Ngày 09 tháng 3 năm 2021

SEGMENT TREE

A. LÝ THUYẾT:

1. Interval Tree max với cập nhật tăng

```
struct INCMAXIT {
    vector<int> it, dt;

    void Init (int n) {
        it.resize (4 * n, 0);
        dt.resize (4 * n, 0);
    }

    void Update (int r, int k, int l, int u, int v, int val) {
        if (v < k || u > l)
            return;
        if (u <= k && l <= v) {
            dt[r] += val;
            return;
        }
        int g = (k + l) / 2;
        dt[2 * r] += dt[r];
        dt[2 * r + 1] += dt[r];
        dt[r] = 0;
        Update (2 * r, k, g, u, v, val);
        Update (2 * r + 1, g + 1, l, u, v, val);
        it[r] = max (it[2 * r] + dt[2 * r], it[2 * r + 1] + dt[2 * r + 1]);
    }

    int Get (int r, int k, int l, int u, int v) {
        if (v < k || u > l)
            return -INF;
        if (u <= k && l <= v)
            return it[r] + dt[r];
        int g = (k + l) / 2;
        dt[2 * r] += dt[r];
        dt[2 * r + 1] += dt[r];
        dt[r] = 0;
        int tL = Get (2 * r, k, g, u, v);
        int tR = Get (2 * r + 1, g + 1, l, u, v);
        it[r] = max (it[2 * r] + dt[2 * r], it[2 * r + 1] + dt[2 * r + 1]);
        return max (tL, tR);
    }
};
```

2. Interval Tree tổng với cập nhật tăng

```
struct INCSUMIT {
    vector<int64_t> it, dt;

    void Init (int n) {
        it.resize (4 * n, 0);
        dt.resize (4 * n, 0);
    }

    void Update (int r, int k, int l, int u, int v, int val) {
```

```
        if (v < k || u > l)
            return;
        if (u <= k && l <= v) {
            dt[r] += val;
            return;
        }
        int g = (k + l) / 2;
        dt[2 * r] += dt[r];
        dt[2 * r + 1] += dt[r];
        dt[r] = 0;
        Update (2 * r, k, g, u, v, val);
        Update (2 * r + 1, g + 1, l, u, v, val);
        it[r] = (it[2 * r] + (g - k + 1) * dt[2 * r]) +
                (it[2 * r + 1] + (l - g) * dt[2 * r + 1]);
    }

    int64_t Get (int r, int k, int l, int u, int v) {
        if (v < k || u > l)
            return 0;
        if (u <= k && l <= v)
            return it[r] + (l - k + 1) * dt[r];
        int g = (k + l) / 2;
        dt[2 * r] += dt[r];
        dt[2 * r + 1] += dt[r];
        dt[r] = 0;
        int64_t tL = Get (2 * r, k, g, u, v);
        int64_t tR = Get (2 * r + 1, g + 1, l, u, v);
        it[r] = (it[2 * r] + (g - k + 1) * dt[2 * r]) +
                (it[2 * r + 1] + (l - g) * dt[2 * r + 1]);
        return tL + tR;
    }
};
```

3. Interval Tree max với cập nhật thay thế

```
struct REPMAXIT {
    vector<int> it, dt, nho;

    void Init (int n) {
        it.resize (4 * n, -INF);
        dt.resize (4 * n);
        nho.resize (4 * n, 0);
    }

    void Update (int r, int k, int l, int u, int v, int val) {
        if (v < k || u > l)
            return;
        if (u <= k && l <= v) {
            nho[r] = 1;
            dt[r] = val;
            return;
        }
        int g = (k + l) / 2;
        if (nho[r]) {
            dt[2 * r] = dt[2 * r + 1] = dt[r];
            nho[2 * r] = nho[2 * r + 1] = 1;
            nho[r] = 0;
        }
```

```
        Update (2 * r, k, g, u, v, val);
        Update (2 * r + 1, g + 1, l, u, v, val);
        int Trai = (nho[2 * r]) ? dt[2 * r] : it[2 * r];
        int Phai = (nho[2 * r + 1]) ? dt[2 * r + 1] : it[2 * r + 1];
        it[r] = max (Trai, Phai);
    }

    int Get (int r, int k, int l, int u, int v) {
        if (v < k || u > l)
            return -INF;
        if (u <= k && l <= v)
            return (nho[r]) ? dt[r] : it[r];
        int g = (k + l) / 2;
        if (nho[r]) {
            dt[2 * r] = dt[2 * r + 1] = dt[r];
            nho[2 * r] = nho[2 * r + 1] = 1;
            nho[r] = 0;
        }
        int tL = Get (2 * r, k, g, u, v);
        int tR = Get (2 * r + 1, g + 1, l, u, v);
        int Trai = (nho[2 * r]) ? dt[2 * r] : it[2 * r];
        int Phai = (nho[2 * r + 1]) ? dt[2 * r + 1] : it[2 * r + 1];
        it[r] = max (Trai, Phai);
        return max (tL, tR);
    }

};
```

