QUI HOẠCH ĐỘNG LỒI DẠNG A

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 $f(x) = \min\{a \cdot x + b : \ a \cdot x + b \in S\}$

Có hai phương pháp để giải quyết bài toán này:

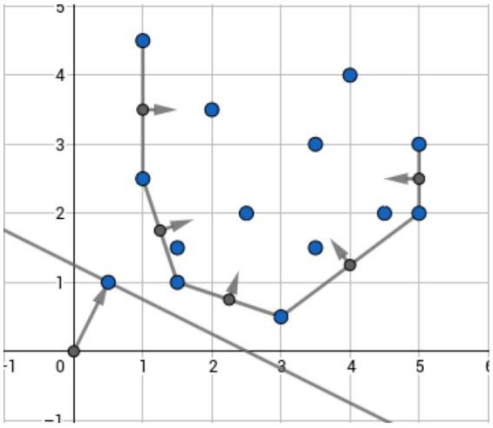
1. Kỹ thuật bao lồi (convex hull trick)
2. Cây Li Chao (Li Chao Tree)

I. Convex hull trick

Kỹ thuật bao lồi được sử dụng với giả thiết các hệ số góc của các đường thẳng được thêm vào không giảm với ý tưởng là duy trì "bao lồi dưới" của các tập đường thẳng.

Trước tiên, ta coi mỗi đường thẳng $y = k \cdot x + b$ như là một điểm có tọa độ (k, b) trên mặt phẳng Đề các 2D. Lúc này tập S có thể coi như là tập điểm trên mặt phẳng và việc trả lời các truy vấn dạng **Q x** đưa về việc tìm điểm $(k, b) \in S$ sao cho tích vô hướng (tích chấm) với $(x, 1)$ là nhỏ nhất (nhắc lại $(k, b) \cdot (x, 1) = k \cdot x + b$).

Dễ thấy một điểm làm cực tiểu giá trị $(k, b) \cdot (x, 1)$ sẽ là một điểm nằm trên bao lồi dưới như hình vẽ minh họa dưới đây:



Giải thích sơ bộ: Những điểm có $k \cdot x + b = P$ nằm trên một đường thẳng vuông góc với véc tơ $(x, 1)$ đi qua $(0, P)$. Khi giá trị P thay đổi thì đường thẳng sẽ "tiếp xúc" với bao lồi.

Như vậy, thay vì duy trì cả tập điểm ta chỉ cần duy trì những điểm nằm trên bao lồi dưới của tập điểm. Vì giá trị k tăng dần nên các đỉnh của bao lồi dưới sắp xếp ngược chiều kim đồng hồ.

Nhận xét rằng vec tơ pháp tuyến các cạnh của bao lồi được sắp xếp theo góc dương (véc tơ pháp tuyến của một cạnh lập với véc tơ pháp tuyến của cạnh tiếp sau nó góc dương) và đỉnh trên bao lồi cần tìm (có giá trị $k \cdot x + b$ nhỏ nhất) sẽ là đầu mút trái của cạnh đầu tiên có góc với véc tơ $(x, 1)$ lớn hơn hoặc bằng 0. Do việc các véc tơ pháp tuyến đã được sắp xếp theo góc dương nên ta có thể tìm điểm bằng kỹ thuật tìm kiếm nhị phân.

a) Duy trì bao lồi dưới

Việc duy trì bao lồi dưới tương tự như trong thuật toán tìm bao lồi: Mỗi khi thêm một điểm, nếu góc tạo bởi cạnh cuối cùng của bao lồi với cạnh mới (từ điểm cuối của bao lồi đến điểm thêm vào) âm thì bỏ đỉnh cuối của bao lồi.... Chú ý rằng khi điểm thêm vào có hoành độ bằng điểm cuối của bao lồi thì do ta đang tìm min nên nếu tung độ nhỏ hơn thì mới thêm vào bao lồi.

