ORIGINAL PAPER

CONVEX HULL PROBLEM, LATTICE POINTS AND APPLICATIONS

DUMITRU FANACHE[1]

Manuscript received: 12.04.2011. Accepted paper: 11.05.2011.

Published online: 10.06.2011.

Abstract. *Problem of finding convex hull is one of the central problems of computational geometry. It appears both applications in economic, financial, environmental, architectural and analytical geometry in specific issues. Latticial point is called (in the plane or in space) at any point whose coordinates are integers. Historically, lattices were investigated since the late 18th century by mathematicians such as Lagrange, Gauss, and later Minkowski. More recently, lattices have become a topic of active research in computer science. They are used an algorithmic tool to solve a wide variety of problems; they have many applications in cryptography and cartography; and they have some unique properties from a computational complexity point of view.*

Keywords: *Lattice point, convex hull, algorithm.*

1. PRELIMINARIES

In most computational geometry problems appears as subproblem determining coordinates of intersection point of two straight lines. The method used to solve this subproblem is the same, with less significant differences for each case.

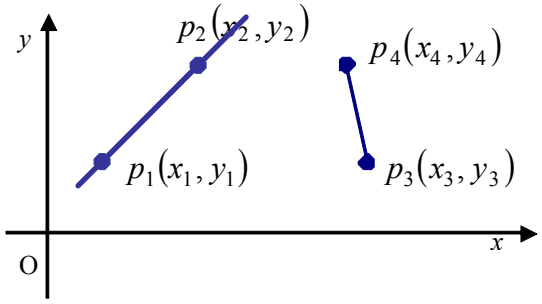


Fig. 1. Defining straight line (segment) by two points.

Both to define right and to define the segment using two distinct points: arbitrary (the straight line) or extreme (to segment) (Fig. 1). Therefore, describing both straight line and segment description may be made of the same data structure, which contains coordinates of two points and, additional, coefficients A, B, C of general equation of straight line. These coefficients will be needed to calculate the coordinates of intersection point. One of the possible ways of defining this structure is:

[1] Valahia University of Targoviste, Faculty of Sciences and Arts, 130082, Targoviste, Romania.
E-mail: dfanache@gmail.com.

```
struct line
{float x1,y1,x2,y2 ;
     float A,B,C;};
```

Let straight line (l) determined by the $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ straight line equation passing through these two points is:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \tag{1.1}$$

From this equation can be calculated coefficients of the right general equation:

$$Ax + By + C = 0 \tag{1.2}$$

by elementary transformation of (1.1) is obtained:

$$x(y_2 - y_1) + y(x_1 - x_2) + (x_2 y_1 - x_1 y_2) = 0 \tag{1.3}$$

From (1.3) resulting formulas of computing for general equation coefficients:

$$A = y_2 - y_1; \quad B = x_1 - x_2; \quad C = x_2 y_1 - x_1 y_2 \tag{1.4}$$

1.1. DETERMINATION OF THE INTERSECTION OF TWO STRAIGHT LINES

Let straight line (p) and (l) determined by pairs of points $p_1(x_1, y_1), p_2(x_2, y_2)$ and that $p_3(x_3, y_3), p_4(x_4, y_4)$. Determine the intersection point q this straight line is reduced to solving the system of equations:

$$\begin{cases} A_1 x + B_1 y + C_1 = 0 \\ A_2 x + B_2 y + C_2 = 0 \end{cases}$$

where $A_1 x + B_1 y + C_1 = 0$ is general equation of straight line p and $A_2 x + B_2 y + C_2 = 0$ is general equation of straight line (l). Coefficients of these equations can be calculated in accordance with relations (I.4). By direct solving of the system will determine if lines are parallel or coincide. This check conditions:

a) $A_1 B_2 = A_2 B_1$ and $A_1 C_2 \neq A_2 C_1$ - lines (p) and (l) are parallel (1.5.a)

b) $A_1 B_2 = A_2 B_1$ and $A_1 C_2 = A_2 C_1$ - lines (p) and (l) coincide (1.5.b)