4. Interval Tree tổng với cập nhật thay thế

```
struct REPSUMIT {
    vector<int64_t> it;
    vector<int> dt, nho;

    void Init (int n) {
        it.resize (4 * n, 0);
        dt.resize (4 * n);
        nho.resize (4 * n, 0);
    }

    void Update (int r, int k, int l, int u, int v, int val) {
        if (v < k || u > l)
            return;
        if (u <= k && l <= v) {
            nho[r] = 1;
            dt[r] = val;
            return;
        }
        int g = (k + l) / 2;
        if (nho[r]) {
            dt[2 * r] = dt[2 * r + 1] = dt[r];
            nho[2 * r] = nho[2 * r + 1] = 1;
            nho[r] = 0;
        }
        Update (2 * r, k, g, u, v, val);
        Update (2 * r + 1, g + 1, l, u, v, val);
        int64_t Trai = (nho[2 * r]) ? 1LL * (g - k + 1) * dt[2 * r] : it[2 * r];
        int64_t Phai = (nho[2 * r + 1]) ? 1LL * (l - g) * dt[2 * r + 1] :
```

```
                                        it[2 * r + 1];
        it[r] = Trai + Phai;
    }

    int64_t Get (int r, int k, int l, int u, int v) {
        if (v < k || u > l)
            return 0;
        if (u <= k &&  l <= v)
            return (nho[r]) ? 1LL * (l - k + 1) * dt[r] : it[r];
        int g = (k + l) / 2;
        if (nho[r]) {
            dt[2 * r] = dt[2 * r + 1] = dt[r];
            nho[2 * r] = nho[2 * r + 1] = 1;
            nho[r] = 0;
        }
        int64_t tL = Get (2 * r, k, g, u, v);
        int64_t tR = Get (2 * r + 1, g + 1, l, u, v);
        int64_t Trai = (nho[2 * r]) ? 1LL * (g - k + 1) * dt[2 * r] : it[2 * r];
        int64_t Phai = (nho[2 * r + 1]) ? 1LL * (l - g) * dt[2 * r + 1] :
                                        it[2 * r + 1];
        it[r] = Trai + Phai;
        return tL + tR;
    }
};
```

B-BÀI TẬP

E. Có bao nhiêu màu? [NCOLOR]

Trong cuộc thi Robocon của khối chuyên tin NT, để kiểm tra hoạt động của các robot, ban tổ chức quyết định cho các robot quét sơn trên một bức tường độ dài N được chia thành N ô đơn vị đánh số từ 1 đến N (ban đầu cả bức tường có màu trắng, màu số 0). Có M robot đánh số từ 1 đến M lần lượt sơn tường. Robot thứ i sẽ quét vôi một đoạn bức tường từ ô a_i đến ô b_i với màu c_i ($1 \le c_i \le K$) (Không có robot nào sơn màu trắng).

Hỏi rằng sau khi M robot thực hiện xong công việc của mình thì trên tường có bao nhiêu màu khác nhau.

Input:

- Dòng đầu tiên ghi 3 số nguyên dương N, M, K ($1 \le N, M, K \le 10^5$)
- M dòng tiếp theo, dòng thứ i ghi ba số a_i, b_i, c_i ($1 \le a_i \le b_i \le N$, $1 \le c_i \le K$)

Output: Một số nguyên duy nhất là số màu khác nhau có trên bức tường.

Example:

Input	Output
5 3 4 1 2 1 1 3 2 2 5 1	2

F. Domino [DOMINO]

Vasya rất thích chơi trò Domino. Người ta thường lập kỷ lục bằng việc chạm nhẹ vào một quân Domino đầu tiên mà làm cho hàng vạn quân Domino bị đổ theo dây chuyền, có điều các Domino thường có hình dạng giống hệt nhau. Vasya không thích thế, cậu dùng các Domino có thể không cùng chiều cao.

Vasya đặt *n* quân Domino dọc theo một trục số (trục nằm theo hướng từ trái sang phải). Domino thứ *i* được đặt ở tọa độ x_i và có chiều cao h_i. Nếu một quân Domino ở tọa độ *x* với chiều cao *h* bị đổ về

bên phải thì nó sẽ làm cho các Domino đặt ở các tọa độ từ x+1 tới x+h-1 bị đổ theo về cùng hướng bên phải.

