MỤC LỤC

A. MỞ ĐẦU 2
1. Lý do chọn đề tài 2
2. Mục đích của đề tài 2
B. NỘI DUNG 2
1. Một số khái niệm cơ bản 2
1.1. Điểm trên hệ trục tọa độ Oxy 2
1.2. Đường thẳng 3
1.3. Tích chéo, tích vô hướng của 2 vector 3
1.4. Phương trình tương quan giữa điểm và đường thẳng, đoạn thẳng 5
2. Một số bài toán hình học 6
2.1. Dạng 1. Mối quan hệ giữa điểm, đoạn thẳng, đa giác. 6
2.2. Dạng 2. Các bài toán về khoảng cách 15
2.3. Dạng 3. Các bài toán về diện tích đa giác 20
3. Bài tập tổng hợp 31
3.1. Bài 1. Bán kính lớn nhất 31
3.2. Bài 2. Đặt chân chống (FLATFORM) 32
3.3. Bài 3. Góc lớn nhất 33
3.4. Bài 4. Trò chơi với Set 35
3.5. Bài 5. Maxim và xe đạp 38
4. Một số bài tập tự luyện 40
C. KẾT LUẬN 41
D. TÀI LIỆU THAM KHẢO 41

CHUYÊN ĐỀ: "PHƯƠNG PHÁP GIẢI MỘT SỐ BÀI TOÁN HÌNH HỌC TRONG TIN HỌC"

A.MỞ ĐẦU

1. Lý do chọn đề tài

Bài toán trong tin học thường rất đa dạng và có nhiều cách để giải. Việc áp dụng cách giải nào còn đòi hỏi sự sáng tạo và khả năng vận dụng của từng người. Để giải quyết những bài toán trong tin học ta có thể sử dụng các chiến lược như: Phương pháp quay lui, phương pháp nhánh cận, phương pháp tham lam, phương pháp chia để chị, phương pháp quy hoạch động …, sử dụng cấu trúc dữ liệu đặc biệt như ngăn xếp, hàng đợi, cây … Ngoài ra còn có một số bài toán sử dụng nhiều kiến thức của toán (số học, tổ hợp, hình học) ... Trong quá trình dạy tôi thấy rằng các em học sinh làm khá tốt những dạng bài trên tuy nhiên khi gặp một bài toán có liên quan đến kiến thức hình học các em thường không làm được, hoặc có làm được nhưng kết quả sai. Lý do có thể là các em khó cài đặt, hoặc không tìm được thuật giải thích hợp, hoặc có thuật giải rồi nhưng kết quả vẫn sai do sai số trong quá trình tính toán. Để có thể giúp học sinh dễ dàng tiếp thu, giải quyết những dạng bài toán đó tôi đã tìm hiểu và viết chuyên đề **"Phương pháp giải một số bài toán hình học trong tin học".**

Trong nội dung chuyên đề này tôi sẽ giới thiệu một số phương pháp giải các bài tập tin học sử dụng kiến thức hình học. Từ đó các em có thể tự tin hơn khi làm việc với các bài toán hình học, tránh được những sai sót không đáng có.

2. Mục đích của đề tài

Có nhiều bài toán hình học phẳng, quan sát bằng mắt thường thì lời giải là hiển nhiên, tuy nhiên muốn lập trình bằng máy tính để giải nó lại khá khó khăn, lại là một vấn đề hoàn toàn khác. Nhưng nếu có kiến thức toán về hình học phẳng tốt, học sinh hoàn toàn có thể làm chủ các bài toán hình học phẳng đó và các lời giải nhiều khi lại là điều khá đơn giản.

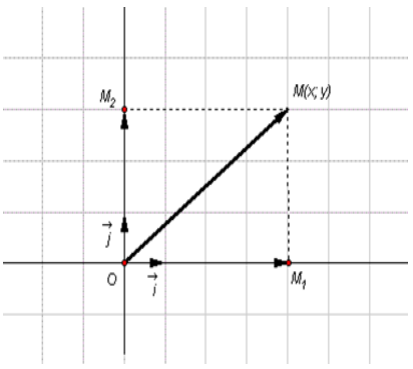
Trong Chuyên đề này, tôi muốn chia sẻ một số kiến thức toán hình học phẳng áp dụng trong tin học và một số vấn đề quan tâm.

Có một số bài toán trình bày ở đây có nhiều cách giải, song tôi chỉ trình bày thuật toán được cho là tối ưu. Toàn bộ code trong chuyên đề được tôi thể hiện bằng ngôn ngữ C++, công thức toán học được soạn thảo bằng phần mềm MathType.

B.NỘI DUNG

1. Một số khái niệm cơ bản

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:

$\overrightarrow{OM} = \overrightarrow{0M_1} + \overrightarrow{0M_2} = x.\vec{i} + y.\vec{j}$

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

```
typedef pair<int,int> Point;   struct Point
                                {
                                    int x;
                                    int y;
```

};

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

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

1.2. Đường thẳng

Đường thẳng trong mặt phẳng xác định khi biết 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: $\overrightarrow{AM} = t.\overrightarrow{AB} \Rightarrow (x_M - x_A; y_M - y_A) = t(x_B - x_A; y_B - y_A)$

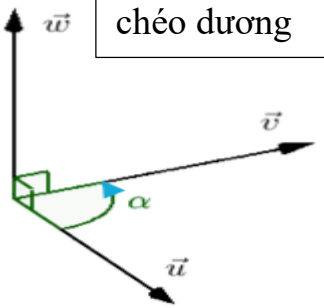
$$\Rightarrow \begin{cases} x_M - x_A = t(x_B - x_A) \\ y_M - y_A = t(y_B - y_A) \end{cases}$$

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.

1.3. Tích chéo, tích vô hướng của 2 vector

1.3.1. 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 Ơclit nhiều chiều): $w = \vec{u} \times \vec{v} = x_u y_v - x_v y_u = \begin{vmatrix} x_u & x_v \\ y_u & y_v \end{vmatrix}$



Quay phải, tích chéo dương

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 $|\vec{u}|.|\vec{v}|.\sin(\vec{u},\vec{v})$. 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:

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

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.

```
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 trai
    if (t<0)        return -1; //quay phai
    return 0; //thang hang
```

}

1.3.2. Tích vô hướng của 2 vector

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

Cài đặt tích vô hướng, tích có hướng bằng kĩ thuật chồng toán tử như sau:

Tích vô hướng (tích chấm)	Tích chéo

int operator *(Point u, Point v)
{
return (u.x*v.x+u.y*v.y);
}
	int operator ^(Point u, Point v)
	{
	return (u.x*v.y-u.y*v.x);
	}

1.3.3. Góc

- Góc tạo bởi hai vector $\vec{u}$, $\vec{v}$ có giá trị lượng giác $\sin(u,v) = \frac{\vec{u} \times \vec{v}}{|\vec{u}|.|\vec{v}|}$.

- Góc tạo bởi tia OA và trục Ox có giá trị lượng giác $\tan AOx = \frac{y_A}{x_A}$

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

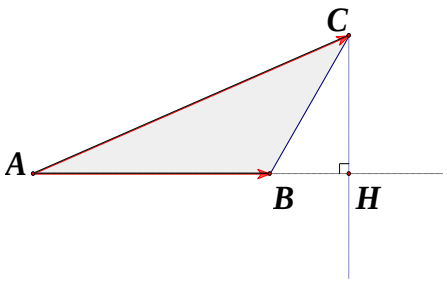
1.3.4. Tam giác

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:

$$S_{\Delta ABC} = \frac{1}{2}\left|\overrightarrow{AB} \times \overrightarrow{AC}\right| = \frac{1}{2}\left|\overrightarrow{AB}\right|.\left|\overrightarrow{AC}\right|.\sin(\overrightarrow{AB},\overrightarrow{AC})$$

```
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);
}
```

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



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

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

1.3.5. 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, A_2, … , A_N. Khi đó **diện tích đại số** của một đa giác không tự cắt được định bởi công thức sau:

$$S = \frac{x_1 - x_2 \quad y_1 + y_2 \quad + \quad x_2 - x_3 \quad y_2 + y_3 \quad + \ldots + \quad x_n - x_1 \quad y_n + y_1}{2}$$

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.

1.3.6. Đường tròn

- 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:

struct circle
{
Point A;
double r;
};
	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: $S = \pi.R^2$

1.4. Phương trình tương quan giữa điểm và đường thẳng, đoạn thẳng

1.4.1.Tương quan giữa điểm và đường thẳng:

Cho đường thẳng (d) có phương trình: ax +by + c = 0 và điểm P(x, y). Phương trình: F(x, y) = a*P.x + b*P.y + c, là phương trình tương quan của điểm P với đường thẳng (d). Khi đó:

* Nếu F(x, y)=0 thì P thuộc (d). Ngược lại nếu F(x, y)<>0 thì P không thuộc (d).
* Nếu điểm P và $Q(x_1;y_1)$ nằm cùng phía (d) thì $F(x_1;y_1)*F(x_2;y_2)>0$
* Nếu điểm P và $Q(x_1;y_1)$ nằm khác phía (d) thì $F(x_1;y_1)*F(x_2;y_2)<0$

Đây là một trong những điều kiện giúp ích cho ta rất nhiều trong giải toán tin hình học và cũng là một phương tiện thiết kế chương trình trong hình học dễ dàng hơn. Chúng ta xây dựng hàm Phuong_Trinh để xác định mối tương quan của đường thẳng với một điểm:

```
void Phuong_trinh (Lines L; Point_Chung P) : Real ;
{
    Phuong_Trinh=L.a*P.x+L.b*P.y+L.c;
}
```

1.4.2. Tương quan của điểm với đoạn thẳng.

Chúng ta biết rằng, đoạn thẳng là một phần đường thẳng. Nên mối tương quan giữa điểm P(x, y) với đoạn thẳng AB, với $A(x_1;y_1)$ và $B(x_2;y_2)$ là:

- P∈[AB] khi đồng thời thoả mãn các điều kiện:

  + F(x, y)=0, tức là: a*x+b*y+c=0, với $a := y_1 - y_2; b := x_2 - x_1; c := -(ax_1 + by_1)$

  + $(x-x_1)*(x-x_2) \le 0$ và $(y-y_1)*(y-y_2) \le 0$ (tương đương với điều kiện: $min(x_1,x_2) \le x \le max(x_1,x_2)$ và $min(y_1,y_2) \le y \le max(y_1,y_2)$

- P∉[AB] khi xảy ra 1 trong các trường hợp sau:

  1. F(x, y)=0 nhưng $(x-x_1)*(x-x_2) > 0$ hoặc $(y-y_1)*(y-y_2) > 0$
  2. F(x, y)<>0 .

Khi đó có thể xây dựng hàm kiểm tra 1 điểm P có thuộc đoạn AB như sau:

```
void thuoc_doan (Point_Chung P ; Point A, B) : Boolean ;
{
    longint a,b,c;
    Real t;
    xac_dinh ABC(A, B, a, b, c) ;
    thuoc_doan =false ;
    t = a * P.x + b*P.y + c ;
    if t<>0 then exit ;
    if ((P.x-A.x)*(P.x-B.x)>0) or ((P.y-A.y) * (P.y-B.y)>0)
    then Exit ;
    thuoc_doan =True;
}
```

2. Một số bài toán hình học

2.1. Dạng 1. Mối quan hệ giữa điểm, đoạn thẳng, đa giác.

Phương pháp: Đây là một trong số dạng bài toán hình học đơn giản nhất. Việc giải bài toán dạng này chủ yếu sử dụng các kiến thức hình học cơ bản (đã trình bày đầy đủ trong phần trên)

a) Bài toán 1: Vị trí tương đối của điểm so với đường thẳng, tia và đoạn thẳng

*** Vị trí tương đối giữa điểm và đoạn thẳng, đường thẳng và tia**

Cho điểm M(x0,y0), A(xA,yA), B(xB,yB). Yêu cầu:

a) Kiểm tra M có thuộc đường thẳng đi qua 2 điểm A, B hay không?

