

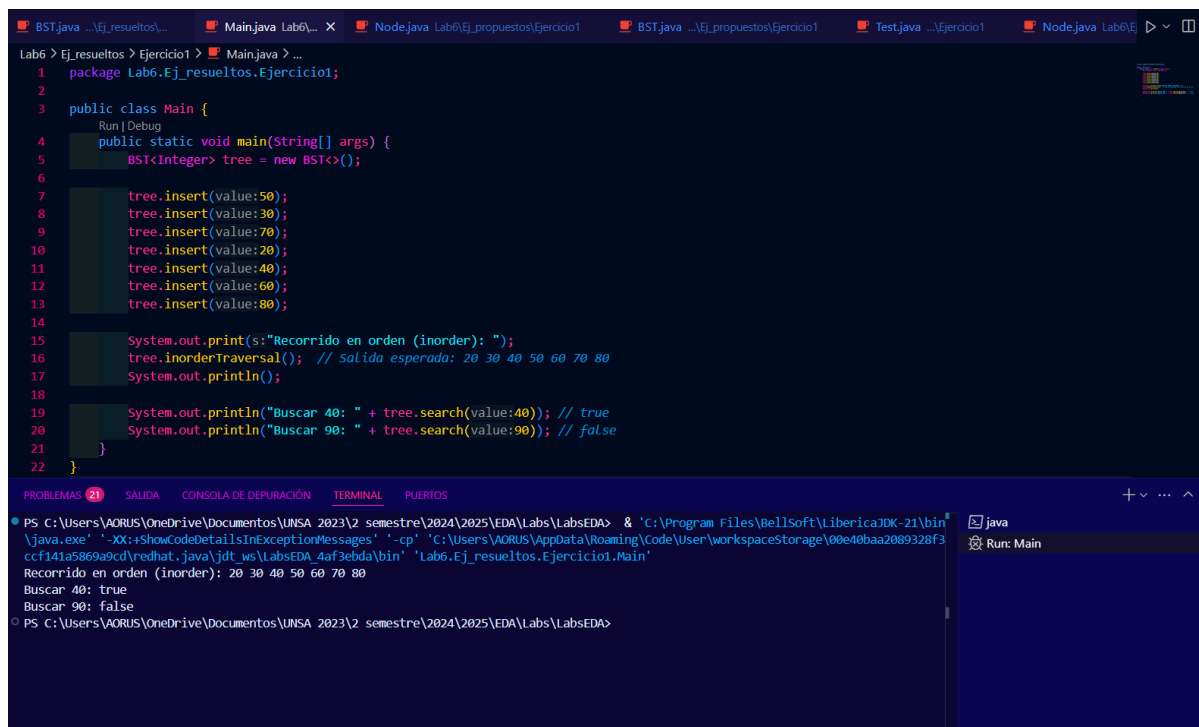
	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

## INFORME DE LABORATORIO

INFORMACIÓN BÁSICA					
ASIGNATURA:	ESTRUCTURA DE DATOS Y ALGORITMOS				
TÍTULO DE LA PRÁCTICA:	ÁRBOL BINARIO SIMPLE Y DE BÚSQUEDA				
NÚMERO DE PRÁCTICA:	06	AÑO LECTIVO:	2025 – A	NRO. SEMESTRE:	Tercero III
FECHA DE PRESENTACIÓN	16/06/2025	HORA DE PRESENTACIÓN	23:59		
INTEGRANTE (s): Davila Flores Mathias Dario				NOTA:	
DOCENTE(s): <ul style="list-style-type: none"> <li>Mg. Ing. Rene Alonso Nieto Valencia.</li> <li>ENLACE GITHUB: <a href="https://github.com/mathiasddf/LabsEDA">https://github.com/mathiasddf/LabsEDA</a></li> </ul>					

