Bài 1: Các khái niệm hình học cơ bản

1. Véc tơ trong mặt phẳng tọa độ

Điểm trong mặt phẳng tọa độ cũng như véc tơ trong mặt phẳng luôn được biểu diễn bởi một cặp hai số thực. Do vậy chúng ta có thể khai báo kiểu dữ liệu sau để mô tả điểm cũng như véc tơ trên mặt phẳng

```
typedef pair<double,double> DD;
```

Các toán tử sau thực hiện phép cộng, trừ và phép nhân vô hướng một số thực với một véc tơ

```
DD operator +(DD u,DD v) {return DD(u.first+v.first,u.second+v.second);}

DD operator -(DD u,DD v) {return DD(u.first-v.first,u.second-v.second);}

DD operator *(double k,DD u) {return DD(k*u.first,k*u.second);}
```

Tích chấm

Tích chấm (dot product) hay còn gọi là tích vô hướng của hai véc tơ $\vec{u}$ và $\vec{v}$, ký hiệu là $\vec{u}.\vec{v}$ là một số thực được tinshbawngf tích độ dài của hai véc tơ $\vec{u}$ và $\vec{v}$ nhân với *cosin* của góc xen giữa hai véc tơ đó. Góc xen giữa hai véc tơ này là góc không định hướng, có số đo từ 0 đến π.

Biểu thức của tích chấm giữa hai véc tơ $\vec{u}=(x_u,y_u)$ và $\vec{v}=(x_v,y_v)$ có thể diễn giải như sau:

$$\vec{u}.\vec{v}=x_u.x_v+y_u.y_v$$

Toán tử dưới đây tính tích chấm của hai véc tơ:

```
double operator ^(DD u,DD v) {return u.first*v.first+u.second*v.second;}
```

Tích chéo

Tích chéo (cross product) của hai véc tơ $\vec{u}$ và $\vec{v}$, ký hiệu là $\vec{u}\times\vec{v}$ là một số thực được tính bằng tích độ dài hai véc tơ $\vec{u}$ và $\vec{v}$ nhân với *sin* của góc xen giữa hai véc tơ đó. Góc xen giữa của hai véc tơ này là góc định hướng, có số đo từ $-\pi$ đến π. Số đo mang dấu dương nếu chiều quay từ $\vec{u}$ đến $\vec{v}$ ngược chiều kim đồng hò và mang dấu âm nếu chiều quay từ $\vec{u}$ đến $\vec{v}$ cùng chiều kim đồng hồ.

Công thức tính tích chéo của hai véc tơ $\vec{u}=(x_u,y_u)$ và $\vec{v}=(x_v,y_v)$ là

$$\vec{u}\times\vec{v}=x_uy_v-x_vy_u$$

Toán tử sau thực hiện tích chéo:

```
double operator *(DD u,DD v) {return u.first*v.second-u.second*v.first;}
```

Về mặt hình học, giá trị tuyệt đối của tích chéo $\vec{u}\times\vec{v}$ là diện tích của hình bình hành OABC trong đó O là gốc tọa độ $\overrightarrow{OA}=\vec{u},\overrightarrow{OC}=\vec{v}$ và $\overrightarrow{OB}=\vec{u}+\vec{v}$

Tích chéo còn một ứng dụng quan trọng trong việc khảo sát chiều: Giả sử ta đi từ điểm A đến điểm B theo đường thẳng và đi tiếp sang điểm C theo đường thẳng. Khi đó

- $\overrightarrow{AB}\times\overrightarrow{BC}>0$ nếu chỗ rẽ tại B là "rẽ trái" (bẻ góc ngược chiều kim đồng hồ)
- $\overrightarrow{AB}\times\overrightarrow{BC}<0$ nếu chỗ rẽ tại B là "rẽ phải" (bẻ góc cùng chiều kim đồng hồ)
- $\overrightarrow{AB}\times\overrightarrow{BC}=0$ có nghĩa là ba điểm A, B, C thẳng hàng

2. Đường thẳng

Đường thẳng trong mặt phẳng tọa độ được xác định bởi hai điểm A, B trên mặt phẳng.