b) Kiểm tra M có thuộc đoạn thẳng AB hay không

c) Kiểm tra M có thuộc tia AB hay không

• Phương pháp:

Đặt $F(X,Y) = (y_A-y_B)X + (x_B-x_B)Y + (xAyB - xByA)$

- Điểm M thuộc đường thẳng AB khi F(x0,y0) = 0
- Điểm M thuộc đoạn thẳng AB khi: F(x0,y0)=0 và Min(xA,xB) ≤ x0 ≤Max(xA,xB) và Min(yA,yB) ≤ y0 ≤ Max(yA,yB)
- Điểm M thuộc tia AB khi F(x0,y0) = 0 và có nghĩa là M phải thoả mãn điều kiện: F(x0,y0) = 0 và (x0-xA)(xB-xA)≥0 và (y0-yA)(yB-yA) ≥0

***Điểm thuộc đa giác:** Cho đa giác không tự cắt $A_1A_2...A_N$ với các đỉnh $A_i(x_i,y_i)$ nguyên. Với điểm $A(x_A,y_A)$ cho trước, hãy xác định xem A có nằm trong đa giác đã cho hay không (Trong trường hợp trên cạnh đa giác xem như nằm trong đa giác)

Ý tưởng:

- Lưu toạ độ các đỉnh đa giác vào mảng A
- Kiểm tra xem điểm A có trùng với đỉnh đa giác
- Kiểm tra xem điểm A có nằm trên cạnh đa giác
- Tìm giao điểm nếu có của tia Ax (Ax//Ox và Ax hướng theo phần dương trục hoành) với các cạnh của đa giác. Trường hợp tia Ax chứa đoạn thẳng cạnh đa giác ta xem như tia Ax có 1 điểm chung với cạnh này. Cụ thể:

+ Giả sử điểm $A(x_0,y_0)$, chọn điểm $B(x_b,y_b)$ với $x_b=x_0+1, y_b=y_0$

+ Kiểm tra tia AB có cắt đoạn thẳng CD bằng cách:

Bước 1: Tìm giao điểm N của 2 đường thẳng AB và CD

a1=yb-ya; b1=xa-xb; c1=ya*xb-xa*yb;

a2=yd-yc; b2=xc-xd; c2=yc*xd-xc*yd;

D=a1*b2-a2*b1; Dx=c2*b1-c1*b2; Dy=a2*c1-a1*c2;

Xác định: Nếu D ≠0 thì toạ độ giao điểm là N(Dx/D,Dy/D)

Bước 2: Kiểm tra N có thuộc tia AM và đoạn thẳng CD hay không.

- Điểm N thuộc đoạn thẳng CD khi: $Min(x_C,x_D) \le x_N \le Max(x_C,x_D)$ và $Min(y_C,y_D) \le yN \le Max(y_C,y_D)$
- Điểm N thuộc tia AB khi $\overrightarrow{AN} = k\overrightarrow{AB}$ có nghĩa là N phải thoả mãn điều kiện: $(x_N-x_A)(x_B-x_A) \ge 0$ và $(y_N-y_A)(y_B-y_A) \ge 0$
- Kiểm tra tia AB chưa cạnh CD hay không bằng cách: (yc=yd)and(yc=yo)
- Đếm số giao điểm, nếu số giao điểm lẻ thì A thuộc đa giác

*** Đếm số điểm có toạ độ nguyên thuộc đa giác**

Cho đa giác gồm n đỉnh (x_1,y_1), (x_2,y_2), ..., (x_n,y_n), biết $(2<n<10^4)$, xi và yi(i=1,...,n) là các số nguyên trong đoạn [-106,106]. Các đỉnh được liệt kê theo thứ tự cùng chiều kim đồng hồ. Viết chương trình tìm số điểm có toạ độ nguyên nằm trong hay trên biên đa giác.

Ý tưởng:

- Tính a,b theo công thức:

$$a = \sum_{i=2}^{n} (x_i - x_{i-1})(y_i + y_{i-1}) + (x_1 - x_n)(y_1 + y_n)$$

$$b = \sum_{i=2}^{n} UCLN(|x_i - x_{i-1}|, |y_i - y_{i-1}|) + UCLN(|x_1 - x_n|, |y_1 - y_n|)$$

- Xác định số điểm có toạ độ nguyên: Sđ=round(abs(a/2)+b/2+1)

Bài 1: Tập hợp các điểm

Nguồn: https://codeforces.com/problemset/problem/277/B

Độ lồi của một tập hợp các điểm trên mặt phẳng là kích thước của tập hợp con lớn nhất của các điểm tạo thành một đa giác lồi. Nhiệm vụ của bạn là xây dựng một tập hợp n điểm với độ lồi chính xác là m. Tập hợp các điểm của bạn không được chứa ba điểm nằm trên một đường thẳng.

Đầu vào: Dòng đơn chứa hai số nguyên n và m ($3 \le m \ 100, \ m \le n \ 2 m$).

Đầu ra: Nếu không có giải pháp, hãy in " -1 ". Mặt khác, in n cặp số nguyên - tọa độ các điểm của bất kỳ tập hợp nào có độ lồi của m. Các tọa độ không vượt quá $|10^8|$.

Ví dụ

input	output
4 3	0 0 3 0 0 3 1 1
6 3	-1
6 6	10 0 -10 0 10 1 9 1 9 -1 0 -2
7 4	176166 6377 709276 539564 654734 174109 910147 434207 790497 366519 606663 21061 859328 886001

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
using namespace std;


int main()
{
	int n,m;
	scanf("%d%d",&n,&m);
	if(n>4&&m==3)
	{
		printf("-1\n");
		return 0;
	}
	for(int i=0;i<m;i++)
	{
		printf("%d %d\n",i,i*i);
	}
	for(int i=0;i<n-m;i++)
	{
```

```
        printf("%d %d\n",i,-i*i-100000);
    }
    return 0;
}
```

Bài 2: Đa giác

Nguồn: https://codeforces.com/problemset/problem/306/D

Nam thích đa giác lồi, đặc biệt nếu tất cả các góc của chúng đều bằng nhau và tất cả các mặt của chúng đều khác nhau. Vẽ cho anh ta bất kỳ đa giác như vậy với số lượng đỉnh đã cho.

Input: Đầu vào chứa một số nguyên n ($3 \leq n \leq 100$) - số lượng các đỉnh đa giác.

Output:

In n dòng, mỗi dòng chứa x_i , y_i là tọa độ các đỉnh theo thứ tự ngược chiều kim đồng hồ. Các tọa độ của các đỉnh không được vượt quá 10^6 trong giá trị tuyệt đối của chúng. Độ dài cạnh phải phù hợp trong giới hạn [1, 1000] (không nhất thiết là số nguyên). So sánh các mặt và góc của đa giác trong quá trình kiểm tra với độ chính xác 10^{-3} . Nếu không có giải pháp, hãy in " No solution" (không có dấu ngoặc kép).

Ví dụ

Input	**output**
số 8	**1.000 0.000** **7.000 0.000** **9.000 2.000** **9.000 3.000** **5.000 7.000** **3.000 7.000** **0.000 4.000** **0.000 1.000**

