UNER Facultad de Ingeniería

Licenciatura en Bioinformática

Bioingeniería

TRABAJO PRÁCTICO N°2 - EJERCICIO 2

ALGORITMOS Y ESTRUCTURAS DE DATOS

Docentes: Rizzato, Juan - Diaz Zamboni, Javier

Alumnos:

Isaac, Priscila Rocio

Jacobo, Nahir

Nista, Nicolás

Fecha de entrega: 07/06

Año lectivo: 2025

Creamos la base de datos Temperaturas_DB, la cual utiliza un **árbol AVL** para almacenar temperaturas asociadas a fechas. Esta estructura garantiza que las operaciones de inserción, búsqueda y eliminación se realicen en O(log n), manteniendo un buen rendimiento incluso con grandes volúmenes de datos. El AVL se mantiene balanceado automáticamente mediante rotaciones cuando es necesario.

Cada nodo del árbol almacena una fecha (clave) convertida a datetime y una temperatura (valor) en ºC. El orden cronológico queda naturalmente definido por la estructura del árbol. Esto permite realizar consultas por rangos de fechas de forma eficiente, aprovechando el recorrido inorden del árbol. Además, el uso de datetime evita errores en las comparaciones de fechas.

Funciones definidas dentro de la clase "Temperaturas DB":

Las funciones que realicen constantes comparaciones entre índices las consideramos de complejidad $O_{(Log\ n)}$, aquellas que solo recorran la lista 1 vez de complejidad $O_{(1)}$ y las que posean estructuras de control como bucles for / while de complejidad $O_{(n)}$.

- **guardar_temperatura**: $O_{(log\ n)}$ → Esto es así porque dentro de la función, *encolar* llama a *insertar* la cual es una función definida en nuestro arbolAVL y tiene orden de complejidad $O_{(log\ n)}$.
- **_init_**: $O_{(1)}$ → Se trata de una asignación.
- **devolver_temperatura**: $O_{(log\ n)}$ → Este método convierte un string en un objeto (O(1)) y luego busca ese string en el árbol AVL. Como el árbol está balanceado, la búsqueda de una clave específica tarda O(log n), ya que en cada nivel se reduce a la mitad el espacio de búsqueda. Por tanto, la complejidad total es O(log n).
- **cantidad_muestras**: $O_{(1)}$ → Devuelve el tamaño del árbol AVL accediendo a la propiedad tamano, que internamente se asume como un contador actualizado cada vez que se inserta o elimina un nodo.
- **borrar_temperatura**: $O_{(log\ n)}$ → Esto es así ya que desencolar llama a la función eliminar de nuestro arbolAVL, la cual tiene $O_{(log\ n)}$.
- **max_temp_rango**: $O_{(n)}$ → Contiene un bucle for de orden de complejidad $O_{(n)}$
- **min_temp_rango**: $O_{(n)}$ → Contiene un bucle for de orden de complejidad $O_{(n)}$
- **temp_extremos_rango**: $O_{(n)}$ → Llama a las funciones min y máx, las cuales conocemos que tienen orden de complejidad $O_{(n)}$.
- **devolver_temperaturas**: $O_{(n)}$ → Recorre n veces mediante bucle for.