**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):**  JOSELIN SHARON CONDORI CATUNTA | | | | **NOTA:** |  |
| **DOCENTE(s):**  Edson Luque Mamani | | | | | |

|  |
| --- |
| **DESARROLLO** |
| Informe del Árbol AVL en Java **1. Objetivo del programa** Crear un árbol binario balanceado (AVL) que permita: - Insertar valores (en este caso, letras convertidas a ASCII). - Buscar elementos. - Obtener el valor mínimo y máximo del árbol. - Mostrar el padre y los hijos de un nodo dado. **2. Fundamentos teóricos** Un Árbol AVL es un tipo especial de árbol binario de búsqueda que se mantiene balanceado automáticamente mediante rotaciones después de cada inserción.  Características: - La diferencia de altura entre subárboles izquierdo y derecho no debe ser mayor a 1. - Utiliza rotaciones simples y dobles para mantenerse balanceado. **3. Estructura del programa**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.    - search(): busca un valor.    - getMin() / getMax(): obtienen los valores extremos.    - parent() / son(): muestran relaciones padre-hijo.     * Factor de equilibrio: El factor de equilibrio en un árbol AVL es una medida que indica cuán   balanceado está un nodo.  factorEquilibrio = altura del subárbol izquierdo - altura del subárbol 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.   **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**  **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. **7. Herramientas utilizadas** - Lenguaje: Java - Entorno: Visual Studio Code / IntelliJ / Eclipse - Compilador: JDK versión 17.0.2 |