SOLUCIÓN Y RESULTADOS
<p><b>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</b></p> <p><b>a. Ejercicios Resueltos:</b></p> <p><b>i. Ejercicio 1:</b> En un editor Java, realizar la integración de los siguientes ejercicios, revisar y mostrar los resultados obtenidos y realizar una explicación del funcionamiento de forma concreta y clara.</p> <ul style="list-style-type: none"> <li>- Implementación de Nodo Genérico. <code>public class Node { private T data; private Node left; private Node right; }</code></li> <li>- Implementación Clase BST Genérico. <code>public class BST&gt; { // Ingrese codigo aqui }</code></li> </ul>

	<p style="text-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>	
<p style="text-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> 2</p>



```

1 package Lab6.Ej_resueltos.Ejercicio1;
2
3 public class Main {
4     public static void main(String[] args) {
5         BST<Integer> tree = new BST<>();
6
7         tree.insert(value:50);
8         tree.insert(value:30);
9         tree.insert(value:70);
10        tree.insert(value:20);
11        tree.insert(value:40);
12        tree.insert(value:60);
13        tree.insert(value:80);
14
15        System.out.print(s:"Recorrido en orden (inorder): ");
16        tree.inorderTraversal(); // Salida esperada: 20 30 40 50 60 70 80
17        System.out.println();
18
19        System.out.println("Buscar 40: " + tree.search(value:40)); // true
20        System.out.println("Buscar 90: " + tree.search(value:90)); // false
21    }
22 }

```

PROBLEMAS (21) SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

PS C:\Users\AORUS\OneDrive\Documents\UNISA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\BellSoft\LibericaJDK-21\bin\java.exe' -XX:+ShowCodeDetailsInExceptionMessages -cp "c:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\60e40baa2089328f3ccf141a5869a9cd\redhat.java\jdk\_ws\LabsEDA\_4af3ebda\bin" 'Lab6.Ej\_resueltos.Ejercicio1.Main'

Recorrido en orden (inorder): 20 30 40 50 60 70 80  
 Buscar 40: true  
 Buscar 90: false



PS C:\Users\AORUS\OneDrive\Documents\UNISA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>

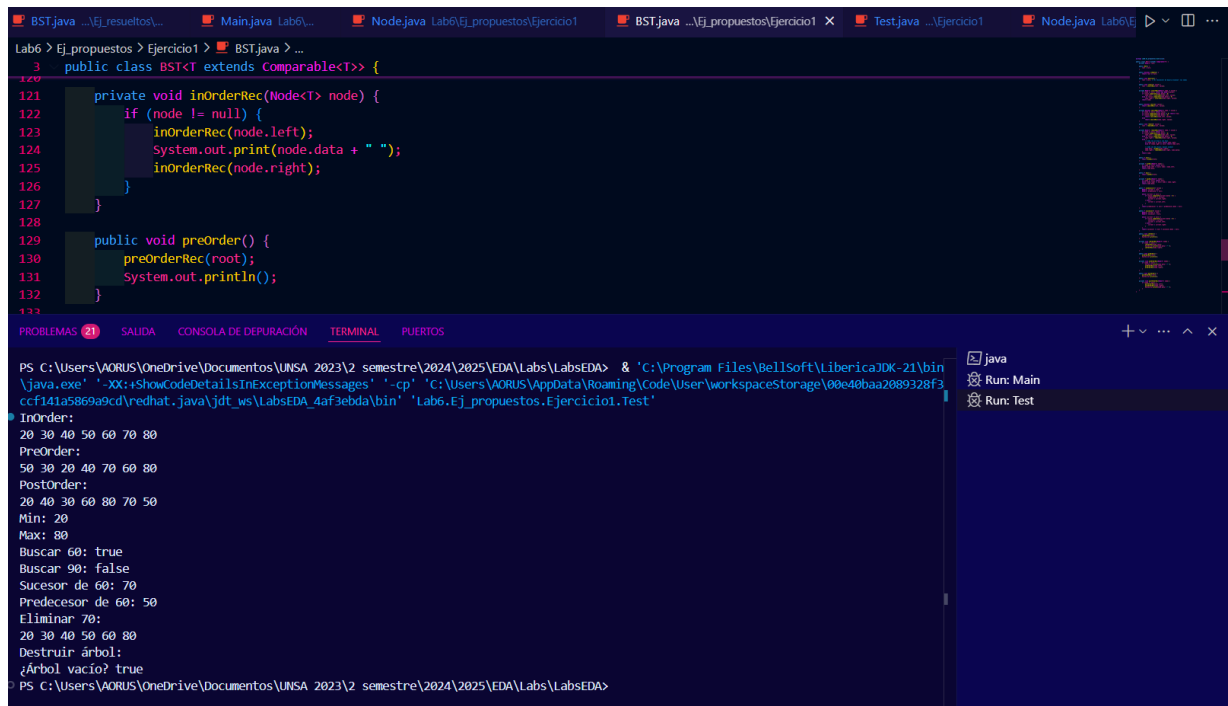
## b. Ejercicios Propuestos:

- i. **EJERCICIO 1:** Implementar un arbol binario de búsqueda ABB o BST agregando todas las operaciones: destroy(), isEmpty(), insert(x), remove(x), search(x), Min(), Max(), Predecesor(), Sucesor(), InOrder, PostOrder(), PreOrder(). Implementar una clase Test para probar los métodos y mostrar los resultados. utilizando clases y métodos genéricos.

Descripción : Se desarrolló una clase genérica BST<T> que implementa un Árbol Binario de Búsqueda con operaciones como: insert, remove, search, min, max, predecesor, sucesor, inOrder, postOrder, preOrder, isEmpty y destroy. También se creó una clase Test para probar todas las funcionalidades mediante ejemplos prácticos.

Objetivo: Implementar un árbol binario de búsqueda genérico en Java con todas sus operaciones fundamentales, y verificar su correcto funcionamiento mediante una clase de prueba.

	<p style="text-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>	
<p style="text-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> 3</p>



```

public class BST<T extends Comparable<T>> {
    private void inOrderRec(Node<T> node) {
        if (node != null) {
            inOrderRec(node.left);
            System.out.print(node.data + " ");
            inOrderRec(node.right);
        }
    }

    public void preOrder() {
        preOrderRec(root);
        System.out.println();
    }
}

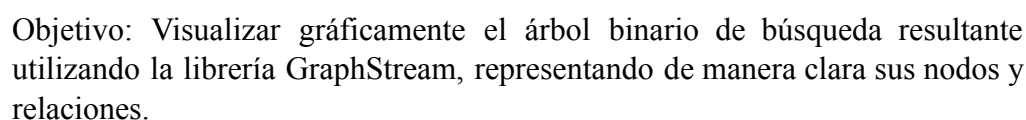
PS C:\Users\AORUS\OneDrive\Documents\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA> & 'C:\Program Files\BellSoft\JDK-21\bin\java.exe' -XX:+ShowCodeDetailsInExceptionMessages -cp 'C:\Users\AORUS\AppData\Roaming\Code\User\workspaceStorage\00e40baa2089328f3ccf141a5969a9cd\redhat.java\jdt_ws\LabsEDA_4af3ebda\bin' 'Lab6.Ej_propuestos.Ejercicio1.Test'

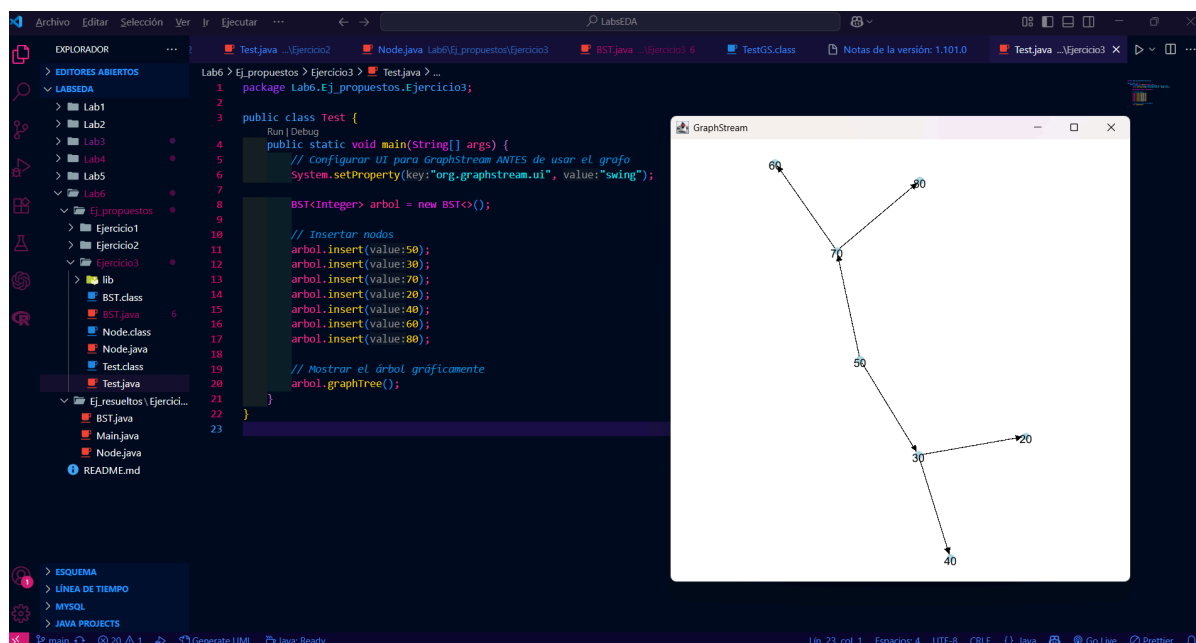
InOrder:
20 30 40 50 60 70 80
PreOrder:
50 30 20 40 70 60 80
PostOrder:
20 40 30 60 80 70 50
Min: 20
Max: 80
Buscar 60: true
Buscar 90: false
Sucesor de 60: 70
Predecesor de 60: 50
Eliminar 70:
20 30 40 50 60 80
Destruir árbol:
¿Árbol vacío? true
PS C:\Users\AORUS\OneDrive\Documents\UNSA 2023\2 semestre\2024\2025\EDA\Labs\LabsEDA>
  