If none of the above conditions is not verified, then there is a single point of intersection of straight lines and whose coordinates can be calculated by formulas:

$$\text{if } A_1 \neq 0 \Rightarrow y_{sol} = \frac{C_1 A_2 - A_1 C_2}{B_2 A_1 - B_1 A_2}, \quad x_{sol} = -\frac{B_1 y_{sol} + C_1}{A_1} \tag{1.6.a}$$

$$\text{if } A_1 = 0 \Rightarrow y_{sol} = -\frac{C_1}{B_1}, \quad x_{sol} = -\frac{B_2 y_{sol} + C_2}{A_2} \tag{1.6.b}$$

1.2. INTERSECTION OF THE STRAIGHT LINE WITH A SEGMENT

The formulas derived in the previous paragraph it possible to calculate the coordinates of the point of intersection of two straight lines. If the intersection of two segments or lines and a segment may be reduced at the intersection of straight lines, but verification of additional conditions.

Let be straight line (l) passing through points p_1, p_2 and (s) segment with extremities points s_1, s_2. Note that if the intersection straight line (l) and the segment (s), segment extremities are positioned in different parts of vector $\vec{p_2 p_1}$. If researched objects do not intersect, then both extremities of the segment are the same part of the vector $\vec{p_2 p_1}$ (Fig. 2).

1.3. POSITION OF THE POINT TO A VECTOR.

Whether vector $\vec{p_2 p_1}$ whose extremities have the coordinates (x_2, y_2) respectively (x_1, y_1) and s point of coordinates (x_3, y_3). For the position s point to vector $\vec{p_2 p_1}$ can be used following determinant used to calculate the area of a triangle when its vertices are given by coordinates.

$$\Delta = \begin{vmatrix} x_2 & y_2 & 1 \\ x_1 & y_1 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \tag{1.7}$$

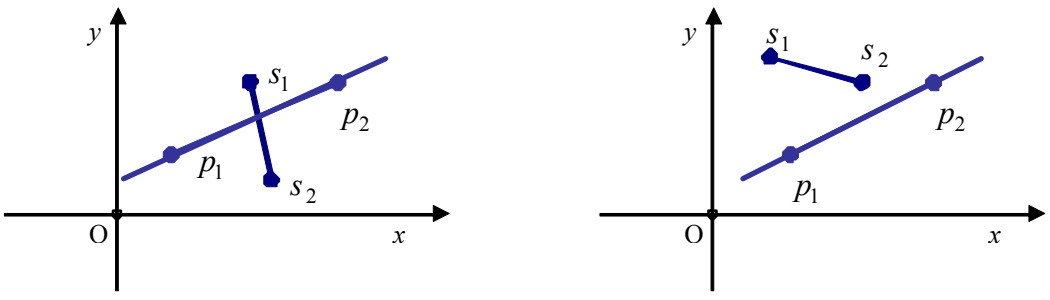


Fig. 2. Segment position to straight line.

Determinant is positive, if the point s is located in right semiplane to vector $\vec{p_2 p_1}$; is null, if point s belongs of this vector; is negative, if s is located left semiplane. It is known

