

MỤC LỤC

I. LÝ THUYẾT VÀ BIỂU DIỄN THUẬT TOÁN CƠ BẢN VỀ HÌNH HỌC PHẪNG	2
1.1. Một số khái niệm cơ bản của hình học phẳng và cách biểu diễn trên máy tính	2
1.2. Một số bài toán cơ bản của hình học phẳng.....	5
1.3. Bao lồi.....	7
1.4. Một số lưu ý khi giải quyết các bài toán hình học	9
II. MỘT SỐ BÀI TOÁN TIN HỌC SỬ DỤNG THUẬT TOÁN HÌNH HỌC CƠ BẢN	10
2.1. Một số bài toán cơ bản	10
2.2. Một số bài toán diễn hình.....	22
2.3. Một số bài toán nâng cao.....	44
2.4. Một số nội dung tự luyện tập	52
C. KẾT LUẬN VÀ KIẾN NGHỊ	Error! Bookmark not defined.
TÀI LIỆU THAM KHẢO	54

LÝ THUYẾT VÀ BIỂU DIỄN THUẬT TOÁN CƠ BẢN VỀ HÌNH HỌC PHẪNG

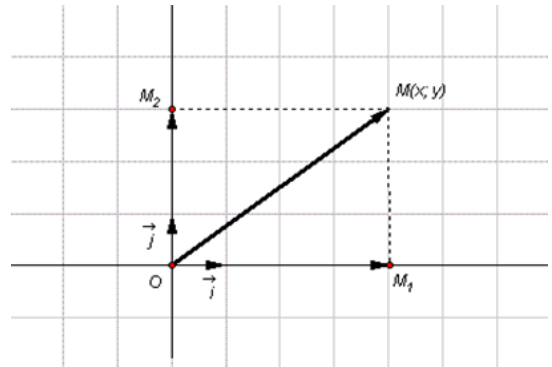
1.1. Một số khái niệm cơ bản của hình học phẳng và cách biểu diễn trên máy tính

1.1.1. Điểm trên hệ trục tọa độ Oxy:

Cho điểm $M(x,y)$ như hình vẽ, x được gọi là hoành độ, y được gọi là tung độ của điểm M trong hệ trục tọa độ Đề Các Oxy , khi đó:

$$\overrightarrow{OM} = \overrightarrow{OM_1} + \overrightarrow{OM_2} = x \cdot \vec{i} + y \cdot \vec{j}$$

- Giả sử x,y là số nguyên, khi đó biểu diễn điểm M theo NNLT có thể dùng một trong hai cách sau:



Cách 1:

```
typedef pair<int,int> Point;
```

Cách 2:

```
struct Point {  
    int x;  
    int y;  
};
```

1.1.2. Khoảng cách giữa 2 điểm:

- Khoảng cách giữa 2 điểm $A(x_A, y_A)$ và $B(x_B, y_B)$, hoặc độ dài của vector \overrightarrow{AB} được tính bằng: $d_{AB} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$

Cài đặt khoảng cách giữa 2 điểm $A(x_A, y_A)$ và $B(x_B, y_B)$ theo NNLT:

```
double dist(Point A, Point B) {  
    return sqrt((B.x-A.x)*(B.x-A.x) + (B.y-A.y)*(B.y-A.y));  
}
```

1.1.3. Đường thẳng:

Đường thẳng trong mặt phẳng xác định khi biết được 2 điểm A, B phân biệt nằm trên đường thẳng đó. Khi đó đường thẳng được xác định là tập hợp các điểm $M(x,y)$ sao cho: $\overrightarrow{AM} = t \cdot \overrightarrow{AB} \Rightarrow (x_M - x_A; y_M - y_A) = t(x_B - x_A; y_B - y_A)$

$$\Rightarrow \begin{cases} x_M - x_A = t(x_B - x_A) \\ y_M - y_A = t(y_B - y_A) \end{cases}$$

- Nếu $t < 0$ thì M nằm ngoài AB về phía A
- Nếu $0 < t < 1$ thì M nằm giữa A, B
- Nếu $t > 1$ thì M nằm ngoài AB về phía B .

1.1.4. Tích chéo của hai vector:

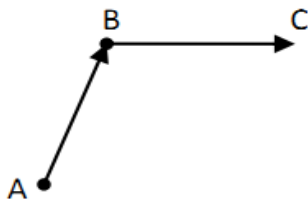
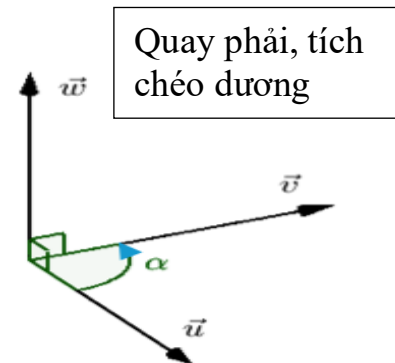
Tích chéo của 2 vector $\vec{u} = (x_u, y_u)$ và $\vec{v} = (x_v, y_v)$ (tích chéo là khái niệm được suy ra từ khái niệm tích có hướng trong không gian vector Oclit nhiều chiều)

$$\text{là: } w = \vec{u} \times \vec{v} = (x_u y_v - x_v y_u) = \begin{vmatrix} x_u & x_v \\ y_u & y_v \end{vmatrix}$$

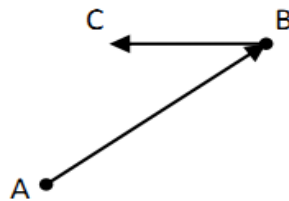
Giá trị của nó bằng định thức của ma trận $\begin{pmatrix} x_u & x_v \\ y_u & y_v \end{pmatrix}$ hoặc tính bằng $|\vec{u}| \cdot |\vec{v}| \cdot \sin(\vec{u}, \vec{v})$. Trong đó góc (\vec{u}, \vec{v}) là góc định hướng, có số đo từ $-\pi$ tới π
→ Giá trị lượng giác \sin của góc định hướng $\alpha = (\vec{u}, \vec{v})$ là:

$$\sin(\vec{u}, \vec{v}) = \frac{\vec{u} \times \vec{v}}{|\vec{u}| \cdot |\vec{v}|}$$

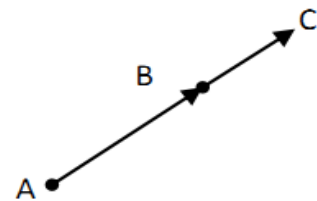
Tích chéo có tác dụng để kiểm tra chiều quay từ \vec{u} đến \vec{v} là chiều quay phải, hay quay trái, ví dụ trong hình vẽ trên là quay trái khi đó $w = \vec{u} \times \vec{v} > 0$; chiều quay từ \vec{u} đến \vec{v} là quay phải nếu tích chéo có giá trị âm; và \vec{u}, \vec{v} thẳng hàng nếu tích chéo của chúng bằng 0.



Rẽ phải



Rẽ trái



Đi thẳng

Minh họa sử dụng tích chéo để xác định chiều quay được biểu diễn theo NNLT như sau:

```
int ccw(Point A, Point B, Point C) {
    double t = (B.x - A.x) * (C.y - A.y) - (B.y - A.y) * (C.x - A.x);
    if (t > 0)
        return 1; //quay trái
    if (t < 0)
        return -1; //quay phải
    return 0; //thang hàng
}
```

1.1.5. Tích vô hướng của hai vector:

Tích vô hướng của 2 vector (hay còn gọi là tích chấm): là một số có giá trị là: $\vec{u} \cdot \vec{v} = |\vec{u}| \cdot |\vec{v}| \cdot \cos(\vec{u}, \vec{v}) = x_u \cdot x_v + y_u \cdot y_v$

Có thể cài đặt tích vô hướng, tích có hướng bằng kĩ thuật chồng toán tử như sau:

<i>Tích vô hướng (tích chấm)</i>	<i>Tích chéo</i>
<pre>int operator *(Point u, Point v) { return (u.x*v.x+u.y*v.y); }</pre>	<pre>int operator ^(Point u, Point v) { return (u.x*v.y-u.y*v.x); }</pre>

1.1.6. Góc:

- Góc tạo bởi hai vector \vec{u}, \vec{v} có giá trị lượng giác $\sin(u, v) = \frac{\vec{u} \times \vec{v}}{|\vec{u}| \cdot |\vec{v}|}$
- Góc tạo bởi tia OA và trục Ox có giá trị lượng giác $\tan \widehat{AOx} = \frac{y_A}{x_A}$

Cài đặt tính số đo của góc tạo bởi tia OA và trục Ox :

```
double goc(Point A) {
    double t=atan2(A.y,A.x);
    if (t<0)
        t=t+2*acos(-1);
    val = 180.0 / PI;
    return t*val;
}
```

1.1.7. Tam giác:

Tam giác được xác định bởi 3 điểm A, B, C , có độ dài 3 cạnh thỏa mãn tất các điều kiện: $a+b>c$; $b+c>a$; $a+c>b$.

Diện tích tam giác được tính thông qua tích có hướng của vector như sau:

$$S_{\Delta ABC} = \frac{1}{2} |\overrightarrow{AB} \times \overrightarrow{AC}| = \frac{1}{2} |\overrightarrow{AB}| \cdot |\overrightarrow{AC}| \cdot \sin(\angle \overrightarrow{AB} \overrightarrow{AC})$$

Cài đặt tính diện tích tam giác ABC :

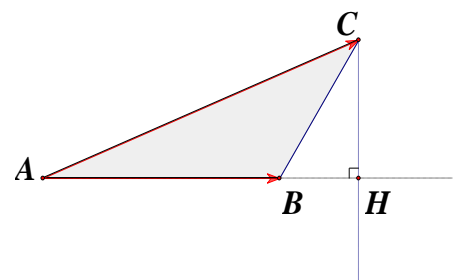
```
double sTriangle(Point A, Point B, Point C) {
    double s=(B.x-A.x)*(C.y-A.y)-(B.y-A.y)*(C.x-A.x);
    return abs(s/2);
}
```

1.1.8. Khoảng cách từ một điểm đến đường thẳng:

Dựa vào diện tích tam giác, ta có thể tính khoảng cách từ điểm C đến đường thẳng d đi qua 2 điểm A, B như sau:

$$CH = \frac{2 \cdot S_{\Delta ABC}}{AB} = \frac{|\overrightarrow{AB} \times \overrightarrow{AC}|}{|\overrightarrow{AB}|}$$

Cài đặt tính khoảng cách từ điểm C đến đường thẳng d đi qua 2 điểm A, B :



```
double dist2(Point A, Point B, Point C){
    return 2*sTriangle(A,B,C)/dist(A,B);
}
```

1.1.9. Đa giác:

Đa giác là một đường gấp khúc khép kín. Trong lập trình, một đa giác được lưu bởi một dãy các đỉnh liên tiếp nhau $A_1(x_1, y_1), A_2(x_2, y_2), \dots, A_N(x_N, y_N)$. Khi đó *diện tích đại số* của một đa giác không tự cắt được định bởi công thức sau:

$$S = \frac{(x_1 - x_2)(y_1 + y_2) + (x_2 - x_3)(y_2 + y_3) + \dots + (x_n - x_1)(y_n + y_1)}{2}$$

Nếu $S > 0$ thì ta đã liệt kê các đỉnh theo chiều ngược chiều kim đồng hồ.

Nếu $S < 0$ thì ta đã liệt kê các đỉnh theo chiều ngược chiều kim đồng hồ.

Còn $|S|$ chính là diện tích của đa giác.

Công thức trên dễ dàng chứng minh bằng cách đi lần lượt theo các đỉnh biên của đa giác, tại mỗi đỉnh kẻ đường thẳng đứng xuống trục Ox , chia đa giác thành các hình thang vuông với hai đáy song song với trục tung Oy để tính diện tích.

1.1.10. Đường tròn:

Đường tròn là tập hợp các điểm cách đều một điểm cho trước (gọi là tâm). Đường tròn biểu diễn thông qua tọa độ tâm và bán kính đường tròn. Đường tròn tâm A bán kính r kí hiệu toán học là: (A, r) . Tương tự điểm thì có 2 cách biểu diễn với đường tròn như sau:

Cách 1

```
struct circle{
    Point A;
    double r;
};
```

Cách 2

```
typedef pair<pair<int,int>,double> circle;
```

Một điểm nằm trong đường tròn khi khoảng cách của của điểm đó đến tâm đường tròn nhỏ hơn hoặc bằng bán kính. Ngược lại, khoảng cách tới tâm lớn hơn bán kính thì nó nằm ngoài đường tròn.

Hai đường tròn có điểm chung nếu khoảng cách giữa 2 tâm nhỏ hơn tổng hai bán kính và ngược lại.

