PROBLEMA 2

Para el desarrollo del problema 2, el cual requería de la implementación de una base de datos en memoria principal "**Temperaturas_DB**"**,** la estructura seleccionada fue un árbol AVL.

En la carpeta "apps" se encuentra disponible un código que prueba y verifica el correcto funcionamiento de los métodos de **Temperaturas_DB.**

Tabla de complejidades

A continuación se presenta una tabla con el análisis del orden de complejidad Big-O para cada uno de los métodos implementados en la clase, explicando brevemente el análisis de los mismos.

MÉTODO	ORDEN DE COMPLEJIDAD
guardar_temperatura	**O(log n)** La operación dominante es la inserción de un nuevo par (fecha, temperatura) en el ArbolAVL. Dado que el árbol AVL mantiene su equilibrio, la altura del árbol es siempre logarítmica respecto al número n de mediciones. Por lo tanto, agregar un elemento tiene complejidad O(log n). Las validaciones y la conversión de fecha son operaciones de tiempo constante que no afectan la complejidad del método.
devolver_temperatura	**O(log n)** Similar a guardar_temperatura, la operación más costosa es la búsqueda de una clave (fecha) en el ArbolAVL. Las búsquedas en un AVL tienen una complejidad de O(log n), por lo que esto determina la complejidad del método.
max_temp_rango	**O(log n + k)** Este método hace uso del método encontrar_min_max_rango del ArbolAVL, el cual está diseñado para visitar sólo los nodos qué caen dentro del rango de fechas, estén estas en el árbol o no: primero realiza una búsqueda O(log n) para encontrar las fechas solicitadas, y luego visita los k nodos en el rango para encontrar el mínimo y máximo. Por lo tanto, este determina la complejidad del método de la base de datos, con una complejidad O(log n +k), donde k es el número de temperaturas (valores) dentro del rango dado.

min_temp_rango	**O(log n + k)** Este es el mismo caso que el método anterior, sólo qué en lugar de devolver sólo el máximo valor encontrado, retorna el mínimo.
temp_extremos_rango	**O(log n + k)** Al igual qué los dos anteriores, este también hace uso del método encontrar_min_max_rango, pero devuelve ambos valores encontrados.
borrar_temperatura	**O(log n)** El algoritmo elimina una medición de temperatura, invocando al método **__delitem__** del ArbolAVL, que a su vez llama al método eliminar. La eliminación de un nodo es una operación O(log n), ya que hace una búsqueda en el árbol, (O(log n)), y una eliminación del nodo con complejidad constante.
devolver_temperaturas	**O(log n + klogk)** El método devolver_valores_rango del AVL recolecta las mediciones dentro del rango. Esta operación es O(logN+k), ya que busca los límites del rango y luego visita cada uno de los k nodos dentro del mismo. Sin embargo, después de obtener la lista, la ordena con sorted().. Si hay k elementos en el rango, ordenar estos k elementos toma O(klogk) tiempo. Por lo tanto, la complejidad dominante es el tiempo de ordenamiento.
cantidad_muestras	**O(1)** Devuelve la cantidad total de mediciones utilizando el método __len__ de un árbol AVL, ya que sólo devuelve el tamaño.