```
#include<cmath>
#include<cstdio>
using namespace std;
int main()
{
    int n;
    scanf("%d",&n);
    if(n<5)
        return puts("No solution"),0;
    double arg=0, t=M_PI*2/n, L=100, X= -L, Y=0;
    for(int i=1;i<n;i++)
    {
        X+=cos(arg)*L,Y+=sin(arg)*L;
        arg+=t;
        printf("%lf %lf\n",X,Y);
        L+=0.002;
    }
    printf("%lf 0\n",X-Y/tan(arg));
```

}

b) Bài toán 2. Giao của các đoạn thẳng, đường thẳng và tia

* Tìm giao điểm của 2 đoạn thẳng

- Bài toán: Cho 2 đoạn thẳng AB và CD với A(x_1,y_1), B(x_2,y_2), C(x_3,y_3), D(x_4,y_4). Tìm giao điểm (nếu có) của 2 đoạn thẳng
- Phương pháp:

B_1. Tìm giao điểm M của 2 đường thẳng AB và CD

B_2. Kiểm tra M có thuộc đồng thời cả 2 đoạn AB và CD hay không. Nếu có đó là giao điểm cần tìm, ngược lại kết luận không có.

* Giao điểm của 2 đường thẳng:

- Bài toán: Cho 2 đường thẳng có phương trình $a_1x+b_1y+c_1=0$ và $a_2x+b_2y+c_2=0$. Tìm giao điểm (nếu có) của 2 đường thẳng trên.
- Phương pháp:

B_1. Tính $D = a_1b_2 - a_2b_1$, $Dx = c_2b_1 - c_1b_2$, $Dy = a_2c_1 - a_1c_2$

B_2. Xét 3 khả năng:

+ Nếu D=Dx=Dy=0 thì kết luận 2 đường thẳng trùng nhau
+ Nếu D=0 và ((Dx≠0) hoặc (Dy≠0)) thì kết luận 2 đường thẳng song song
+ Nếu D≠0 thì kết luận 2 đường thẳng cắt nhau tại điểm có (Dx/D, Dy/D)

* Tìm giao điểm của tia và đoạn thẳng

- Bài toán: Cho tia AM chứa điểm B (khác A) và đoạn thẳng CD với A(x_1,y_1), B(x_2,y_2), C(x_3,y_3), D(x_4,y_4). Tìm giao điểm (nếu có) của tia AM với đoạn thẳng CD.
- Phương pháp:

B_1. Tìm giao điểm N của 2 đường thẳng AB và CD

B_2. Kiểm tra N có thuộc tia AM và đoạn thẳng CD hay không. Nếu có đó là giao điểm cần tìm, ngược lại kết luận không có.

Bài 1. Đường thẳng cắt nhau

Cho n đường thẳng A_iB_i ($1 \le i \le n$) phân biệt với A_i, B_i là các điểm cho trước. Hãy thông báo ra màn hình các cặp đường thẳng đôi một cắt nhau.

Dữ liệu: Cho trong file DL.INP gồm N dòng (N không biết trước). Dòng thứ i ghi 4 số thực x_{Ai} y_{Ai} x_{Bi} y_{Bi}. Các số trên cùng một dòng ghi cách nhau ít nhất một dấu cách.

DL.INP	KQ.σŒT
4 0 0 3 3 4 0 2 1	8

*** Ý tưởng:**

- Mỗi đường thẳng được đặc trưng bởi 3 thông số a,b,c được xác định:

a:=(y1-y2); b:=(x2-x1) ; c:=x1*y2-x2*y1;

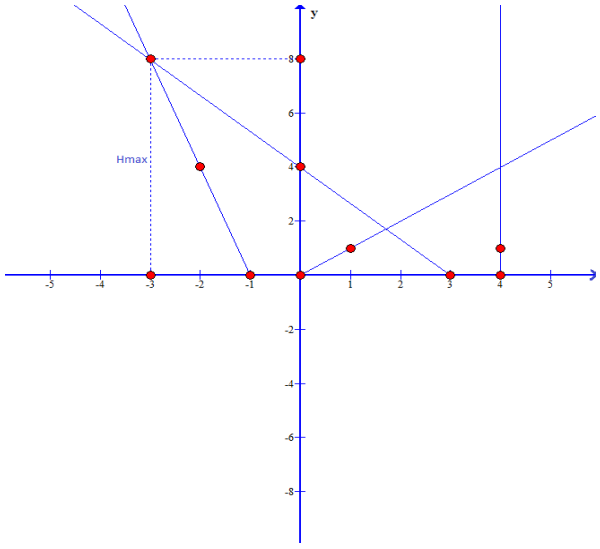
- Hai đường thẳng cắt nhau khi: D:=a1*b2-a2*b1 <> 0;

Bài 2. Giao điểm cao nhất

Trong lễ hội bắn pháo hoa năm nay. Tiết mục trình diễn ánh sáng trong lễ khai mạc của chủ nhà Đà Nẵng, có N tia laser được chiếu lên trời nhờ vào các đèn chiếu có công suất rất lớn, vì vậy các tia laser này có thể đi rất xa. Các tia laser được chiếu

lên nằm trên cùng một mặt phẳng thẳng đứng nên nếu không có 2 tia laser nào song song với nhau thì 2 tia laser hoặc là cắt nhau hoặc là không cắt nhau.

Các tia laser được biểu diễn bởi 3 số nguyên: Một số là tọa độ X ở dưới đất của ngọn đèn chiếu, 2 số còn lại là tọa độ của một điểm nào đó thuộc tia laser này. Biết rằng không có 2 tia laser nào song song với nhau và cũng không có tia Laser nào trùng với mặt đất (mặt đất được coi như là đường thẳng y = 0). Không có đèn chiếu nào đặt cùng một vị trí trên tục tọa độ.



Yêu cầu: Các nhà tổ chức buổi trình diễn muốn bạn cho biết với các tia laser sẽ được chiếu lên trời như trong kế hoạch thì 2 tia laser nào cắt nhau tại điểm cao nhất.

Dữ liệu vào: Từ tệp văn bản 'H_MAX.INP' gồm:

- Dòng 1 ghi số nguyên N là số tia laser ($2 \le n \le 100000$).
- N dòng, mỗi dòng ghi 3 số nguyên x_i, z_i, t_i với ý nghĩa là tia laser thứ i đi qua 2 điểm (x_i , 0) và (z_i , t_i) ($|x_i|, |z_i| < 10^6$, $0 < t_i \le 10^6$).

Kết quả: Ghi ra tệp văn bàn 'H_MAX.OUT' như sau:

- Trong trường hợp không có 2 tia laser nào cắt nhau thì ghi ra duy nhất một số -1.
- Nếu tồn tại 2 tia Laser cắt nhau thì ghi số H_{max} là độ cao lớn nhất giao điểm của 2 tia laser. (h_{max} được ghi với độ chính xác 3 chữ số sau dấu phẩy).

Ví dụ:

H_MAX.INP	H_MAX.OUT
2 -1 -2 4 2 3 5	-1

H_MAX.INP	H_MAX.OUT
4 -1 -2 4 0 1 1 3 0 4 4 4 1	8.000

HD Thuật toán:

- Cách 1: Duyệt chọn ra 2 tia, tìm điểm giao và tính chiều cao. Trong các chiều cao đó chọn ra cặp 2 tia có chiều cao lớn nhất. Độ phức tạp thuật toán ~ $O(N^2)$. Làm cách này thí sinh có không quá 40% số điểm của bài thi.

- Cách 2: Bạn nên chú ý tới giá trị N – số tia laser, nó rất lớn ($N \le 10^5$). Ta gọi góc tạo bởi một tia laser với trục nằm ngang (chiều dương là chiều ngược chiều kim đồng hồ) là **góc của tia đó**.

Nhận xét: Nếu tia i và tia j giao nhau (với điều kiện góc của tia j là góc có giá trị gần với góc của tia i nhất) thì độ cao giao điểm của hai tia đó sẽ lớn hơn độ cao giao điểm của tia i với các tia khác tia j. Từ đó ta có thuật toán:

- Sắp xếp các tia theo thứ tự tăng dần (hoặc giảm dần) của góc tạo bởi các tia.
- Duyệt 1 lần theo thứ tự và xét chiều cao giao điểm của hai tia cạnh nhau theo thứ tự mảng đó.
- Lưu lại giá trị lớn nhất tìm được.

Độ phức tạp thuật toán chủ yếu nằm ở bước sắp xếp các tia ~ O(NLogN).

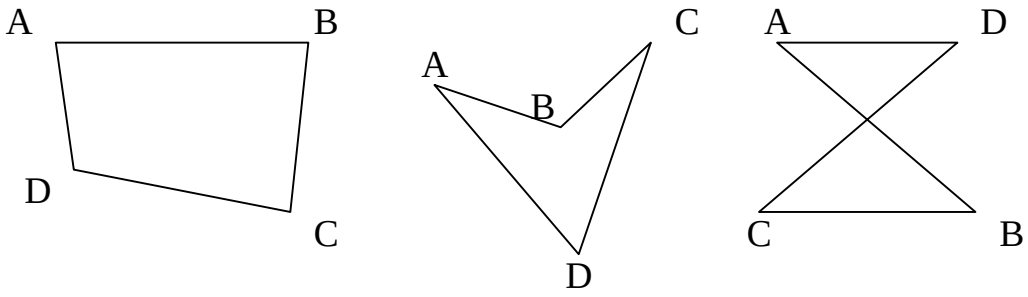
Code và test trong file đính kèm

Bài 3. Loại hình tứ giác

Trong hệ trục toạ độ Oxy, cho 4 điểm A(x1,y1), B(x2,y2), C(x3,y3), D(x4,y4). Viết chương trình in ra thông báo 4 điểm có tạo thành 1 tứ giác ABCD hay không? Nếu là tứ giác thì là tứ giác gì trong các loại hình tứ giác sau:

- Tứ giác lồi(1); - Tứ giác lõm(2); - Tứ giác tự cắt(3).

Tính diện tích tứ giác trong các trường hợp.



Tứ giác lồi Tứ giác lõm Tứ giác tự cắt

Dữ liệu vào file TUGIAC.INP: Gồm 4 dòng, mỗi dòng là toạ độ lần lượt của bốn đỉnh A, B, C, D.

Kết quả ra file TUGIAC.OUT: Nếu bốn điểm tạo thành tứ giác thì: Dòng thứ nhất ghi ra số tương ứng với mỗi loại tứ giác; Dòng thứ 2 ghi ra diện tích tứ giác. Nếu bốn điểm trên không là tứ giác ghi ra số 0.

Ví dụ:

TUGIAC.INP	TUGIAC.OUT
1 1 4 2 5 3 2 4	1 6

Thuật toán:

- Xét bốn điểm A, B, C, D là tứ giác khí không có 3 điểm bất kỳ cùng nằm trên đường thẳng.

- Nếu là tứ giác, xét lần lượt các trường hợp tứ giác lồi, rồi đến tứ giác tự cắt còn là tứ giác lõm.

+ Trường hợp tứ giác lồi chỉ cần xét AC khác phía với BD và ngược lại.

+ Trường hợp tứ giác tự cắt chỉ cần xét AB khác phía với CD và ngược lại. Trường hợp này ta có công thức tính diện tích riêng.

Hai trường hợp tứ giác lồi và tứ giác lõm áp dụng công thức tính diện tích đa giác như đã trình bày ở trên.

Code và test trong file đính kèm

Bài 4. Trò chơi đếm hình bình hành

Cho (N<=2000) điểm trên mặt phẳng, đếm số hình bình hành tạo bởi N điểm .

Input :

- số N
- N điểm nguyên tọa độ <=100,>=-100

Output: Số hình bình hành

Example

input

```
4
0 1
1 0
1 1
2 0
```

output

```
1
```

HD Thuật toán: bài này là 1 bài khá đơn giản , với mỗi cặp điểm trong N điểm, gọi C là trung điểm của cặp điểm đó, CNT là số lượng cặp điểm có C là trung điểm. như vậy kết quả của bài toán là $\sum(CNT_i*(CNT_i -1)/2)$

CODE:

```
#include<cstdio>
#include<map>
int i,j,n,ans,x[2005],y[2005];
std::map<std::pair<int,int>,int>a;
int main()
{
	scanf("%d",&n);
	for(i=1;i<=n;i++)
	{
		scanf("%d%d",&x[i],&y[i]);
		for(j=1;j<i;j++)
			ans+=a[std::make_pair(x[i]+x[j],y[i]+y[j])]++;
	}
	printf("%d",ans);
}
```

Bài 5. Hai hình vuông

Cho hai hình vuông, một hình vuông có các cạnh song song với trục tọa độ, và hình kia có các cạch tạo với trục tọa độ góc 45^0, kiểm tra xem hai hình vuông có phần chung không. Hai hình được gọi là có phần chung khi tồn tại ít nhất một điểm thuộc cả 2 hình (có thể có 1 điểm chung hoặc vô số điểm chung).

Dữ liệu vào: gồm 2 dòng, mỗi dòng có 4 cặp số nguyên là tọa độ của 2 hình vuông tương ứng. Dòng thứ nhất là đỉnh của hình vuông có cạnh song song với trục

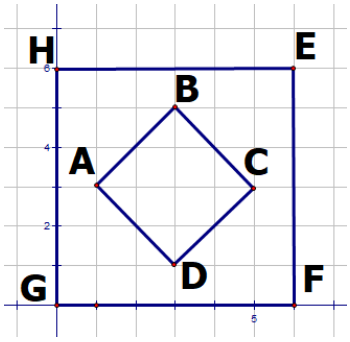
tọa độ, dòng thứ hai chứa các đỉnh của tam giác lệch góc 45^0 với hệ trục tọa độ. Lưu ý các đỉnh được cho theo lần lượt theo thứ tự ngược chiều hoặc thuận chiều kim đồng hồ.

Dữ liệu ra: Nếu 2 hình vuông có phần chung in ra "YES" ngược lại in "NO"

Ví dụ:

2SQUARES.INP	2SQUARES.OUT
0 0 6 0 6 6 0 6 1 3 3 5 5 3 3 1	YES
0 0 6 0 6 6 0 6 7 3 9 5 11 3 9 1	NO
6 0 6 6 0 6 0 0 7 4 4 7 7 10 10 7	YES
0 0 6 0 6 6 0 6 8 4 4 8 8 12 12 8	YES

Hình cho ví dụ 1:



Thuật toán 1: Trong bài trên, có thể quy về bài toán xác định điểm nằm trong đa giác bằng cách kiểm tra xem có điểm nào trong các điểm A,B,C,D có nằm trong hình vuông GHEF và có điểm nào trong các điểm G, H, E, F nằm trong hình vuông ABCD hay không. Nếu có tồn tại điểm như vậy thì 2 hình vuông có phần chung, ngược lại thì không.

Thuật toán 2: Kiểm tra các đoạn thẳng giao nhau, nếu có cạnh nào đó của hình vuông thứ nhất giao với cạnh nào đó của hình vuông thứ 2 thì chúng có điểm chung.

Thuật toán 3: Dựa theo khoảng cách tâm của 2 hình vuông chiếu trên trục Ox và Oy, nếu có 1 khoảng cách lớn hơn tổng của nửa AC và HE thì không giao nhau

```
#include<bits/stdc++.h>
using namespace std;
int px,py,sx,sy,len,lo,a,b,ans;
int main() {
 freopen("2SQUARES.INP","r",stdin);
 freopen("2SQUARES.OUT","w",stdout);
 for (int i=1; i<=4; i++) {
  cin>>a>>b;
  px+=a;
  py+=b;
 }
 px/=2,py/=2;
 lo=abs(a-px);
 for (int i=0; i<4; i++) {
  cin>>a>>b;
  sx+=a;
  sy+=b;
 }
 sx/=2,sy/=2;
 len=abs(sx-a)+abs(sy-b);
 ans=max(0,abs(px-sx)-lo)+max(0,abs(py-sy)-lo);
 if (ans<=len)
```

```
 cout<<"YES";
else
 cout<<"NO";
return 0;
}
```

2.2.Dạng 2. Các bài toán về khoảng cách

*Bài toán: 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).

```
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*).

Bài 1: Đặt trạm BTS1

BTS là một cơ sở hạ tầng viễn thông được sử dụng nhằm tạo thông tin liên lạc không dây giữa các thiết bị thuê bao viễn thông tại các khu dân cư. Các thiết bị thuê bao có thể là điện thoại di động, thiết bị internet không dây,… Nhằm nâng cấp dịch vụ mạng điện thoại di động tại đất nước XYZ, người ta mới nghĩ ra phương án hợp nhất tất cả các nhà cung cấp dịch vụ di động lại thành một thể thống nhất lại các vị trí đặt trạm BTS. Là kĩ sư phụ trách kĩ thuật cho quá trình này, Bờm tự hỏi: với mỗi vị trí đặt trạm BTS sau khi hợp nhất có bao nhiêu vị trí khu dân cư sẽ nằm ngoài vùng phủ sóng của trạm mới khi đã biết tọa độ của các vị trí này. Coi vùng phủ sóng di động của trạm BTS dạng hình tròn với tâm là tọa độ đặt trạm BTS.

