaciones si es necesario.

```
NodoAVL insert(NodoAVL nodo, int valor) {
    if (nodo == null)
        return new NodoAVL(valor);

    if (valor < nodo.valor)
        nodo.izquierdo = insert(nodo.izquierdo, valor);
    else if (valor > nodo.valor)
        nodo.derecho = insert(nodo.derecho, valor);
    else
        return nodo;

    nodo.altura = 1 + Math.max(altura(nodo.izquierdo), altura(nodo.derecho));
    int balance = factorEquilibrio(nodo);

    // Rotaciones
    if (balance > 1 && valor < nodo.izquierdo.valor)
        return rotacionDerecha(nodo);

    if (balance < -1 && valor > nodo.derecho.valor)
        return rotacionIzquierda(nodo);

    if (balance > 1 && valor > nodo.izquierdo.valor) {
        nodo.izquierdo = rotacionIzquierda(nodo.izquierdo);
        return rotacionDerecha(nodo);
    }

    if (balance < -1 && valor < nodo.derecho.valor) {
        nodo.derecho = rotacionDerecha(nodo.derecho);
        return rotacionIzquierda(nodo);
    }

    return nodo;
}
```

- search(): busca un valor.

```
boolean search(NodoAVL nodo, int valor) {  
    if (nodo == null) return false;  
    if (nodo.valor == valor) return true;  
    return valor < nodo.valor ? search(nodo.izquierdo, valor) : search(nodo.derecho, valor);  
}
```

- getMin() / getMax(): obtienen los valores extremos.

```
int getMin(NodoAVL nodo) {  
    while (nodo.izquierdo != null)  
        nodo = nodo.izquierdo;  
    return nodo.valor;  
}  
  
int getMax(NodoAVL nodo) {  
    while (nodo.derecho != null)  
        nodo = nodo.derecho;  
    return nodo.valor;  
}
```

- parent() / son(): muestran relaciones padre-hijo.

```
NodoAVL parent(NodoAVL nodo, int valor, NodoAVL padre) {  
    if (nodo == null)  
        return null;  
    if (nodo.valor == valor)  
        return padre;  
    return valor < nodo.valor ?  
        parent(nodo.izquierdo, valor, nodo) :  
        parent(nodo.derecho, valor, nodo);  
}  
  
void son(NodoAVL nodo, int valor) {  
    if (nodo == null) return;  
  
    if (nodo.valor == valor) {  
        if (nodo.izquierdo != null)  
            System.out.println("Hijo izquierdo: " + nodo.izquierdo.valor);  
        if (nodo.derecho != null)  
            System.out.println("Hijo derecho: " + nodo.derecho.valor);  
        return;  
    }  
  
    if (valor < nodo.valor)  
        son(nodo.izquierdo, valor);  
    else  
        son(nodo.derecho, valor);  
}
```

- Factor de equilibrio: El factor de equilibrio en un árbol AVL es una medida que indica cuán balanceado está un nodo.



$\text{factorEquilibrio} = \text{altura del subárbol izquierdo} - \text{altura del subárbol derecho}$

```
int factorEquilibrio(NodoAVL nodo) {  
    if (nodo == null)  
        return 0;  
    return altura(nodo.izquierdo) - altura(nodo.derecho);  
}
```

### 3.3 Clase Main

En el main, se convierte cada letra de una palabra (por ejemplo "UNSA") en su valor ASCII y se inserta en el árbol.

```
1 public class Main {  
2     public static void main(String[] args) {  
3         String palabra = "UNSA";  
4         ArbolAVL arbol = new ArbolAVL();  
5  
6         NodoAVL raiz = null;  
7         System.out.println("Insertando valores ASCII de: " + palabra);  
8  
9         for (char c : palabra.toCharArray()) {  
10             int ascii = (int) c;  
11             System.out.println("Letra: " + c + " -> ASCII: " + ascii);  
12             raiz = arbol.insert(raiz, ascii);  
13         }  
14  
15         System.out.println("\n- Búsqueda de 'S' (83): " + arbol.search(raiz, 83));  
16         System.out.println("- Mínimo: " + arbol.getMin(raiz));  
17         System.out.println("- Máximo: " + arbol.getMax(raiz));  
18  
19         NodoAVL padre = arbol.parent(raiz, 78, null);  
20         if (padre != null)  
21             System.out.println("- Padre de 78: " + padre.valor);  
22         else  
23             System.out.println("- Padre de 78: No encontrado");  
24  
25         System.out.println("- Hijos de 85:");  
26         arbol.son(raiz, 85);  
27     }  
28 }
```

	<p align="center"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLE-001	<b>Página:</b> 5

## 4. Pruebas realizadas

- Inserté la palabra "UNSA" y verifiqué el árbol AVL.
- Probé la búsqueda del número ASCII correspondiente a 'S'.
- Usé los métodos getMin() y getMax() para confirmar los extremos del árbol.
- Probé parent() para hallar el padre del nodo 'N'.
- Usé son() para ver los hijos del nodo 'U'.

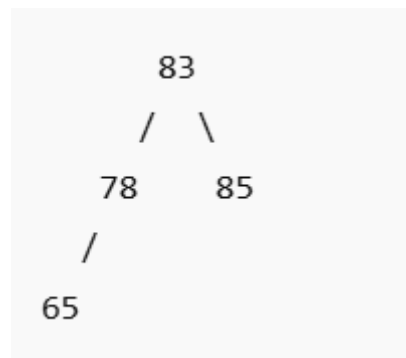
## 5. Resultados esperados

```

19      NodoAVL padre = arbol.parent(raiz, 78, null);
...
Console x
<terminated> Main [Java Application] C:\Users\ASUS\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.f
Insertando valores ASCII de: UNSA
Letra: U -> ASCII: 85
Letra: N -> ASCII: 78
Letra: S -> ASCII: 83
Letra: A -> ASCII: 65



- Búsqueda de 'S' (83): true
- Mínimo: 65
- Máximo: 85
- Padre de 78: 83
- Hijos de 85:

```



## 6. Conclusiones

- Aprendí cómo funcionan los árboles AVL y cómo se balancean automáticamente.
- Entendí que convertir caracteres en números facilita operaciones lógicas.
- Las rotaciones mantienen eficiente la búsqueda y la inserción.
- El código me ayudó a practicar estructuras de datos avanzadas y recursividad.

	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p align="center"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 6</p>

## 7. Herramientas utilizadas

- Lenguaje: Java
- Entorno: Visual Studio Code / IntelliJ / Eclipse
- Compilador: JDK versión 17.0.2