Diện tích hình tròn: $S = \pi \cdot R^2$

1.2. Một số bài toán cơ bản của hình học phẳng

1.2.1. Biểu diễn tuyến tính:

Cho ba vectơ \vec{a}, \vec{b} và \vec{c} hãy tìm hai số p, q để: $\vec{c} = p\vec{a} + q\vec{b}$

Khi đó hai số p, q được tính bằng công thức sau:

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

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

+ Nếu $D = \vec{a} \times \vec{b} \neq 0$ thì có duy nhất một cách biểu diễn tuyến tính vector \vec{c} qua vector \vec{a} và \vec{b}

+ Nếu $D = 0$ thì hai vector song song với nhau, khi đó

$(p, q) = (nan; nan)$ nếu \vec{c} song song với \vec{a} và \vec{b}

$(p, q) = (inf, inf)$ nếu \vec{c} không song song với \vec{a} và \vec{b}

1.2.2. Giao điểm của hai đường thẳng:

Phát biểu bài toán: Cho ba vector \vec{a}, \vec{b} và \vec{c} hãy tìm hai số p, q để: $\vec{c} = p\vec{a} + q\vec{b}$

Trên mặt phẳng tọa độ Đề-các cho hai đường thẳng với phương trình tổng quát: $A_1x + B_1y + C_1 = 0$

$$A_2x + B_2y + C_2 = 0$$

Hãy tìm giao điểm của hai đường thẳng đã cho

Cách giải: Đặt $\vec{u} = (A_1; A_2)$, $\vec{v} = (B_1; B_2)$ và $\vec{w} = (-C_1; -C_2)$ bài toán trở thành biểu diễn vector \vec{w} qua tổ hợp tuyến tính của hai vector \vec{u} và \vec{v} : $\vec{w} = x\vec{u} + y\vec{v}$ và biện luận cho giá trị giao điểm tìm được

1.2.3. Tìm giao điểm của hai đoạn thẳng:

Cho bốn điểm A, B, C, D trên Oxy . Hãy cho biết hai đoạn thẳng AB và CD có giao nhau không, nếu có cho biết tọa độ giao điểm

Thuật toán 1:

+ Viết phương trình đường thẳng đi qua AB và CD

+ Tìm giao điểm M

+ Kiểm tra xem M có nằm trên AB hay không

Với $A(x_A, y_A)$ và $B(x_B, y_B)$ thì phương trình đường thẳng đi qua A, B là:

$$(x - x_A)(y_B - y_A) = (y - y_A)(x_B - x_A)$$

$$\Rightarrow (y_B - y_A)x + (x_A - x_B)y + (x_A y_A - x_A y_B + y_A x_B - x_A y_A) = 0$$

$$\Rightarrow (y_B - y_A)x + (x_A - x_B)y + (y_A x_B - x_A y_B) = 0(d_1)$$

Tương tự như vậy ta tìm phương trình đường thẳng qua CD sau đó tìm giao điểm M bằng cách giải hệ hai phương trình bậc nhất hai ẩn sử dụng định thức, sau đó kiểm tra xem M có nằm giữa A, B hay không theo (theo 1.1.3. Đường thẳng).

Nhận xét: Cách làm trên cần thực hiện nhiều phép tính nên không hiệu quả, dẫn đến tăng sai số.

Thuật toán 2:

Nếu M là giao điểm duy nhất của AB và CD thì $\exists p, q \in [0, 1]$ để:

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

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

Như vậy, ta chỉ cần biểu diễn tuyến tính \overrightarrow{AC} qua \overrightarrow{AB} và \overrightarrow{DC} , sau khi tìm được p và q (theo 1.2.1. Biểu diễn tuyến tính) ta kiểm tra $p, q \in [0,1]$ và tìm tọa độ điểm M như sau:

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

Hướng dẫn cài đặt chương trình tìm giao điểm của 2 đoạn thẳng:

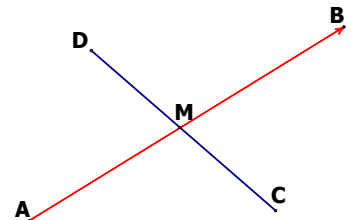
```
double p=(double) ((C-A)^(C-D))/((B-A)^(C-D));
double q=(double) ((C-A)^(B-A))/((C-D)^(B-A));
if (0<p & p<1 & 0<q & q<1) {
    Point M=A+p*(B-A); //Giao của AB, CD là điểm M
}
```

Tương tự như trên ta có thể tìm giao điểm của 1 đường thẳng và 1 tia tương tự như tìm giao điểm của 2 đoạn thẳng như trên, chỉ khác điều kiện bây giờ là:

$$p \geq 0; 0 \leq q \leq 1$$

Chương trình tìm giao điểm của tia AB và đoạn thẳng CD cũng được cài đặt tương tự:

```
double p=(double) ((C-A)^(C-D))/((B-A)^(C-D));
double q=(double) ((C-A)^(B-A))/((C-D)^(B-A));
if (0<p & 0<q & q<1) {
    Point M=A+p*(B-A); //Giao của tia AB với đoạn CD là điểm M
}
```



1.2.4. Tìm hai điểm gần nhau nhất:

Cho tập hợp Q gồm n điểm. Tìm cặp điểm trong Q có khoảng cách gần nhất.

Một thuật toán sơ đẳng nhất là ta đi tính tất cả các khoảng cách có thể được tạo ra từ 2 điểm trong Q , sau đó tìm khoảng cách lớn nhất trong đó. Độ phức tạp của thuật toán như vậy là $O(n^2)$.

Cách cài đặt cơ bản để tìm hai điểm gần nhau nhất :

```
double len=dist(a[1],a[2]);
for (int i=1; i<n; i++)
    for (int j=i+1; j<=n; j++) {
        if (dist(a[i],a[j])<len)
            len=dist(a[i],a[j]);
    }
```

Có cách tiếp cận khác nhằm giảm độ phức tạp thuật toán còn $O(n \log n)$ sử dụng phương pháp chia để trị. Với bài toán kích thước n ta chia làm 2 bài toán con kích thước $n/2$ và kết hợp kết quả trong thời gian $O(n)$.

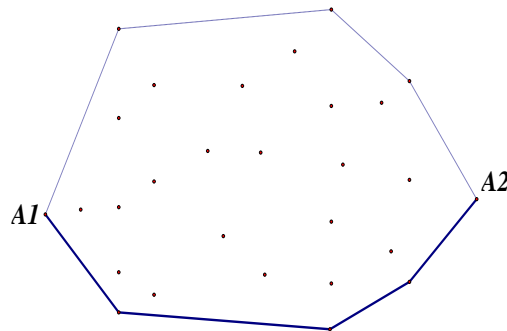
1.3. Bao lồi

1.3.1. Thuật toán tìm bao lồi:

Khái niệm bao lồi: 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 biên của nó.

Có nhiều thuật toán tìm bao lồi khác nhau như: thuật toán bọc gói, quét Graham,... Trong tài liệu này xét một cách tìm bao lồi đơn giản hơn bằng Thuật toán *chuỗi đơn điệu (Monotone chain)* chỉ sử dụng các phép toán vector đơn giản.

Độ phức tạp chung của thuật toán $O(n \log n)$, trong đó sắp xếp mất $O(n \log n)$, xác định bao lồi mất $O(n)$.



Giả sử n điểm đã cho là $A_1, A_2, A_3, \dots, A_n$. Không mất tổng quát ta có thể xem các điểm này được xếp theo hoành độ tăng dần. Nếu hoành độ bằng nhau thì chúng được sắp xếp theo tung độ tăng dần. Khi đó ta chắc chắn A_1 (điểm cực trái) và A_n (điểm cực phải) thuộc bao lồi. Các điểm còn lại được chia thành hai phần:

- Các điểm ở nửa trên hay còn gọi là **chuỗi trên** (đường nét đứt trên hình), các điểm này thỏa mãn: $\overrightarrow{A_1 M} \times \overrightarrow{M A_n} < 0$

- Các điểm ở nửa dưới hay còn gọi là **chuỗi dưới** (đường nét liền trên hình), các điểm này thỏa mãn: $\overrightarrow{A_1 M} \times \overrightarrow{M A_n} > 0$

Hướng dẫn cài đặt thuật toán tìm bao lồi :

```
//Point a[maxn]; đa giác ban đầu đã cho
//Point c[maxn]; các đỉnh bao lồi đánh số cùng chiều kim đồng hồ

sort(a+1, a+n+1);
// Tìm bao lồi
m=2;
c[1]=a[1];
c[2]=a[2];
for(int i=3; i<=n; i++) { //tìm chuỗi trên O(n)
    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--) { //tìm chuỗi dưới O(n)
    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;

```

1.3.2. Kiểm tra điểm thuộc đa giác:

Phát biểu bài toán: Cho đa giác được xác định bởi $A[\maxn]$ được liệt kê lần lượt theo một thứ tự nào đó, kiểm tra xem điểm $B(x_B, y_B)$ có thuộc đa giác đó hay không.

Thuật toán 1: Áp dụng được với mọi loại đa giác không tự cắt.

- Nếu điểm B nằm trên biên của đa giác, được tính là thuộc đa giác.
- Nếu điểm B nằm trong đa giác thì kẻ một tia bất kì xuất phát từ B sao cho không đi qua đỉnh hoặc chứa cạnh nào của đa giác. Tia này sẽ cắt đa giác số lẻ lần. Để tìm giao điểm của tia và đoạn thẳng áp dụng phần 3.3.

Độ phức tạp thuật toán $O(n)$.

Thuật toán 2: Chỉ áp dụng với đa giác lồi

- Nếu điểm B nằm trên biên của đa giác, được tính là thuộc đa giác.
- Nếu B nằm trong đa giác thì tổng diện tích tạo thành từ tam giác với 1 đỉnh là B và một cạnh là cạnh của đa giác bằng tổng diện tích đa giác.

Độ phức tạp là $O(n)$.

Thuật toán 3: Chỉ áp dụng với đa giác lồi

Có cách khác để kiểm tra điểm có thuộc bao lồi hay không với độ phức tạp $O(\log n)$ bằng cách chập nhị phân.

Thuật toán 4: Áp dụng đối với mọi loại đa giác không tự cắt

Dựa theo tổng góc định hướng tạo bởi B với các cặp đỉnh liên tiếp của đa giác. Nếu trị tuyệt đối của tổng đó bằng 2π thì B nằm trong đa giác.

Độ phức tạp: $O(n)$.

1.4. Một số lưu ý khi giải quyết các bài toán hình học

Thứ nhất, như việc giải quyết các bài toán tin khác, học sinh phải xác định đúng Input và Output của bài toán.

Thứ hai:

- Với bài tập đơn giản, chỉ cần vẽ hình học ra và áp dụng những công thức hình học cơ bản là được. Loại bài tập này, giải thuật cũng đơn giản.

- Với bài tập khó, phải xác định, tìm cách biểu diễn và quản lý tốt các đối tượng hình học. Loại bài tập này, giải thuật phức tạp hơn.

Thứ ba, viết chương trình máy tính để biến Input thành Output cần tìm.

Các bài toán hình học thường gây khó khăn vì:

- Dễ bỏ sót các trường hợp đặc biệt:
 - + Trường hợp một đường thẳng là thẳng đứng (có hệ số góc là vô hạn)
 - + Trường hợp các điểm là thẳng hàng với nhau
 - + Trường hợp đa giác là lõm.
- Có thể gây ra lỗi về độ chính xác sau dấu phẩy động nên dù thuật toán đúng nhưng quá trình tính toán vẫn dẫn đến kết quả sai.
 - + Cần nắm vững nguyên lý lưu trữ biến và tính toán trong máy tính để có thể ước lượng được sai số tính toán.
 - + Khi so sánh hai giá trị với nhau, không được dùng dấu “==”, mà phải xét trị tuyệt đối hiệu hai giá trị với một giá trị Epsilon nào đó. Ở đây, Epsilon là một số tương đối bé, tùy vào yêu cầu của bài toán mà ta lựa chọn cho phù hợp.

Ví dụ:

Không được dùng:

`if (x1 == x2) ...`

mà phải dùng:

`if (abs(x1 - x2) < Eps) ...`

Lưu ý:

Trong C++, sử dụng đơn vị radian với góc.

Quy đổi góc α độ thành radian: $\alpha = \alpha * \frac{PI}{180}$.

Thủ thuật: $PI = \text{acos}(-1)$.

II. MỘT SỐ BÀI TOÁN TIN HỌC SỬ DỤNG THUẬT TOÁN HÌNH HỌC CƠ BẢN

2.1. Một số bài toán cơ bản

2.1.1. Bài 1: Vị trí tương đối giữa hai đoạn thẳng

2.1.1.1. Phát biểu bài toán:

Trong hệ trục tọa độ ĐềCác, cho 2 đoạn thẳng AB và CD xác định bởi tọa độ các điểm $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$, $D(x_4, y_4)$. Xét vị trí tương đối của hai đoạn thẳng trên xem chúng trùng nhau tại một điểm (1), trùng nhau nhiều điểm (2), cắt nhau (3), hay không có điểm chung nào (4)?

- Dữ liệu vào file HAIDOANTHANG.INP: Dòng đầu là tọa độ của A và B; dòng thứ hai là tọa độ của C và D.

- Kết quả ra file HAIDOANTHANG.OUT ghi ra số 1, hoặc 2, hoặc 3, hoặc 4 tương ứng các trường hợp trên.

Ví dụ:

HAIDOANTHANG.INP	HAIDOANTHANG.OUT
5 0 0 5 0 0 5 5	3

2.1.1.2. Thuật toán:

- Tính hệ số ptđt đi qua AB và CD;
- Tính $d = a_1 * b_2 - a_2 * b_1$; dx, dy;
- Nếu $d = 0$ {
 - + nếu $dx \neq 0$ hoặc $dy \neq 0 \Rightarrow 4$;
 - + nếu $dx = 0$ và $dy = 0$ xét các trường hợp A với CD, B với CD, C với AB, D với AB nếu nằm trong $\Rightarrow 2$;
 - còn lại nếu trùng vào các mốc $\Rightarrow 1$;
- Còn lại Nếu $d \neq 0$, kiểm tra 2 đoạn thẳng cắt nhau, ta xét nếu $tdt(A, B, C) * tdt(A, B, D) < 0$ và $tdt(C, D, A) * tdt(C, D, B) < 0 \Rightarrow 3$;
- còn lại $\Rightarrow 4$.

