HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

THUẬT TOÁN ỨNG DỤNG

THUẬT TOÁN HÌNH HỌC

ONE LOVE. ONE FUTURE.

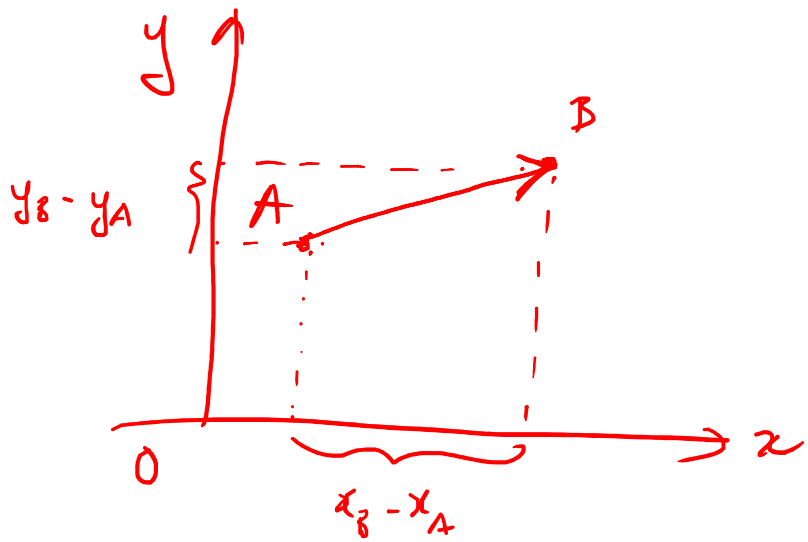
NỘI DUNG

- Công thức cơ bản
- Tìm bao lồi
- Kiểm tra 1 điểm nằm trong đa giác lồi

Công thức cơ bản

- Điểm

```
struct Point {
    double x, y;
};
```



- Đường thẳng $ax + by + c = 0$

```
struct Line {
    double a, b, c;
};
```

- Vector $\overrightarrow{AB}$ của hai điểm $A(x_A, y_A)$ và $B(x_B, y_B)$

$$\overrightarrow{AB} = (\underbrace{x_B - x_A}, \underbrace{y_B - y_A})$$

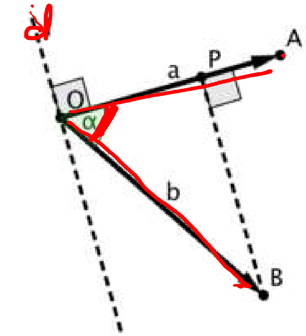
Công thức cơ bản

- 3 điểm $A(x_A, y_A)$, $B(x_B, y_B)$ và $C(x_C, y_C)$ thẳng hàng khi:
- $\overrightarrow{AB} = k \times \overrightarrow{AC}$
- $x_B - x_A = k \times (x_C - x_A)$
- $y_B - y_A = k \times (y_C - y_A)$
- Để tránh phép chia cho 0: $\boxed{(x_A - x_B) \times (y_A - y_C) = (x_A - x_C) \times (y_A - y_B)}$

Công thức cơ bản

- Tích vô hướng của $\overrightarrow{OA}(x_a, y_a)$ và $\overrightarrow{OB}(x_b, y_b)$
- $\overrightarrow{OA} \cdot \overrightarrow{OB} = x_a x_b + y_a y_b = |\overrightarrow{OA}||\overrightarrow{OB}| \cos\alpha = \sqrt{x_a^2 + y_a^2}\sqrt{x_b^2 + y_b^2} \cos\alpha$,