Đặt $\vec{u} = \overrightarrow{AB} = (x_b - x_a, y_b - y_a)$. $\vec{u}$ được gọi là véc tơ chỉ phương của đường thẳng . Khi đó đường thẳng AB có thể coi là tập hợp các điểm M(*x*,*y*) có tọa độ thỏa mãn:

$$\begin{cases} x = x_a + (x_b - x_a).t \\ y = y_a + (y_b - y_a).t \end{cases} \quad \text{với } t \in \mathbb{R} \qquad (*)$$

(*) được gọi là phương trình tham số của đường thẳng. Chú ý quan trọng:

- Nếu $t<0$ thì M nằm ngoài đoạn thẳng AB về phía A
- Nếu $0 \le t \le 1$ thì M nằm trên đoạn thẳng AB. $t=0$ thì M≡A còn $t=1$ thì M≡B
- Nếu $t>1$ thì M nằm ngoài đoạn thẳng AB về phía B

Ngoài các điểm nằm trên đường thẳng AB, các điểm còn lại được chia thành hai phần

- Các điểm C có $\overrightarrow{AB} \times \overrightarrow{BC} > 0$ tạo thành nửa dương của mặt phẳng (nửa nằm phía tay trái của đường thẳng AB khi đi từ A đến B)
- Các điểm C có $\overrightarrow{AB} \times \overrightarrow{BC} < 0$ tạo thành nửa âm của mặt phẳng (nửa nằm phía tay phải của đường thẳng AB khi đi từ A đến B)

3. Góc

Chú ý rằng giá trị π được tính trong C++ là **acos(-1).**

Trong nhiều bài toán. Chúng ta cần phải xác định góc của một véc tơ $\vec{u} = (x, y)$ so với chiều dương của trục hoành. Hàm **atan2(y,x)** trong C++ làm điều này. Tuy nhiên kết quả trả về là giá trị góc nằm trong đoạn $[-\pi, \pi]$. Do đó muốn giá trị góc trả về trong đoạn $[0, 2\pi]$ chúng ta có hàm sau:

```
double goc(DD v) {
    double t=atan2(V.second,v.first);
    if (t<0) t=t+2*acos(-1);
    return t;
}
```

Phần lớn các trường hợp, góc thường chỉ dùng để so sánh (>,=,<). Nếu sử dụng trực tiếp các hàm có sẵn của C++ thường sẽ gặp sai số. Một trong những cách khắc phục là xây dựng hàm tính góc riêng (chỉ dùng để so sánh). Hàm dưới đây làm điều này (nó trả về giá trị của góc nằm trong 0..360):

```
double theta(DD v) {
    if (v.first==0 && v.second==0) return 400.0;
    double t=v.second/(abs(v.first)+abs(v.second));
    if (v.first<0) t=2-t; else
        if (v.second<0) t+=4;
    return 90.0*t;
}
```

4. Đa giác

Đa giác là đường gấp khúc khép kín. Trong lập trình, một đa giác được mô tả bởi một dãy các đỉnh liên tiếp nhau $A_1, A_2, ..., A_n$.

Diện tích đại số của một đa giác có thể được tính bằng công thức sau:

$$S = \frac{1}{2}\sum_{i=1}^{n} \overrightarrow{OA_i} \times \overrightarrow{OA_{i+1}} = \frac{1}{2}\sum_{i=1}^{n} \left(x_i . y_{i+1} - x_{i+1} . y_i \right) = \frac{1}{2}\sum_{i=1}^{n} \left(x_{i-1} - x_{i+1} \right) y_i$$

Ở đây qui ước $x_{n+1} = x_0 = x_1$ còn $y_{n+1} = y_0 = y_1$

Dãy đỉnh của đa giác có thể được lưu cùng hoặc ngược chiều kim đồng hồ (*cùng chiều kim đồng hồ có nghĩa là khi đi dọc theo biên của đa giác phần nằm phía tay phải là phần mặt phẳng bên trong đa giác*) dựa vào dấu của S như sau:

- Nếu S>0 thì các điểm liệt kê ngược chiều kim đồng hồ
- Nếu S<0 thì các điểm liệt kê cùng chiều kim đồng hồ

Bài 2: Một số bài toán cơ bản

1. Biểu diễn tuyến tính

Cho ba vector $\vec{a},\vec{b},\vec{c}$. Hãy tìm hai số thực *p, q* để

$$\vec{c} = p\vec{a} + q\vec{b} \quad (1)$$

Nhân hai vế của (1) lần lượt với $\vec{a}$ và $\vec{b}$ ta có:

$$\vec{c}\times\vec{a} = p\vec{a}\times\vec{a} + q\vec{b}\times\vec{a} = q\vec{b}\times\vec{a} \Rightarrow q = \frac{\vec{a}\times\vec{c}}{\vec{a}\times\vec{b}} = \frac{Dy}{D}$$

$$\vec{c}\times\vec{b} = p\vec{a}\times\vec{b} + q\vec{b}\times\vec{b} = p\vec{a}\times\vec{b} \Rightarrow p = \frac{\vec{c}\times\vec{b}}{\vec{a}\times\vec{b}} = \frac{Dx}{D}$$

