Problema de Contar Segmentos - Solución

Este documento describe una solución eficiente para el problema de contar cuántos puntos enteros en el eje x están cubiertos por exactamente k segmentos.

1. **Estructura de los segmentos**:

Dado un conjunto de puntos en el eje x, cada par de puntos define un segmento cerrado que cubre todos los puntos entre ellos. El número de segmentos que cubren un punto depende de su posición con respecto a los extremos.

2. **Precomputación de puntos cubiertos**:

En lugar de contar cuántos segmentos cubren cada punto uno por uno, aprovechamos la simetría del problema para precomputar cuántos puntos están cubiertos por k segmentos. Usamos un mapa (diccionario) donde las claves son el número de segmentos y los valores son la cantidad de puntos cubiertos.

3. **Consultas**:

Después de precomputar cuántos puntos están cubiertos por cada cantidad de segmentos, respondemos a las consultas simplemente buscando el valor correspondiente en el mapa.

Código en C++

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
```

```
void solve() {
    int n, q;
    cin >> n >> q;

    vector<int> x(n);
    vector<ll> queries(q);

    // Leer los puntos y las consultas
    for (int& e : x) cin >> e;
    for (ll& e : queries) cin >> e;

    // Mapa para contar cuántos puntos están cubiertos por exactamente k segmentos
    map<ll, ll> segment_count;

    // Calcular cuántos puntos están cubiertos por k segmentos
    ll total_segments = n - 1;  // Inicialmente, los puntos extremos
    ll remaining_segments = n - 2;  // Para los siguientes puntos
    segment_count[total_segments]++;  // Los extremos siempre están en n-1 segmentos

    for (int i = 1; i < n; ++i) {
        total_segments += remaining_segments;
        remaining_segments -= 2;
        segment_count[total_segments]++;
    }

    // Ahora procesamos las consultas
```

```
    for (int i = 0; i < q; ++i) {
        ll k = queries[i];
        // Si k está en el mapa, imprimimos el valor correspondiente
        if (segment_count.count(k)) {
            cout << segment_count[k] << " ";
        } else {
            // Si no está, significa que no hay puntos cubiertos por exactamente k segmentos
            cout << 0 << " ";
        }
    }
    cout << endl;
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);

    int t;
    cin >> t;
    while (t--) {
        solve();
    }

    return 0;
}
```

Explicación del Código:

El código en C++ lee múltiples casos de prueba y precomputa cuántos puntos están cubiertos por cada número de segmentos usando un mapa. Luego, responde a las consultas de manera eficiente en tiempo constante O(1) por consulta.

Complejidad:

El tiempo de ejecución es O(n + q) por caso de prueba, donde n es el número de puntos y q es el número de consultas. El uso de un mapa garantiza que las búsquedas de las consultas sean rápidas.