Inspectores de hacienda

Autor del problema: Gerard Orriols Giménez (Miembro del comité científico)

Resumen del enunciado

Procesar dos tipos de queries a una lista de números:

- Modificar un segmento añadiendo un valor a cada elemento.
- Calcular el mínimo de los valores en una posición a lo largo de un periodo de tiempo

Explicación de la solución

- Si todos los cambios afectan a todas las casas, usar un *segment tree* de mínimo a lo largo del tiempo.
- Si los cambios sólo afectan a partir de una cierta casa, actualizar esta información procesando las casas de izquierda a derecha usando lazy propagation.

También existe una solución online usando una técnica estándar.

Solución en C++

```
#include <iostream>
#include <vector>

using namespace std;
using ll = long long;

const ll inf = 1e18;

struct ST {
  vector<ll> v, lazy;
  int sz;
  ST(int n) {
        sz = 1;
        while (sz < n) sz *= 2;
        v = vector<ll>(2 * sz, 0);
        lazy = vector<ll>(2 * sz, 0);
  }
  void add(ll x, int l, int r) {
        return add(x, l, r, 0, sz, 1);
  }
  void add(ll x, int l, int r, int i, int j, int p) {
        if (j <= l or i >= r) return;
        if (i >= l and j <= r) {
        v[p] += x;
        lazy[p] += x;
        return;
        }
        prop(p);
        int mid = (i + j) / 2;
        add(x, l, r, i, mid, 2 * p);
        add(x, l, r, mid, j, 2 * p + 1);
```

```
        v[p] = min(v[2 * p], v[2 * p + 1]);
    }
    void prop(int p) {
        v[2 * p] += lazy[p];
        v[2 * p + 1] += lazy[p];
        lazy[2 * p] += lazy[p];
        lazy[2 * p + 1] += lazy[p];
        lazy[p] = 0;
    }
    ll minim(int l, int r) {
        return minim(l, r, 0, sz, 1);
    }
    ll minim(int l, int r, int i, int j, int p) {
        if (j <= l or i >= r) return inf;
        if (i >= l and j <= r) return v[p];
        prop(p);
        int mid = (i + j) / 2;
        return min(minim(l, r, i, mid, 2 * p), minim(l, r, mid, j, 2 * p + 1));
    }
    ll debug(int l) {
        return minim(l, l + 1);
    }
};

int main() {
    cin.tie(0);
    ios::sync_with_stdio(false);
    int n, q;
    cin >> n >> q;

    int m = 0;
    vector<int> start, stop, order, current(n, -1);
    vector<vector<int>> inspec(n);
    vector<vector<pair<int, int>>> events(n);
    for (int i = 0; i < n; ++i) {
        int a;
        cin >> a;
        events[i].emplace_back(0, a);
        if (i + 1 < n) events[i + 1].emplace_back(0, -a);
    }

    int t = 1;
    while (q--) {
        char c;
        cin >> c;
        if (c == 'C') {
        //  Cartero pasa con factura/nomina de x entre las casas l y r.
        int l, r, x;
        cin >> l >> r >> x; --l;  // add x to [l, r)
        events[l].emplace_back(t, x);
        if (r < n) events[r].emplace_back(t, -x);
        ++t;
        } else if (c == 'I') {
        //  Inicia a investigar la casa i.
```

```
            int i;
            cin >> i; --i;
            current[i] = m;
            start.push_back(t - 1);
            stop.push_back(-1);
            inspec[i].push_back(m);
            m++;
            } else {  // c == 'F'
            //  Finaliza la investigacion de la casa i y escribe el minimo.
            int i;
            cin >> i; --i;
            int ins = current[i];
            current[i] = -1;
            stop[ins] = t;
            order.push_back(ins);
            }
      }

      ST tree(t);
      vector<ll> ans(m);
      for (int i = 0; i < n; ++i) {
            for (auto p : events[i]) {
            tree.add(p.second,  p.first, t);
            }
            for (int ins : inspec[i]) {
            ans[ins] = tree.minim(start[ins], stop[ins]);
            }
      }
      for (int i = 0; i < m; ++i) {
            cout << ans[order[i]] << endl;
      }
}
```

Autor de la solución: Gerard Orriols Giménez (Miembro del comité científico)