Dữ liệu vào:

- Dòng đầu tiên ghi tọa độ, bán kính phủ sóng trạm BTS mới.
- Dòng tiếp theo ghi số lượng thuê bao di động *n (n<=10^6)*
- Các dòng tiếp theo là tọa độ (x_i, y_i) của các điểm dân cư, $|x_i| \le 100; |y_i| \le 100$

Dữ liệu ra:

- Số lượng thuê bao di động nằm ngoài vùng phủ sóng của trạm BTS mới.

Ví dụ:

BTS1.INP	BTS1.OUT	Giải thích
0 0 5 5 2 2 -3 3 4 -3 -6 -6 1 1	1	Vị trí tọa độ (-6;-6) nằm ngoài vùng phủ sóng.

Thuật toán: Khá đơn giản, chỉ cần kiểm đếm các điểm có khoảng cách đến trạm BTS lớn hơn vùng phủ sóng.

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

```
const int maxn=1e6;
struct Point {
   double x, y;
};
Point a,b;
int dem,n,r;
int distSq(Point A, Point B) {    // binh phuong khoang cach
   return (A.x-B.x)*(A.x-B.x)+(A.y-B.y)*(A.y-B.y);
}
int main() {
   freopen("BTS1.INP","r",stdin);
   freopen("BTS1.OUT","w",stdout);
   cin>>a.x>>a.y>>r>>n;
   dem=0;
   for (int i=1; i<=n; i++) {
      cin>>b.x>>b.y;
      if (distSq(a,b)>r*r)
         dem++;
   }
   cout<<dem;
}
```

Bài 2: Tìm vị trí đặt loa thông báo

Trong kỳ thi học sinh giỏi môn Tin có N học sinh tại sân trường XYZ. Có thể coi sân trường là một hệ trục tọa độ, mỗi học sinh có một tọa độ x_i, y_i. Ban tổ chức cần đứng tại một vị trí thật thuận tiện để có thể loa thông báo cho N em học sinh. Cho rằng loa có thể thông báo cho các em học sinh trong phạm vi R nghe thấy. Hãy xác định R nhỏ nhất để có thể thông báo cho N học sinh để có thể đi thi đúng giờ.

Input: LOA.INP

+ Dòng 1 ghi n ($n \leq 100$)
+ n dòng tiếp theo, dòng thứ i ghi hai số nguyên x_i, y_i thể hiện tọa độ của một điểm

Output: LOA.OUT Một số thực với 3 chữ số thập phân là kết quả cần tìm.

LOA.INP	LOA.OUT
3 0 0 0 1 1 1	0.707

Hướng dẫn thuật toán:

Duyệt 3 điểm khác nhau và tìm bán kính nhỏ nhất chứa 3 điểm đó. Khi đó bán kính R cần tìm chính là bán kính R lớn nhất.

Chương trình: LOA.CPP

```
#include <bits/stdc++.h>
using namespace std;
struct Point
{    long x, y; };
double dis(Point A, Point B);
```

```
double bankinh(Point A, Point B, Point C);
int main()
{
   Point A[101];
   double res,r;
   long n,i,j,k;
   freopen("loa.inp","r", stdin);
   freopen("loa.out", "w", stdout);
   res=0.0;
   cin >> n;
   for (i=1;i<=n;i++)
      cin >> A[i].x >> A[i].y;
   for(i=1;i<=n;i++)
   for(j=i+1;j<=n;j++)
   for(k=j+1;k<=n;k++)
   {
      r= bankinh(A[i],A[j],A[k]);
      if(r>res) res = r;
   }
   cout << fixed << setprecision(3) << res;
   return 0;
}
double dis(Point A, Point B)
{
   return sqrt( (B.x - A.x)*(B.x - A.x) + (B.y- A.y)*(B.y-A.y));
}
double bankinh(Point A, Point B, Point C)
{
   double a,b,c,p,S;
   a=dis(A,B);
   b=dis(B,C);
   c=dis(A,C);
   if(a*a>b*b+c*c) return (a/2);
   if(b*b>c*c+a*a) return (b/2);
   if(c*c>a*a+b*b) return (c/2);
   p=(a+b+c)/2;
   S=sqrt(p*(p-a)*(p-b)*(p-c));
   return (a*b*c/4/S);
}
```

Bài 3. Trung tâm du lịch

Trên một tỉnh XYZ có N địa điểm du lịch. Có thể coi là N điểm trên mặt phẳng tọa độ, mỗi điểm du lịch có tọa độ x_i, y_i. Chủ tịch hội đồng quản trị công ty ABC cần đặt bến tại một trong N điểm du lịch sao cho tổng khoảng cách từ điểm đó đến các điểm du lịch còn lại là nhỏ nhất. Coi khoảng cách giữa hai điểm du lịch chính bằng khoảng cách giữa hai điểm trên mặt phẳng tọa độ. Nếu có nhiều điểm như vậy bạn hãy chọn điểm có số hiệu nhỏ nhất.

Input: DULICH.INP

+ Dòng 1 ghi n (n ≤100)

+ n dòng tiếp theo, dòng thứ i ghi hai số nguyên x_i, y_i thể hiện tọa độ của một điểm

Output: DULICH.OUT

Dòng duy nhất ghi hai số, số đầu là số hiệu của điểm tìm được và số thứ hai là số thực thể hiện tổng khoảng cách đến các điểm còn lại (3 chữ số thập phân)

DULICH.INP	DULICH.OUT
1 6 29	1 0.000

Hướng dẫn thuật toán:

Tính khoảng cách giữa hai điểm du lịch là khoảng cách giữa hai điểm trong mặt phẳng tọa độ Đề-các. Ta thử đặt bến tại các điểm và tính khoảng cách đến các điểm còn lại, lưu lại điểm đặt và tổng khoảng cách ngắn nhất từ điểm đó đến các điểm còn lại.

Chương trình: DULICH.CPP

```
#include <bits/stdc++.h>
using namespace std;
struct Point
{
   long x,y;
};
Point A[1001];
double res;
long i,j,n,vt;
double dis(long i, long j);
int main()
{
   freopen("DULICH.inp","r",stdin);
   freopen("DULICH.out","w", stdout);
   cin >> n ;
   for(i=1;i<=n;i++)
      cin >> A[i].x >> A[i].y;
   res=10000000 ;vt=0;
   for(i=1;i<=n;i++) // xet cac diem i
   {
      double s=0;
      for(j=1;j<=n;j++) // duyet tat ca cac diem
         if (i != j) s+= dis(i,j);
      if (s < res)
      {
         res= s;
         vt = i;
      }
   }
```

```
    cout << vt << " "; cout << fixed<< setprecision(3) << res;
    return 0;
}
double dis(long i, long j)
{
    return sqrt( (A[i].x - A[j].x)*(A[i].x - A[j].x) + (A[i].y - A[j].y) * (A[i].y - A[j].y) );
}
```

Bài 4. Xây dựng đường đi ở các khu chung cư

Trong thành phố X có N khu chung cư, khu chung cư thứ i có tọa độ x_i, y_i. Lãnh đạo thành phố X rất quan tâm đến N khu chung cư này và có ý định xây một đường cao tốc song song với trục hoành. Khi đó chắc chắn từ N khu chung cư sẽ phải làm thêm đường từ chung cư của mình đến đường cao tốc song song với trục tung. Mỗi khu chung cư làm một đường (không chung nhau). Hỏi tổng độ dài các đường cần làm nhỏ nhất là bao nhiêu (Hai đường có thể trùng nhau trên mặt phẳng tọa độ - khi đó một cái nằm trên và cái kia nằm dưới)

Input: CHUNGCU.INP

+ Dòng 1 ghi n ($n \leq 100$)
+ n dòng tiếp theo, dòng thứ i ghi hai số nguyên x_i, y_i thể hiện tọa độ của một điểm

Output: CHUNGCU.OUT

Ghi một số nguyên duy nhất là đáp số tìm được.

CHUNGCU.INP	CHUNGCU.OUT
5 -2282 2142 -2886 -228 436 -2782 991 -2468 1378 -4259	8955

Hướng dẫn thuật toán:

+ Sắp xếp các điểm theo tung độ
+ Đường cao tốc cần đặt chính là tung độ giữa các điểm vừa sắp xếp
+ Tính tổng quãng đường từ khu chung cư đến đường cao tốc.

Chương trình: CHUNGCU.CPP

```
#include <bits/stdc++.h>
using namespace std;
struct Point
{
    long x, y;
};
long n,i,j;
Point A[1001];
int main()
{
    freopen("CHUNGCU.inp" , "r", stdin);
```

```
    freopen("CHUNGCU.out" , "w", stdout);
    cin >> n;
    for ( i=1 ; i <=n ; i++) cin >> A[i].x >> A[i].y;
    for ( i=1; i<= n-1; i++)
       for( j=i+1; j <=n ; j++)
       if ( A[i].y > A[j].y) swap(A[i] , A[j]);
    long yo;
    if (n % 2 ==0) yo= A[n/2].y; else yo = A[(n+1)/2].y;
    long res=0;
    for ( i=1; i <=n ;i++) res += abs( A[i].y -yo);
    cout << res;
    return 0;
}
```

2.3. Dạng 3. Các bài toán về diện tích đa giác

a)Bài toán 1: Tính diện tích đa giác

Phương pháp: Giả sử cho đa giác có n đỉnh và toạ độ các đỉnh lưu vào mảng a. Để tính diện tích đa giác ta làm như sau:

Bước 1. Gắn thêm đỉnh phụ: a[n+1].x:=a[1].x; a[n+1].y:=a[1].y;

Bước 2. Diện tích đa giác tính theo công thức:

$$S = \left| \sum_{i=1}^{n} (a[i+1].x - a[i].x)(a[i+1].y + a[i].y)/2 \right|$$

Lưu ý: Có thể áp dụng công thức khác để tính diện tích trong các trường hợp đặc biệt.

- Nếu đa giác là tam giác (n=3) thì diện tích tính theo công thức:

$$S = \sqrt{(a+b+c)(a+b-c)(a+c-b)(b+c-a)}/4$$

- Nếu đa giác là hình chữ nhật (n=4) có các cạnh là a,b thì diện tích là: S=ab
- Nếu đa giác là hình vuông (n=4) có cạnh là a thì diện tích là: S=a2
- Nếu đa giác là hình tròn có bán kính R thì diện tích là ΠR^2

Bài 1. Xác định diện tích đa giác

Cho đa giác lồi N đỉnh $A_1A_2A_3...A_N$-1AN với các đỉnh $A_i(x_i,y_i)$ có toạ độ nguyên. Hãy tính diện tích đa giác trên.

DL.INP	Kết quả
5 - 8 -8 0 0 1 0 -2 4 -5 0	44.00

Dữ liệu: Cho trong file DL.INP gồm 2 dòng

- Dòng 1: Chứa số nguyên dương N
- Dòng 2: Chứa 2xN số nguyên dương x_1 y_1 x_2 y_2...x_N y_N là toạ độ các đỉnh của đa giác. Mỗi số ghi cách nhau một dấu cách.

Kết quả: Xuất ra màn hình diện tích đa giác.