Như vậy $\vec{c}$ chỉ biểu diễn được duy nhất nếu như $D = \vec{a}\times\vec{b} \neq 0$. Ta có hàm sau trả về bộ (*p*,*q*) như là một cặp số thực DD:

```
DD SolveSLE(DD a,DD b,DD c) {
    double D=a*b;
    return DD(c*b/D,a*c/D);
}
```

2 Tìm giao điểm của hai đường thẳng.

Trên mặt phẳng tọa độ cho hai đường thẳng với phương trình tổng quát:

$$A_1x + B_1y + C_1 = 0$$
$$A_2x + B_2y + C_2 = 0$$

Hãy xác định giao điểm của hai đường thẳng này.

Đặt $\vec{u} = (A_1, A_2)$, $\vec{v} = (B_1, B_2)$, $\vec{w} = (-C_1, -C_2)$ Bài toán trở thành bài toán biểu diễn vector $\vec{w}$ qua tổ hợp tuyến tính của hai véc tơ $\vec{u},\vec{v}$:

$$\vec{w} = x\vec{u} + y\vec{v}$$

Hàm C++ dưới đây để tìm giao điểm của hai đường thẳng. Nó trả về cặp (-oo,-oo) nếu như hai đường thẳng không cắt nhau và trả về cặp (oo,oo) nếu hai đường thẳng trùng nhau. Ở đây oo được định nghĩa là số thực dương đủ lớn:

```
#define oo 1e+15
```

Hàm được viết như sau:

```
DD GiaoDuongThang(double a1,double b1,double c1,double a2,double b2,double c2) {
    DD u=DD(a1,a2), v=DD(b1,b2), w=DD(-c1,-c2);
    double D=u*v;
    if (D!=0) return DD(w*v/D,u*w/D);
    if (u*w==0) return DD(oo,oo); else return DD(-oo,-oo);
}
```

Ta xét trường hợp khi các đường thẳng đã cho được xác định bởi một điểm và véc tơ chỉ phương: *Tím giao điểm của đường thẳng đi qua $A(x_a,y_a)$ có véc tơ chỉ phương $\vec{u}$ với đường thẳng đi qua $B(x_b, y_b)$ có véc tơ chỉ phương $\vec{v}$.*

Gọi M là giao điểm của hai đường thẳng trên ta có

$$\overrightarrow{AM} = p\vec{u}$$
$$\overrightarrow{BM} = q\vec{v}$$

Trừ tương ứng hai vế ta có:

$$\overrightarrow{AB} = \overrightarrow{AM} - \overrightarrow{BM} = p\vec{u} - q\vec{v} = p\vec{u} + q\left(-\vec{v}\right)$$

Bài toán qui về tìm biểu diễn tuyến tính của $\overrightarrow{AB}$ qua $\vec{u}$ và $-\vec{v}$. Sau khi tìm được (*p*,*q*) thì tọa độ điểm M được tìm do công thức $\overrightarrow{OM} = \overrightarrow{OA} + p\vec{u}$ Hàm dưới đây thực hiện điều này:

```
DD GiaoDuongThang(DD A,DD u,DD B,DD v) {
    DD w=B-A;
    double D=u*v;
    if (D!=0) {
        DD t=DD(w*v/D,u*w/D);
        return A+t.first*u;
    }
    if (u*w==0) return DD(oo,oo);
    return DD(-oo,-oo);
}
```

3. Tìm giao điểm của hai đoạn thẳng

Bài toán tiếp theo ta xét là cho bốn điểm A, B, C, D trên mặt phẳng. Hãy cho biết hai đoạn thẳng AB và CD có giao điểm duy nhất hay không? Nếu có cho biết tọa độ giao điểm?.