Ta có các hàm sau:

a.1 Các hàm cơ bản trên véc tơ

```
// Hàm hình học cơ bản
typedef pair<int, int> point;

point operator - (point a, point b) {
    return {a.X - b.X, a.Y - b.Y};
}

int dot (point a, point b) {
    return a.X * b.X + a.Y * b.Y;
}

int cross (point a, point b) {
    return a.X * b.Y - a.Y * b.X;
}
```

a.2 Biến mô tả bao lồi dưới

```
vector<point> hull, vecs;
// hull : các đỉnh của bao lồi dưới
// vecs : các phép tơ pháp tuyến của các cạnh
```

a.3 Hàm thêm điểm vào bao lồi dưới

```
// Hàm thêm một đỉnh vào bao lồi
void add_line (int k, int b) {
    point nw = {k, b};

    // Bao lồi rỗng → thêm đỉnh đầu tiên
    if (hull.empty()) {
        hull.push_back (nw);
        return;
    }

    // Nếu đỉnh cùng hoành độ nhưng lớn hơn → bỏ khỏi bao lồi
    if (nw.X == hull.back().X && nw.Y >= hull.back().Y)
        return;

    // Bao lồi đánh số cùng ngược kim đồng hồ
    while (!vecs.empty() && dot (vecs.back(), nw - hull.back()) < 0) {
        hull.pop_back();
        vecs.pop_back();
    }

    // Thêm đỉnh vào bao lồi
    if (!hull.empty()) {
        vecs.push_back ({- (nw - hull.back()).Y, (nw - hull.back()).X});
    }
    hull.push_back (nw);
}
```

b) Tính giá trị cực tiểu

Góc của các vector pháp tuyến với $(x, 1)$ có tính chất: **đấu của góc chuyển từ dương sang âm**.

Do vậy Ta có thể sử dụng kỹ thuật tìm kiếm nhị phân để làm điều này:

```
int get(int x) {
    point query = {x, 1};
    auto it = lower_bound(vecs.begin(), vecs.end(), query,
```

```
                        [](point a, point b) {
                            return cross(a, b) > 0;
                        });
    return dot(query, hull[it - vecs.begin()]);
}
```

II. Các dạng qui hoạch động lồi loại A

Bài toán: Cho dãy vô hạn $f_1, f_2, \ldots$ được xác định bởi:

- $f_1 = c$
- $f_i = \min\limits_{1 \le j < i}(\max)\{f_j + b_j \cdot a_i\}$. Ở đây dãy $b_1, b_2, \ldots$ hoặc không giảm hoặc không tăng

Yêu cầu: Tính f_n

Ta có dạng code chung dạng như sau:

```
f[1]=c;
add_line(b[1], f[1]);
for(int i=2;i<=n;++i) {
    f[i]=get(a[i]);
    add_line(b[i], f[i]);
}
```

Trong đó các hàm add_line có các chỉnh đổi version như dưới đây:

a. Bài toán tìm min với $b_1 \le b_2 \le \cdots$

Bao lồi trong trường hợp này quay ngược chiều kim đồng hồ. Góc giữa các véc tơ pháp tuyến với $(x, 1)$ chuyển từ dấu dương sang dấu âm.

```
// Hàm thêm một đỉnh vào bao lồi
void add_line (int k, int b) {
    point nw = {k, b};

    // Bao lồi rỗng → thêm đỉnh đầu tiên
    if (hull.empty()) {
        hull.push_back (nw);
        return;
    }

    if (nw.X == hull.back().X && nw.Y >= hull.back().Y)
        return;

    // Bao lồi đánh số cùng ngược kim đồng hồ
    while (!vecs.empty() && dot (vecs.back(), nw - hull.back()) < 0) {
        hull.pop_back();
        vecs.pop_back();
    }

    // Thêm đỉnh vào bao lồi
    if (!hull.empty()) {
        vecs.push_back ({- (nw - hull.back()).Y, (nw - hull.back()).X});
    }
    hull.push_back (nw);
}
```

```
int get(int x) {
    point query = {x, 1};
    auto it = lower_bound(vecs.begin(), vecs.end(), query,
                          [](point a, point b) {
                        return cross(a, b) > 0;
                          });
    return dot(query, hull[it - vecs.begin()]);
}
```