* **Ý tưởng:**

- Lưu toạ độ các đỉnh đa giác vào mảng toado
- Sử dụng công thức tính diện tích đa giác:

```
#include <conio.h>
#include <math.h>
#include <iostream>
```

```
using namespace std;

class point{
private:
   float x,y;
public:
   void setPoint(float a,float b){
      x = a;y = b;
   };
   friend float calculateAreaTriangle(point A,point B,point C){
      return (0.5*abs((C.x-A.x)*(B.y-A.y)-(B.x-A.x)*(C.y-A.y)));
   };
};

class polygon{
private:
   point *p;
   int n;
   float Area;
public:
   polygon(int m){
      n = m;
      p = new point[m];
      Area = 0;
   };
   void setPolygon(){
      float a,b;
      for(int i=0;i<n;i++){
         cout<<"x("<<i+1<<") = ";          cin>>a;
         cout<<"y("<<i+1<<") = ";          cin>>b;
         p[i].setPoint(a,b);
      }
      for(int i=0;i<n-1;i++){
         Area = Area + calculateAreaTriangle(p[0],p[i],p[i+1]);
      }
   };
   void testInPolygon(point P){
      float sum = 0;
      for(int i=0;i<n-1;i++){
         sum = sum + calculateAreaTriangle(p[i],p[i+1],P);
      }
      sum = sum + calculateAreaTriangle(p[n-1],p[0],P);
      if(abs(Area-sum)<=0.01)
         cout<<"diem nam trong hinh";
      else          cout<<"diem nam ngoai hinh";
   };
```

```
    ~polygon(){
       delete []p;
    };
};

void main(){
    int m;
    float a,b;
    cout<<"nhap so dinh cua da giac: ";
    cin>>m;
    polygon A(m);
    A.setPolygon();
    point P;
    cout<<"nhap diem can xet: ";
    cout<<"x = ";
    cin>>a;
    cout<<"y = ";
    cin>>b;
    P.setPoint(a,b);
    A.testInPolygon(P);
}
```

Bài 2. Dãy hình chữ nhật

Trong mặt phẳng toạ độ trực chuẩn, cho N hình chữ nhật có các cạnh song song với trục toạ độ. Mỗi HCN được xác định bởi toạ độ đỉnh dưới bên trái và đỉnh trên bên phải của nó. Hãy đưa ra dãy các hình chữ nhật theo thứ tự tăng dần diện tích .

Dữ liệu: Cho trong file HCN.inp gồm N+1 dòng.

- Dòng 1. Chứa số N

-Dòng i+1 ($1 \le i \le N$): Ghi 4 số nguyên x_1, y_1, x_2 ,y_2 lần lượt là toạ độ đỉnh dưới bên trái và đỉnh trên bên phải của HCN i. (Các số ghi trên một dòng cách nhau ít nhất một dấu cách)

HCN.inp	HCN.out
5 -3 -4 0 -2 -6 -4 0 0 -6 -2 -3 0 0 0 7 7 -6 0 0 7	-3 -4 0 -2 -6 -2 -3 0 -6 -4 0 0 -6 0 0 7 0 0 7 7

Kết quả: Ghi vào tệp HCN.out dãy các hình chữ nhật sau khi sắp xếp.

Ý tưởng:

- Lưu toạ độ các đỉnh đa giác vào mảng a
- Tính diện tích hình chữ nhật theo công thức:

$$s = |(x_2 - x_1)(y_2 - y_1)|$$

- Sắp xếp mảng a tăng dần theo diện tích

Code và test trong file đính kèm

b)Bài toán 2: Xác định diện tích phủ bởi các hình chữ nhật

Phương pháp: Giả sử có n hình chữ nhật. Để tính diện tích phủ bởi n hình chữ nhật ta làm như sau:

Bước 1. Sử dụng a,b lần lượt là các mảng lưu hoành độ và tung độ các đỉnh hình chữ nhật (mỗi hình chữ nhật chỉ cần lưu toạ độ 2 đỉnh đối diện qua tâm hình chữ nhật).

Bước 2. Sắp xếp mảng a,b theo thứ tự tăng dần

Bước 3. Lần lượt kiểm tra các hình chữ nhật có toạ độ đỉnh trên bên phải (x_{i+1},y_{i+1}) và toạ độ đỉnh dưới bên phải là (x_i,y_i) với $1 \le i \le n-1$. Nếu hình chữ nhật này thuộc một trong các hình chữ nhật ban đầu thì cộng thêm vào phần diện tích đang cần tìm diện tích của hình chữ nhật con này.

Bài 3: Diện tích phủ bởi các hình chữ nhật

Trên mặt phẳng tọa độ, một hình chữ nhật với các cạnh song song với các trục toạ độ được xác định bởi hai điểm đối tâm: đỉnh góc trên bên trái và đỉnh góc dưới bên phải. Cho N hình chữ nhật song song với các trục toạ độ. Phủ S của các hình chữ nhật có diện tích nhỏ nhất chứa N hình chữ nhật đã cho.

Dữ liệu vào: Đọc từ tệp PHUCN.INP có cấu trúc:

- Dòng đầu tiên chứa N ($N \le 30$);
- Trong N dòng tiếp theo, mỗi dòng ghi 4 số là toạ độ của hai đỉnh đối tâm của một hình chữ nhật, các số này là các số nguyên có trị tuyệt đối không quá 100.

Kết quả: Ghi ra tệp văn bản PHUCN.OUT

- Dòng 1 ghi toạ độ hai đỉnh đối tâm của phủ S các hình chữ nhật
- Dòng 2 ghi diện tích của phần hình S không nằm trong hình chữ nhật nào trong N hình đã cho

PhuCN.INP	PhuCN.OUT
3 10 30 40 10 20 20 60 0 0 40 30 0	0 40 60 0 500

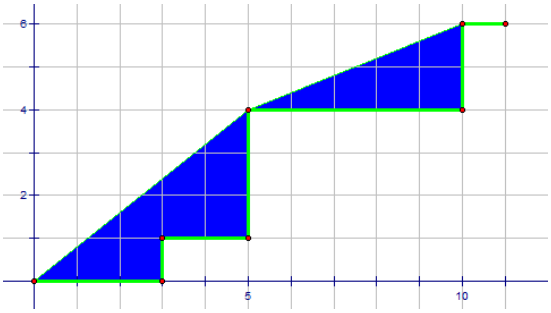
- Ý tưởng:
- Xác định hình chữ nhật H nhỏ nhất bao tất cả các hình chữ nhật ban đầu:

Gọi minx,maxx lần lượt là hoành độ nhỏ nhất và lớn nhất trong các hoành độ các đỉnh hình chữ nhật đã cho; miny, maxy lần lượt là tung độ nhỏ nhất và lớn nhất trong các tung độ các đỉnh hình chữ nhật đã cho. Khi đó hình H có toạ độ đỉnh dưới trái là (minx,miny) và đỉnh trên phải là (max,maxy). Đó là phủ S cần tìm.

- Tính diện tích hình H là (maxx-minx)(maxy-miny)
- Tính diện tích s phủ bởi các hình chữ nhật (đã nêu rõ ở phương pháp chung)
- Phần diện tích cần tìm là: s1:=abs((maxx-minx)*(maxy-miny))-s
- Chương trình:

Code và test trong file đính kèm

Bài 4. Đa giác



Trên mặt phẳng lưới toạ độ Đê các, xuất phát từ điểm (0, 0) người ta vẽ một đường gấp khúc có các cạnh song song với trục toạ 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 toạ độ. Hình bên tương ứng với chương trình vẽ là RRRURRUUURRRRRUUR.

Yêu cầu: Từ điểm (0,0) nhìn theo đường chéo đến một điểm A mà các điểm còn lại đều nằm phía dưới đường chéo ta được một hình tô đậm, lặp lại như vậy nhưng điểm nhìn là từ điểm A .. Tính tổng diện tích phần tô

Input: Vào từ file văn bản POLYGON.INP gồm nhiều dòng, mỗi 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. Ch-¬ng tr×nh kÕt thóc b»ng lÖnh S.

Output: Đưa ra file văn bản POLYGON.OUT các diện tích tìm được, mỗi kết quả trên một dòng, là một số thực với 3 chữ số sau dấu chấm thập phân.

Ví dụ:

POLYGON.INP
RRRURRUUURRRRRUURS RUURS

POLYGON.OUT
13.000 1.000

Hướng dẫn thuật toán:

+Xác định tọa độ các điểm khi thay đổi U, R và dừng lại khi gặp S

+ Bắt đầu từ (0,0) tìm A điểm mà từ đó mọi điểm đều nằm phía dưới OA. Ta được phần tô thứ nhất và có thể tính diện tích và lặp lại như vậy bắt đầu từ A …

Chương trình : POLYGON.CPP

```
#include <bits/stdc++.h>
#define Point TVector
using namespace std;
struct Point
{
   float x, y;
};
Point A[2000];
int n;
float res;
int dau,cuoi;
void xuli();
void docdl();
float dtdagiac(Point C[2000],int d, int c)
{
   Point B[2000];
   for ( int i = 1; i <= n; i++) B[i] = C[i];
   B[d-1] = B[c]; B[c+1] = B[d];
   float s = 0;
   for ( int i = d; i <= c; i++)
      s += B[i].y *(B[i-1].x - B[i+1].x );
   s = s/2;
   return ( fabs(s));
}
Point Vector(float x, float y)
{
   Point tmp;
```

```
    tmp.x = x; tmp.y = y;
    return tmp;
}
float tichcheo( TVector u, TVector v)
{
    return ( u.x*v.y - u.y*v.x );
}
int main()
{
    docdl();
    return 0;
}
void xuli()
{
    dau = 1;
    while ( A[dau].x == A[dau+1].x) dau++;
    cuoi = 3;
    while ( dau < n && cuoi <= n)
    {
    for ( int i = cuoi + 1; i <= n; i++)
    {
        TVector AB, BC;
        AB = Vector(A[cuoi].x - A[dau].x , A[cuoi].y - A[dau].y);
        BC = Vector(A[i].x - A[cuoi].x , A[i].y - A[cuoi].y  );
        if ( A[i].y != A[cuoi].y)
            if ( tichcheo(AB,BC) > 0) cuoi = i;
    }
    res += dtdagiac(A,dau,cuoi);
    dau = cuoi; cuoi = dau + 2;
    }
}
void docdl()
{
    char ch ;
    string s;
    freopen("POLYGON.INP","r", stdin);
    freopen("POLYGON.OUT","w", stdout);
    char ch1;
    int x,y;
    while ( cin >> s )
    {
        x = y = n = 0; ch1 = ' ';
        cout << s << endl;
        for ( int i = 0; i < s.length(); i++)
        {   if ( s[i] == 'S') break;
            else if ( s[i] != ch1)
```

```
            {
               n++; A[n].x = x;
               A[n].y = y;
            }
            if ( s[i] == 'R') x++;
            if ( s[i] == 'U') y++;
            ch1 = s[i];
         }
         n++; A[n].x = x; A[n].y = y;
         res = 0;
         xuli();
         cout << fixed << setprecision(3) << res << endl;
      }
   }
```

Bài 5: Diện tích phủ bởi các hình tròn

Trên mặt phẳng cho N hình tròn. Tính diện tích phần mặt phẳng bị phủ bởi các hình tròn trên.

INP.BL3	Kết quả:
3 0 0 5 1 1 5 7 0 1	Dien tich: 95.60

Dữ liệu: Cho trong file INP.BL3 dòng đầu là số lượng hình tròn, từ dòng thứ 2 trở đi mỗi dòng chứa 3 số nguyên dương là tọa độ x, y của tâm và bán kính của từng hình tròn (các số trên cùng một dòng ghi cách nhau ít nhất 1 dấu cách)

Kết quả: Xuất ra màn hình

- Ý tưởng:

- Tìm hình chữ nhật nhỏ nhất có các cạnh song song với các trục toạ độ và chứa toàn bộ N hình tròn

- Chia hình chữ nhật này thành lưới các ô vuông có cạnh 0.1 đơn vị, với mỗi ô thuộc hình chữ nhật kiểm tra xem ô này có thuộc vào hình tròn nào đó hay không, nếu có thì tăng diện tích cần tính lên 0.01 đơn vị.

Bài 6: Chia đất

Có một mảnh đất, trên biên của nó có cắm một số cột mốc để mảnh đất có thể xem như một đa giác có biên gồm các đoạn thẳng kề nhau nối các mốc này. Bắt đầu từ một cột mốc nào đó được đánh số 1, người ta đi vòng quanh mảnh đất theo một chiều xác định để đánh số các cột mốc kế tiếp (2, 3, ..., N) theo thứ tự được gặp.