```

ii. **EJERCICIO 2:** Implementar una árbol binario de búsqueda ABB o BST donde ingrese por teclado una palabra y como resultado debe mostrar el arbol de acuerdo al valor decimal correspondiente de su código ASCII. Utilizando los métodos del ejercicio anterior.

Descripción: Se desarrolló un programa que solicita al usuario ingresar una palabra por teclado. Cada letra es transformada a su código ASCII y esos valores son insertados en un BST, reutilizando los métodos del ejercicio anterior. Posteriormente, se muestra el recorrido del árbol para visualizar su estructura.

Objetivo: Construir un árbol binario de búsqueda a partir de los valores ASCII de una palabra ingresada por teclado, utilizando los métodos ya implementados en el ejercicio anterior..





## II. SOLUCIÓN DEL CUESTIONARIO

a. ¿Cuáles fueron las dificultades que encontraste al desarrollar los ejercicios propuestos? por ejemplo, poca documentación, complejidad del lenguaje, etc.

- **Integración de librerías externas:** Una de las principales dificultades fue la correcta integración de la librería GraphStream. Al principio, surgieron errores como `MissingDisplayException` debido a que no se encontraba el paquete de visualización (`gs-ui-swing`) o no se configuró correctamente la propiedad del sistema (`org.graphstream.ui`).
- **Falta de documentación clara:** La documentación de GraphStream en su versión 2.0 es limitada y no siempre especifica para tareas comunes como visualizar árboles binarios, lo cual exigió buscar ejemplos en foros, GitHub o documentación de versiones anteriores.
- **Estructura de carpetas y paquetes en Java:** Otra complicación fue la estructura del proyecto y la ejecución de clases con nombres calificados (`Lab6.Ej_propuestos.Ejercicio3.Test`). Fue necesario compilar y ejecutar

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

con rutas y classpaths específicos, lo que puede resultar confuso sin experiencia previa.

- **Complejidad de la recursividad en Java:** La implementación de métodos recursivos como insert, remove, buildGraph y recorridos en orden también puede ser un reto para quienes no dominan bien la lógica recursiva.
- **Errores de ejecución vs. compilación:** Algunos errores no aparecieron en tiempo de compilación, sino al ejecutar (por ejemplo, cuando display() fallaba), lo cual complicó el proceso de depuración.

