1 O problema

O problema analisado é o problema D (*Static Range Queries*) do Contest *Usaco Guide Problem Submission*, disponível em `https://codeforces.com/gym/102951/problem/D`.

Problema. *There is an array a of length 10^9, initially containing all zeroes.*
First perform N updates of the following form:

- *Given integers l, r, and v, add v to all values $a_l, \ldots, a_{r-1}$.*

Then, answer Q queries of the following form:

- *Given integers l and r, print the sum $a_l + \ldots + a_{r-1}$.*

Input
Line 1: The two space-separated integers N and Q ($1 \leq N, Q \leq 10^5$)
Lines $2 \ldots N+1$: Each line contains three space-separated integers l, r, and v, corresponding to an update ($0 \leq l < r \leq 10^9$, $v \leq 10^4$).
Lines $N+2 \ldots N+Q+1$: Each line contains two space-separated integers l and r, corresponding to a query ($0 \leq l < r \leq 10^9$).

Output
Print Q lines, the answers to the queries in the same order as given in the input.

Solução. Seja b um vetor de tamanho 10^9, contendo inicialmente apenas zeros, e para cada query de update (l, r, v):

- Incremente $b[l]$ por v: $b[l] \leftarrow b[l] + v$.
- Se $r \neq 10^9$, decremente $b[r]$ por v: $b[r] \leftarrow b[r] - v$.

Então é fácil mostrar que

$$a[i] = \sum_{0 \leq j \leq i} b[j]$$

(para todo $0 \leq i < 10^9$),isto é, a soma de prefixos de b resulta no vetor original.

Sejam $0 \leq p_0 < \ldots < p_{k-1} < 10^9$ as posições não-nulas de b, e defina $p_k = 10^9$. Assim, para todo $0 \leq i < 10^9$

$$\sum_{0 \leq j \leq i} a[j] = \sum_{0 \leq j \leq i} \sum_{0 \leq l \leq j} b[l] = \sum_{0 \leq l \leq i} (i + 1 - l) b[l]$$

pois cada $b[l]$ aparece uma vez na soma, para cada j que satisfaz $l \leq j \leq i$.

Porém, note que b possui apenas $O(n)$ posições não-nulas, e $n << 10^9$. Portanto, podemos realizar **compressão de coordenadas**

Primeiramente, defina para todo $0 \leq i \leq 10^9 - 1$

$$f(i) = \begin{cases} \max\left(\{j | p_j \leq i\}\right) \text{ se } i \geq p_0 \\ -1, \text{ do contrário} \end{cases}$$

É evidente que $f(i) < k$ (estritamente) pois $p_k = 10^9 > i$

Ademais, defina $x_j := b[p_j]$ e $y_j = \sum_{0 \leq l \leq j} x_l$ (j-ésimo prefixo) para todo $0 \leq j \leq k$. Assim, segue que

$$\begin{aligned} \sum_{0 \leq j \leq i} a[j] &= \sum_{0 \leq j \leq f(i)} (i + 1 - p_j) b[p_j] = \sum_{0 \leq j \leq f(i)} (i + 1 - p_j) x_j \\ &= \left(\sum_{0 \leq j \leq f(i)} (p_{f(i)} - p_j) x_j \right) + (i + 1 - p_{f(i)}) \left(\sum_{0 \leq j \leq f(i)} x_j \right) \\ &= \left(\sum_{0 \leq j \leq f(i)} \sum_{j \leq l < f(i)} (p_{l+1} - p_l) x_j \right) + (i + 1 - p_{f(i)}) \left(\sum_{0 \leq j \leq f(i)} x_j \right) \end{aligned}$$

Assim, podemos notar que, na primeira parcela da soma, cada $p_{l+1} - p_l$, para $0 \leq l < f(i)$, aparece acompanhado por um fator de $\sum_{0 \leq j \leq l} x_j = y_l$. Logo, invertendo a ordem da soma temos

$$\sum_{0 \leq j \leq i} a[j] = \sum_{0 \leq l < f(i)} (p_{l+1} - p_l) y_l + (i + 1 - p_{f(i)}) y_{f(i)} = \sum_{0 \leq l \leq f(i)} (p_{l+1} - p_l) y_l - (p_{f(i)+1} - i - 1) y_{f(i)}$$