→ $\cos\alpha = \dfrac{x_a x_b + y_a y_b}{\sqrt{x_a^2+y_a^2}\sqrt{x_b^2+y_b^2}}$

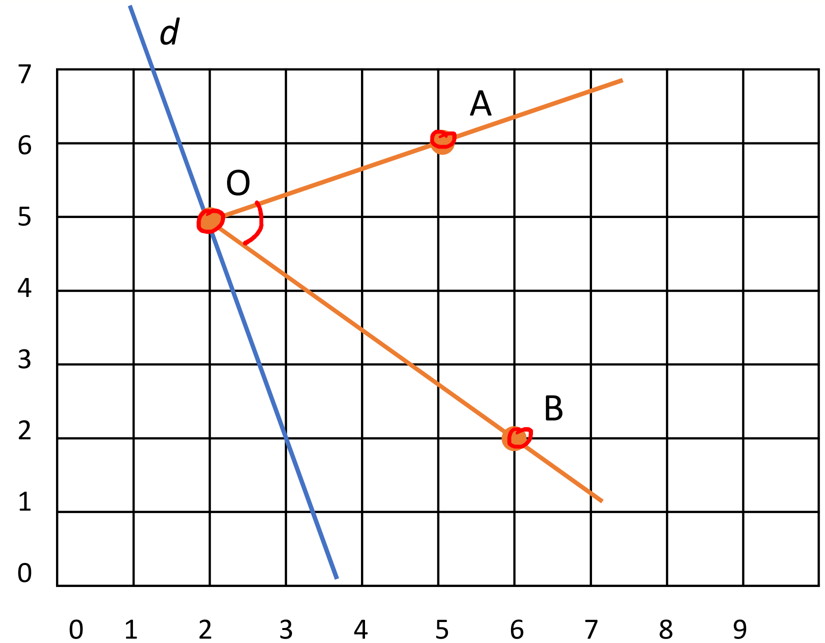


- Vẽ đường thẳng d vuông góc với $\overrightarrow{OA}$, dựa vào $\cos\alpha$ ta có:
  - Nếu $\overrightarrow{OA} \cdot \overrightarrow{OB} > 0$ thì A và B cùng phía so với đường thẳng d
  - Nếu $\overrightarrow{OA} \cdot \overrightarrow{OB} = 0$ thì B nằm trên đường thẳng d
  - Nếu $\overrightarrow{OA} \cdot \overrightarrow{OB} < 0$ thì A và B khác phía so với đường thẳng d

Công thức cơ bản

```
struct Point {
    int x, y;
    Point(int x, int y) : x(x), y(y) {}
};
double dist(Point &a, Point &b) {
    long long x = a.x - b.x; long long y = a.y - b.y;
    return sqrt(1LL * x*x + 1LL * y*y);
}
long long dot_product(Point &O, Point &A, Point &B) {
    long long xa = A.x - O.x; long long ya = A.y - O.y;
    long long xb = B.x - O.x; long long yb = B.y - O.y;
    return 1LL * xa * xb + 1LL * ya * yb;
}
int main(){
    Point O(2,5); Point A(5,6);  Point B(6,2);
    double cos = dot_product(O,A,B)*1.0/(dist(O,A)*dist(O,B));
    cout << "cos = " << cos << endl;
}
```

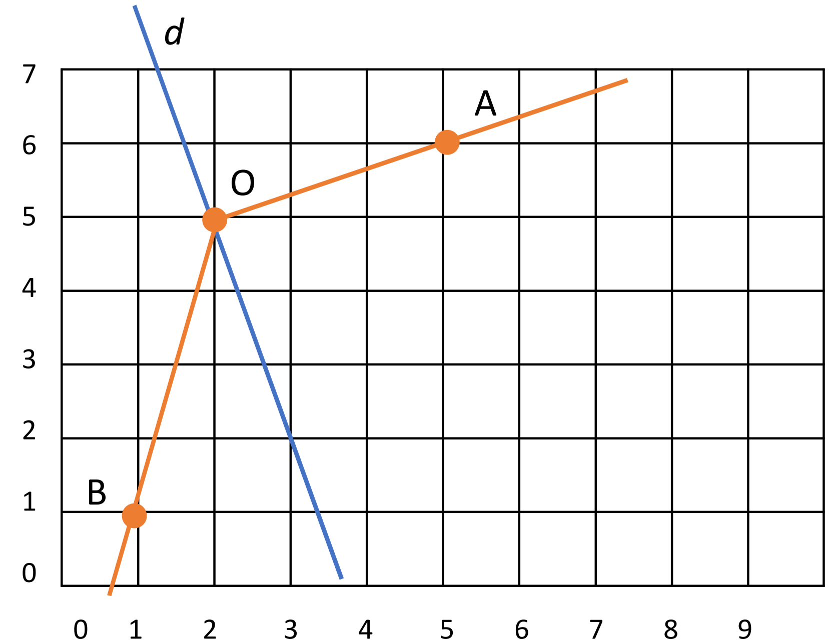
$\vec{OA} \cdot \vec{OB}$




```
cos = 0.56921
```

