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

INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	ESTRUCTURA DE ALGORITMOS				
TÍTULO DE LA PRÁCTICA:	ÁRBOLES DE BÚSQUEDA				
NÚMERO DE PRÁCTICA:	04	AÑO LECTIVO:	2DO	NRO. SEMESTRE:	III
INTEGRANTE (s):				NOTA:	
JOSELIN SHARON CONDORI CATUNTA					
DOCENTE(s): Edson Luque Mamani					

DESARROLLO
<h1>Implementación de Árboles de Búsqueda Binarios (BST)</h1> <h2>Introducción</h2> <p>En este trabajo se implementó un Árbol de Búsqueda Binario (BST) para manejar caracteres de una palabra ingresada en mayúsculas. Cada nodo del árbol representa un carácter, y el árbol se construye considerando el valor decimal de su código ASCII para organizar los nodos.</p> <h2>Objetivo</h2> <p>El objetivo principal fue desarrollar un BST que permita realizar las siguientes operaciones básicas:</p> <ul style="list-style-type: none"> insertar(): Agregar un nuevo nodo en el árbol. buscar(): Buscar un nodo con un valor específico. obtenerMin(): Obtener el nodo con el valor mínimo (menor código ASCII). obtenerMax(): Obtener el nodo con el valor máximo (mayor código ASCII). padre(): Encontrar el nodo padre de un nodo dado. hijo(): Obtener los hijos izquierdo y derecho de un nodo dado.

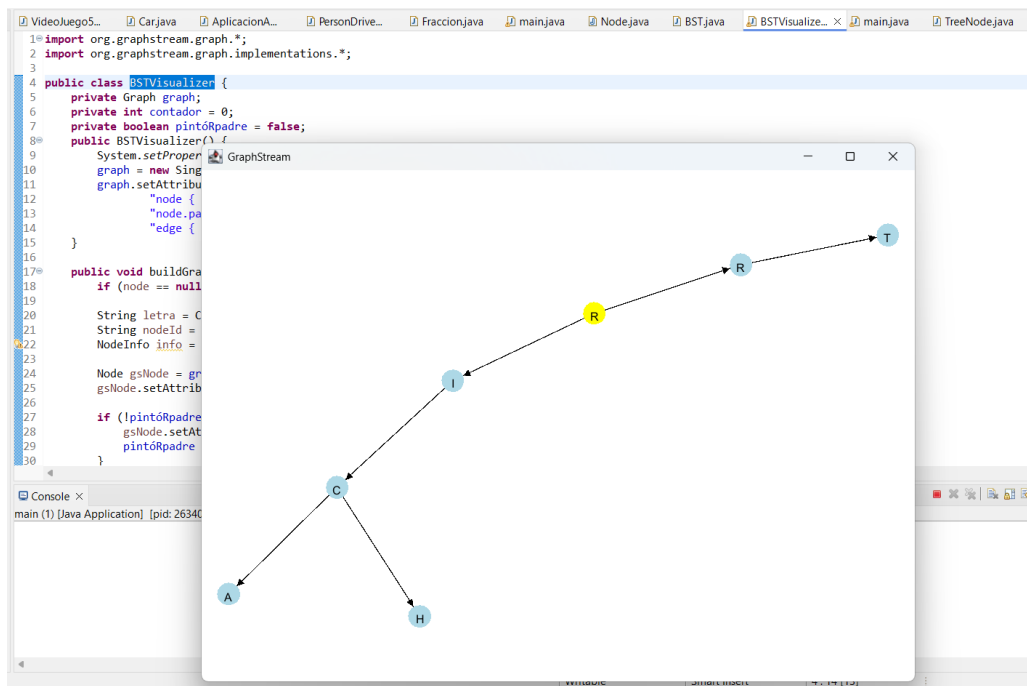
Metodología



1. **Entrada:** Se recibe una sola palabra en mayúsculas.
2. **Construcción del BST:** Para cada letra de la palabra, se inserta un nodo en el árbol. La posición de cada nodo depende del valor decimal de su código ASCII, lo que permite mantener el orden binario.
3. **Operaciones implementadas:**
 - o insertar(): Inserta nodos manteniendo la propiedad del BST.
 - o buscar(): Retorna si un nodo con valor dado existe en el árbol.
 - o obtenerMin(): Retorna el nodo con el menor valor ASCII.
 - o obtenerMax(): Retorna el nodo con el mayor valor ASCII.
 - o padre(): Busca y retorna el padre de un nodo específico.
 - o hijo(): Retorna los hijos izquierdo y derecho de un nodo específico.

Resultados

Se realizó la construcción del árbol con la palabra "**RICHART**", cuyos códigos ASCII se usaron para ordenar los nodos.

- Mínimo: 65 (letra: A)
- Máximo: 84 (letra: T)
- Búsqueda: El nodo 'A' (65) se encontró correctamente.
- Nodo 'R' (82) identificado como nodo padre.
- Hijos de 'R': Izquierdo: 'I' (73), Derecho: 'T' (84).



	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 3</p>

Visualización

Para mejorar la comprensión, se visualizó el BST utilizando la librería **GraphStream**, donde:

- El nodo padre se resaltó en amarillo.
- Los nodos hijos se mostraron en azul claro.
- Se dibujaron las conexiones dirigidas entre padres e hijos.

Conclusiones

- La implementación del BST usando el valor ASCII permitió un orden consistente para la inserción y búsqueda.
- Las operaciones básicas funcionaron correctamente, demostrando la utilidad del BST para organizar y consultar datos.
- La visualización gráfica fue un apoyo valioso para validar la estructura del árbol.
- La principal dificultad estuvo en manejar la visualización personalizada de nodos, lo cual se solucionó con estilos CSS específicos para nodos padres e hijos.