Yêu cầu: Với mỗi Domino thứ i bị Vasya làm đổ về bên phải, hãy xác định số z_i các Domino bị đổ.

Input: Dòng đầu là số nguyên dương n ($1 \le n \le 10^5$). Dòng thứ i trong số n dòng sau chứa hai số nguyên ngăn cách nhau bởi một dấu cách là tọa độ x_i và độ cao h_i của Domino thứ i ($-10^8 \le x_i \le 10^8$, $2 \le h_i \le 10^8$). Không có hai Domino nào ở cùng một tọa độ.

Output: Trên một dòng n số z_i tìm được, các số ghi ngăn cách nhau bởi một dấu cách.

Example:

Input	Output
4 16 5 20 5 10 10 18 2	3 1 4 1

G. Các biển quảng cáo [POSTERS]

Dọc theo đại lộ Tây-Đông của thành phố là dãy ***n*** ngôi nhà được xây dựng sát nhau đánh số từ 1 đến ***n*** theo hướng từ tây sang đông. Tất cả các ngôi nhà đều quay mặt về phía nam và có độ sâu (tính từ mặt đường vào) bằng nhau. Như vậy nếu nhìn từ hướng bắc xuống phía sau các ngôi nhà lập thành bức tường trải dài từ tây sang đông. Ngôi nhà thứ ***i*** có độ cao là h_i và chiều rộng (theo mặt đường) là w_i .

Chính quyền thành phố quyết định dán toàn bộ bức tường phía bắc các ngôi nhà bằng các tấm poster (áp phích). Ban lãnh đạo muốn biết cần tối thiểu bao nhiêu tấm poster để có thể che phủ toàn bộ mặt phía bắc này của dãy nhà. Các tấm poster có dạng hình chữ nhật với các cạnh dọc và ngang (so với mặt đất). Các poster không thể đặt chồng lên nhau nhưng có thể có điểm chung (cạnh chung).

Hãy tính số poster tối thiểu để làm điều này.

Input:

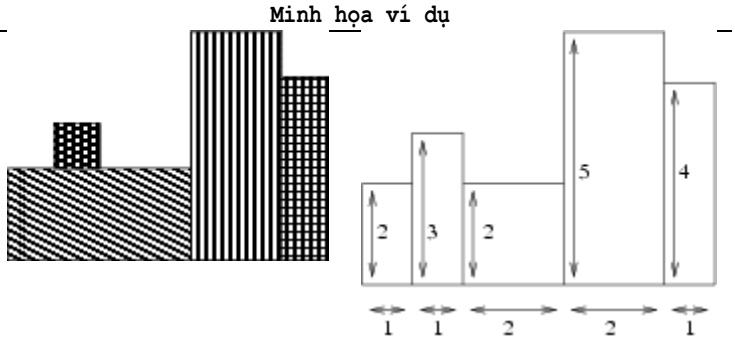
- Dòng đầu tiên ghi ***n*** là số ngôi nhà ($1 \le n \le 250000$)
- ***n*** dòng tiếp theo, dòng thứ ***i*** ghi hai số nguyên w_i và h_i là chiều rộng và chiều cao của ngôi nhà thứ ***i*** ($1 \le w_i, h_i \le 10^6$)

Output: Một số nguyên duy nhất là số poster tối thiểu tìm được.

Example:

Input	Output
5 1 2 1 3 2 2 2 5 1 4	4

Minh họa ví dụ



H. Tổng lớn nhất [MSUM]

Cho dãy n số nguyên $A = (a_1, a_2, \dots, a_n)$ ($0 \le a_i \le 10^8; 2 \le n \le 10^5$). Hãy thực hiện một số truy vấn trên mảng như sau:

- **U i x:** Gán cho a_i giá trị x ($1 \le i \le n; 0 \le x \le 10^8$)
- **Q u v:** Tìm hai chỉ số i và j sao cho $u \le i, j \le v, i \ne j$ và $a_i + a_j$ lớn nhất. Chỉ cần in ra tổng $a_i + a_j$ tìm được

Input:

- Dòng đầu tiên chứa số nguyên dương n
- Dòng thứ hai chứa n số nguyên $a_1, a_2, \dots, a_n$
- Dòng thứ ba chứa số nguyên dương Q số truy vấn cần thực hiện ($Q \le 10^5$)

- Q dòng cuối, mỗi dòng mô tả một truy vấn theo qui cách trên.

Output: Với mỗi truy vấn dang Q u v in ra kết quả tìm được.

Example:

input	output
5 1 2 3 4 5 6 Q 2 4 Q 2 5 U 1 6 Q 1 5 U 1 7 Q 1 5	7 9 11 12