Defina $z_i = \sum\limits_{0 \le l \le i} (p_{l+1} - p_l) y_l$. Então

$$\sum_{0 \le j \le i} a[j] = z_{f(i)} - y_{f(i)}(p_{f(i)+1} - i - 1)$$

Portanto, a resposta de uma query (l, r) é

$$query(l,r) = \sum_{l \le i < r} a[i] = \begin{cases} z_{f(r-1)} - y_{f(r-1)}(p_{f(r-1)+1} - r) \text{ se } l = 0 \\ z_{f(r-1)} - y_{f(r-1)}(p_{f(r-1)+1} - r) - z_{f(l-1)} + y_{f(l-1)}(p_{f(l-1)+1} - l) \text{ se } l > 0 \end{cases}$$

□

Solução. [Complexidade] Podemos calcular os vetores p, x em $O(n \log n)$ da seguinte forma:

> Criamos dois vetores de inteiros $p[], x[]$ vazios.
> Criamos um vetor de pares de inteiros $points[]$, inicialmente vazio.
> Para todo $0 \le i < n$
>   Lemos uma query de update l, r, v.
>   Adicionamos o par $\{l, v\}$ a $points[]$.
>   Adicionamos o par $\{r, -v\}$ a $points[]$.
> **Ordenamos** $points$
> Para todo $0 \le i < 2n$
>   Se $p[]$ não é vazio e $points[i].first$ é igual ao último elemento de $p[]$, incrementamos o último elemento de $x[]$ por $points[i].second$. Do contrário, adicionamos $points[i].first$ ao final de $p[]$ e $points[i].second$ ao final de $x[]$.

Ademais, podemos pré-computar os vetores y, z. Por fim, para toda query (l, r), podemos encontrar $f(l-1)$ e $f(r-1)$ em $O(\log |p|) = O(\log n)$ (uma vez que $|p| \le 2n$), e o resto é computado em tempo constante.

Assim, a complexidade final é $\boldsymbol{O((n+q)\log n)}$.

□

2 Implementação

```
#include <bits/stdc++.h>
using namespace std;
#define int long long

int query(int l, vector<int>& y, vector<int>& z, vector<int>& p)
{
    // ans + 1 < |p|
    int lo = 0, hi = p.size() - 2, ans = -1;
    while (lo <= hi)
    {
        int mid = lo + (hi - lo)/2;
        if (p[mid] > l)
        {
            hi = mid - 1;
        } else
        {
            ans = mid;
            lo = mid + 1;
        }
    }
    if (ans == -1)
    {
        return 0;
    } else
    {
        return z[ans] - y[ans] * (p[ans + 1] - l - 1);
    }
}
```

```
signed main()
{
    int n, q;
    cin >> n >> q;
    pair<int, int> points[2*n];

    int l, r , v;
    for (int i = 0; i < n; i++)
    {
        cin >> l >> r >> v;
        points[2*i] = make_pair(l, v);
        points[2*i + 1] = make_pair(r, -v);
    }
    sort(points, points + 2 * n);

    vector<int> x, p;
    for (int i = 0; i < 2*n; i++)
    {
        if (p.size() > 0 && points[i].first == p[p.size() - 1])
        {
            x[x.size() - 1] += points[i].second;
        } else
        {
            p.emplace_back(points[i].first);
            x.emplace_back(points[i].second);
        }
    }
    p.emplace_back(1e9);
    x.emplace_back(0);

    for (int i = 1; i < (int)x.size(); i++)
    {
        x[i] += x[i-1];
    }
    vector<int> z((int)x.size() - 1);
    for (int i = 0; i < (int)x.size() - 1; i++)
    {
        z[i] = x[i] * (p[i + 1] - p[i]);
        if (i > 0) z[i] += z[i-1];
    }

    for (int i = 0; i < q; i++)
    {
        cin >> l >> r;
        int ans = 0;
        if (r > 0) ans += query(r - 1, x, z, p);
        if (l > 0) ans -= query(l - 1, x, z, p);
        cout << ans << '\n';
    }
}
```