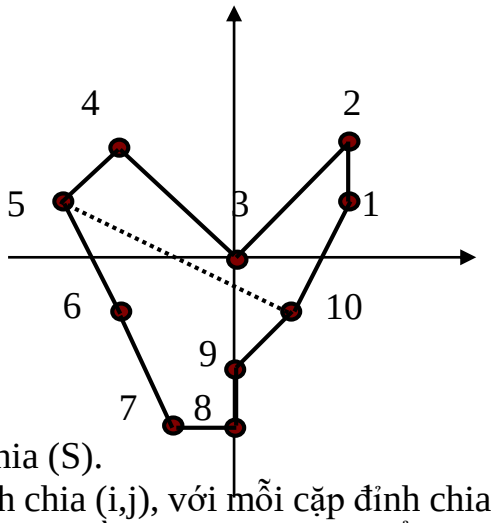
Yêu cầu: Hãy phân chia mảnh đất thành hai phần bằng một đoạn thẳng nối hai cột nào đó để độ chênh lệch diện tích của hai phần được chia là nhỏ nhất. Đoạn thẳng chia không được phần nào nằm ngoài mảnh đất và không được chứa cột mốc nào ngoài hai cột mốc đầu mút của nó.

Dữ liệu vào file CHIADAT.INP: Dòng đầu là số N (4<=N<=1000); N dòng tiếp theo, mỗi dòng ghi hai số nguyên x, y là các cặp toạ độ một cột mốc (theo thứ tự các cột từ 1 đến N). Các số trên một dòng được ghi cách nhau ít nhất một dấu cách.

Kết quả ra file CHIADAT.OUT: Một dòng duy nhất gồm hai số ghi cách nhau ít nhất một dấu cách, là số hiệu của hai cột mốc tạo thành đoạn thẳng chia.

Ví dụ:

CHIADAT.INP	CHIADAT.OUT
10 2 1 2 2 0 0 -2 2 -3 2 -2 -2 -1 -3 0 -3 0 -2 1 -2	4 8



Thuật toán:

- Tính diện tích ban đầu của mảnh đất cần chia (S).

- Ta dùng hai vòng lặp để duyệt mọi cặp đỉnh chia (i,j), với mỗi cặp đỉnh chia ta xét mọi đỉnh nằm trong đoạn từ i tời j xem có cùng về phía với đường thẳng đi qua hai điểm i, j không, tương tự với các đỉnh còn lại cũng nằm cùng về một phía với đường thẳng đi qua điểm i, j. Nếu cả hai điều kiện này đều sảy ra thì ta tính diện tích một phần (S1) ví dụ tính phần từ i đến j, sau đó so sánh với S/2, nếu S1 nào gần với S/2 nhất thì đó chính là kết quả của bài toán.

Code và test trong file đính kèm

c) Bài toán 3: Xác định đa giác nhỏ nhất bao tất cả các điểm, đa giác đã cho

Phương pháp: Cho N điểm A_1, A_2, ..., A_N trên mặt phẳng. Để xác định một đa giác không tự cắt chứa một số điểm đã cho và bao tất cả các điểm còn lại ta làm như sau:

Bước 1. Tìm điểm có tung độ nhỏ nhất. Điểm đó sẽ là đỉnh đa giác

Bước 2. Giả sử ta đã chọn được điểm P_M. Tìm điểm P_i sao cho góc hợp bởi P_MP_i và trục hoành là nhỏ nhất và đồng thời góc này phải lớn hơn góc hợp bởi P_MP_{M-1} và trục hoành. Điểm P_i sẽ là một đỉnh của đa giác.

Bước 3. Lấy kết quả là dãy các đỉnh P tìm được.

Lưu ý: Với bài toán tìm đa giác bao nhau thì cần ghi nhớ đa giác a bao đa giác b khi mọi điểm trong đa giác b đều nằm trong đa giác a.

Bài 7: Đa giác không tự cắt

Cho N điểm A_1, A_2, ..., 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 thực hiện các công việc sau đây: 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 và có chu vi nhỏ nhất. Hãy tính diện tích đa giác này.

Dữ liệu: cho trong tệp HCN.INP gồm n+1 dòng

+ Dòng 1: Chứa số N

+ Dòng i+1 ($1 \le i \le N$): Ghi 2 chữ số nguyên x_i, y_i là toạ độ đỉnh A_i. Các số trên cùng một dòng cách nhau một khoảng trắng.

HCN.inp	HCN.out
5 0 1 4 4 0 4 4 0 2 2	4 15.12 14.00 4 4 0 4 0 1 4 0

Kết quả: Xuất ra tệp HCN.Out

+ Dòng 1: Ghi 3 số K, V, S với K là số đỉnh đa giác tìm được, V là chu vi, S là diện tích của nó.

+ Dòng i+1($1 \le i \le K$): Ghi toạ độ của đỉnh đa giác.

***Ý tưởng:**

- Tìm điểm có tung độ nhỏ nhất. Điểm đó sẽ là đỉnh đa giác

- Giả sử ta đã chọn được điểm P_M. Tìm điểm P_i sao cho góc hợp bởi P_MP_i và trục hoành là nhỏ nhất và đồng thời góc này phải lớn hơn góc hợp bởi P_MP_{M-1} và trục hoành. Điểm P_i sẽ là một đỉnh của đa giác.

Code và test trong file đính kèm

Bài 8: Đa giác bao nhau

Cho N đa giác thoả mãn các tính chất

- Với 2 đa giác bất kỳ luôn có một đa giác mà mọi điểm của nó nằm trong đa giác kia.

- Các cạnh của chúng không có điểm chung.

Bài toán đặt ra là: Với mỗi đa giác i, có bao nhiêu đa giác bao nó? (i nằm trong bao nhiêu đa giác)

Dữ liệu vào: Ghi trong tập tin văn bản Dagiac.Inp.

- Dòng đầu tiên ghi số tự nhiên N ($3 \le N \le 10000$).

- Trên N dòng tiếp theo: Dòng thứ i+1 ghi thông tin về đa giác có số hiệu thứ i. Bao gồm số đầu tiên Si là số đỉnh của đa giác ($S_i \ge 3$), S_icặp số nguyên tiếp theo lần lượt là hoành độ và tung độ các đỉnh của đa giác. Các số trên cùng dòng cách nhau bởi ít nhất một khoảng trắng.

Dagiac.INP	Dagiac.OUT
4 4 1 1 15 1 15 8 1 8 4 9 3 9 6 4 6 4 3 4 3 2 11 2 11 7 3 7 3 8 4 8 5 6 5	0 2 1 3

Dữ liệu ra: Ghi trong tập tin Dagiac.Out

- Gồm N dòng

- Dòng thứ i: Ghi số lượng đa giác bao đa giác i.

Ý tưởng:

- Sử dụng các mảng a,vt,kq (với a[i] lưu giá trị hoành độ nhỏ nhất của các đỉnh của đa giác thứ i, vt[i] chỉ đa giác thứ i, mảng kq lưu kết quả)

- Thực hiện sắp xếp các đa giác theo thứ tự tăng dần của giá trị hoành độ nhỏ nhất của các đỉnh của các đa giác.

- Do theo điều kiện bài toán là với 2 đa giác bất kỳ luôn có một đa giác mà mọi điểm của nó nằm trong đa giác kia nên KQ[vt[i]] =i-1

Code và test trong file đính kèm

Bài 9. Trò chơi với tam giác

Cho N điểm với tọa độ nguyên, trong đó mọi tam giác nào với đỉnh là 3 trong n điểm đã cho có diện tích không vượt quá S. Nhiệm vụ của bạn là tìm 1 tam giác có diện tích không vượt quá 4S, các đỉnh có tọa độ nguyên, bao gồm N điểm đã cho.

- Dữ liệu đảm bảo không có 3 điểm nào thẳng hàng

Input

- 2 số nguyên N,S(N<=5000,S<=10^{18})
- N dòng sau mỗi dòng 2 số nguyên

Output

- Tọa độ của tam giác tìm được

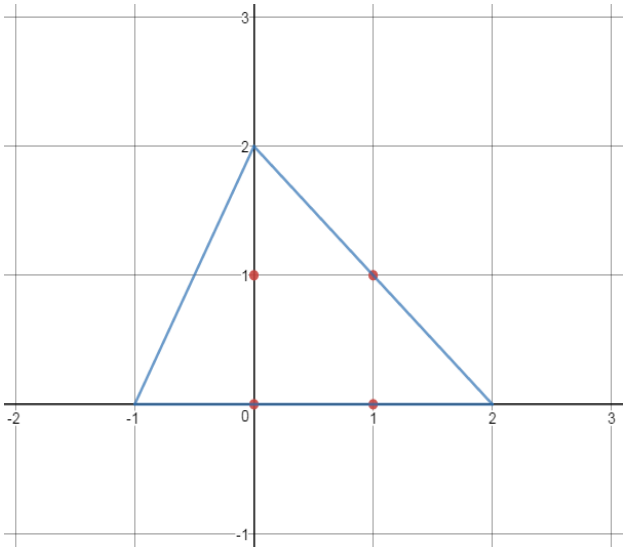
Example

input

```
4 1
0 0
1 0
0 1
1 1
```

output

```
-1 0
2 0
0 2
```



HD Thuật toán :

Tìm tam giác có diện tích lớn nhất, tam giác cần tìm có các cạnh nhận tọa độ của tam giác có diện tích lớn nhất làm trung điểm . Chúng ta dễ thấy tam giác này đương nhiên diện tích không vượt quá 4S và bao gồm N điểm . Giả sử có 1 điểm nằm ngoài tam giác này, vẽ đường cao đến các cạnh của tam giác có diện tích lớn nhất , luôn tồn tại 1 tam giác có diện tích lớn hơn => mâu thuẫn. Vậy bài toán chỉ đơn thuần tìm tam giác với diện tích lớn nhất .

Code:

```
#include<bits/stdc++.h>
usingnamespace std;
constint N =5010;

longlong n, s, x[N], y[N];
longlong ar(int a,int b,int c){
```

```
return abs((x[b]- x[a])*(y[c]- y[a])-(y[b]- y[a])*(x[c]- x[a]));
}

int main(){
scanf("%lld%lld",&n,&s);
for(int i =0; i < n;++i) scanf("%lld%lld", x + i, y + i);
int a =0, b =1, c =2;
while(1){
bool g =1;
for(int i =0; i < n;++i){
if(ar(i, b, c)> ar(a, b, c)) a = i, g =0;
if(ar(a, i, c)> ar(a, b, c)) b = i, g =0;
if(ar(a, b, i)> ar(a, b, c)) c = i, g =0;
}
if(g)break;
}
printf("%lld %lld\n%lld %lld\n%lld %lld\n",
x[a]+ x[b]- x[c], y[a]+ y[b]- y[c],
x[b]+ x[c]- x[a], y[b]+ y[c]- y[a],
x[c]+ x[a]- x[b], y[c]+ y[a]- y[b]);
return0;
}
```

Bài 10: Hình chữ nhật bao nhau

Cho N hình chữ nhật trên mặt phẳng mà các cạnh song song với các trục toạ độ. Biết hình chữ nhật i bao hình chữ nhật j nếu cả 4 đỉnh của hình chữ nhật j đều nằm trong hình chữ nhật i hoặc nằm trên cạnh của hình chữ nhật i.

Một dãy các hình chữ nhật được gọi là hình chữ nhật bao nhau chiều dài k (k≥1) nếu dãy này gồm các hình chữ nhật H1, H2, ..., Hk sao cho hình chữ nhật i bao hình chữ nhật i+1 với i=1 ... (k-1). Hãy tìm số k lớn nhất nói trên.

Dữ liệu vào: Được cho trong tập tin HCN.INP

- Dòng thứ nhất ghi số N (1≤N≤1000).
- N dòng tiếp theo, dòng thứ i ghi 4 số nguyên x1, y1, x2, y2 (-10000< x1,y1,x2,y2<10000) lần lượt là hoành độ, tung độ các đỉnh trái trên, phải dưới của hình chữ nhật.