Công thức cơ bản

```
struct Point {
    int x, y;
    Point(int x, int y) : x(x), y(y) {}
};
double dist(Point &a, Point &b) {
    long long x = a.x - b.x; long long y = a.y - b.y;
    return sqrt(1LL * x*x + 1LL * y*y);
}
long long dot_product(Point &O, Point &A, Point &B) {
    long long xa = A.x - O.x; long long ya = A.y - O.y;
    long long xb = B.x - O.x; long long yb = B.y - O.y;
    return 1LL * xa * xb + 1LL * ya * yb;
}
int main(){
    Point O(2,5); Point A(5,6);  Point B(1,1);
    double cos = dot_product(O,A,B)*1.0/(dist(O,A)*dist(O,B));
    cout << "cos = " << cos << endl;
}
```



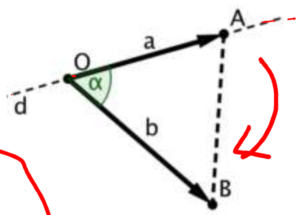

```
cos = -0.536875
```

Công thức cơ bản

- Tích có hướng của $\overrightarrow{OA}(x_a, y_a)$ và $\overrightarrow{OB}(x_b, y_b)$
- $\overrightarrow{OA} \times \overrightarrow{OB} = \boxed{x_a y_b - y_a x_b} = |\overrightarrow{OA}||\overrightarrow{OB}| \sin\alpha$

$$\overrightarrow{OA} \times \overrightarrow{OB} = x_a y_b - y_a x_b = \sqrt{x_a^2 + y_a^2}\sqrt{x_b^2 + y_b^2} \sin\alpha\,,$$

$$\boxed{\sin\alpha = \frac{x_a y_b - y_a x_b}{\sqrt{x_a^2 + y_a^2}\sqrt{x_b^2 + y_b^2}}}$$