Nếu M là giao điểm của hai đoạn thẳng AB và CD thì sẽ tồn tại duy nhất một cặp số thực $p,q \in [0,1]$ để:

$$\begin{cases} \overrightarrow{AM} = p\overrightarrow{AB} \\ \overrightarrow{CM} = q\overrightarrow{CD} \end{cases}$$

Từ đây suy ra:

$$\overrightarrow{AC} = p\overrightarrow{AB} + q\overrightarrow{DC}$$

Do đó ta chỉ cần tìm biểu diễn tuyến tính của $\overrightarrow{AC}$ qua $\overrightarrow{AB}$ và $\overrightarrow{DC}$, sau khi có cặp số *p*,*q* ta kiểm tra điều kiện $p,q \in [0,1]$ và tính tọa độ điểm M theo công thức:

$\overrightarrow{OM} = \overrightarrow{OA} + p\overrightarrow{AB}$

Dưới đây là hàm thực hiện ý tưởng trên. Chú ý rằng trường hợp không có giao điểm (hai đoạn thẳng không cắt nhau) xảy ra khi:

- Hoặc không tìm được *p*,*q*
- Hoặc tìm được *p*,*q* duy nhất nhưng không thỏa mãn điều kiện $p,q \in [0,1]$
- Hoặc tìm được vô số cặp (*p*,*q*) khi đó bốn điểm A, B, C, D thẳng hàng và A, B không nằm trên đoạn CD và C, D không nằm trên đoạn AB.

Hàm dưới đây kiểm tra hai đoạn thẳng có cắt nhau hay không. Nếu cắt nhau thì cho ra tọa độ một giao điểm

```
bool Cat(DD A,DD B,DD C,DD D,DD &M) {
    DD u=B-A, v=C-D, w=C-A;
    double t=u*v;
    if (t!=0) {
        DD pq=DD(w*v/t,u*w/t);
```

```
        double p=pq.first, q=pq.second;
        M=A+p*u;
        if (0<=p && p<=1 && 0<=q && q<=1) return true;
        else return false;
    }
    if (u*w!=0) return false;
    if (((A-C)^(B-C))<=0) {M=C;return true;}
    if (((A-D)^(B-D))<=0) {M=D;return true;}
    if (((C-A)^(D-A))<=0) {M=A;return true;}
    if (((C-B)^(D-B))<=0) {M=B;return true;}
    return false;
}
```

4. Đo góc giữa hai véc tơ

Ta cần xác định góc giữa hai véc tơ $\vec{u}$ và $\vec{v}$. Theo định nghĩa:

$$\sin(\vec{u},\vec{v}) = \frac{\vec{u}\times\vec{v}}{|\vec{u}|.|\vec{v}|} \quad \text{do đó} \quad \tan(\vec{u},\vec{v}) = \frac{\vec{u}\times\vec{v}}{\vec{u}.\vec{v}}$$

$$\cos(\vec{u},\vec{v}) = \frac{\vec{u}.\vec{v}}{|\vec{u}|.|\vec{v}|}$$

Sử dụng hàm **atan2(y,x)** ta có hàm sau trả về góc dương giữa hai véc tơ (nằm trong $[-\pi,\pi]$):

```
double Rad(DD u,DD v) {return atan2(u*v,u^v);}
```

5. Cắt đa giác

Cho một đa giác không tự cắt $A_1A_2 \dots A_n$ và một đường thẳng (M, $\vec{v}$). Hãy tìm phần giao của đa giác trên với nửa dương của đường thẳng.

Nói chung phần giao của một đa giác không tự cắt với nửa dương của đường thẳng không phải là đa giác không tự cắt theo đúng định nghĩa của nó. Tuy nhiên bằng cách mở rộng khái niệm "không tự cắt" (cho phép một cạnh nằm trên một cạnh khác, một đỉnh nằm trên một cạnh nhưng không có hai cạnh liên tiếp "xuyên qua" một cạnh) ta có thể coi giao của một đa giác không tự cắt với nửa dương của một đường thẳng là một đa giác không tự cắt.