HCN.INP	HCN.OUT
6 1 5 2 2 2 4 3 3 1 5 5 2 4 3 8 1 5 6 8 4 6 6 8 5	2

Kết quả: Được ghi vào tệp văn bản HCN.OUT gồm một dòng chứa số nguyên duy nhất là số k tìm được hoặc số -1 nếu không tồn tại số k thoả điều kiện đề bài.

*** Ý tưởng:**

- Tính diện tích các hình chữ nhật (HCN)
- Sắp xếp lại các HCN theo thứ tự không giảm của diện tích các HCN
- Lập hàm kiểm tra HCN i bao HCN j, thoả mãn điều kiện:
  (x1[i]<=x1[j]) and (y1[i]>=y1[j]) and (x2[i]>=x2[j]) and (y2[i]<=y2[j])
- Xác định số lượng các HCN bao HCN i và lưu vào phần tử mảng kq[i] biết rằng: nếu thì kq[i]:=kq[j]+1

Code và test trong file đính kèm

3. Bài tập tổng hợp

3.1. Bài 1. Bán kính lớn nhất

Bạn được một công ty du lịch giao cho việc thiết kế một địa điểm du lịch mới. Ở địa điểm này, các du khách sẽ được ngồi trên một chiếc bè di chuyển trên một dòng sông và ngắm quang cảnh thiên nhiên xung quanh. Dòng sông đã được thiết kế song. Nó được giới hạn bởi hai hình đa giác, hình đa giác lớn nằm bên ngoài hình đa giác nhỏ. Lòng sông chính là phần nằm giữa hai đường biên đa giác (nói cách khác là tất cả những điểm nằm trong đa giác lớn mà không nằm trong đa giác nhỏ).

Chiếc bè phải được thiết kế hình tròn để có thể dễ dàng xoay khi di chuyển, tạo cảm giác thú vị cho du khách. Đồng thời chiếc bè cũng cần phải có kích thước lớn nhất có thể để phục vụ được nhiều du khác một lúc nhất. Chiếc bè sẽ đi một vòng quanh đa giác nhỏ và để không bị mắc kẹt giữa hai bờ sông, nó phải có một bán kính hợp lý.

Bạn hãy tính bán kính lớn nhất của chiếc bè sao cho du khách có thể đi một vòng quanh đa giác nhỏ mà không bị mắc kẹt.

Input: polygon.inp

- Nhóm dòng đầu tiên mô tả đa giác nhỏ với cấu trúc như sau: Dòng đầu ghi số nguyên dương N1 là số đỉnh của đa giác nhỏ. N1 dòng sau mỗi dòng ghi hai số nguyên dương (x_i, y_i) là tọa độ của một đỉnh thuộc đa giác nhỏ. Nhóm dòng thứ hai mô tả đa giác lớn theo cấu trúc tương tự.
- Mỗi đa giác có ít nhất 3 đỉnh và không quá 100 đỉnh. Các tọa độ có giá trị tuyệt đối không quá 1000. Các đỉnh được liệt kê theo chiều kim đồng hồ hoặc ngược lại. Hai cạnh thuộc hai đa giác không có điểm chung. Đa giác lớn đảm bảo bao hoàn toàn đa giác nhỏ.

Output: polygon.out

- Ghi ra kết quả tìm được với sai số không quá 10^{-6}.

Ví dụ:

Input	Output
4 -5 -5 5 -5 5 5 -5 5 4 -10 -10 -10 10 10 10 10 -10	2.5

Giải thích: Hai đa giác đều là hai hình vuông có tâm ở gốc tọa độ. Đa giác nhỏ là hình vuông cạnh 10 và đa giác lớn là hình vuông cạnh 20.

Hướng dẫn: Kết quả bài toán là ½ khoảng cách nhỏ nhất trong các khoảng cách giữa mọi cặp cạnh với một cạnh thuộc đa giác nhỏ và cạnh còn lại thuộc đa giác lớn. Khoảng cách giữa hai đoạn thẳng là khoảng cách nhỏ nhất giữa hai điểm tương ứng thuộc đường thẳng đó. Ta chỉ cần xét các đầu mút và đưa về bài toán tìm khoảng cách từ 1 điểm đến một đoạn thẳng. Bài này dùng tích vô hướng trong lúc

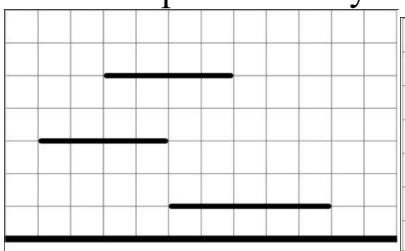
kiểm tra chân đường cao có nằm trong đoạn thẳng kia không. Rồi dùng tích có hướng để tính diện tích. Cuối cùng chỉ cần dùng phép căn cho việc tính khoảng cách nên sai số rất nhỏ.

Code và test trong file đính kèm

3.2. Bài 2. Đặt chân chống (FLATFORM)

Trong một trò chơi cần phải đặt các miếng phẳng nằm ngang tại các vị trí cho trước. Các miếng phẳng này không thể đặt trong không khí nên mỗi miếng cần phải có một cặp chân chống đỡ nó. Các chân chống này có thể được đặt trên nền nhà hoặc đặt trên mặt một miếng phẳng khác.

Bạn được cho vị trí các miếng phẳng trong hệ trục tọa độ tương tự như hình bên trái của hình dưới đây. Mỗi miếng phẳng được xác định bởi độ cao của nó (khoảng cách đến nền nhà) và tọa độ hai điểm đầu mút của miếng theo phương nằm ngang. Tất cả các chân chống được đặt cách hai đầu mút một nửa đơn vị độ dài tương tự như hình bên phải dưới đây:



Example level with three platforms. The lowest is at altitude 1, the next at altitude 3, and the third at 5.

The total length of support pillars needed is 14.

Hãy xác định tổng độ dài của các chân chống dùng để đỡ các miếng phẳng nói trên.

Input: File PLATFORM.INP

- Dòng đầu tiên ghi số nguyên dương N ($1 \le N \le 100$) là số lượng các miếng phẳng.
- N dòng tiếp theo, mỗi dòng mô tả một miếng phẳng gồm 3 số nguyên Y, X_1, X_2 lần lượt là độ cao, hoành độ mép trái và mép phải của miếng phẳng. Các tọa độ là các số nguyên dương không vượt quá 10000 thỏa mãn $X_2 > X_1 + 1$ (tức mỗi miếng phẳng có độ rộng tối thiểu là 2)

Dữ liệu vào thỏa mãn không có hai miếng phẳng nào phủ chồng lên nhau

Output: File PLATFORM.OUT

Một số nguyên duy nhất là tổng độ dài của các chân chống cần đỡ

PLATFORM.INP	PLATFORM.OUT
3 1 5 10 3 1 5 5 3 7	14

HD Thuật toán: Đặt chân chống (FLATFORM)

- Với mỗi flatform chúng ta ghi nhớ bằng bộ ba số (y,x1,x2) trong đó y là độ cao, x1 là đoạn đơn vị đầu tiên và x2 là đoạn đơn vị cuối cùng theo trục hoành.
- Sắp xếp các y tăng dần
- Với flatform thứ i tính độ cao cần thiết và cập nhật độ cao theo trục hoành.
- Với N nhỏ thì chỉ cần lưu trữ mảng 1 chiều theo trục x là mảng "mặt sàn" hiện thời và thao tác cập nhật có thể dùng 1 vòng for

- Ta có thể mở rộng cho N lớn hơn bằng cách sử dụng cấu trúc dữ liệu IT

Code và test trong file đính kèm

3.3. Bài 3. Góc lớn nhất

Cho N điểm $A_1,A_2,A_3,...,A_N$ trên mặt phẳng. Các điểm đều có toạ độ thực và không có 3 điểm bất kỳ trong chúng thẳng hàng. Khi đó ta sẽ xác định một đa giác không tự cắt có đỉnh là một trong số các điểm đã cho và chứa tất cả những điểm còn lại và có chi vi nhỏ nhất. Trong đa giác đó hãy tìm góc lớn nhất tính bằng rad

Dữ liệu vào: GOCMAX.INP gồm N+1 dòng:

+ Dòng 1: Chứa số N (N ≤200)
+ Các dòng tiếp theo mỗi dòng chứa 2 số thực là toạ độ của một điểm.

Kết quả: Ghi ra file GOCMAX.OUT như sau:

Kết quả là một số thực có 3 chữ số phần thập phân

GOCMAX.INP	GOCMAX.OUT
3 0 0 5 0 0 2.5	1.571

Hướng dẫn thuật toán:

+ Tìm bao lồi chứa N điểm đó
+ Tìm góc lớn nhất của bao lồi đó

```
#include <bits/stdc++.h>
#define Point TVector
using namespace std;
const int maxn = 1000;
const float PI = 3.1416;
struct Point
{
   float x, y;
};
int n, m;
Point P[maxn +1];
int bao[maxn + 1];
void doc()
{
   freopen("GOCMAX.INP", "r", stdin);
   freopen("GOCMAX.OUT","w", stdout);
   cin >> n;
   for ( int i = 1; i <= n; i++) cin >> P[i].x >> P[i].y;
}
float goc(Point p1, Point p2)
{
   float g, dx, dy;
   dx = p2.x - p1.x; dy = p2.y - p1.y;
   if ( dx == 0)
      if ( dy > 0) g = PI/2;
```

```
      else g = 3*PI/2;
   else
   {
      g = atan(abs(dy)/abs(dx));
      if ( dx < 0)
         if ( dy > 0) g = PI - g;
         else g = PI + g;
      else
         if ( dy < 0) g = 2*PI - g;
   }
   return g;
}
void xuly()
{
   int i, j, t;
   float r, r1, r2;
   float g2, g1;
   //Tim deim co tung do nho nhat
   j = 1;
   for ( i = 2; i <= n; i++)
      if ( P[i].y < P[j].y) j = i;
   // Bao goi
   for ( int i = 1; i <= n; i++) bao[i] = i;
   bao[n+1] = j;
   P[n+1] = P[j];
   m = 0 ; g2 = 0;
   do
   {
      m++;
      swap(bao[m],bao[j]);
      g1 = g2; g2 = 2*PI;
      for ( i = m +1 ; i <= n + 1; i++)
      {
         float g = goc(P[bao[m]], P[bao[i]]);
         if ( g >= g1 && g < g2)
         {
            j = i;
            g2 = g;
         }
      }
   }while ( j != n + 1);
}
TVector Vector(float x, float y)
{
   TVector tmp;
   tmp.x = x; tmp.y = y;
```

```
    return tmp;
}
float tichcham(TVector u, TVector v)
{
    return ( u.x * v.x + u.y*v.y);
}
float tichcheo(TVector u, TVector v)
{
    return ( u.x*v.y - v.x*u.y);
}
float Rad(TVector u, TVector v)
{
    return (atan2(tichcheo(u,v),tichcham(u,v)));
}
void ghi()
{
    //cout << m << endl;
    //for ( int i = 1; i <= m; i++) cout << P[bao[i]].x << ","  << P[bao[i]].y << endl;
    P[0] = P[bao[m]]; P[bao[m+1]] = P[bao[1]];
    float res = 0;
    for ( int i = 1; i <= m; i++)
    {
        TVector u, v;
        u = Vector(P[bao[i-1]].x - P[bao[i]].x, P[bao[i-1]].y - P[bao[i]].y);
        v = Vector(P[bao[i+1]].x - P[bao[i]].x, P[bao[i+1]].y - P[bao[i]].y);
        float res1 = Rad(v,u);
        if ( res < res1) res = res1;
    }
    cout << fixed << setprecision(3) << res << endl;
}
int main()
{
    doc();    xuly();    ghi();    return 0;
}
```

3.4. Bài 4. Trò chơi với Set

Bạn được cho 1 Set rỗng, mỗi phần tử của set gồm 1 cặp số nguyên. Có N truy vấn thuộc 3 loại:

1. Đưa cặp số (a, b) vào set
2. Loại bỏ cặp số đã được thêm vào ở truy vấn thứ I (truy vấn thứ I thuộc loại 1)
3. Cho số nguyên q, tìm giá trị q*a+b lớn nhất trong set .

