Análisis de Complejidad de Métodos - Clase Temperaturas_DB

A continuación se presenta una tabla con el análisis del orden de complejidad temporal para cada uno de los métodos implementados en la clase Temperaturas_DB. El análisis está basado en el uso de un árbol AVL como estructura subyacente, lo cual garantiza operaciones balanceadas con complejidad logarítmica en el peor caso.

Método	Complejidad	Justificación
guardar_temperatura	O(log n)	Inserta o actualiza un nodo en el árbol AVL, manteniendo balanceo.
devolver_temperatura	O(log n)	Busca una fecha específica en el árbol, siguiendo ramas equilibradas.
max_temp_rango	O(k + log n)	Recorre k nodos dentro del rango y accede logarítmicamente al árbol.
min_temp_rango	O(k + log n)	Similar al anterior, pero buscando el mínimo en el rango.
temp_extremos_rango	O(k + log n)	Agrupa los extremos mínimo y máximo dentro del mismo recorrido del rango.
borrar_temperatura	O(log n)	Elimina un nodo manteniendo el balance del árbol AVL.
devolver_temperaturas	O(k + log n)	Devuelve en orden todas las temperaturas del rango solicitado.
cantidad_muestras	O(n)	Cuenta recursivamente los nodos del árbol. Podría optimizarse almacenando el total.

Análisis General

La clase Temperaturas_DB está basada en un árbol AVL, lo cual garantiza que la altura del árbol se mantenga en O(log n) en todo momento. Gracias a esto, las operaciones de inserción, búsqueda y eliminación logran una eficiencia óptima, incluso cuando la cantidad de datos almacenados es considerable.

Los métodos que recorren rangos de fechas, como max_temp_rango, min_temp_rango, temp_extremos_rango y devolver_temperaturas, tienen una complejidad combinada de O(k + log n), donde 'k' representa la cantidad de nodos visitados dentro del rango, y log n corresponde al tiempo necesario para alcanzar el inicio del recorrido en el árbol.

El método cantidad_muestras actualmente realiza un recorrido completo del árbol para contar los nodos, lo que implica una complejidad lineal O(n). No obstante, esta operación podría optimizarse si se mantiene un contador interno que se actualice en cada inserción o borrado.

En resumen, el uso de un AVL garantiza una estructura balanceada y un acceso eficiente, cumpliendo con los requerimientos de rendimiento que se esperan en una base de datos en memoria principal.