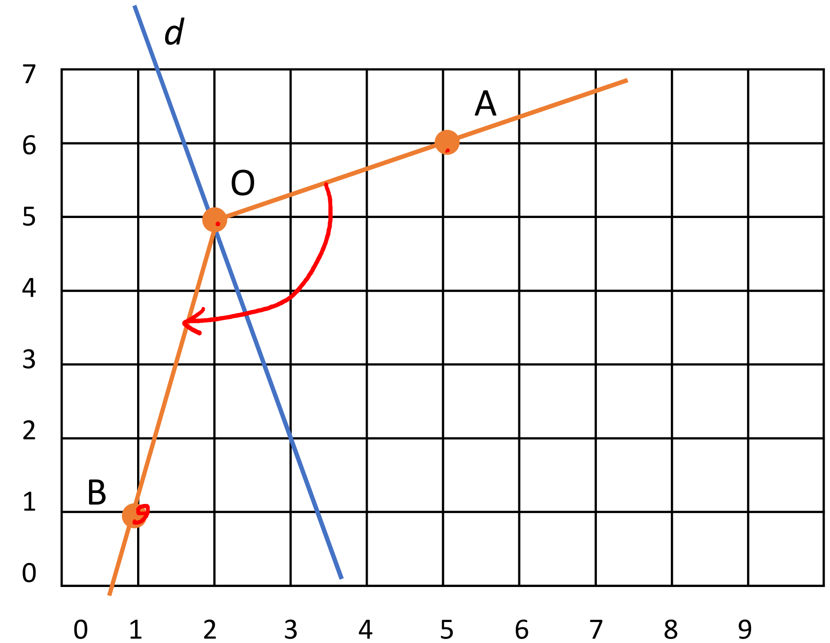


- Vẽ đường thẳng d trùng với $\overrightarrow{OA}$, dựa vào $\sin\alpha$ ta có:
  - Nếu $\overrightarrow{OA} \times \overrightarrow{OB} > 0$ thì B ở bên trái so với đường thẳng d (hướng xoay từ tia $\overrightarrow{OA}$ đến tia $\overrightarrow{OB}$ là ngược chiều kim đồng hồ)
  - Nếu $\overrightarrow{OA} \times \overrightarrow{OB} = 0$ thì B nằm trên đường thẳng d
  - Nếu $\boxed{\overrightarrow{OA} \times \overrightarrow{OB} < 0}$ thì B nằm bên phải so đường thẳng d (hướng xoay từ tia $\overrightarrow{OA}$ đến tia $\overrightarrow{OB}$ là cùng chiều kim đồng hồ)
- Trị tuyệt đối của tích có hướng của hai vector $\overrightarrow{OA}$ và $\overrightarrow{OB}$ bằng hai lần diện tích tam giác OAB.

Công thức cơ bản

```
struct Point {
    int x, y;
    Point(int x, int y) : x(x), y(y) {}
};
double dist(Point &a, Point &b) {
    long long x = a.x - b.x; long long y = a.y - b.y;
    return sqrt(1LL * x*x + 1LL * y*y);
}
long long cross_product(Point &O, Point &A, Point &B) {
    //tich vo huong 2 vector (O,A).(O,B)
    long long xa = A.x - O.x; long long ya = A.y - O.y;
    long long xb = B.x - O.x; long long yb = B.y - O.y;
    return 1LL * xa * yb - 1LL * ya * xb;
}
int main(){
    Point O(2,5); Point A(5,6); Point B(1,1);
    double sin = cross_product(O,A,B)*1.0/(dist(O,A)*dist(O,B));
    cout << "sin = " << sin << endl;
}
```

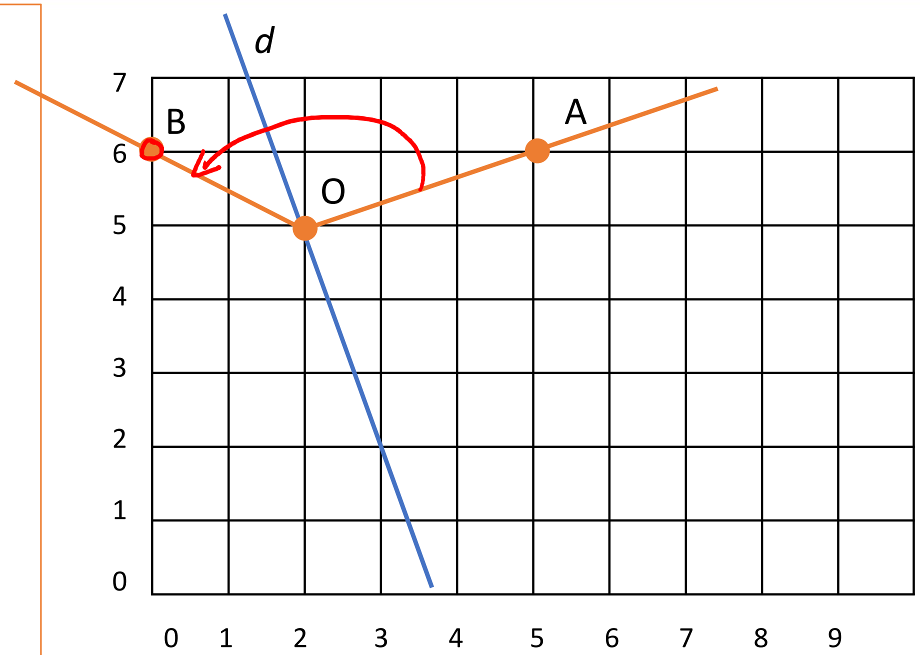
$\vec{OA} \times \vec{OB}$




```
sin = -0.843661
```

Công thức cơ bản

```
struct Point {
    int x, y;
    Point(int x, int y) : x(x), y(y) {}
};
double dist(Point &a, Point &b) {
    long long x = a.x - b.x; long long y = a.y - b.y;
    return sqrt(1LL * x*x + 1LL * y*y);
}
long long cross_product(Point &O, Point &A, Point &B) {
    //tich vo huong 2 vector (O,A).(O,B)
    long long xa = A.x - O.x; long long ya = A.y - O.y;
    long long xb = B.x - O.x; long long yb = B.y - O.y;
    return 1LL * xa * yb - 1LL * ya * xb;
}
int main(){
    Point O(2,5); Point A(5,6); Point B(0,6);
    double sin = cross_product(O,A,B)*1.0/(dist(O,A)*dist(O,B));
    cout << "sin = " << sin << endl;
}
```




```
sin =  0.707107
```