b. Bài toán min với $b_1 \geq b_2 \geq \cdots$

Bao lồi trong trường hợp này quay cùng chiều kim đồng hồ. Góc giữa các vector pháp và $(x, 1)$ có dấu chuyển từ dương sang âm.

```
// Hàm thêm một đỉnh vào bao lồi
void add_line (int k, int b) {
    point nw = {k, b};

    // Bao lồi rỗng → thêm đỉnh đầu tiên
    if (hull.empty()) {
        hull.push_back (nw);
        return;
    }

    if (nw.X == hull.back().X && nw.Y >= hull.back().Y)
        return;

    // Bao lồi đánh số cùng chiều kim đồng hồ
     while (!vecs.empty() && dot (vecs.back(), nw - hull.back()) > 0) {
         hull.pop_back();
         vecs.pop_back();
     }

    // Thêm đỉnh vào bao lồi
    if (!hull.empty()) {
        vecs.push_back ({- (nw - hull.back()).Y, (nw - hull.back()).X});
    }
    hull.push_back (nw);
}

int get(int x) {
    point query = {x, 1};
    auto it = lower_bound(vecs.begin(), vecs.end(), query,
                          [](point a, point b) {
                        return cross(a, b) > 0;
                          });
    return dot(query, hull[it - vecs.begin()]);
}
```

c. Bài toán max với $b_1 \leq b_2 \leq \cdots$

Bao lồi trong trường hợp này quay cùng chiều kim đồng hồ. Góc giữa các vector pháp và $(x, 1)$ có dấu chuyển từ âm sang dương.

```
// Hàm thêm một đỉnh vào bao lồi
void add_line (int k, int b) {
    point nw = {k, b};

    // Bao lồi rỗng → thêm đỉnh đầu tiên
    if (hull.empty()) {
        hull.push_back (nw);
        return;
    }

    if (nw.X == hull.back().X && nw.Y <= hull.back().Y)
        return;

    // Bao lồi đánh số cùng chiều kim đồng hồ
     while (!vecs.empty() && dot (vecs.back(), nw - hull.back()) > 0) {
         hull.pop_back();
         vecs.pop_back();
     }

    // Thêm đỉnh vào bao lồi
    if (!hull.empty()) {
        vecs.push_back ({- (nw - hull.back()).Y, (nw - hull.back()).X});
    }
    hull.push_back (nw);
}

int get(int x) {
    point query = {x, 1};
    auto it = lower_bound(vecs.begin(), vecs.end(), query,
                          [](point a, point b) {
                        return cross(a, b) < 0;
                          });
    return dot(query, hull[it - vecs.begin()]);
}
```

c. Bài toán max với $b_1 \geq b_2 \geq \cdots$

Bao lồi trong trường hợp này quay ngược chiều kim đồng hồ. Góc giữa các vector pháp và $(x, 1)$ có dấu chuyển từ âm sang dương.

```
// Hàm thêm một đỉnh vào bao lồi
void add_line (int k, int b) {
    point nw = {k, b};

    // Bao lồi rỗng → thêm đỉnh đầu tiên
    if (hull.empty()) {
        hull.push_back (nw);
        return;
    }

    if (nw.X == hull.back().X && nw.Y <= hull.back().Y)
        return;

    // Bao lồi đánh số ngược chiều kim đồng hồ
     while (!vecs.empty() && dot (vecs.back(), nw - hull.back()) < 0) {
         hull.pop_back();
```

```
        vecs.pop_back();
    }

    // Thêm đỉnh vào bao lồi
    if (!hull.empty()) {
        vecs.push_back ({- (nw - hull.back()).Y, (nw - hull.back()).X});
    }
    hull.push_back (nw);
}

int get(int x) {
    point query = {x, 1};
    auto it = lower_bound(vecs.begin(), vecs.end(), query,
                          [](point a, point b) {
                        return cross(a, b) < 0;
                          });
    return dot(query, hull[it - vecs.begin()]);
}
```