2.1.1.3. Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
fstream fi, fo;
struct diem
{
    double x, y;
};
diem a, b, c, d;
double a1, b1, c1, a2, b2, c2, dd, dx, dy;
void dt(diem a, diem b, double &a1, double &b1, double &c1)
{
```

```

        a1=b.y-a.y;
        b1=a.x-b.x;
        c1=-a.x*b.y+b.x*a.y;
    }
    double tdt(diem a,double a1,double b1,double c1)
    {
        return (a1*a.x+b1*a.y+c1);
    }
    int main()
    {
        fi.open("Haidoanthang.inp",ios::in);
        fo.open("Haidoanthang.out",ios::out);
        fi>>a.x>>a.y>>b.x>>b.y;
        fi>>c.x>>c.y>>d.x>>d.y;
        dt(a,b,a1,b1,c1);
        dt(c,d,a2,b2,c2);
        dd=a1*b2-a2*b1;
        dx=-c1*b2+c2*b1;
        dy=-a1*c2+a2*c1;
        if (dd==0)
        {
            if (dx!=0||dy!=0) fi<<4;
            else
            {
                if (((a.x-c.x)*(c.x-b.x)>0||(a.y-c.y)*(c.y-
b.y)>0)||
                    ((a.x-d.x)*(d.x-b.x)>0||(a.y-d.y)*(d.y-
b.y)>0)||
                    ((c.x-a.x)*(a.x-d.x)>0||(c.y-a.y)*(a.y-
d.y)>0)||
                    ((c.x-b.x)*(b.x-d.x)>0||(c.y-b.y)*(b.y-
d.y)>0)) fo<<2;
                else
                if ((a.x==c.x&&a.y==c.y)|| (b.x==c.x&&b.y==c.y)||
(a.x==d.x&&a.y==d.y)|| (b.x==d.x&&b.y==d.y)) fo<<1;
                else fo<<4;
            } }
        else

```

```

        {if
(tdt(c,a1,b1,c1)*tdt(d,a1,b1,c1)<0&&tdt(a,a2,b2,c2)*tdt(b,a2,b2,c
2)<0) fo<<3;
        else

if(tdt(c,a1,b1,c1)==0||tdt(d,a1,b1,c1)==0||tdt(a,a2,b2,c2)==0||td
t(b,a2,b2,c2)==0) fo<<1;
        else fo<<4;
        }
        return 0;
    }

```

2.1.2. Bài 2: Loại hình tứ giác

2.1.2.1. Đề bài

Cho tứ giác lồi ABCD có tọa độ lần lượt là: (x1,y1), (x2,y2), (x3,y3), (x4,y4). Hãy xét xem tứ giác ABCD là hình gì? Trong các hình sau: Hình bình hành (1); Hình thoi (2); Hình chữ nhật (3); Hình vuông (4); Hình tứ giác thường (5) ?

- Dữ liệu vào file **TUGIACLOI.INP** gồm 4 dòng, mỗi dòng là số tọa độ của các điểm đã nêu ở trên.

- Kết quả ra file **TUGIACLOI.OUT** ghi ra số 1, hoặc 2, hoặc 3, hoặc 4, hoặc 5 tương ứng các trường hợp trên.

Ví dụ:

TUGIACLOI.INP	TUGIACLOI.OUT
0 0 0 5 5 5 5 0	4

2.1.2.2. Thuật toán:

Với điều kiện tứ giác là hình bình hành khi có 2 cặp cạnh đối song song và bằng nhau (xét hai cặp vector bằng nhau) thì xét tiếp:

```

{ + là hình vuông nếu có 4 cạnh bằng nhau, đường chéo bằng nhau (4);
+ còn không : là hình chữ nhật nếu đường chéo bằng nhau (3);
+ còn không : là hình thoi nếu có 4 cạnh bằng nhau (2);
+ còn lại : hình bình hành (1)
}

```

else còn lại là hình tứ giác thường.

2.1.2.3. Chương trình:

```
#include <bits/stdc++.h>
using namespace std;
fstream fi,fo;
struct diem
{
    int x,y;
};
diem a,b,c,d;
bool ss(diem a,diem b, diem c, diem d)
{
    if (b.x-a.x==c.x-d.x&&b.y-a.y==c.y-d.y) return true;
    else return false;
}
int kc(diem a, diem b)
{
    return ((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y));
}
int main()
{
    fi.open("Tugiacloi.inp",ios::in);
    fo.open("Tugiacloi.out",ios::out);
    fi>>a.x>>a.y>>b.x>>b.y>>c.x>>c.y>>d.x>>d.y;
    if(ss(a,b,c,d)&&ss(b,c,d,a))//đk HBH
    {
        if
(kc(a,b)==kc(b,c)&&kc(b,c)==kc(c,d)&&kc(a,c)==kc(b,d)) fo<<4;
        else if (kc(a,c)==kc(b,d)) fo<<3;
        else if (kc(a,b)==kc(b,c)&&kc(b,c)==kc(c,d)) fo<<2;
        else fo<<1;
    }
    else fo<<5;
    return 0;
}
```

2.1.3. Bài 3. Tìm số mảnh cắt

2.1.3.1. Đề bài

Một mảnh giấy hình chữ nhật được cắt bởi những nhát kéo. Cho biết toạ độ của mảnh giấy cũng như các nhát cắt, hãy xác định số mảnh được cắt rời.

Giả thiết mảnh giấy được đặt trong một hệ toạ độ sao cho các mép giấy song song với các trục toạ độ, góc dưới trái của nó trùng với điểm (0; 0) và góc trên phải của nó trùng với điểm (m; n). Mỗi nhát cắt được xác định bởi hai đầu mút trên biên của mảnh giấy sao cho đảm bảo đoạn thẳng nối hai đầu mút này thực sự cắt mảnh giấy.

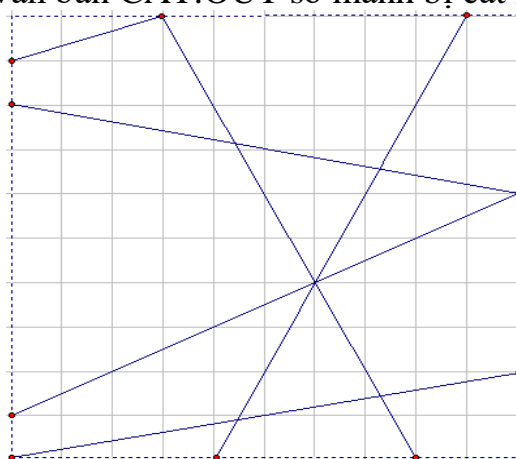
Dữ liệu vào cho trong file văn bản CAT.INP gồm:

- Dòng đầu ghi hai giá trị nguyên dương m và n.
- Dòng tiếp theo ghi số nhát cắt k.
- Các dòng tiếp theo, mỗi dòng ghi toạ độ của một nhát cắt gồm 4 số: 2 số đầu là hoành độ và tung độ của một đầu mút và 2 số sau là hoành độ và tung độ của đầu mút còn lại.

Các toạ độ trong file dữ liệu đều là những số nguyên và được ghi cách nhau ít nhất một dấu trắng nếu trên cùng một dòng.

Giới hạn: m, n, k ≤ 1000.

Kết quả ghi ra file văn bản CAT.OUT số mảnh bị cắt rời.



Ví dụ: Hình vẽ trên mô tả một mảnh giấy bị cắt bởi 6 nhát kéo thành 13 mảnh, tương ứng với các file vào, ra dưới đây:

CAT.INP	CAT.OUT
10 10	13
6	
3 10 0 9	
8 0 3 10	
0 0 10 2	
0 8 10 6	
9 10 4 0	
10 6 0 1	

2.1.3.2. Thuật toán

Hướng dẫn:

+ Bài toán này phù hợp để rèn luyện kỹ năng lập trình các bài toán hình học cho học sinh giỏi tin học vào thời điểm học xong các phép toán hình học cơ bản, giúp rèn luyện được kỹ năng kiểm tra vị trí tương đối giữa điểm và đoạn thẳng, giữa đoạn thẳng và đoạn thẳng

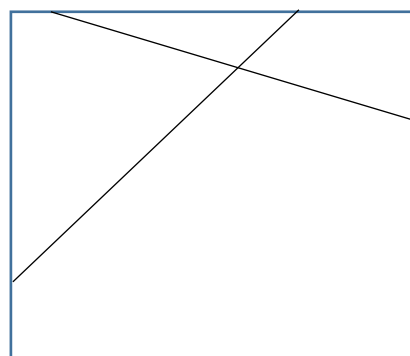
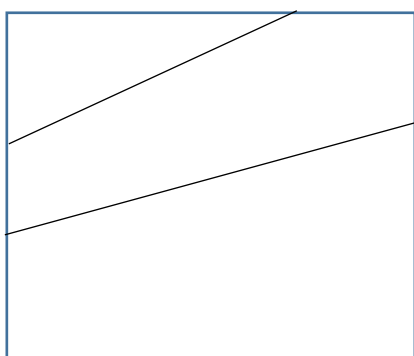
+ Thuật toán của bài toán khá dễ tìm, giáo viên yêu cầu học sinh khảo sát với số nhất cắt từ nhỏ đến lớn. Với mỗi nhất cắt, học sinh sẽ dễ dàng thấy số mảnh được tách rời phụ thuộc vào số giao điểm của nhất cắt hiện tại với các nhất cắt trước đó.

+ Lưu ý rằng giao điểm không nằm trên biên và nếu có nhiều giao điểm trùng nhau thì giao điểm ấy chỉ được tính một lần.

Cụ thể:

+ $i=1$: Xét nhất cắt đầu tiên ta có kết quả $res = 2$;

+ $i=2$: Xét đến nhất cắt thứ 2, sẽ có hai khả năng xảy ra là số giao điểm bằng 0 hoặc số giao điểm bằng 1.



=> Kết quả $res = 3$ (nếu $sgd=0$); hoặc $res = 4$ (nếu $sgd=1$);

+ $i=3$: Xét đến nhất cắt thứ 3, sẽ có 6 trường hợp xảy ra. Do khi có 2 nhất cắt có 2 trường hợp xảy ra, với mỗi trường hợp đó lại có 3 khả năng xảy ra là: $sgd=0$; $sgd=1$; $sgd=2$;

	Giá trị tìm được	Kết quả
1	$res=3$; $sgd=0$;	$res=4$;
2	$res=3$; $sgd=1$;	$res=5$;
3	$res=3$; $sgd=2$;	$res=6$;
4	$res=4$; $sgd=0$;	$res=5$;
5	$res=4$; $sgd=1$;	$res=6$;
6	$res=4$; $sgd=2$;	$res=7$;

Chỉ cần khảo sát đến đây đã có thể tìm được công thức tính kết quả của bài toán qua việc tích lũy dần các mảnh tương ứng với số lượng nhất cắt lớn dần:

res = res + sgd + 1;

Trong đó, sgd là số lượng giao điểm của nhất cắt thứ i với i-1 nhất cắt trước nó.

2.1.3.3. *Chương trình tham khảo*

```
#include <bits/stdc++.h>
#define maxn 1000007
#define maxk 1000
using namespace std;
typedef struct point {long x, y;};
typedef struct fpoint {double x, y;};
typedef struct line {point p1,p2;};
line a[maxk];
fpoint gd[maxn];
int m,n,k;
long dem,res,sl;
void nhap()
{   int i;    point A,B;
    cin>>m>>n;
    cin>>k;
    for (i=1;i<=k;i++)
    {   cin>>A.x>>A.y>>B.x>>B.y;
        a[i].p1 = A; a[i].p2 = B;    }
}
void ptdt(point p1, point p2, long &a, long &b, long &c)
{   a = p1.y - p2.y;
    b = p2.x - p1.x;
    c = - (a*p1.x+b*p1.y); }
void giao2dt(point p1,point p2,point p3,point p4,int &kq,
fpoint &g)
{   double d,dx,dy;
    long a1,b1,c1,a2,b2,c2;
    ptdt(p1,p2,a1,b1,c1);
    ptdt(p3,p4,a2,b2,c2);
    d =  a1*b2 - a2*b1;
    dx = c2*b1 - c1*b2;
    dy = a2*c1 - a1*c2;
    if (d!=0) {g.x = dx/d; g.y = dy/d; kq = 1; return;}
    else
```

```

        if (dx!=0 || dy!=0) {kq=0; return;}
        else kq=2;
    }
    int ccw(point p1, point p2, point p3)
    {
        int dx1=p2.x-p1.x;
        int dy1=p2.y-p1.y;
        int dx2=p3.x-p1.x;
        int dy2=p3.y-p1.y;
        if (dy1*dx2 < dy2*dx1) return 1;
        if (dy1*dx2 > dy2*dx1) return -1;
        return 0;
    }
    bool intersect(line l1, line l2)
    {
        int k1=ccw(l1.p1,l1.p2,l2.p1);
        int k2=ccw(l1.p1,l1.p2,l2.p2);
        int k3=ccw(l2.p1,l2.p2,l1.p1);
        int k4=ccw(l2.p1,l2.p2,l1.p2);
        return ((k1*k2<0) && (k3*k4<0));
    }
    void process()
    {
        int i;
        res=1;
        for (i=1;i<=k;i++)
        {
            dem=0; sl=0;
            for (int j=1; j<i;j++)
            {
                bool ok = intersect(a[i],a[j]);
                if (ok==1)
                {
                    int kq;
                    fpoint g;
                    giao2dt(a[i].p1,a[i].p2,a[j].p1,a[j].p2,kq,g);
                    if (sl==0) {sl++; gd[sl] = g; dem++;}
                    else
                    {
                        bool ok1=1;
                        for (int h=1;h<=sl;h++)
                        if ((gd[h].x==g.x) && (gd[h].y==g.y))
                        {ok1=0;break;}
                        if (ok1==1) {sl++; gd[sl]=g; dem++;}
                    }
                }
            }
        }
    }

```