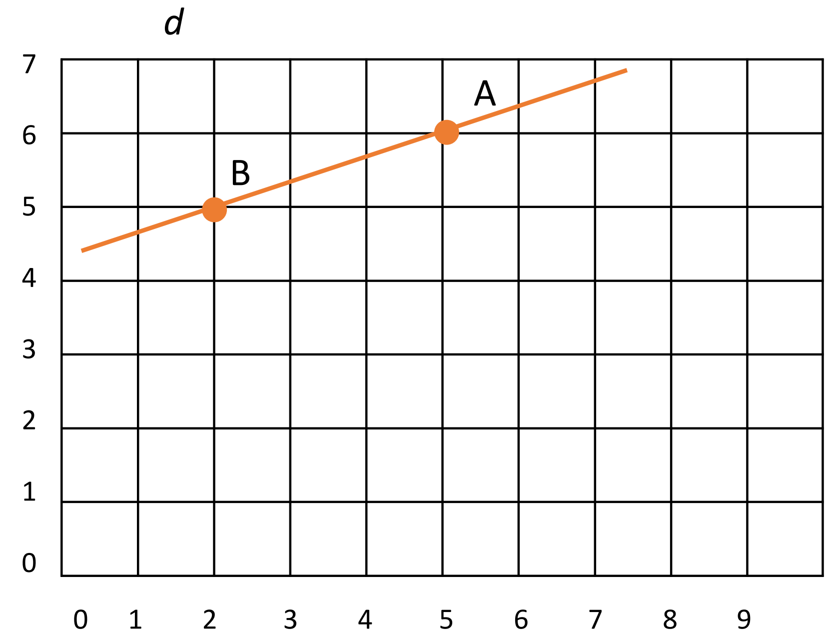
Công thức cơ bản

- Gọi đường thẳng l_{AB} đi qua hai điểm $A(x_A, y_A)$ và $B(x_B, y_B)$
  - Vector $\overrightarrow{AB} = (x_B - x_A, y_B - y_A)$
  - Vector vuông góc $\vec{v} = (y_A - y_B, x_B - x_A)$
  - Mọi điểm $P(x, y)$ nằm trên đường thẳng l_{AB} thì có $\overrightarrow{AP} \cdot \vec{v} = 0$

    $(x - x_A)(y_A - y_B) + (y - y_A)(x_B - x_A) = 0$
  - Phương trình đường thẳng l_{AB} là $ax + by + c = 0$

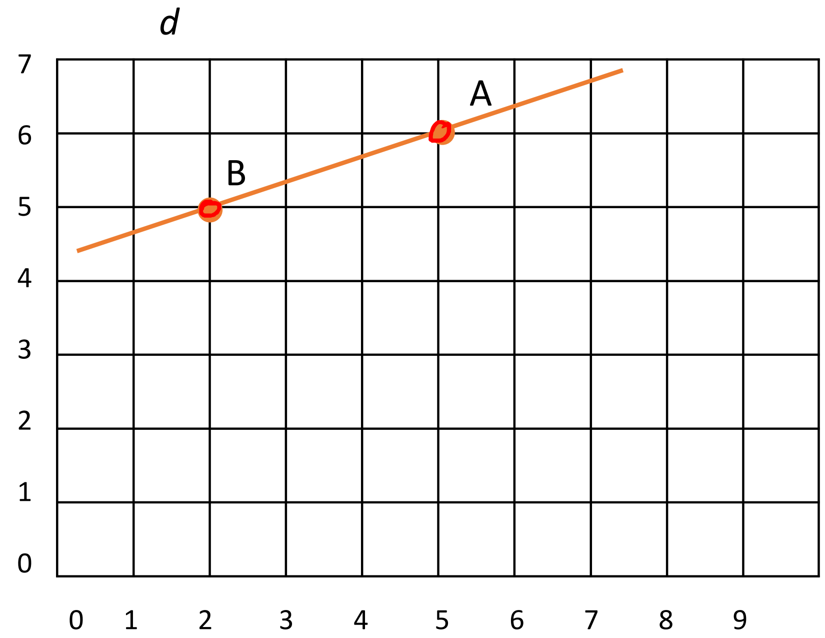
$$a = y_A - y_B$$
$$b = x_B - x_A$$
$$c = x_A y_B - y_A x_B$$