Lần lượt xét $\overrightarrow{MA_i}\times\vec{v}$ với $i = 1,2,\dots,n+1$ ($A_{n+1} \equiv A_1$). Ta có bảng kết luận về đỉnh của đa giác giao như sau:

$\overrightarrow{MA_{i-1}}\times\vec{v}$	$\overrightarrow{MA_i}\times\vec{v}$	Kết luận
0	+	Điểm A_i thêm vào đỉnh đa giác giao
0	-	Bỏ qua A_i
0	0	Điểm A_i thêm vào đỉnh đa giác giao
+	+	Điểm A_i thêm vào đỉnh đa giác giao
+	0	Điểm A_i thêm vào đỉnh đa giác giao
+	-	Giao của đoạn $A_{i-1}A_i$ với (M,$\vec{v}$) thêm vào đa giác giao
-	+	Giao của đoạn $A_{i-1}A_i$ với (M,$\vec{v}$) và A_i thêm vào đa giác giao
-	0	Điểm A_i thêm vào đỉnh đa giác giao
-	-	Bỏ qua A_i

Kết luận: Nếu $\overrightarrow{MA_{i-1}} \times \vec{v}$ và $\overrightarrow{MA_i} \times \vec{v}$ trái dấu nhau thì luôn thêm giao điểm của đoạn $A_{i-1}A_i$ với (M,$\vec{v}$) vào đỉnh của đa giác giao. Tiếp theo, nếu $\overrightarrow{MA_i} \times \vec{v} \geq 0$ thì A_i luôn là đỉnh của đa giác giao.
Đoạn chương trình dưới đây thực hiện ý tưởng trên:

```
input: Đa giác không tự cắt A1A2 … An, Điểm M và vector v
output: Đa giác giao B1B2 … Bm

m=0;
double sign1, sign2=(A[1]-M)*v;
if (sign2>=0) B[++m]=A[1];
for(int i=2;i<=n+1;i++) {
    sign1=sign2;
    sign2=(A[i]-M)*v;
    if (sign1*sign2<0) {
        DD t=giao(A[i-1],A[i],M,v);
        B[++m]=t;
    }
    if (sign2>=0) B[++m]=A[i]);
}
```

Bài 3: Bao lồi

1. Bao lồi của tập điểm

Định nghĩa: Cho tập hợp *n* điểm trên mặt phẳng. Bao lồi của tập này được định nghĩa là đa giác lồi có diện tích nhỏ nhất với các đỉnh thuộc tập đã cho và chứa tất cả *n* điểm ở bên trong hoặc trên biên của nó.

Có nhiều thuật toán tìm bao lồi khác nhau như thuật toán bọc gói, quét Grasham,… Ở đây chúng ta xét một cách tìm bao lồi đơn giản hơn chỉ sử dụng các phép tính véc tơ đã trình bày ở trên.

Giả sử *n* điểm đã cho là $A_1, A_2, \dots, A_n$. Không mất tổng quát ta có thể xem các điểm này được sắp xếp theo hoành độ tăng dần. Nếu hoành độ bằng nhau thì chúng được sắp xếp theo tung độ tăng dần (sắp xếp tự nhiên theo kiểu *pair* trong C++). Khi đó các điểm A_1 và A_n chắc chắn thuộc bao lồi (điểm cực trái và điểm cực phải). Các điểm còn lại được chia thành hai phần:

- Các điểm ở phần trên. Các điểm này thỏa mãn $\overrightarrow{A_1M} \times \overrightarrow{MA_n} < 0$
- Các điểm ở phần dưới. Các điểm này thỏa mãn $\overrightarrow{A_1M} \times \overrightarrow{MA_n} > 0$ hay $\overrightarrow{A_nM} \times \overrightarrow{MA_1} < 0$

Đoạn chương trình dưới đây cho phép xây dựng bao lồi từ tập *n* điểm theo cách trên:

```
input: int n, a[maxn] - đa giác ban đầu đã cho
output: int m, c[maxn] - bao lồi với m là số đỉnh, các đỉnh đánh số cùng chiều kim ĐH

sort(a+1,a+n+1);
// Tim bao loi
m=2; c[1]=a[1]; c[2]=a[2];
for(int i=3;i<=n;i++) {
    c[++m]=a[i];
    while (m>2 && (c[m-1]-c[m-2])*(c[m]-c[m-1])>=0) {m--; c[m]=c[m+1];}
}
int m0=m;
c[++m]=a[n-1];
for(int i=n-2;i>=1;i--) {
    c[++m]=a[i];
    while (m-m0>1 && (c[m-1]-c[m-2])*(c[m]-c[m-1])>=0) {m--; c[m]=c[m+1];}
}
--m;
```

Chú ý: Các đỉnh của bao lồi là các đỉnh thực sự của đa giác lồi, tức là không có ba đỉnh liên tiếp thẳng hàng. Tuy nhiên trong trường hợp cần phải xây dựng bao lồi sao cho tất cả các đỉnh trên biên đều là đỉnh của bao lồi thi chỉ việc thay `(c[m-1]-c[m-2])*(c[m]-c[m-1])>=0` bằng điều kiện:
`(c[m-1]-c[m-2])*(c[m]-c[m-1])>0`