```

    }
    }
    }
    res = res + dem + 1;
}
}
int main()
{
    freopen("cat.inp", "r", stdin);
    freopen("cat.out", "w", stdout);
    nhap();
    process();
    cout<<res;
    return 0;
}

```

2.1.4. Bài 4. Đếm bộ ba điểm thẳng hàng

2.1.4.1. Đề bài

OICamp đang dự định lấy bài toán hình học để đưa vào cuộc thi marathon tuần thứ tư. Các admin đang sinh test cho bài toán này. Một bộ test được đánh giá là khó nếu có nhiều những bộ 3 điểm thẳng hàng. Sau khi bộ test được sinh xong, các admin cần kiểm tra xem có bao nhiêu bộ 3 điểm thẳng hàng. Tuy vậy, do đã mệt mỏi sau khi viết chương trình sinh test, các admin cần bạn giúp đỡ trong việc này.

Dữ liệu: Đọc từ file TRIPOINT.INP có cấu trúc dạng:

- + Dòng thứ nhất ghi số N là số điểm mà trình sinh test sinh ra.
- + N dòng tiếp theo, mỗi dòng ghi tọa độ của một điểm.

Kết quả: Ghi ra file TRIPOINT.OUT một số duy nhất là số bộ ba điểm thẳng hàng.

Giới hạn: $1 \leq N \leq 2000$. Tọa độ các điểm có trị tuyệt đối không quá 10000.

Ví dụ:

TRIPOINT.INP	TRIPOINT.OUT
6	3
0 0	
0 1	
0 2	
1 1	

1 2	
2 2	

2.1.4.2. Thuật toán

Hướng dẫn:

- + Bài toán rèn luyện cho học sinh tư duy toán học tổ hợp để tìm ra đáp số và tư duy hình học liên quan đến hệ số góc của đường thẳng;
- + Thuật toán duyệt có sử dụng yếu tố hình học và toán học;
- + Có thể mở rộng thành các bài toán tương tự như *Đếm số bộ k điểm thẳng hàng*.

Thuật toán 1. Thuật toán vét cạn với độ phức tạp là $O(n^3)$

Duyệt mọi bộ 3 điểm, với mỗi bộ 3 điểm, kiểm tra 3 điểm đó có thẳng hàng với nhau hay không? Nếu có thẳng hàng thì tăng res lên 1.

Thuật toán 2. Thuật toán hình học kết hợp tổ hợp với độ phức tạp $O(n^2)$

Học sinh có thể làm 1 số khảo sát đơn giản để phát hiện thuật toán giải bài toán:

- + Ba điểm A, B, C thẳng hàng khi và chỉ khi 2 đường thẳng AB và AC có cùng hệ số góc.
- + Nếu có m điểm thẳng hàng với nhau thì *số bộ 3 điểm thẳng hàng* đúng bằng số cách chọn ra 3 điểm bất kỳ trong m điểm đó, *bằng số tổ hợp chập 3 của m phần tử*.
- + Nếu có nhiều dãy các điểm thẳng hàng thì kết quả được tích lũy bằng tổng các kết quả của các dãy đó.

Cụ thể:

- + Cố định một điểm (giả sử điểm A) rồi tính hệ số góc của đường thẳng đi qua điểm A và tất cả các điểm còn lại.
- + Sắp xếp lại mảng hệ số góc
- + Đếm số lượng số hệ số góc bằng nhau, nếu có m_i số hệ số góc k_i bằng nhau thì tăng kết quả lên $k_i * \frac{k_i - 1}{2}$.

Bằng phân tích nêu trên ta có thuật toán giải bài toán:

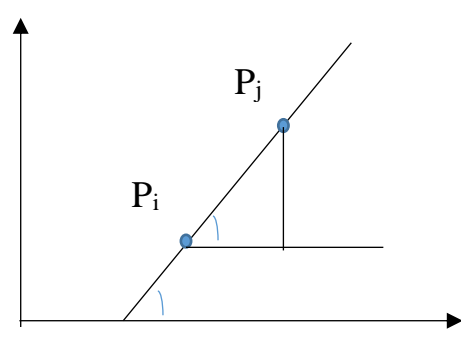
Với mỗi điểm bắt đầu của 1 bộ ba điểm thẳng hàng i (i=1..n):

+ Duyệt mọi điểm j khác i ($j=i+1, \dots, n$), với mỗi cặp điểm (i, j) , tính góc hợp bởi đường thẳng đi qua 2 điểm $(p[i], p[j])$ với trục hoành Ox .

$$dx = p[j].x - p[i].x;$$

$$dy = p[j].y - p[i].y;$$

$$goc[] = dy/dx;$$

(nếu $dx=0$ thì đường thẳng song song với Ox , gán $goc[] = +\infty$;) 

- + Sắp xếp véc tơ goc theo thứ tự tăng dần về độ lớn.
- + Đếm số lượng điểm thẳng hàng với nhau có điểm bắt đầu thứ i là m
- + Tính số bộ ba điểm thẳng hàng tương ứng với i và tích lũy vào res ;
- + Giả code:

```
for(int i = 1; i ≤ N - 2; i++){
    for(j = i + 1 → N)
        Tính mảng hệ số góc goc[];
    Sắp xếp lại mảng goc[];
    Đếm trên mảng goc[] số lượng số hệ số góc giống nhau
    res +=  $k_i * \frac{k_i - 1}{2}$ .
}
```

2.1.4.3. Chương trình tham khảo

```
#include<bits/stdc++.h>
#define ll long long
#define fi first
#define se second
#define pb push_back
using namespace std;
int n;
const int nmax=2e3+10;
struct point{ll x,y;} p[nmax];
ll res=0;
typedef pair<float,int> ii;
vector<ii> g;
int ccw(point p1,point p2,point p3)
{ ll dx1=p2.x-p1.x;
  ll dx2=p3.x-p1.x;
  ll dy1=p2.y-p1.y;
  ll dy2=p3.y-p1.y;
```

```

        if (dy1*dx2<dy2*dx1) return 1;
        if (dy1*dx2>dy2*dx1) return -1;
        return 0;
    }
    int main()
    { ios_base::sync_with_stdio(0);cin.tie(0);
      freopen("tripoint.inp","r",stdin);
      freopen("tripoint.out","w",stdout);
      cin>>n;
      for (int i=1;i<=n;i++)      {   cin>>p[i].x>>p[i].y;      }
      for (int i=1;i<=n;i++)
      {   g.clear();
          for (int j=i+1;j<=n;j++)
          {   ll dx=p[j].x-p[i].x;
              ll dy=p[j].y-p[i].y;
              if (dx==0) g.pb(ii(LLONG_MAX,j));
              else g.pb(ii((float)dy/dx,j));}
          sort(g.begin(),g.end());
          int sl=1;
          point p1=p[i],p2=p[g[0].se];
          for (int j=1;j<g.size();j++)
          {   if (ccw(p1,p2,p[g[j].se])==0) sl++;
              else      {   res=res+1ll*sl*(sl-1)/2;      sl=1;
p2=p[g[j].se];   }
              }
          res=res+1ll*sl*(sl-1)/2;
      }
      cout<<res;
      return 0;}

```

2.2. Một số bài toán điển hình

2.2.1. Bài 5. METERAIN - Mưa thiên thạch

2.2.1.1. Phát biểu bài toán:

Phú ông nhận được thông tin về một trận mưa thiên thạch sắp ập xuống trái đất. Không những thế, Phú ông còn biết tọa độ của vị trí điểm rơi của mỗi một thiên thạch. Phú ông nhờ Cuội xác định xem có bao nhiêu thiên thạch có thể rơi xuống cánh đồng của ông ta. Cánh đồng của Phú ông có dạng một hình đa giác

lỗi được xác định bởi danh sách các đỉnh được liệt kê theo thứ tự ngược chiều kim đồng hồ.

Yêu cầu: Xác định xem trong tập cho trước các điểm rơi của thiên thạch, có bao nhiêu điểm nằm trong cánh đồng của Phú ông. Các điểm nằm trên biên của cánh đồng không được tính là điểm nằm trong cánh đồng.

Input

- Dòng đầu tiên là số nguyên n ($3 \leq n \leq 5000$) là số đỉnh của đa giác lồi mô tả cánh đồng của Phú ông.
- Mỗi dòng trong n dòng tiếp theo chứa cặp tọa độ của một đỉnh của đa giác lồi.
- Dòng tiếp theo là số nguyên m ($2 \leq m \leq 5000$) - số thiên thạch rơi xuống.
- Mỗi dòng trong số m dòng cuối cùng chứa 2 số là tọa độ điểm rơi của một thiên thạch.

Các tọa độ là các số nguyên có trị tuyệt đối không quá 10^6 .

Output

Ghi ra m dòng, mỗi dòng tương ứng với 1 điểm rơi của thiên thạch. Ghi "YES" nếu điểm rơi của thiên thạch nằm trong cánh đồng và ghi "NO" nếu trái lại.

Ví dụ:

Input	Output
4	NO
2 4	NO
8 4	YES
6 8	YES
4 6	
4	
3 5	
4 7	
5 5	
6 7	

2.2.1.2. Hướng dẫn:

Ta sử dụng tìm kiếm nhị phân cho mỗi truy vấn.

Gọi điểm đang xét là p . Ta chọn hai cạnh nối điểm 1 với điểm n và điểm 2 là 2 cạnh chốt. Sau đó, gọi l và r là 2 điểm mà cạnh nối từ 1 đến l và r thỏa mãn điểm p đang xét nằm giữa 2 cạnh này (Ban đầu $l = 2$, $r = n$). Ta sẽ kiểm tra cạnh nối điểm 1 với điểm $\frac{l+r}{2}$ (mid) nằm cùng phía với cạnh nào. Nếu cùng phía với l

thì $l = mid$, và ngược lại. Làm như vậy đến khi $r - l = 1$. Khi đó ta xét xem $S_{l|r}$ có bằng $S_{l|p} + S_{l|p} + S_{p|r}$ hay ko ($S_{l|r}$ là diện tích tạo bởi 3 điểm l , điểm p và điểm r). Đây chính là câu trả lời của bài toán.

2.2.1.3. Chương trình tham khảo:

```
#include<bits/stdc++.h>
#define maxn 5005
#define maxC 1000000000
#define MOD (1e9 + 7)
#define mp make_pair
#define PB push_back
#define F first
#define S second
#define pii pair<int, int>
#define Task "metarain"
#define ll long long
using namespace std;
int n, Query;
pii a[maxn];

void setup() {
    cin >> n;
    for(int i=1 ; i<=n ; ++i) cin >> a[i].F >> a[i].S;
}

pii Vect(pii a, pii b) {
    return mp(b.F - a.F, b.S - a.S);
}

ll CCW(pii x, pii y, pii mete) {
    pii a = Vect(x, y);
    pii b = Vect(y, mete);
    return (1ll * a.F * b.S - 1ll * a.S * b.F);
}

ll dientich(pii a, pii b, pii c) {
    return abs((b.F - a.F) * (b.S + a.S) + (c.F - b.F) * (c.S + b.S) + (a.F - c.F) * (a.S + c.S));
}
```



```

bool calc(pii mete) {
    int r = n, l = 2;
    while(r - l > 1) {
        int mid = (r + l) >> 1;
        if(CCW(a[l], a[mid], mete) > 0) l = mid;
        else r = mid;
    }
    if(l == 2 && CCW(a[l], a[2], mete) <= 0) return 0;
    if(r == n && CCW(a[l], a[n], mete) >= 0) return 0;
    ll s1 = dientich(a[l], mete, a[r]);
    ll s2 = dientich(a[l], a[l], mete);
    ll s3 = dientich(mete, a[l], a[r]);
    ll S = dientich(a[l], a[l], a[r]);
    if(!s3) return 0;
    return (s1 + s2 + s3 == S);
}

int main() {
    ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
    // freopen(Task".inp", "r", stdin);
    // freopen(Task".out", "w", stdout);
    setup();
    cin >> Query;
    while(Query--) {
        pii mete;
        cin >> mete.F >> mete.S;
        bool ok = calc(mete);
        cout << (ok == true ? "YES" : "NO") << '\n';
    }
    return 0;
}

```

2.2.2. Bài 6: Bao lồi

2.2.2.1. Phát biểu bài toán

Cho N ($N \leq 1000$) điểm a_1, a_2, \dots, a_N trên mặt phẳng. Các điểm đều có toạ độ nguyên và không có 3 điểm bất kỳ trong chúng thẳng hàng. Hãy viết chương trình xác định một đa giác không tự cắt có đỉnh là một số điểm trong các điểm đã cho và chứa tất cả các điểm còn lại. Hãy tính chu vi và diện tích đa giác này.

- Dữ liệu vào file BAOLLOI.INP: Dòng đầu ghi số N, các dòng tiếp theo ghi tọa độ các điểm.

- Kết quả ra file BAOLLOI.OUT: Gồm 3 số M, V, S, trong đó M là số đỉnh của đa giác, V là chu vi và S là diện tích của đa giác tìm được, lấy độ chính xác 3 chữ số sau dấu chấm thập phân.

Ví dụ:

BAOLOI.INP	BAOLOI.OUT	Giải thích
8	6 39.631 105.000	<i>Bao lồi gồm các đỉnh:</i>
7 1		7 1
16 4		2 8
2 3		5 11
13 7		11 12
11 12		16 4
5 11		
2 8		
7 5		

2.2.2.2. Thuật toán:

- Sắp xếp các điểm theo tung độ tăng dần.
- Chọn điểm đầu tiên là điểm $a[1]$, $vt=1$;
- Duyệt tất cả các điểm để chọn điểm tiếp theo (i), nếu $ptdt(vt,i)$ mà tất cả các điểm còn lại đều cùng 1 phía thì điểm i là đỉnh tiếp theo của bao lồi, $vt=i$, lặp lại cho đến khi không chọn được điểm nào thêm.

2.2.2.3. Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
fstream fi,fo;
const int nm=1e4+1;
struct diem
{
    double x,y;
};
diem a[nm];
vector<diem>b;
int i,j,n,d;
```

```

bool c[nm], kt;
double a1, b1, c1, p;

double kc(double x1, double y1, double x2, double y2)
{
    return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}

bool ss(diem a, diem b)
{
    return (a.y < b.y);
}

bool cungphia(diem d1, diem d2)
{
    int j=0;
    double kq, a1, b1, c1;
    a1=d1.y-d2.y; b1=d2.x-d1.x; c1=d1.x*d2.y-d2.x*d1.y;
    do
    {
        j++;
        kq=a1*a[j].x+b1*a[j].y+c1;
    } while (kq==0 && j<n);
    if(kq<0)
    {
        for(int k=j+1; k<=n; k++)
            if(a1*a[k].x+b1*a[k].y+c1>0) return 0;
    }
    else
    {
        for(int k=j+1; k<=n; k++)
            if(a1*a[k].x+b1*a[k].y+c1<0) return 0;
    }
    return 1;
}

void dtcvdg()
{
    double dt=0, cv=0;
    for(i=0; i<b.size()-1; i++)
    {
        dt=dt+(b[i].x-b[i+1].x)*(b[i].y+b[i+1].y)/2;
        cv=cv+kc(b[i].x, b[i].y, b[i+1].x, b[i+1].y);
    }
}