  - Ví dụ: đường thẳng đi qua A(5, 6) và B(2, 5) có phương trình: (6-5)x + (2-5)y + (5x5 – 6x2) = 0 hay x – 3y + 13 = 0



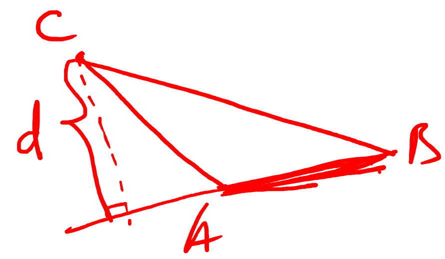
Công thức cơ bản

```
struct Point {
    int x, y;
    Point(int x, int y) : x(x), y(y) {}
};
struct Line{
    int a,b,c;
};
void makeLine(Point& A, Point& B, Line& L){
    L.a = A.y - B.y;
    L.b = B.x - A.x;
    L.c = A.x*B.y - A.y*B.x;
}
int main(){
    Point A(5,6);  Point B(2,5);  Line L;
    makeLine(A,B,L);
    cout << L.a << "x" << " + " << L.b << "y + " << L.c << " = 0“;
}
```



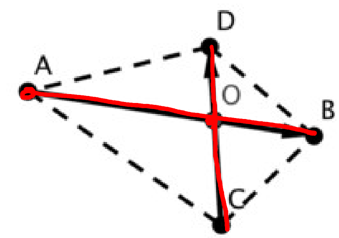
Công thức cơ bản

- Đường thẳng l_{AB} đi qua hai điểm $A(x_A, y_A)$ và $B(x_B, y_B)$
  - $(y_A - y_B)x + (x_B - x_A)y + (x_A y_B - y_A x_B) = 0$ (1)
- Khoảng cách từ điểm $C(x_C, y_C)$ đến đường thẳng l_{AB} là d
- Diện tích tam giác ABC là:
  - $S_{ABC} = \frac{|\overrightarrow{AB} \times \overrightarrow{AC}|}{2} = \frac{|\overrightarrow{AB}| \times d}{2} \Rightarrow d = \frac{|\overrightarrow{AB} \times \overrightarrow{AC}|}{|\overrightarrow{AB}|}$
- $\frac{|\overrightarrow{AB} \times \overrightarrow{AC}|}{|\overrightarrow{AB}|} = \frac{|(x_B - x_A)(y_C - y_A) - (y_B - y_A)(x_C - x_A)|}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}}$ (2)
- Từ (1) và (2), nếu đường thẳng l có phương trình là $ax + by + c = 0$ thì khoảng cách từ điểm $P(x_P, y_P)$ xuống đường thẳng l sẽ là:
- $dist(l, P) = \frac{|ax_P + by_P + c|}{\sqrt{a^2 + b^2}}$



Công thức cơ bản

- Hai đoạn thẳng AB và CD cắt nhau khi:
  - C và D không nằm cùng phía so với đường thẳng l_{AB}: $(\overrightarrow{AB} \times \overrightarrow{AC})(\overrightarrow{AB} \times \overrightarrow{AD}) \leq 0$
  - A và B không nằm cùng phía so với đường thẳng l_{CD}: $(\overrightarrow{CD} \times \overrightarrow{CA})(\overrightarrow{CD} \times \overrightarrow{CB}) \leq 0$
- Tọa độ giao điểm $O(x_O, y_O)$ của hai đường thẳng:
  - Phương trình đường thẳng l_{AB}: $a_1x + b_1y + c_1 = 0$
  - Phương trình đường thẳng l_{CD}: $a_2x + b_2y + c_2 = 0$



