Laboratorio 2: Sequential File vs AVL File

Profesor: Heider Sanchez | ACLs: Sebastian Loza, Ana Maria Accilio

1. Introducción

El propósito de este laboratorio es implementar y comparar el desempeño de dos estructuras de almacenamiento de datos en memoria secundaria:

1. Archivo Secuencialmente Ordenado (Sequential File)
2. Archivo organizado como Árbol Binario de Búsqueda Balanceado (AVL File)

Los estudiantes analizarán el tiempo de acceso y evaluación de eficiencia en las operaciones fundamentales:

- insert(record)
- search(key)
- remove(key)
- rangeSearch(init_key, end_key)

Requerimientos de implementación:

- La implementación de ambos métodos será en **Python**.
- Utilizar **archivos binarios** con registro de longitud fija.
- Medición y comparación de los **tiempos de acceso** en ambos métodos.
- Usar la siguiente **estructura de registro** para representar una venta de producto:

Campo	Tipo de Dato
ID de la venta	`int (4 bytes)`
Nombre producto	`string (30 bytes)`
Cantidad vendida	`int (4 bytes)`
Precio unitario	`float (4 bytes)`
Fecha de venta	`string (YYYY-MM-DD)`

2. Desarrollo

Parte 1: Implementación del Archivo Secuencial

1. Carga de datos desde un archivo csv (sales_dataset.csv).
2. Función para **insertar** nuevos registros usando espacio auxiliar. El archivo original debe reconstruirse con el espacio extra cuando este ultimo exceda k registros.

3. Función de **búsqueda secuencial** por ID de venta.
4. Función para **eliminar** un registro marcándolo como eliminado. En la reconstrucción del archivo de datos no se debe considerar los registros eliminados lógicamente.
5. Función para la **búsqueda por rango** el cual debe retornar todos los elementos entre un rango especificado.

Parte 2: Implementación del Archivo AVL

1. Carga de datos desde un archivo csv (sales_dataset.csv).
2. Función para **insertar** nuevos registros actualizando correctamente los punteros de jerarquía.
3. Función para **buscar** una venta específica utilizando la estructura del AVL.
4. Función para **eliminar** un registro y reestructurar el BST en el archivo.
5. Función para la **búsqueda por rango** el cual debe retornar todos los elementos entre un rango especificado.

Parte 3: Evaluación de Desempeño

- Medir el tiempo de ejecución para:
    a. **Inserción de registros**
    b. **Búsqueda de ventas especificas**
    c. **Búsqueda por rango de ventas**
    d. **Eliminación de registros**
- Comparar los tiempos de acceso y **analizar los resultados**, utilizar una gráfica.

---

3. Análisis y Conclusión

- □ Comparar los tiempos obtenidos para cada método.
- □ Evaluar en qué escenarios conviene usar cada método.
- □ Reflexionar sobre la importancia de la organización de archivos en almacenamiento eficiente.

---

4. Entregable

- **Código fuente en Python** con las implementaciones.
- **Informe con los resultados del experimento** incluyendo el analisis y la discución.