CÂY LI CHAO

Bài toán: Gọi S là tập hợp các hàm tuyến tính dạng $y = a \cdot x + b$. Khởi đầu $S = \emptyset$. Viết chương trình thực hiện các truy vấn thuộc một trong hai loại dưới đây:

- **I a b:** Thêm hàm $y = a \cdot x + b$ vào tập S
- **Q x**: Tính giá trị của hàm $f(x) = \max\{a \cdot x + b : \ a \cdot x + b \in S\}$

Trong trường hợp tổng quát, khi hệ số góc các đường thẳng được thêm vào không có rằng buộc tăng hoặc giảm dần chúng ta sử dụng một cấu trúc dữ liệu dạng cây tương tự như segment tree quản lý các đường thẳng. Cấu trúc này là **câu Li Chao.**

Ta sẽ xây dựng một segment tree (ST) lưu trữ các đường thẳng sao cho có thể tìm ra được đường thẳng nhận giá trị max tại hoành độ x một cách nhanh chóng. Mỗi nút của ST sẽ quản lý một đoạn của trục hoành và lưu trữ tại nút này một đường thẳng sao cho **nếu ta đi từ lá (hoành độ x) đến gốc thì một trong các đường thẳng được lưu trữ trong các nút đi qua sẽ là đường thẳng nhận giá trị max tại x.**

Cấu trúc cây có dạng:

```
typedef long long ftype;
typedef pair<ftype, ftype> point;

struct Tnode {
    point line;           // Đường thẳng lưu tại nút
    int L, R;             // Nút quản lý [L,R)
    int Left, Right;      // Địa chỉ bên trái và bên phải

    Tnode (point _line = {0, -oo}, int _l = xmin, int _r = xmax)
    : line (_line), L (_l), R (_r) {
        Left = Right = -1;
    }
};

vector<Tnode> LiChao;     // Vector lưu các nút của cây
int root;                 // Địa chỉ nút gốc
```

a) Khởi tạo cây:

```
Tnode nw = Tnode ({0, -oo}, xmin, xmax);
LiChao.push_back (nw);
root = 0;
```

b) Thêm một đường thẳng vào cây

```
// Hàm tính giá trị đường thẳng p tại x
ftype f (point p, int x) {
    return p.first * x + p.second;
}

// Hàm thêm một đường thẳng vào cây
void add_line (int v, point nw) {
    int l = LiChao[v].L, r = LiChao[v].R;
    // Nếu đường thẳng thêm luôn nằm dưới đường thẳng nút → bỏ đi
    if (f (nw, l) <= f (LiChao[v].line, l) && f (nw, r - 1) <= f (LiChao[v].line, r - 1))
        return;
    // Đường thẳng thêm nằm trên → Thay thế đường thẳng nút bằng đường thẳng mới
    if (f (nw, l) >= f (LiChao[v].line, l) && f (nw, r - 1) >= f (LiChao[v].line, r - 1))
    {
```

```
        LiChao[v].line = nw;
        return;
    }
    int m = (l + r) / 2;
    bool lef = f (nw, l) > f (LiChao[v].line, l);
    bool mid = f (nw, m) > f (LiChao[v].line, m);

   // Nếu tại điểm giữa đường thẳng mới luôn lớn hơn → đổi chỗ, đẩy đường
   // thẳng cũ xuống một trong hai bên trái hoặc phải
    if (mid) {
        swap (LiChao[v].line, nw);
    }

    if (r - l == 1)
        return;

    if (lef != mid) {
        //Cây trái khác rỗng, đẩy xuống. Ngược lại thêm nút trái và dừng
        if (LiChao[v].Left != -1) {
            add_line (LiChao[v].Left, nw);
        } else {

            Tnode node = Tnode (nw, l, m);
            LiChao.push_back (node);
            LiChao[v].Left = LiChao.size() - 1;
        }
    } else {
        // Cây phải khác rỗng, đẩy xuống. Ngược lại thêm nút phải và dừng
        if (LiChao[v].Right != -1) {
            add_line (LiChao[v].Right, nw);
        } else {
            Tnode node = Tnode (nw, m, r);
            LiChao.push_back (node);
            LiChao[v].Right = LiChao.size() - 1;
        }
    }
}
```

c) Truy vấn giá trị max tại điểm x

```
int get (int v, int x) {
    int l = LiChao[v].L, r = LiChao[v].R;
    int m = (l + r) / 2;
    int fval = f (LiChao[v].line, x);
    if (r - l == 1)
        return fval;
    if (x < m) {// x thuộc cây trái
        if (LiChao[v].Left == -1)
            return fval;
        return max (fval, get (LiChao[v].Left, x));
    } else { // x thuộc cây phải
        if (LiChao[v].Right == -1)
            return fval;
        return max (fval, get (LiChao[v].Right, x));
    }
}
```