Método	Complejidad	Explicación
guardar_temperatura	O(log n)	Inserta en el árbol AVL, que garantiza altura logarítmica mediante rebalanceo.
devolver_temperatura	O(log n)	Búsqueda binaria en árbol AVL balanceado.
borrar_temperatura	O(log n)	Eliminación en árbol AVL, con rotaciones posibles para mantener balance.
cantidad_muestras	O(n)	Recorre todo el árbol para contar nodos. No usa contador auxiliar.
devolver_temperaturas	O(k + log n)	Búsqueda del rango (O(log n)) + recorrido in-order en ese rango (k elementos).
max_temp_rango	O(k + log n)	Recorre nodos dentro del rango (k), búsqueda inicial es logarítmica.
min_temp_rango	O(k + log n)	Igual al caso anterior, con comparación mínima.
temp_extremos_rango	O(k + log n)	Invoca los dos métodos anteriores, por lo tanto se mantiene esta complejidad.

Conclusiones:

El uso de un árbol AVL garantiza que todas las operaciones básicas sobre el conjunto de fechas (inserción, búsqueda y eliminación) se realicen con complejidad logarítmica en el peor caso.

Las complejidades más relevantes son:

- guardar_temperatura, devolver_temperatura, borrar_temperatura → O(log n)
  Debido al uso del árbol AVL balanceado.
- cantidad_muestras → O(n)
  Ya que recorre todos los nodos para contarlos. Esto podría optimizarse si se mantiene un contador de nodos.
- Consultas por rango (devolver_temperaturas, min_temp_rango, max_temp_rango, temp_extremos_rango) → O(k + log n)
  Donde k es la cantidad de fechas dentro del rango pedido.
  Esto es eficiente, porque el recorrido se restringe solo a la porción relevante del árbol.