**b. ¿Explique cómo es el algoritmo que implemento para obtener el árbol binario de búsqueda con la librería Graph Stream? Recuerdar que puede agregar operaciones sobre la clase BST.**

El algoritmo implementado para visualizar el Árbol Binario de Búsqueda con GraphStream se estructura en dos partes principales:

**a) Preparación y configuración del grafo**

Se crea un objeto SingleGraph, se ajustan propiedades visuales y se inicia la visualización:



```
Graph graph = new SingleGraph("Árbol BST");
graph.setStrict(false);
graph.setAutoCreate(true);
graph.setAttribute("ui.stylesheet", "node { fill-color: lightblue; text-size: 18px; }");
System.setProperty("org.graphstream.ui", "swing");
graph.display();
```

Esto prepara el entorno gráfico donde se mostrará el árbol.

**b) Construcción recursiva del grafo a partir del árbol**

Se utiliza un método recursivo buildGraph(Graph graph, Node<T> current, Node<T> parent) que:

- Agrega cada nodo del árbol al grafo como un Node de GraphStream.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-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> 7</p>

- Si el nodo tiene un padre, se crea una Edge (arista) dirigida desde el padre hacia el hijo.
- Repite el proceso recursivamente para los subárboles izquierdo y derecho.

```
private void buildGraph(Graph graph, Node<T> current,
Node<T> parent) {

    String currId = current.data.toString();

    graph.addNode(currId).setAttribute("ui.label", currId);

    if (parent != null) {

        String parentId = parent.data.toString();

        String edgeId = parentId + "-" + currId;

        graph.addEdge(edgeId, parentId, currId, true);

    }

    if (current.left != null)

        buildGraph(graph, current.left, current);

    if (current.right != null)

        buildGraph(graph, current.right, current);

}
```

Este método permite representar gráficamente la estructura jerárquica del BST, conectando visualmente los nodos con sus respectivos hijos.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-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> 8</p>

### III. CONCLUSIONES

- La implementación de un árbol binario de búsqueda (BST) utilizando clases genéricas en Java permitió comprender a fondo el funcionamiento interno de estructuras jerárquicas, así como las operaciones básicas de inserción, eliminación, búsqueda y recorrido.
- Al desarrollar los tres ejercicios, se logró aplicar de manera progresiva conceptos teóricos en situaciones prácticas: desde la implementación de un BST básico, hasta su aplicación con datos basados en códigos ASCII y finalmente su visualización gráfica.
- El uso de la librería externa **GraphStream** facilitó la representación visual del árbol, permitiendo una mejor comprensión de la estructura generada. Sin embargo, también se presentaron dificultades debido a la escasa documentación en español y la necesidad de configurar correctamente el entorno y los paquetes UI necesarios.
- La experiencia reforzó la importancia de la modularización del código, al reutilizar las operaciones del BST en todos los ejercicios, así como la utilidad de aplicar principios de programación genérica para lograr soluciones flexibles y escalables.
- Finalmente, este laboratorio fortaleció habilidades clave como el manejo de bibliotecas externas, la resolución de errores de compilación y ejecución, y la adaptación de estructuras de datos clásicas a requerimientos específicos, como el ingreso de texto o la visualización gráfica.

### REFERENCIAS Y BIBLIOGRAFÍA

- Deitel, P. J., & Deitel, H. M. (2014). *Java: Cómo programar* (10.<sup>a</sup> ed.). Pearson Educación.
- Goodrich, M. T., Tamassia, R., & Goldwasser, M. H. (2013). *Estructuras de datos y algoritmos en Java* (6.<sup>a</sup> ed.). Wiley.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms* (3rd ed.). MIT Press.
- GraphStream Team. (2023). *GraphStream – A dynamic graph library*. <https://graphstream-project.org/>
- Oracle. (2024). *The Java™ tutorials*. <https://docs.oracle.com/javase/tutorial/>
- Stack Overflow. (n.d.). *Various discussions and solutions for configuring external libraries in Java projects*. <https://stackoverflow.com>