```

```

    }
    dt=abs(dt);
    fo<<setprecision(3)<<fixed<<cv<<" "<<dt;
}
int main()
{
    fi.open("BAOLOI.INP",ios::in);
    fo.open("BAOLOI.OUT",ios::out);
    fi>>n;
    for(i=1;i<=n;i++)
        fi>>a[i].x>>a[i].y;
    sort(a+1,a+1+n,ss);
    b.push_back(a[1]);c[1]=1;
    do
    {
        kt=false;
        for(i=2;i<=n;i++)
            if(c[i]==0&&cungphia(b[b.size()-1],a[i]))
            {
                b.push_back(a[i]);
                c[i]=1;
                kt=true;
            }
    }while(kt);
    b.push_back(a[1]);
    fo<<b.size()-1<<" ";
    dtcvdg();
    return 0;
}

```

2.2.3. Bài 7: Loại đa giác

2.2.3.1. Phát biểu bài toán

Cho đa giác A gồm có n đỉnh A_1, A_2, \dots, A_n ($3 \leq n \leq 1000$), tọa độ các đỉnh là số nguyên có giá trị tuyệt đối không vượt quá 30.000. Hãy xác định xem đa giác là đa giác gì trong các loại đa giác sau: Đa giác lồi (1); Đa giác lõm (2); Đa giác tự cắt (3).

Dữ liệu vào file **DAGIAC.INP**: Dòng đầu là số n; n dòng tiếp theo lần lượt là tọa độ các đỉnh của đa giác A.

Kết quả ra file **DAGIAC.OUT**: Dòng đầu là câu trả lời với các trường hợp đa giác như trên; dòng thứ 2 là nếu là trường hợp (1), (2) thì tính diện tích đa giác; còn lại trường hợp (3) ghi số 0. (*Lấy độ chính xác 3 chữ số sau dấu chấm thập phân*)

Ví dụ:

DAGIAC.INP	DAGIAC.OUT
4	1
0 0	25.000
0 5	
5 5	
5 0	

2.2.3.2. Thuật toán:

Xét các trường hợp:

- Đa giác lồi (xét cùng phía): Nếu thay tất cả các các đỉnh còn lại các phương trình đường thẳng (a_1, a_2) , (a_2, a_3) , ... (a_n, a_{n+1}) $a_{n+1} = a_1$ đều cùng dấu \Rightarrow đa giác lồi;

- Đa giác tự cắt: Duyệt tất cả các cặp cạnh (a_i, a_{i+1}) , (a_j, a_{j+1}) với $j = i + 2 \dots n$, nếu có cặp cạnh nào đó cắt nhau \Rightarrow đa giác tự cắt.

Còn lại là đa giác lõm.

2.2.3.3. Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
fstream fi,fo;
const int nm=1002;
struct diem
{
    double x,y;
};
diem a[nm];
bool kt;
int i,j,n;

double tdt(diem a,diem b,diem c)
{ double a1,b1,c1;
  a1 = a.y-b.y;
```

```

        b1 = b.x-a.x;
        c1 = a.x*b.y-a.y*b.x;
        return (a1*c.x+b1*c.y+c1);
    }
void dtdg()
{
    double dt=0;
    for(int i=1;i<=n;i++)
        dt=dt+(a[i].x-a[i+1].x)*(a[i].y+a[i+1].y)/2;
    dt=abs(dt);
    fo<<setprecision(3)<<fixed<<dt;
}
bool ktloi(int i, int j)
{
    int k,vt;
    for(k=1;k<=n;k++)
        if(tdt(a[i],a[j],a[k])*tdt(a[i],a[j],a[k+1]))<0)
return 0;
    return 1;
}
bool ktcac(int i, int j)
{
    if(tdt(a[i],a[i+1],a[j])*tdt(a[i],a[i+1],a[j+1]))<0
    && tdt(a[j],a[j+1],a[i])*tdt(a[j],a[j+1],a[i+1]))<0)
return 1;
    return 0;
}
bool ktac()
{
    int i,j;
    for(i=1;i<n-1;i++)
        for(j=i+2;j<=n;j++)
            if(ktcac(i,j)) return 1;
    return 0;
}
int main()
{
    fi.open("DAGIAC.INP",ios::in);
    fo.open("DAGIAC.OUT",ios::out);
    fi>>n;
    for(i=1;i<=n;i++)
        fi>>a[i].x>>a[i].y;
    a[n+1]=a[1];

```

```

kt=1;
for(i=1;i<=n;i++)
    if(ktloi(i,i+1)==0){kt=0; break;}
if(kt)
{
    fo<<1<<endl;
    dtdg();
}
else
{
    if(kttc()) fo<<3;
    else
    {
        fo<<2<<endl;
        dtdg();
    }
}
return 0;
}

```

2.2.4. Bài 8. Thửa đất lớn nhất

2.2.4.1. Phát biểu bài toán

Bờm lại thắng Phú ông trong một cuộc đánh cược và theo thỏa thuận từ trước, Phú ông buộc phải cho Bờm một thửa đất trong phần đất đai rộng lớn của mình. Bản đồ phần đất của Phú ông có thể coi là một mặt phẳng với hệ trục tọa độ Descartes vuông góc Oxy trên đó đánh dấu n ($n \geq 3$) cột mốc hoàn toàn phân biệt và không đồng thời thẳng hàng, cột mốc thứ i có tọa độ (x_i, y_i) . Bờm được chọn ba cột mốc trong số đó để nhận thửa đất có dạng hình tam giác có ba đỉnh là vị trí ba cột mốc được chọn.

Yêu cầu: Hãy giúp Bờm chọn ba cột mốc để nhận được thửa đất có diện tích lớn nhất.

Dữ liệu: Vào từ file văn bản TRILAND.INP có cấu trúc:

- + Dòng 1 chứa số nguyên dương $(3 \leq \leq 3000)$;
- + Các dòng tiếp theo, dòng thứ i chứa hai số nguyên x_i, y_i
($\forall i: |x_i|, |y_i| \leq 10^9$) cách nhau bởi dấu cách.

Kết quả: Ghi ra file văn bản TRILAND.OUT diện tích của thửa đất Bờm sẽ nhận theo phương án tìm được. Diện tích này phải ghi dưới dạng số thực với đúng 1 chữ số sau dấu chấm thập phân.

TRILAND.INP	TRILAND.OUT
8	11.5
1 1	
1 2	
1 5	
2 2	
3 1	
3 3	
4 1	
6 6	

Bài toán này tiếp tục củng cố hàm tính diện tích của 1 đa giác, giúp rèn luyện kỹ năng lập trình thuật toán tìm bao lồi của một tập hợp điểm; kỹ năng lựa chọn thuật toán và nâng cao kỹ năng kiểm soát lỗi trong một chương trình dài, có sự tổng hợp nhiều phép toán hình học

2.2.4.2. Thuật toán

Cách duyệt trâu: Thuật toán xét tất cả bộ 3 điểm rồi tính diện tích tam giác ($O(n^3)$) có thể đạt 50% số điểm. Chú ý phải tính diện tích tam giác bằng tích chéo hoặc theo kiểu diện tích hình thang, dùng công thức Heron dính sai số nhiều và chậm.

Cách chuẩn:

Tìm bao lồi của n điểm, một nhận xét quan trọng đó là tam giác có diện tích lớn nhất phải có 3 đỉnh là 3 đỉnh của bao lồi. Thuật toán thực hiện như sau:

Đánh số các đỉnh của đa giác lồi theo chiều thuận (ngược chiều kim đồng hồ),

Xét mọi cặp điểm i, j , ($i < j$) với mỗi cặp, tìm điểm k ($j < k$) tạo ra diện tích tam giác (i, j, k) lớn nhất và ghi nhận lại. Giả sử bao lồi có đỉnh:

for $i := 1$ to $m - 2$ do

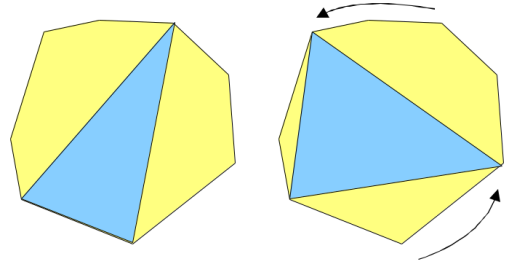
for $j := i + 1$ to $m - 1$ do

«Tìm k : Diện tích tam giác (i, j, k) lớn nhất và ghi nhận»;

Bây giờ ta tìm cách xác định nhanh chỉ số k ứng với mỗi giá trị j chạy từ $i+1$ tới $m-1$.

Với giá trị đầu tiên bằng $i + 1$, nếu ta xét lần lượt $k = j + 1, j + 2, \dots, m$ thì diện tích tam giác (i, j, k) tăng dần lên giá trị cực đại rồi giảm dần xuống. Lý do là đi xa dần đường thẳng $i - j$ rồi lại đi gần về đường thẳng (i, j) khi k chạy trên bao lồi. Vậy giá trị k ứng với vị trí $j = i + 1$ có thể tìm như sau: Đặt $k = j + 2$, chừng nào điểm $k + 1$ tạo ra diện tích lớn hơn diện tích (i, j, k) ta tăng k lên 1 đơn vị (Hình bên trái).

Với giá trị j' tiếp theo (bằng $j + 1$), k' tương ứng với nó được tìm bằng cách tương tự, từ vị trí k của bước trước, tăng dần k lên chừng nào thấy diện tích vẫn tăng lên, ta thu được giá trị k' sau vòng lặp (hình bên phải)



Thuật toán có thể viết:

```
for i := 1 to m - 2 do
{
    k := i + 2;
    for j := i + 1 to m - 1 do
    {
        while (k < m) and («Tăng k lên cho diện tích không
        nhỏ hơn») k := k + 1;
        «Ghi nhận diện tích tam giác (i, j, k)»;
    }
}
```

«In ra diện tích lớn nhất trong các lần ghi nhận»;

Độ phức tạp tính toán có thể đánh giá qua số lần thực hiện phép toán $k = k + 1$, với mỗi giá trị i , k được khởi tạo bằng $i + 2$ và không bao giờ vượt quá m , tức là với mỗi giá trị i , số lần tăng k không quá $m - i - 2$. Vì i chạy từ 1 tới $m - 2$. Độ phức tạp của thuật toán là $O(m^2)$.

2.2.4.3. Chương trình tham khảo

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int n,m;
const int nmax=3001;
struct point {ll x,y; } p[nmax],d[5];
struct ii {point p; int i; } a[nmax];
long double res=0;
ll dis(point p1,point p2)
```

```

{    return (p2.x-p1.x)*(p2.x-p1.x)+(p2.y-p1.y)*(p2.y-p1.y);}
float theta(point p1,point p2)
{   ll dx=p2.x-p1.x;
    ll dy=p2.y-p1.y;
    ll ax=abs(dx);
    ll ay=abs(dy);
    float t;
    if (dx==0&&dy==0) t=0;
    else t=(float) dy/(ax+ay);
    if (dx<0) t=2-t;
    else if (dy<0) t=t+4;
    return (t*90.0);
}
bool compare(ii a1,ii a2)
{   if (theta(a[1].p,a1.p)<theta(a[1].p,a2.p)) return 1;
    if
(theta(a[1].p,a1.p)==theta(a[1].p,a2.p)&&dis(a[1].p,a1.p)<dis(a[1
].p,a2.p))
    return 1;
    return 0; }
bool compare2(ii a1,ii a2)
{   if (dis(a[1].p,a1.p)>dis(a[1].p,a2.p)) return 1;
    return 0; }
void simpleclosepath()
{   int i,j=1;
    for (i=2;i<=n;i++)
        if
(a[i].p.y<a[j].p.y||(a[i].p.y==a[j].p.y&&a[i].p.x<a[j].p.x)) j=i;
    swap(a[1],a[j]);
    sort(a+2,a+n+1,compare);
    int k=n;
    while (theta(a[1].p,a[k].p)==theta(a[1].p,a[k-1].p)) k--;
    sort(a+k,a+n+1,compare2); }
int ccw(point p1,point p2,point p3)
{   ll dx1=p2.x-p1.x;
    ll dx2=p3.x-p1.x;
    ll dy1=p2.y-p1.y;
    ll dy2=p3.y-p1.y;
    if (dy1*dx2<dy2*dx1) return 1;