Mỗi điểm được đưa vào bao lồi và lấy ra khỏi bao lồi nhiều nhất một lần ở phần trên và một lần ở phần dưới nên thuật toán trên (chưa tính thời gian sắp xếp) có độ phức tạp O(n)

2. Bao lồi trên và bao lồi dưới của các đường thẳng

Bài toán: Cho *n* đường thẳng có phương trình:

$$y = a_i x + b_i \ \ (i = 1,2, \dots, n)$$

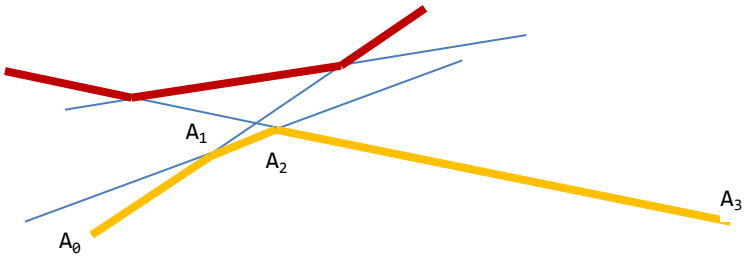
Và *m* giá trị $x_1, x_2, \dots, x_m$ hãy tính giá trị của các hàm:

$$f(x) = min\{a_1x + b_1, a_2x + b_2, \dots, a_nx\_b_n\}$$
$$g(x) = max\{a_1x + b_1, a_2x + b_2, \dots, a_nx + b_n\}$$

Tại các giá trị x đã cho.

Ở đây chúng ta không xem xét thuật toán đơn giản O(nm). Mục đích của chúng ta là xây dựng được thuật toán O((m+n)log n) để trả lời tất cả các truy vấn.

Trước tiên chúng ta tìm cách xây dựng đồ thị của các hàm $f(x), g(x)$ (hình vẽ dưới):



Đường màu đỏ đậm ở bên trên là đồ thị hàm $g(x)$ và đồ thị bên dưới đậm màu vàng là đồ thị $f(x)$. Một nhận xét quan trọng là đồ thị hai hàm này đều có tính chất "lồi" (tức có thể là cạnh của một đa giác lồi nào đó) ta nói đồ thị của $g(x)$ là "bao lồi trên" còn đồ thị của $f(x)$ là "bao lồi dưới"

Xem xét "bao lồi dưới". Có thể thấy $f(x)$ là đường gấp khúc trong đó mỗi đoạn thẳng của đường này nằm trên một đường thẳng khác nhau và điều quan trọng hơn là nếu tính từ trái sang phải thì độ dốc của các đường thẳng chứa đoạn thẳng là giảm dần. Do vậy một cách tự nhiên, khi tìm bao lồi dưới chúng ta sắp xếp các đoạn thẳng theo độ dốc giảm dần (a_i).

Ta xây dựng đường gấp khúc dưới bằng cách xây dựng Danh sách các đường thẳng $l_1 l_2 \dots l_m$ mà đường bao lồi dưới lần lượt đặt trên nó. Trước tiên l_1 là đường thẳng đầu tiên. Giả sử đã xây dựng được $l_1 l_2 \dots l_k$ và đường thẳng xét tiếp là i. Gọi M giao của i và l_{k-1}, N là giao của i và l_k. Có 2 trường hợp xảy ra:

- Hoành độ của M nhỏ hơn hoành độ của N. Khi đó đường thẳng i được thêm vào danh sách
- Hoành độ của M lớn hơn hoặc bằng hoành độ của N. Khi đó đường thẳng l_k bị loại ra khỏi danh sách trước khi thêm đường thẳng i,…

Do vậy ta có thể tổ chức một cơ chế ngăn xếp để quét các đường thẳng theo hệ số góc giảm dần.