Input:

- Dòng đầu số N (N<=10^5) số lượng truy vấn.
- Đầu mỗi dòng trong N dòng tiếp theo là số nguyên t(1<=t<=3) loại truy vấn
- Truy vấn loại 1 thì sao đó là 2 số nguyên a, b

- Truy vấn loại 2 thì sau đó là số nguyên I, dữ liệu đảm bảo rằng I nhỏ hơn số lượng truy vấn ở thời điểm đó. Ví dụ ta đang sẽ đến truy vấn số 3 thì i<3 và dữ liệu cũng đảm bảo rằng truy vấn số I thuộc loại 1 và chưa từng bị loại bỏ trước đó.
- Truy vấn loại 3: tiếp theo đó là số nguyen q ($|q|<=10^9$)

Output:

Kết quả cho mỗi truy vấn loại 3, mỗi truy vấn 1 dòng, nếu set rỗng in ra "EMPTY SET".

Example

input

```
7
3 1
1 2 3
3 1
1 -1 100
3 1
2 4
3 1
```

output

```
EMPTY SET
5
99
5
```

Code:

```
#include<cstdio>
#include<vector>
#include<algorithm>
#definepb push_back
usingnamespace std;
constint N =2010100;
struct point{
	longlong x, y;
	longlongoperator*(const point &t)const
	{
		return x*t.y-y*t.x;
	}
	pointoperator-(const point &t)const
	{
		point tmp;
		tmp.x=x-t.x;
		tmp.y=y-t.y;
		return tmp;
	}
	longlongoperator&(constlonglong&t)const
	{
		return x*t+y;
```

```
        }
        booloperator<(const point &t)const
        {
                if(x==t.x)          return y<t.y;
                return x<t.x;
        }
    }p[N],stack[N];
    vector<point> vec[N];
    int n,i,typ[N],q[N],cnt,L[N],R[N],id[N],a;
    longlong ans[N], inf;
    void put(int x,int a,int b,int l,int r,point p)
    {
        if((a<=l) &&(r<=b))
        {
                vec[x].pb(p);
                return;
        }
        int m=(l+r)>>1;
        if(a<=m) put(2*x,a,b,l,m,p);
        if(m+1<=b) put(2*x+1,a,b,m+1,r,p);
    }
    longlong query(longlong x,int top)
    {
        int l=1,r=top;
        if(top==0)          return-inf;
    while(l<r)
    {
    int m1=(2*l+r)/3;
    int m2=(l+2*r+2)/3;
    if((stack[m1]&x)<(stack[m2]&x))
                l=m1+1;
    else        r=m2-1;
    }
    return(stack[l]&x);
    }
    void solve(int x,int l,int r)
    {
        int m=(l+r)>>1,i;
        sort(vec[x].begin(),vec[x].end());
        int top=0;
        for(i=0;i<vec[x].size();i++)
        {
                while((top>1)&&((vec[x][i]-stack[top])*(stack[top]-stack[top-
1])<=0)) top--;
                stack[++top]=vec[x][i];
        }
```

```
        for(i=l;i<=r;i++)
        if(typ[i]==3)
        {
                ans[i]=max(ans[i],query(q[i],top));
        }
        if(l<r)
        {
                solve(2*x,l,m);
                solve(2*x+1,m+1,r);
        }
}
int main()
{
        scanf("%d",&n);
        inf=2100000000;
        inf=inf*inf;
        for(i=1;i<=n;i++)
        {
                ans[i]=-inf;
                scanf("%d",&typ[i]);
                if(typ[i]==1)
                {
                        cnt++;scanf("%I64d%I64d",&p[cnt].x,&p[cnt].y);
                        L[cnt]=i;R[cnt]=n;id[i]=cnt;
                }
                else
                if(typ[i]==2)
                {
                        scanf("%d",&a);
                        R[id[a]]=i-1;
                }
                else
                        scanf("%d",&q[i]);
        }
        for(i=1;i<=cnt;i++)
                put(1,L[i],R[i],1,n,p[i]);
        solve(1,1,n);
        for(i=1;i<=n;i++)
        if(typ[i]==3)
        if(ans[i]==-inf)
                printf("EMPTY SET\n");
        else
                printf("%I64d\n",ans[i]);
}
```

Bài 5: Maxim và xe đạp

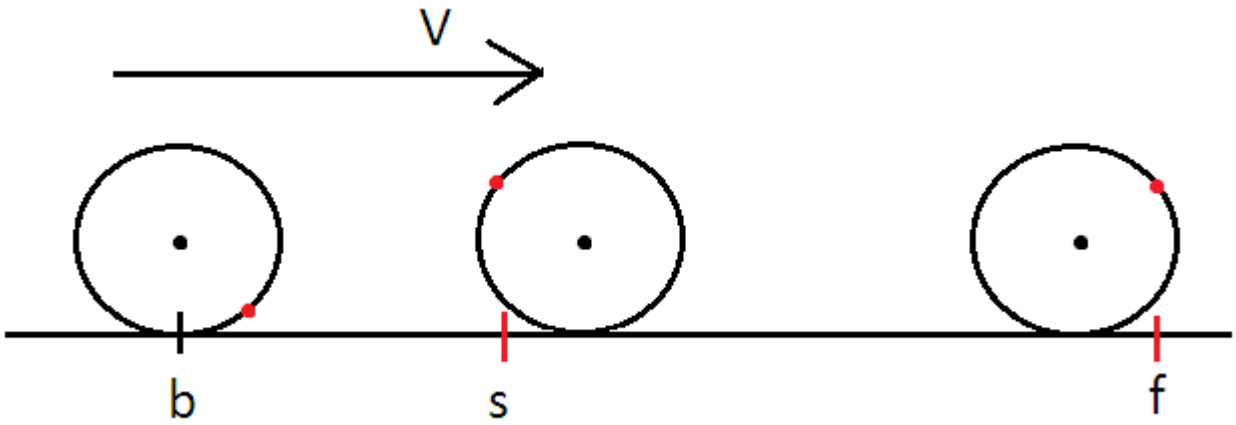
https://codeforces.com/problemset/problem/594/B

Trong nhiều tháng, Maxim đã đến làm việc trên chiếc xe đạp yêu thích của mình. Và gần đây anh ấy đã quyết định rằng anh ấy đã sẵn sàng tham gia vào các cuộc thi của người đi xe đạp.

Ông biết rằng năm nay có n cuộc thi sẽ diễn ra. Trong cuộc thi thứ i , người tham gia phải nhanh nhất có thể hoàn thành một chuyến đi dọc theo một đường thẳng từ điểm s_i đến điểm f_i ($s_i < f_i$).

Đo thời gian là một quá trình phức tạp liên quan đến việc sử dụng cảm biến đặc biệt và bộ đếm thời gian. Hãy nghĩ về bánh trước của một chiếc xe đạp như một vòng tròn bán kính r . Chúng ta hãy bỏ qua độ dày của lốp xe, kích thước của cảm biến và tất cả các hiệu ứng vật lý. Cảm biến được đặt trên vành bánh xe, nghĩa là trên một số điểm cố định trên một vòng tròn bán kính r . Sau đó, bộ đếm di chuyển giống như điểm đã chọn của vòng tròn, tức là di chuyển về phía trước và xoay quanh tâm của vòng tròn.

Lúc đầu, mỗi người tham gia có thể chọn **bất kỳ** điểm $b_{i\ nào}$, sao cho chiếc xe đạp của anh ta hoàn toàn phía sau vạch xuất phát, nghĩa là, $b_i < s_i - r$. Sau đó, anh ta bắt đầu chuyển động, ngay lập tức tăng tốc đến tốc độ tối đa của mình và tại thời điểm ts_i , khi tọa độ của cảm biến bằng với tọa độ bắt đầu, bộ đếm thời gian bắt đầu. Người đi xe đạp thực hiện một chuyến đi hoàn chỉnh, di chuyển với tốc độ tối đa của mình và tại thời điểm tọa độ của cảm biến bằng với tọa độ kết thúc (thời điểm tf_i), bộ đếm thời gian hủy kích hoạt và ghi lại thời gian cuối cùng. Do đó, bộ đếm ghi lại rằng người tham gia đã thực hiện một chuyến đi hoàn chỉnh trong thời gian $tf_i - ts_i$.



Maxim giỏi toán và anh ta nghi ngờ rằng tổng kết quả không chỉ phụ thuộc vào tốc độ tối đa v của anh ta, mà còn phụ thuộc vào sự lựa chọn của anh ta về điểm ban đầu b_i . Bây giờ Maxim đang yêu cầu bạn tính toán cho mỗi n cuộc thi thời gian tối thiểu có thể được đo bằng bộ đếm thời gian. Bán kính của bánh xe đạp của anh ta bằng r .

Đầu vào

- Dòng đầu tiên chứa ba số nguyên n , r và v ($1 \le n$ 100 000, $1 \le r$, $v \le 10^9$) số lần thi đấu, bán kính của bánh trước của xe đạp Max và tốc độ tối đa của anh ta, tương ứng.

- *N* dòng tiếp theo chứa các mô tả của các cuộc thi. Dòng thứ *i* chứa hai số nguyên s_i và f_i ($1 \le s_i < f_i \le 10^9$) - tọa độ bắt đầu và tọa độ kết thúc của cuộc thi thứ *i* .

Đầu ra

In *n* số thực, số thứ *i* phải bằng thời gian tối thiểu có thể được đo bằng bộ đếm thời gian. Câu trả lời của bạn sẽ được coi là chính xác nếu lỗi tuyệt đối hoặc tương đối của nó sẽ không vượt quá 10^{-6} .

Cụ thể: giả sử rằng câu trả lời của bạn bằng *a* và câu trả lời của bồi thẩm đoàn là *b* . Chương trình kiểm tra sẽ xem xét câu trả lời của bạn chính xác nếu $\frac{|a-b|}{\max(1,b)} \le 10^{-6}$.

Ví dụ

input	**output**
2 1 2 1 10 5 9	3.849644710502 1.106060157705

```
#include<bits/stdc++.h>
#define eps 1e-8
using namespace std;
double n,r,v;
double s,f;

int main()
{
	scanf("%lf%lf%lf",&n,&r,&v);
	while(n--)
		{
		scanf("%lf%lf",&s,&f);
		double L=0,R=1e10;
		for(int o_o=1;o_o<=200;o_o++)
			{
			double mid=(L+R)/2;
			double t=mid*v+r*fabs(sin(v*mid/r));
			if(t*2>f-s)R=mid;
			else L=mid;
			}
		printf("%.9f\n",L+R);
		}
	return 0;
}
```

4. Một số bài tập tự luyện

http://vn.spoj.com/problems/VMCXG/
http://vn.spoj.com/problems/PRAVO/

http://vn.spoj.com/problems/VORAIN/
http://vn.spoj.com/problems/BALLGMVN/
https://codeforces.com/problemset/problem/717/I
https://codeforces.com/problemset/problem/1071/E
https://codeforces.com/problemset/problem/549/E
https://codeforces.com/problemset/problem/388/E

C. KẾT LUẬN

Với kinh nghiệm còn hạn chế, bài viết mới chỉ dừng ở mức độ tập hợp một số bài toán có thể giải quyết bằng phương pháp hình học. Một số bài chưa đi sâu phân tích cụ thể để đưa ra thuật toán chi tiết, cũng như chưa tập hợp đầy đủ các bài toán có thể giải bằng phương pháp này. Bên cạnh đó còn vấn đề cần phải có một phương pháp truyền đạt như thế nào để học sinh tích cực, chủ động lĩnh hội được các kiến thức chuyên sâu. Tôi hy vọng nhận được sự giúp đỡ, góp ý của đồng nghiệp để bài viết được hoàn thiện hơn, góp phần nhỏ cho công tác bồi dưỡng học sinh giỏi.

Tôi xin chân thành cảm ơn!

D. TÀI LIỆU THAM KHẢO

1. Tài liệu tập huấn phát triển chuyên môn giáo viên Tin học - Nhiều tác giả;
2. VNOI - Olympic tin học Việt Nam
3. Website: http://vn.spoj.pl; https://codeforces.com