```

```

        if (dy1*dx2>dy2*dx1) return -1;
        return 0;
    }
void graham()
{   simpleclosepath();
    n=n+1;
    a[n]=a[1];
    m=2;
    for (int i=3;i<=n;i++)
    {   while (ccw(a[m].p,a[m-1].p,a[i].p)>=0) m--;
        m++;
        if (m<0) m=2;
        swap(a[m],a[i]);   }
    m=m-1;
    for (int i=1;i<=m;i++) p[i]=a[i].p;}
long double area()
{   long double s=0;
    d[4]=d[1];
    for (int i=1;i<=3;i++)
        s=s+(d[i].y+d[i+1].y)*(d[i].x-d[i+1].x);
    s=abs(s);
    return s; }
void xuli()
{   for (int i=1;i<=m-2;i++)
        for (int j=i+1;j<=m-1;j++) {
            d[1]=p[i];   d[2]=p[j];
            long double dt=0;
            for (int k=j+1;k<=m;k++)
            {   d[3]=p[k];
                long double tg=area();
                if (dt<tg) dt=tg;
                else if (dt>tg) break;}
            res=max(res,dt);
        }
}
int main()
{   ios_base::sync_with_stdio(0);cin.tie(0);
    freopen("triland.inp","r",stdin);

```

```

    freopen("triland.out", "w", stdout);
    cin>>n;
    for (int i=1;i<=n;i++) cin>>a[i].p.x>>a[i].p.y, a[i].i=i;
    graham();
    xuli();
    cout<<setiosflags(ios::fixed);
    cout<<setprecision(1)<<(res/2);
    return 0;
}

```

2.2.5. Bài 9. Câu chuyện người lính (military.*)

2.2.5.1. Phát biểu bài toán

“Tôi vẫn nhớ chiến trường Điện Biên năm đó rất ác liệt, rất nhiều người lính đã ngã xuống. Tại vùng căn cứ này, địch cho xây dựng lô cốt, hàng rào dây thép gai rất nhiều, vòng trong nối vòng ngoài, tạo thành nhiều vòng bảo vệ ...”. Đó là dòng hồi tưởng của 1 người lính già đã từng tham gia chiến dịch Tây Bắc lịch sử. Lần theo những trang sử được ghi chép lại, người ta biết rằng tướng Đờ Cát lúc đầu chưa chọn vị trí để đặt sở chỉ huy mà tìm cách thiết lập các vòng bảo vệ bằng dây thép gai nối các cứ điểm lại với nhau, sau đó sẽ chọn đặt sở chỉ huy tại vị trí an toàn nhất là ở vị trí mà có nhiều vòng bảo vệ bao quanh nhất. Mỗi 1 vòng bảo vệ là 1 đa giác không tự cắt tạo thành bằng cách nối 1 số cứ điểm lại với nhau bằng dây thép gai, 1 cứ điểm thuộc về không quá 1 vòng bảo vệ, các vòng bảo vệ phải được thiết lập sao cho giữa 2 vòng bảo vệ bất kỳ X và Y thì phần diện tích chung của X và Y = Min (diện tích X, diện tích Y) hoặc = 0. Trên mặt phẳng toạ độ, các cứ điểm được coi như các điểm có toạ độ nguyên. Bạn hãy xác định xem, sở chỉ huy của tướng Đờ Cát sẽ được bảo vệ tối đa bởi mấy vòng bảo vệ.

Input: đọc từ file văn bản MILITARY.inp gồm:

- + Dòng 1: số nguyên N là số cứ điểm. ($1 \leq N \leq 4000$).
- + N dòng tiếp theo, dòng thứ i gồm 2 số nguyên x_i, y_i tương ứng là toạ độ của cứ điểm i. Các toạ độ đều là số nguyên dương ≤ 10000 .

Output: ghi ra file văn bản MILITARY.out một dòng duy nhất ghi ra số lượng vòng bảo vệ tối đa mà sở chỉ huy của tướng Đờ Cát có thể được bao bọc.

Example:

MILITARY.inp	MILITARY.out	Giải thích
4 100 100 200 100 100 200	1	<i>Ta nối cứ điểm 1, 2, 3, 4 lại tạo thành 1 vòng bảo vệ, đặt trụ sở chỉ huy bên trong thì ra được đáp án. Ngoài ra còn có các phương án khác là nối cứ điểm</i>

300 300		<i>1, 2, 3 tạo thành 1 vòng bảo vệ, nối cứ điểm 2, 3, 4 thành 1 vòng bảo vệ, ... nhưng tất cả các phương án này thì khi chọn vị trí đặt trụ sở chỉ huy thì vẫn tối đa = 1</i>
---------	--	---

2.2.5.2. Thuật toán

- Ứng dụng thuật toán tìm bao lồi;
- Tìm 1 bao lồi của tập hợp N điểm. Sau đó xóa nó đi khỏi mảng. Rồi lại tiếp tục tìm bao lồi cho những điểm còn lại cho đến khi chỉ còn 1 điểm hoặc các điểm còn lại tạo thành 1 đường thẳng;
- Kết quả của bài toán là số lần tìm bao lồi của bài toán.

2.2.5.3. Chương trình tham khảo

```
#include <bits/stdc++.h>
#define ii pair<int, int>
#define X first
#define Y second
const int N = 4004;
using namespace std;
int n, res;
ii O;
ii a[N]; int s[N];
bool chk[N];
void operator -= (ii& A, ii B) {A.X -= B.X; A.Y -= B.Y;}
bool CCW(ii A, ii B, ii C) {C -= B; B -= A; return B.X * C.Y > B.Y * C.X;}
void Graham_Scan() {
    int i, k, t;
    sort(a + 1, a + 1 + n);
    k = 1;
    for(i = 1; i <= n; i++) {
        while (k > 2 && CCW(a[s[k - 2]], a[s[k - 1]], a[i])) k--;
        s[k++] = i;
    }
    for(i = n, t = k + 1; i; i--) {
        while (k > t && CCW(a[s[k - 2]], a[s[k - 1]], a[i])) k--;
        s[k++] = i;
    }
    for(i = 1; i <= n; i++) chk[i] = false;
    for(i = 1; i < k; i++) chk[s[i]] = true;
    int m = 0;
    for(i = 1; i <= n ; i++) if (!chk[i])
        a[++m] = a[i];
}
```

```

        n = m;
    }
    int main()
    {
        freopen("MILITARY.inp", "r", stdin);
        freopen("MILITARY.out", "w", stdout);
        cin>>n;
        for(int i = 1; i <= n; i++) cin>>a[i].X>>a[i].Y;
        while (n > 2) {res++; Graham_Scan();}
        cout<<res;
        return 0;
    }

```

2.2.6. Bài 10. Nông trại táo (APPLES.*)

2.2.6.1. Phát biểu bài toán

Sau nhiều năm theo cha khởi nghiệp làm nông, hai anh em trai nhà nọ đã phát triển khu vườn cây ăn quả sau nhà trở thành một nông trại táo rộng lớn gồm có n cây táo. Nông trại của họ khá đặc biệt, người ta không thể tìm thấy 3 gốc cây nào nằm thẳng hàng.

Người cha cảm thấy đến lúc cần được nghỉ ngơi để hưởng thụ tuổi già nên quyết định giao toàn bộ nông trại cho 2 người con. Tuy nhiên, để tránh những tranh chấp về sau, người cha dự định làm một hàng rào đi qua 2 gốc cây tạo thành một đường phân cách chia nông trại thành 2 phần đều nhau về số lượng cây táo.

Việc chia đều các cây táo cho 2 người con trai không làm ông băn khoăn vì số lượng cây táo là một số chẵn. Nhưng việc chọn ra 2 cây trong nông trại để làm hàng rào là điều không đơn giản đối với ông.

Yêu cầu: Cho tọa độ của n gốc cây táo trong nông trại. Hãy chỉ ra 2 cây táo được chọn để làm hàng rào sao cho chia nông trại thành 2 phần bằng nhau về số lượng cây táo.

Input: đọc vào từ file văn bản APPLES.INP gồm:

- + Dòng đầu chứa số nguyên chẵn n ($2 \leq n \leq 10^6$) – số lượng cây táo.
- + Dòng thứ i trong n dòng tiếp theo chứa cặp số nguyên x_i, y_i ($|x_i|, |y_i| \leq 10^6$) – tọa độ của gốc cây táo thứ i ($1 \leq i \leq n$). Dữ liệu đảm bảo không có 2 tọa độ trùng nhau, không có 3 gốc cây nằm trên cùng một đường thẳng.

Output: Ghi ra tập tin văn bản APPLES.OUT gồm 2 số nguyên là thứ tự của 2 cây táo được chọn. Nếu có nhiều phương án thực hiện thì đưa ra một phương án bất kỳ.

Example:

APPLES.INP	APPLES.OUT
6	2 5

3 5	
1 3	
3 1	
6 1	
8 3	
6 5	

2.2.6.2. Thuật toán

- Đầu tiên ta cần xét bài toán xác định chiều của tam giác trong mặt phẳng tọa độ (chiều của tam giác được định nghĩa là chiều của các cạnh tam giác, có thể là $1 \rightarrow 2 \rightarrow 3$ hoặc ngược lại : $1 \rightarrow 3 \rightarrow 2$). Ta xác định chiều tam giác bằng cách tính diện tích của tam giác thông qua tọa độ 3 đỉnh.

- Gọi 3 đỉnh của tam giác có tọa độ là (x_1, y_1) , (x_2, y_2) , (x_3, y_3) . Công thức tính diện tích tam giác :

Có biểu thức $f = (x_2 - x_1) \cdot (y_2 + y_1) + (x_3 - x_2) \cdot (y_3 + y_2) + (x_1 - x_3) \cdot (y_1 + y_3)$

→ Diện tích tam giác : $S = |f / 2|$.

- Để xét chiều của tam giác (có thể hiểu là chiều 3 cạnh nối giữa 3 đỉnh 1, 2, 3), ta chỉ quan tâm đến biểu thức f . Nếu biểu thức f có giá trị dương ($f > 0$) thì chiều của tam giác là chiều kim đồng hồ (chiều từ $1 \rightarrow 2 \rightarrow 3$ là chiều kim đồng hồ), còn nếu f có giá trị âm ($f < 0$) thì ngược lại ($1 \rightarrow 2 \rightarrow 3$ ngược chiều kim đồng hồ).

- Dựa vào bài toán xác định chiều tam giác vừa nêu, ta áp dụng vào bài Apples.

- Ta tìm điểm có tọa độ trái nhất và dưới nhất (đặt là $A[1]$). Giờ chỉ việc tìm 1 điểm có tọa độ mà nằm giữa các điểm còn lại (đặt là $A[g]$). Nếu nối $A[1]$ và $A[g]$ thì ta sẽ được đường thẳng chia đôi tập hợp. Làm sao để tìm $A[g]$? Cách đơn giản nhất là sắp xếp lại các điểm còn lại rồi lấy điểm ở giữa.

- Tiêu chí sắp xếp : xét dần các cặp điểm $(A[2], A[3])$, $(A[3], A[4])$, ... $(A[n - 1], A[n])$, với mỗi cặp điểm $(A[i], A[i + 1])$, nếu diện tích tam giác $(A[1], A[i], A[i + 1]) < 0$, tức là $A[1] \rightarrow A[i] \rightarrow A[i + 1]$ ngược chiều kim đồng hồ, ta đổi chỗ $A[i]$ và $A[i + 1]$.

- Sau khi đã sắp xếp theo tiêu chí trên, điểm $A[g]$ cần tìm là $A[n / 2 + 1]$.

- Kết quả bài toán là $A[1]$ và $A[g]$.

2.2.6.3. Chương trình tham khảo

```
#include<bits/stdc++.h>
using namespace std;
```

```

const long long maxn=1e6+1;
long long
n,x[maxn],y[maxn],minY=1000001,minX=1000001,minI,dem[maxn];
long long check(long long a, long long b, long long c)
{return (x[b]-x[a])*(y[b]+y[a])+(x[c]-
x[b])*(y[c]+y[b])+(x[a]-x[c])*(y[a]+y[c]);
}

int main()
{ ios_base::sync_with_stdio(false);
  cin.tie(NULL);
  freopen("APPLES.INP","r",stdin);
  freopen("APPLES.OUT","w",stdout);
  cin>>n;
  x[0]=0;
  y[0]=0;
  for(long long i=1;i<=n;i++)
  {   cin>>x[i]>>y[i];
      if(y[i]==minY)
      {   if(x[i]<minX)
          {
              minX=x[i];
              minI=i;
          }
      }
      if(minY>y[i])
      {
          minY=y[i];
          minX=x[i];
          minI=i;
      }
  }
  for(long long i=1;i<=n;i++)
  {   if(i!=minI)
      {for(long long j=i+1;j<=n;j++)
          {
              if(j!=minI)
              {
                  if(check(minI,i,j)>0) dem[i]++;
                  else dem[j]++;
              }
          }
      }
      if(dem[i]==n/2-1)
      {
          cout<<minI<<" "<<i;

```



```

        break;
    }
}
}

```

2.2.7. BÀI 11. BẢO VỆ THÀNH PHỐ (CITY.*)

2.2.7.1. Phát biểu bài toán

Ở vương quốc Hòa Bình, có N thành phố được đánh số từ 1 đến N . Thành phố thứ i nằm ở vị trí có tọa độ (x_i, y_i) . Tuy thế giới vẫn đang trong thời gian hòa bình, vị vua của đất nước tin rằng họ phải luôn sẵn sàng bảo vệ đất nước. Quốc vương quyết định xây một tường thành khép kín có dạng một đa giác để bảo vệ một số thành phố của vương quốc. Các thành phố nằm trong và trên biên giới của tường thành này sẽ có thể yên tâm phát triển trong yên bình. Để đảm bảo cho sự vững chắc của nền kinh tế, phải có ít nhất K thành phố được bảo vệ. Tuy nhiên, việc xây dựng một tường thành kiên cố là vô cùng tốn kém, vì vậy, Quốc vương muốn chọn phương án xây tường thành có chu vi nhỏ nhất.

