Informe Trabajo Práctico N°2

Problema 2: Temperaturas_DB

Enunciado:

Kelvin Kevin, debe consultar frecuentemente en una base de datos la temperatura del planeta tierra dentro de un rango de fechas. A su vez, el científico registra y agrega **mediciones** a la base de datos. Esta medición está conformada por el valor de temperatura en ºC (flotante) y la fecha de registro, la cual deberá ser ingresada como “dd/mm/aaaa” (string). (Internamente sugerimos emplear objetos de tipo *datetime*).
Se solicita implementar una base de datos en memoria principal denominada **Temperaturas_DB,** utilizando un **árbol AVL** para el almacenamiento de datos, que permita realizar las siguientes operaciones:
guardar_temperatura(temperatura, fecha)
devolver_temperatura(fecha)

max_temp_rango(fecha1, fecha2)

min_temp_rango(fecha1, fecha2)

temp_extremos_rango(fecha1, fecha2)

borrar_temperatura(fecha)

devolver_temperaturas(fecha1, fecha2)

cantidad_muestras()

Solución:

Para resolver el problema, se desarrolló una estructura modular basada en la implementación de un **árbol AVL**, que permite almacenar y consultar de forma eficiente las temperaturas registradas en distintas fechas.

El sistema se organizó en los siguientes módulos:

- **`AVL.py`**: define la clase `NodoAVL`, que representa cada nodo del árbol con su fecha, temperatura, hijos izquierdo y derecho, y su altura para mantener el equilibrio.
- **`temperaturas_AVL.py`:** implementa la clase Temperaturas_DB, que realiza operaciones de inserción, búsqueda, eliminación y consulta por rangos de fechas, aprovechando las propiedades del árbol AVL. Aplica rotaciones simples y dobles para conservar el balance del árbol

- **main.py:** se encarga de cargar los datos desde un archivo de texto muestras , crea la base de datos y muestra resultados.

Análisis de complejidad:

Métodos implementados	Orden de complejidad Big-O	Análisis
guardar_temperatura(temperatura, fecha)	O (log n)	Inserta un nodo recorriendo desde la raíz hasta la posición correcta (altura ≈ log n). Las rotaciones para mantener el balance son O(1).
devolver_temperatura(fecha)	O (log n)	Es una búsqueda por clave(fecha) donde compara la clave con la del nodo actual y decide si va a la izquierda o derecha. Altura del AVL ≈ log n
max_temp_rango(fecha1, fecha2)	O (n)	Revisa todos los nodos dentro de un rango hasta encontrar el máximo, por lo que en el peor caso recorrería todos los nodos.
min_temp_rango(fecha1, fecha2)	O(n)	Similar al caso anterior, solo que en este caso buscaría el mínimo.
temp_extremos_rango(fecha1, fecha2)	O(n)	Llama secuencialmente a min_temp_rango y max_temp_rango; la complejidad se mantiene.
borrar_temperatura(fecha)	O (log n)	La eliminación (_borrar) busca el nodo, lo reemplaza si tiene dos hijos y realiza rebalanceo. Todas estas operaciones dependen de la altura del árbol.
devolver_temperaturas(fecha1, fecha2)	O (log n + k)	Primero busca donde empieza el rango O (log n), luego recorre las temperaturas que están dentro del rango O(k)
cantidad_muestras()	O(1)	devuelve el contador de las temperaturas guardadas.