Dưới đây là chương trình minh họa:

```
input: L[1], L[2], …, L[n] - các đường thẳng cho bởi cặp số thực (a,b)
output: x[1], x[2], …, x[m+1] - hoành độ các điểm của đường gấp khúc và ret[1], ret[2],
…,ret[m] -cặp (a, b) của các đường thẳng bắt đầu tại x[1], x[2], …, x[m]

#include <bits/stdc++.h>
#define FT first
#define SC second
#define maxn 100005
#define oo 1e15

using namespace std;
typedef pair<double,double> DD;

int n, m, s[maxn];
DD L[maxn];
double x[maxn];
DD ret[maxn];
```

```
inline bool cmp(DD u,DD v) {return (u>v);}
inline double xgiao(DD u,DD v) {return (v.SC-u.SC)/(u.FT-v.FT);}
inline bool xoa(DD u, DD v, DD w) {return (xgiao(u,w)>=xgiao(v,w));}

int main() {
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%lf%lf",&L[i].FT,&L[i].SC);
    sort(L+1,L+n+1,cmp);
    m=0;
    for(int i=1;i<=n;i++) {
        while ((m>0 && L[i].FT==L[s[m]].FT) || (m>1 && xoa(L[s[m-1]],L[s[m]],L[i]))) m--;
        s[++m]=i;
    }
    // Tinh ket qua vao double x[..], ret[...]
    x[1]=-oo;
    for(int i=2;i<=m;i++) x[i]=xgiao(L[s[i-1]],L[s[i]]);
    x[m+1]=oo;
    for(int i=1;i<=m;i++) ret[i]=DD(L[s[i]].FT,L[s[i]].SC);
    // In ket qua
    for(int i=1;i<=m;i++) printf("%0.3lf %0.3lf %0.3lf\n",x[i],ret[i].FT,ret[i].SC);
}
```

Khi đã có bao lồi dưới. Thì việc tính giá trị của $f(x)$ tại một điểm $x = x_0$ được thực hiện bằng cách tìm kiếm nhị phân trên mảng x[…] giá trị lớn nhất ≤ x_0 và sau đó tính trực tiếp nhờ ret[…]:

```
double calc(double x0) {
    int u=upper_bound(x+1,x+m+1,x0)-x-1;
    return ret[u].FT*x0+ret[u].SC;
}
```

Đối với bao lồi trên thì cách làm hoàn toàn tương tự. Điểm khác biệt duy nhất là sắp xếp các đường thẳng theo hệ số góc tăng dần, nếu hệ số góc bằng nhau thì sắp xếp theo b tăng dần (sắp xếp tự nhiên của *pair* trong C++)

3. Kiểm tra một điểm có nằm trong đa giác lồi

Bài toán: Cho điểm M trên mặt phẳng và đa giác lồi $A_1A_2 \dots A_n$ với các đỉnh đánh số cùng chiều kim đồng hồ. Hãy kiểm tra xem M nằm trong, nằm trên biên hay nằm ngoài đa giác trên

Cố định điểm A_1 chúng ta đưa việc kiểm tra M có nằm trong đa giác về việc kiểm tra M có nằm trong tam giác $A_1A_iA_{i+1}$ hay không bằng cách chặt nhị phân như sau:

Đầu tiên nhận xét rằng nếu $\overrightarrow{A_1A_n} \times \overrightarrow{A_1M}$ và $\overrightarrow{A_1A_2} \times \overrightarrow{A_1M}$ cùng dấu thì M chắc chắn nằm ngoài đa giác. Do vậy ta chỉ xét trường hợp khi hai đại lượng trên khác dấu. Việc tìm kiếm nhị phân sẽ thực hiện như sau:

```
int NamTrong(DD M) {
    double t1=ccw(a[1],a[2],M);
    double t2=ccw(a[1],a[n],M);
    if (t1*t2>0) return -1; // nam ngoai
    if (t1==0) return (((M-a[1])^(M-a[2]))<=0) ? 0 : -1;
    if (t2==0) return (((M-a[1])^(M-a[n]))<=0) ? 0 : -1;
    int dau=2, cuoi=n;
    while (cuoi-dau>1) {
```

```
        int giua=(dau+cuoi)/2;
        int t=ccw(a[1],a[giua],M);
        if (t==0) {
            int r=((M-a[1])^(M-a[giua]));
            return (r<0) ? 1: ((r>0) ? -1 : 0);
        }
        if (t*t1>0) dau=giua; else cuoi=giua;
    }
    int t3=ccw(a[giua],a[giua+1],M);
    if (t3==0) return 0;
    return (t1*t3>0) ? 1:-1;
}
```

Hàm trên sẽ cho giá trị 1, 0, -1 tùy theo điểm M nằm trong, nằm trên biên hay nằm ngoài đa giác. Chi phí toàn bộ quá trình kiểm tra là O(log *n*)