Cho tọa độ của N thành phố, và số nguyên dương K , nhiệm vụ của bạn là tìm một tường thành bao quanh ít nhất K thành phố với chu vi nhỏ nhất.

- **Input:** đọc từ file văn bản CITY.inp gồm:
 - + Dòng thứ nhất gồm 2 số nguyên dương N, K .
 - + N dòng tiếp theo: mỗi dòng chứa 2 số nguyên x_i, y_i là tọa độ của một thành phố trong vương quốc.
- **Output:** ghi ra file văn bản CITY.out một số duy nhất là độ dài nhỏ nhất của tường thành.

Giới hạn:

- + $1 \leq K \leq N \leq 100$. Trong 35% số test có $1 \leq K \leq N \leq 25$.
- + Các tọa độ có giá trị tuyệt đối không vượt quá 10^4

Example:

CITY.inp	CITY.out
5 4 1 1 2 5 3 2 3 3	8.0645

4 1	

2.2.7.2. Thuật toán

- Tìm đa giác lồi bao tập gồm n đỉnh ($n \geq k$);
- Tính chu vi của đa giác lồi trên;
- Duyệt tất cả các đỉnh i, j với mỗi đỉnh i khác j thỏa mãn $y[j] > y[i]$ hoặc $y[j] = y[i]$ và $x[j] > x[i]$;
 + Tính hoành độ đỉnh m và tung độ đỉnh $m+1$: $xx[m] = x[j] - x[i]$, $yy[m++] = y[j] - y[i]$;
- + Với mỗi tập các điểm lớn hơn hoặc bằng k thì cập nhật chu vi nhỏ nhất;

2.2.7.3. Chương trình tham khảo

```
#include <bits/stdc++.h>
#define sqr(x) (1LL*(x)*(x))
using namespace std;
const double eps = 1e-8, oo = 1e9;
long long area(int x1, int y1, int x2, int y2, int x3, int y3)
{
    long long res = 1LL * (x1 - x2) * (y1 + y2);
    res += 1LL * (x2 - x3) * (y2 + y3);
    res += 1LL * (x3 - x1) * (y3 + y1);
    if (res < 0) res = -res;
    return res;
}
int inside(int x, int y, int x1, int y1, int x2, int y2, int x3, int y3)
{
    long long sumArea = area(x, y, x1, y1, x2, y2);
    sumArea += area(x, y, x1, y1, x3, y3);
    sumArea += area(x, y, x3, y3, x2, y2);
    return area(x1, y1, x2, y2, x3, y3) == sumArea;
}
double dist(int x, int y, int xx, int yy)
{
    return sqrt(sqr(x - xx) + sqr(y - yy));
}
double solve(int x[], int y[], int n, int k)
{
    if (k == 1) return 0;

    double f[105][105];
```

```

int cntOnSlope[105] = {0};
for (int i = 1; i < n; i++)
    for (int j = i + 1; j < n; j++)
    {
        double u = atan2(y[i], x[i]), v = atan2(y[j], x[j]);
        long long distI = sqr(x[i]) + sqr(y[i]), distJ =
sqr(x[j]) + sqr(y[j]);
        if ((fabs(u - v) < eps && distI > distJ) || u > v +
eps)
            swap(x[i], x[j]), swap(y[i], y[j]);
    }
for (int i = 2; i < n; i++)
{
    double u = atan2(y[i], x[i]), v = atan2(y[i - 1], x[i -
1]);
    if (fabs(u - v) < eps) cntOnSlope[i] = cntOnSlope[i - 1]
+ 1;
}
for (int i = 0; i <= n; i++)
    for (int j = 0; j <= k; j++)
        f[i][j] = oo;
for (int i = 1; i < n; i++)
    f[i][min(cntOnSlope[i] + 2, k)] = dist(0, 0, x[i], y[i]);
for (int i = 1; i < n; i++)
    for (int j = i + 1; j < n; j++)
    {
        double u = atan2(y[i], x[i]), v = atan2(y[j], x[j]);
        if (fabs(u - v) < eps) continue;
        int cntIn = 0;
        for (int q = 1; q < n; q++)
            cntIn += inside(x[q], y[q], 0, 0, x[i], y[i],
x[j], y[j]);
        for (int q = 1; q <= k; q++)
            if (f[i][q] < oo)
            {
                int qq = min(q + cntIn - cntOnSlope[i] -
1, k);
                f[j][qq] = min(f[j][qq], f[i][q] + dist(x[i],
y[i], x[j], y[j]));}
    }
double res = oo;
for (int i = 1; i < n; i++) res = min(res, f[i][k] + dist(0, 0,
x[i], y[i]));
return res;}

int main()
{
    int n, k, x[105], y[105];
    double ans = oo;
    cin >> n >> k;
    for (int i = 0; i < n; i++) cin >> x[i] >> y[i];

```

```

for (int i = 0; i < n; i++)
{
    int xx[105], yy[105], m = 1;
    for (int j = 0; j < n; j++)
        if (i != j && (y[j] > y[i] || (y[j] == y[i] && x[j]
> x[i])))
            xx[m] = x[j] - x[i], yy[m++] = y[j] - y[i];
    if (m >= k) ans = min(ans, solve(xx, yy, m, k));
}
cout<<ans;
return 0;
}

```

2.3. Một số bài toán nâng cao

2.3.1. Bài 12: Dây tam giác bao nhau

2.3.1.1. Phát biểu bài toán:

Cho N tam giác được cho bởi toạ độ các đỉnh $A_i(x1_i, y1_i)$, $B_i(x2_i, y2_i)$, $C_i(x3_i, y3_i)$, ($i=1..N$). Tam giác i được gọi là bao tam giác j nếu 3 đỉnh của tam giác j đều thuộc miền trong của tam giác i . Hãy tìm dãy các tam giác bao nhau có số lượng lớn nhất?

- Dữ liệu vào file DTAMGIAC.INP: Dòng đầu ghi số N ($1 < N \leq 1000$); Trong N dòng tiếp theo dòng thứ $i+1$ ghi toạ độ 3 đỉnh của tam giác thứ i .

- Kết quả ra file TDAMGIAC.OUT: Số lượng dãy các tam giác bao nhau nhiều nhất nếu có; trường hợp không có dãy các tam giác bao nhau ghi số 0.

Ví dụ:

DTAMGIAC.INP						DTAMGIAC.OUT	Giải thích
5						2	Có 2 tam giác bao nhau là: Tam giác số 1 bao tam giác số 3.
1	1	14	1	1	7		
2	3	7	3	2	9		
5	1	3	4	8	2		
-4	2	2	5	8	3		
7	6	9	8	9	2		

2.3.1.2. Thuật toán:

- Đọc dữ liệu, tính diện tích các tam giác.
- Sắp xếp diện tích các tam giác tăng dần.
- Chọn dãy tam giác theo thuật toán dãy con tăng dài nhất.

(Xét tam giác j nằm trong tam giác $i \Rightarrow$ xét 3 đỉnh của tam giác j nằm trong tam giác $i \Rightarrow$ xét từng đỉnh của j . Nếu thay các đỉnh vào các ptđt đi qua các đỉnh của i đều cùng dấu thì đỉnh này sẽ nằm trong.)

2.3.1.3. Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
fstream fi,fo;
const int nm=1e4+1;
struct tamgiac
{
    double x1,y1,x2,y2,x3,y3,s;
};
tamgiac a[nm];
int i,j,n,b[nm],c[nm],maxb=0;
double a1,b1,c1,p;

double kc(double x1,double y1,double x2,double y2)
{
    return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
}
bool ss(tamgiac x, tamgiac y)
{
    return (x.s<y.s);
}
double tdt(double x1,double y1,double x2,double y2,double x3,double y3)
{
    return ((y2-y1)*x3+(x1-x2)*y3-x1*y2+x2*y1);
}
bool ktdtg(tamgiac a,double x, double y)
{
    if((tdt(a.x1,a.y1,a.x2,a.y2,x,y)>=0 &&
        tdt(a.x2,a.y2,a.x3,a.y3,x,y)>=0 &&
        tdt(a.x3,a.y3,a.x1,a.y1,x,y)>=0)||
        (tdt(a.x1,a.y1,a.x2,a.y2,x,y)<=0 &&
        tdt(a.x2,a.y2,a.x3,a.y3,x,y)<=0 &&
        tdt(a.x3,a.y3,a.x1,a.y1,x,y)<=0)) return true;
    return false;
}
bool kttgtg(tamgiac a, tamgiac b)
{
    if(ktdtg(a,b.x1,b.y1)&&ktdtg(a,b.x2,b.y2)&&ktdtg(a,b.x3,b.y3))
return true;
    return false;
}
int main()
{
    fi.open("DTAMGIAC.INP",ios::in);
    fo.open("DTAMGIAC.OUT",ios::out);
    fi>>n;
    for(i=1;i<=n;i++)
    {
```

```

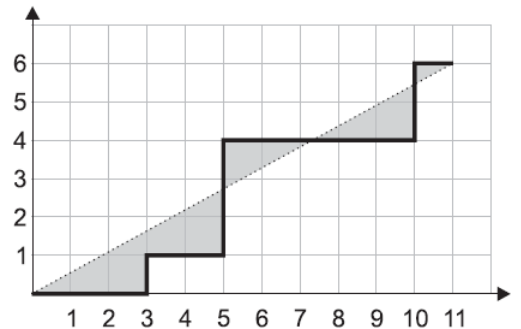
fi>>a[i].x1>>a[i].y1>>a[i].x2>>a[i].y2>>a[i].x3>>a[i].y3;
a1=kc(a[i].x1,a[i].y1,a[i].x2,a[i].y2);
b1=kc(a[i].x2,a[i].y2,a[i].x3,a[i].y3);
c1=kc(a[i].x1,a[i].y1,a[i].x3,a[i].y3);
p=(a1+b1+c1)/2;
a[i].s=sqrt(p*(p-a1)*(p-b1)*(p-c1));
}
sort(a+1,a+1+n,ss);
b[1]=1;c[1]=0;
for(i=2;i<=n;i++)
{
for(j=i-1;j>=1;j--)
if(kttgtg(a[i],a[j])&&b[i]<b[j])
{
b[i]=b[j];
c[i]=j;
}
b[i]++;
maxb=max(maxb,b[i]);
}
fo<<maxb;
return 0;

```

2.3.2. Bài 13. Diện tích đa giác tự cắt

2.3.2.1. Phát biểu bài toán:

Trên mặt phẳng lưới tọa độ Đề các, xuất phát từ điểm $(0, 0)$ người ta vẽ một đường gấp khúc có các cạnh song song với trục tọa độ theo quy tắc sau: bút vẽ được điều khiển bằng chương trình là một xâu các ký tự U, R. Gặp lệnh U bút vẽ sẽ chuyển lên trên một đơn vị, còn khi gặp lệnh R bút vẽ sẽ chuyển sang phải một đơn vị. Khi hết chương trình bút vẽ được kéo thẳng về gốc tọa độ. Hình bên tương ứng với chương trình vẽ là RRRURRUUURRRRUUR.



Yêu cầu: Tìm diện tích bị giới hạn bởi đường gấp khúc và đường thẳng. Trên hình bên, miền cần tính diện tích được tô đậm.

Dữ liệu: Vào từ file văn bản **AREA.INP** gồm một dòng chứa một xâu các ký tự R, U, xác định một chương trình vẽ. Bút vẽ luôn chuyển động trong phạm vi lưới kích thước 1000×1000 .

Kết quả: Đưa ra file văn bản **AREA.OUT** diện tích tìm được, kết quả lấy độ chính xác 3 chữ số sau dấu chấm thập phân.

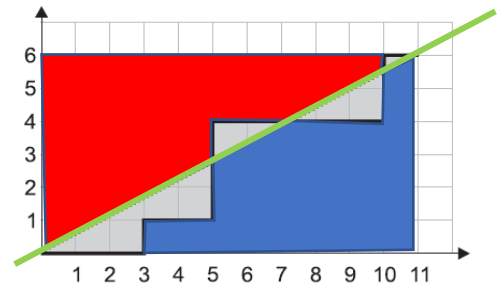
Ví dụ:

AREA.INP	AREA.OUT
RRRURRUUURRRRRRUUR	8.515

2.3.2.2. Thuật toán:

- Đặt $s = x_x * y_y$, diện tích hình chữ nhật có cạnh x_x, y_y ; (x_x là hoành độ max theo ox , y_y là tung độ max theo oy)

- Đặt s_1 = diện tích hình màu đỏ; là các điểm nằm phía trên đường thẳng đi qua hai điểm $(0,0)$ và (x_x, y_y) .



- Đặt s_2 = diện tích hình màu xanh; là các điểm nằm phía dưới đường thẳng đi qua hai điểm $(0,0)$ và (x_x, y_y) .

- Diện tích cần tính bằng $s - s_1 - s_2$;

2.3.2.3. Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
const int nm=1e5+1;
struct diem
{
    double x,y;
};
vector<diem> a,b,c;
int n,i,d;
double ss,s1,s2,a1,b1,c1,a2,b2,c2,xx,yy,dd,dx,dy,x,y;
fstream fi,fo;
string s;
double tdt(diem a,diem b,diem c)
{ double a1,b1,c1;
  a1 = a.y-b.y;
  b1 = b.x-a.x;
  c1 = a.x*b.y-a.y*b.x;
  return (a1*c.x+b1*c.y+c1);
}
bool ktcac(int i, int j)
{
    if(tdt(a[0],a[n],a[i])*tdt(a[0],a[n],a[j])<0
        &&tdt(a[i],a[j],a[0])*tdt(a[i],a[j],a[n])<0) return 1;
    return 0;
}
double dientich(vector<diem> a)
{ double dt=0;
```

```

        for(int i=0;i<a.size()-1;i++)
            dt=dt+(a[i].x-a[i+1].x)*(a[i].y+a[i+1].y)/2;
        return abs(dt);
    }
int main ()
{
    fi.open("AREA.INP",ios::in);
    fo.open("AREA.OUT",ios::out);
    fi>>s;
    s=s+'A';
    a.push_back({0,0});
    d=1;
    xx=0; yy=0;
    for(i=0;i<s.length()-1;i++)
        if(s[i]==s[i+1]) d++;
        else
        {
            if(s[i]=='U') yy=yy+d;
            else
            if(s[i]=='R') xx=xx+d;
            a.push_back({xx,yy});
            d=1;
        }
    n=a.size()-1;
    a1=a[n].y-a[0].y; b1=a[0].x-a[n].x; c1=a[n].x*a[0].y-
a[0].x*a[n].y;
    b.push_back(a[0]);
    c.push_back(a[0]);
    for(i=1;i<n;i++)
    {
        if(tdt(a[0],a[n],a[i])>=0) b.push_back(a[i]);
        if(tdt(a[0],a[n],a[i])<=0) c.push_back(a[i]);
        if(ktcat(i,i+1))
        {
            a2=a[i+1].y-a[i].y; b2=a[i].x-a[i+1].x;
c2=a[i+1].x*a[i].y-a[i].x*a[i+1].y;
            dd=a1*b2-a2*b1; dx=-c1*b2+c2*b1; dy=-a1*c2+a2*c1;
            x=dx/dd; y=dy/dd;
            b.push_back({x,y});
            c.push_back({x,y});
        }
    }
    b.push_back(a[n]);
    b.push_back({a[0].x,a[n].y});
    b.push_back(a[0]);
    c.push_back(a[n]);
    c.push_back({a[n].x,a[0].y});
    c.push_back(a[0]);
}

