**📘 Uso de lower\_bound y upper\_bound en C++ y su aplicación al problema Codeforces 474B - Worms**

**🧩 1. Concepto general**

En C++, las funciones lower\_bound() y upper\_bound() son parte de la librería <algorithm> y sirven para realizar **búsquedas binarias eficientes** en arreglos **ordenados**.

**⚙️ Sintaxis**

auto it = lower\_bound(v.begin(), v.end(), x);

auto it = upper\_bound(v.begin(), v.end(), x);

Donde:

* v es un vector o arreglo **ordenado** (ascendente).
* x es el valor que buscas.
* Devuelve un **iterador** (no un índice) que puedes transformar así:
* int pos = it - v.begin(); // índice (0-based)

**🔍 Diferencias clave**

| **Función** | **Qué devuelve** | **Ejemplo (v = {1, 3, 3, 5})** | **Resultado para x = 3** |
| --- | --- | --- | --- |
| lower\_bound | primer elemento **≥ x** | apunta al primer 3 | índice 1 |
| upper\_bound | primer elemento **> x** | apunta al 5 | índice 3 |

**⏱️ Complejidad**

* Tiempo: **O(log n)**
* Espacio: **O(1)**  
  Ideal cuando necesitas muchas búsquedas rápidas en un arreglo ordenado.

| **Necesitas…** | **Usa…** |
| --- | --- |
| Primer índice donde v[i] ≥ x | lower\_bound |
| Primer índice donde v[i] > x | upper\_bound |
| Contar cuántas veces aparece x en un vector | upper\_bound - lower\_bound |
| Insertar x manteniendo orden sin duplicados | lower\_bound y verifica si \*it == x |

Ejemplo:

int solve(int x){

    return upper\_bound(w.begin(), w.end(), x) - w.begin() + 1;

}