$$x_O = \frac{c_2b_1 - c_1b_2}{a_1b_2 - a_2b_1}$$

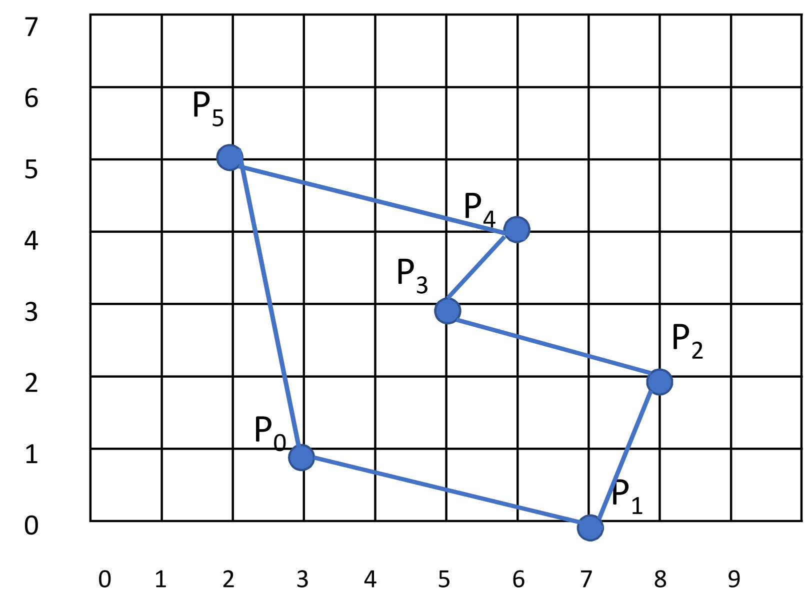
$$y_O = \frac{c_1a_2 - c_2a_1}{a_1b_2 - a_2b_1}$$

Công thức cơ bản

```
struct Point {
    int x, y;
    Point(int x, int y) : x(x), y(y) {}
};
struct Line{
    int a,b,c;
};
void makeLine(Point& A, Point& B, Line& L){
    L.a = A.y - B.y;    L.b = B.x - A.x;    L.c = A.x*B.y - A.y*B.x;
}
void intersection(Line& L1, Line& L2){
    double x = (L2.c*L1.b - L1.c*L2.b)*1.0/(L1.a*L2.b - L2.a*L1.b);
    double y = (L1.c*L2.a - L2.c*L1.a)*1.0/(L1.a*L2.b - L2.a*L1.b);
    cout << "Giao diem = (" << x << "," << y << ")" << endl;
}
int main(){
    Point A(3,1);  Point B(6,4);  Line LAB;
    Point C(2,5); Point D(7,0); Line LCD;
    makeLine(A,B,LAB); makeLine(C,D,LCD);    intersection(LAB,LCD);
}
```

Công thức cơ bản

- Một đa giác được tạo thành bởi 1 đường gấp khúc không tự cắt với các cạnh P_0P_1, P_1P_2, P_2P_3, …, $P_{n-1}P_0$
- Trong đó đỉnh P_i có tọa độ (x_i, y_i)
  - Cố định một đỉnh P_0
  - Tính tổng S:
  - $S = \sum_{i=1}^{n-2} \overrightarrow{P_0P_i} \times \overrightarrow{P_0P_{i+1}}$
  - Diện tích của đa giác là $\frac{|S|}{2}$



Tìm bao lồi (P.08.13.05)

- Cho một tập n điểm P_i, tìm đa giác lồi có diện tích nhỏ nhất chứa tất cả các điểm đã cho.
- Dữ liệu
  - Dòng 1: chứa số nguyên dương *n* (3 <= *n* <= 100000)
  - Dòng *i*+1 (*i* = 1, 2, …, *n*): chứa 2 số nguyên x_i, y_i là tọa độ của điểm P_i (-1000 <= x_i, y_i <= 1000)
- Kết quả
  - Dòng 1: ghi số nguyên dương *m* là số điểm (đỉnh của đa giác) trên bao lồi tìm được
  - Dòng *i* + 1 (*i* = 1, 2, …, *m*): ghi 2 số nguyên là tọa độ của điểm thứ *i* của bao lồi tìm được

stdin	stdout
6 4 5 5 3 5 6 2 5 8 7 3 7	4 5 3 8 7 3 7 2 5