```



```

ss=xx*yy;
s1=dientich(b);
s2=dientich(c);
fo<<setprecision(3)<<fixed<<ss-s1-s2;
}

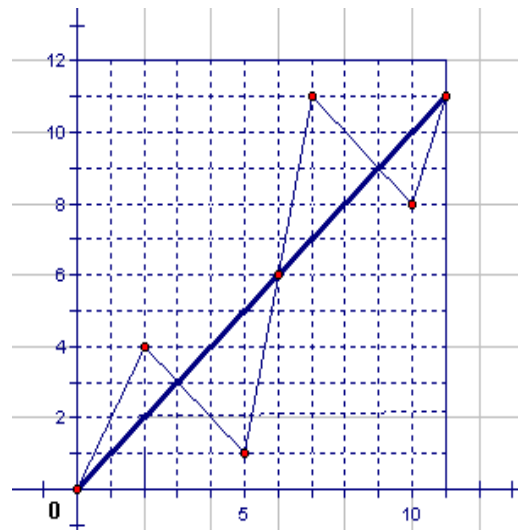
```

2.3.3. BÀI 14. ĐỔI ĐẤT (Đề thi chọn HSG QG năm 2004 -2005)

2.3.3.1. Phát biểu bài toán

Hai bộ lạc Anpha và Bêta sống rất hoà thuận với nhau. Một phần ranh giới của hai bộ lạc là một đường gấp khúc không tự cắt. Đường gấp khúc nhận được bằng cách lần lượt nối N điểm đôi một khác nhau $A_1, A_2, A_3, \dots, A_N$. Điểm A_1 được xác định bởi hoành độ x_1 và tung độ y_1 (x_i là các số nguyên thoả mãn $x_i \leq x_{i+1}$). Phần đất của bộ lạc Anpha nằm ở phía trên đường gấp khúc.

Nhân dịp năm mới, tù trưởng hai bộ lạc quyết định thay đổi đường ranh giới cũ bằng cách xây dựng một đường cao tốc là đường nối thẳng từ A_1 tới A_N và lấy đường cao tốc này làm ranh giới mới. Dĩ nhiên, sự thay đổi này sẽ chuyển một số phần đất của bộ lạc Anpha cho bộ lạc Bêta và ngược lại. Hai tù trưởng thoả thuận phần diện tích chênh lệch do việc thay đổi đường ranh giới sẽ được điều chỉnh trong tương lai bằng một cách khác.



Yêu cầu: Hãy tính diện tích phần đất SA của bộ lạc Anpha trở thành đất của bộ lạc Bêta và diện tích phần đất SB của bộ lạc Bêta trở thành đất của bộ lạc Anpha sau khi thay đổi đường ranh giới giữa hai bộ lạc.

Dữ liệu: Vào từ file văn bản LAND.INP trong đó:

- Dòng đầu chứa số N ($N \leq 100000$)
- Dòng thứ i trong N dòng tiếp theo chứa hai số nguyên x_i và y_i đặt cách nhau ít nhất một dấu cách ($-32000 \leq x_i, y_i \leq 32000$)

Kết quả: Đưa ra file văn bản LAND.OUT trong đó dòng thứ nhất chứa SA, dòng thứ hai chứa SB. Kết quả được lấy chính xác với 4 chữ số sau dấu thập phân.

Ví dụ:

LAND.INP	LAND.OUT
6	8.0000
0 0	9.0000
2 4	
5 1	
7 11	
10 8	
11 11	

2.3.3.2. Thuật toán

Để hoàn thành bài tập tập, học sinh cần kiểm soát tốt chương trình với nhiều phép xử lý hình học cơ bản như:

- + Viết phương trình đường thẳng đi qua 2 điểm phân biệt
- + Kiểm tra đoạn thẳng có cắt đường thẳng không?
- + Xây dựng 1 đa giác không tự cắt
- + Tính diện tích đa giác
- + Vị trí tương đối của một điểm so với đường thẳng

Hướng dẫn:

+ Đường cao tốc là đường nối thẳng từ A_1 tới A_N và lấy đường cao tốc này làm ranh giới mới để phân chia hai bộ lạc. Do đó, ngay phần khởi tạo ta đã xác định được phương trình đường thẳng A_1A_N ;

$Ptdt(A[1], A[N], a, b, c);$

Đặt phương trình đường thẳng $d \equiv A_1A_N$ là $f(x; y) = ax + by + c = 0$;

+ Bộ lạc Alpha nằm phía trên đường gấp khúc còn bộ lạc Beta nằm phía dưới đường gấp khúc. Do đó, SA là tổng diện tích các đa giác nằm phía dưới đường thẳng A_1A_N gọi là miền 2, SB là tổng diện tích các đa giác nằm phía trên đường thẳng A_1A_N gọi là miền 1.

+ Những điểm thuộc miền 1 có $f(x; y) \geq 0$; Những điểm thuộc miền 2 có $f(x; y) < 0$; Những điểm nằm trên đường ranh giới mới có $f(x; y) = 0$;

Thuật toán:

+ Ban đầu $SA = SB = 0$;

+ Điểm bắt đầu đa giác P là điểm A_1

+ Duyệt n điểm từ điểm 2 đến điểm thứ n, với mỗi điểm i:

- Kiểm tra xem đoạn thẳng nối hai điểm liên tiếp thứ i và i+1 có cắt đường thẳng d hay không?
- Nếu không cắt thì kết nạp điểm i vào đa giác P
- Xét điểm kế tiếp, lặp lại 2 thao tác trên cho tới khi gặp đoạn thẳng

$A_i A_{i+1}$ cắt đường thẳng d ; Giao điểm tìm được là đỉnh cuối cùng của đa giác P cần tìm.

- Tính diện tích đa giác P
- Xác định miền chứa đa giác P
- Tính SA hoặc SB

+ Xử lý đa giác P cuối cùng nhận điểm AN là đỉnh cuối.

+ Đưa ra SA, SB

2.3.3.3. Chương trình tham khảo

```
#include <bits/stdc++.h>
#define maxn 10010
using namespace std;
typedef struct point {double x; double y;};
long n, sd;
point d[maxn], p[maxn];
double a, b, c, sa, sb, s;
void nhap()
{ long i;
  cin >> n;
  for (i=1; i<=n; i++) cin >> d[i].x >> d[i].y; }
void ptdt(point p1, point p2, double &a, double &b, double &c)
{ a = p2.y - p1.y;
  b = p1.x - p2.x;
  c = -(a*p1.x + b*p1.y); }
double f(point m)
{ return (a*m.x + b*m.y + c); }
bool cut(point p1, point p2)
{ double f1, f2;
  f1 = f(p1); f2 = f(p2);
  if ((f1==0) && (f2==0)) return 0;
  return (f1*f2<=0); }
void giao2dt(point p1, point p2, point &gd)
{ double a1, b1, c1, dd, dx, dy;
  ptdt(p1, p2, a1, b1, c1);
  dd = a*b1 - a1*b;
  dx = c1*b - c*b1;
  dy = a1*c - a*c1;
  if (dd!=0) {gd.x = dx/dd; gd.y = dy/dd;}}
void dientich(long sd)
{ long i;
  s=0; p[sd+1] = p[1];
  for (i=1; i<=sd; i++) s = s + ((p[i].y + p[i+1].y)*(p[i].x -
p[i+1].x));
  s = abs(s)/2; }
int mien()
{ long i;
  double f1;
```

```

        for (i=2; i<=sd-1;i++)
        {   f1=f(p[i]);
            if (f1!=0)
                {   if (f1>0) return 1;    else return 2;}
        }
    }
void dagiac()
{   int k;
    dientich(sd);
    k = mien();
    if (k==1) sa = sa + s;
        else sb = sb+s;
}
void xuli()
{   point p1,p2,gd;
    long i;
    sa = 0; sb=0;s=0;
    ptdt(d[1],d[n],a,b,c);
    p[1] = d[1];
    sd = 1;
    for (i=2; i<=n-1;i++)
    {   sd++;
        p[sd] = d[i];
        p1 = d[i]; p2=d[i+1];
        if (cut(p1,p2)==1)
        {   giao2dt(p1,p2,gd);
            sd++;
            p[sd]=gd;
            dagiac();
            sd=1;
            p[sd]=gd;
        }
    }
    sd++;
    p[sd]=d[n];
    dagiac();
    cout << setiosflags(ios::fixed)<<setprecision(4);
    cout<<sa<<endl;
    cout<<sb;}
int main()
{   freopen("land.inp","r",stdin);
    freopen("land.out","w",stdout);
    nhap();
    xuli();
    return 0; }

```

2.4. Một số nội dung tự luyện tập

Hiện nay có đa dạng các nguồn tài liệu về lý thuyết và bài tập về thuật toán Hình học được cập nhật trên internet, chúng ta có thể tham khảo thêm nội dung lý thuyết và luyện tập thực hành trên một số website lập trình như:

Lý thuyết:

<https://www.geeksforgeeks.org/geometric-algorithms/>

<https://vnoi.info/wiki/algo/geometry/basic-geometry-1.md>

<https://vnoi.info/wiki/algo/geometry/basic-geometry-2.md>

Luyện tập:

<https://leetcode.com/tag/geometry/>

<https://codeforces.com/problemset?tags=geometry>

<https://vn.spoj.com/problems/POLY4/>

<https://vn.spoj.com/problems/CARPET/>

<https://vn.spoj.com/problems/RAOVUON/>

<https://vn.spoj.com/problems/QBCAKE/>

<https://vn.spoj.com/problems/SCIRCLE/>

<https://vn.spoj.com/problems/QBPIZZA/>

<https://vn.spoj.com/problems/FIRE/>

<https://vn.spoj.com/problems/TRIPOD/>

<https://vn.spoj.com/problems/RECT1/>

TÀI LIỆU THAM KHẢO

1. Bộ giáo dục và Đào tạo, Tài liệu tập huấn Định hướng dạy học các môn Chuyên – Môn Tin học, 2022.
2. Chuyên đề Duyên Hải, *Hình học phẳng trong Tin học*, 2021.
3. Hồ Sĩ Đàm, Đỗ Đức Đông, Lê Minh Hoàng, Trần Đỗ Hùng, Nguyễn Thanh Hùng, Tài liệu Chuyên Tin Học Quyển 3, NXB Giáo dục, 2017.
4. Lê Minh Hoàng, *Giải thuật và lập trình*, Đại học Sư phạm Hà Nội, 2006.
5. Antti Laaksonen, *Competitive Programmer's Handbook*, Helsinki, 2018.
6. Steven Halim, Felix Halim, *Competitive Programming 3*, Handbook for ACM ICPC and IOI contestants, 2013.
7. <https://www.geeksforgeeks.org/geometric-algorithms/>
8. <https://vnoi.info/wiki/algo/geometry/basic-geometry-1.md>
9. <https://vnoi.info/wiki/algo/geometry/basic-geometry-2.md>
10. <https://leetcode.com/tag/geometry/>
11. <https://codeforces.com/problemset?tags=geometry>
12. <https://vn.spoj.com/>