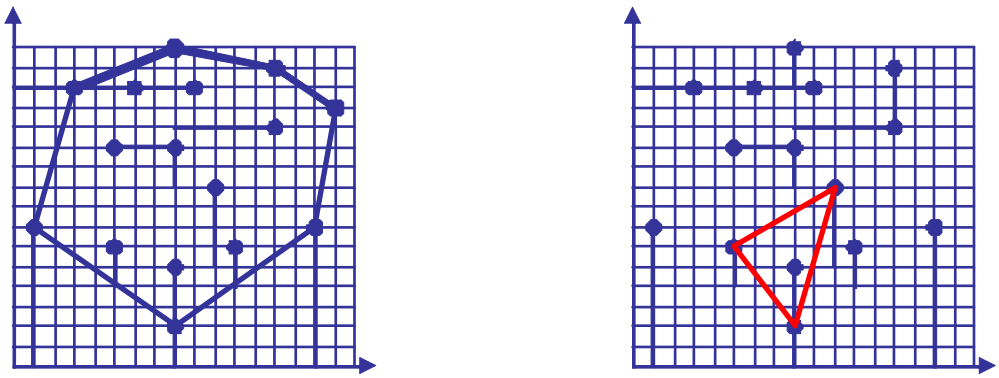
that Δ can be evaluated simply by the triangle method (known as the *Sarrus's rule*), we present the evaluation function of:

```
float Sarrus(point p1;point p2;point p3)
{float k=
 p1.x*p2.y+p2.x*p3.y+p1.y*p3.x-p3.x*p2.y-p3.y*p1.x-p1.y*p2.x;
 return k ;}
```

Expression Sarrus(p2,p1,s1)*Sarrus(p2,p1,s2) will be of value: *positive*, if straight line (l) and segment (s) *not intersect* (both extremities of the segment are on the same side of straight line, the determinant values are of the same sign); *zero*, if at least one between extremities *belong of straight line* (for extremities of segment (s) what belong straight line determinant is zero); *negative*, if the straight line (l) and segment (s) *intersects* (segment extremities are located on different parts of the vector, the determinant values have signs different).Therefore, if the straight (l) determined of p_1 and p_2 points and segment (s) with extremities s_1 and s_2 expression Sarrus(p2,p1,s1)*Sarrus(p2,p1,s2) has negative value, then the straight (l) and segment (s) intersect, and coordinates of the point of intersection can be calculated according to formulas (1.6). If the value expression is null, check directly that of extremities of segment belongs the straight line. If both extremities of the segment are on straight line, then whole segment is content of straight line.

2. CONVEX HULL PROBLEM

The notion of the convex hull in the plane is intuitively simple. For a set points S of the plane, convex hull $Q(S)$ is the set of vertices of convex polygon with the smallest area containing all points of the set S (Fig. 3a).



(a) **(b)**

Fig. 3. Convex hull of a set points (a); Determination of inside point (b).

2.1. BASIC ALGORITHM.

Let be the set of points on the plane $S=\{p_1,p_2,\cdots,p_N\}$. Each element p_i of the set is described by its Cartesian coordinates (x_i,y_i). An intuitive method for determining of the

convex hull requires the removal of initial set of all its interior points. Basic algorithm assume on two simple statements:

(a) the convex hull of a set points S is formed of extreme points of the set S;

(b) an point $p \in S$ is not extreme point if and only if at least one triangle, determined by points $p_i, p_j, p_k \in S$, so p to belong (Fig. 3 b).

Based on this affirmation, the inside points of set are excluded by researching belonging each point to each triangle generated by the set points. So much the pseudocode and algorithm are very simple.

Pseudocode

Pas 1 $C \leftarrow S$

Pas 2. for all $p_i \in S$

for all $p_j \in S, p_j \neq p_i$

for all $p_k \in S, p_k \neq p_j, p_k \neq p_j$

for all $p \in S, p \neq p_i, p \neq p_j, p \neq p_k$

if $p \in \Delta p_i p_j p_k$ then $C \leftarrow C \setminus \{p\}$

2.2. THE BELONG OF POINT TO A TRIANGLE

Condition $p \in \Delta p_i p_j p_k$ can be verified by determining the point position p from each of the vectors $\vec{p_i p_j}$, $\vec{p_j p_k}$, $\vec{p_k p_i}$. If the sign of values returned from function calls `Sarrus(pi,pj,p), Sarrus(pj,pk,p, Sarrus(pk,pi,p)` is the same, then $p \in \Delta p_i p_j p_k$. Therefore, verification membership point a triangle can be achieved in a number of operations bounded by a constant. Instructions for the step 2 of algorithm generate a number of operations proportional with N^4, what determines the final complexity of the algorithm $O(N^4)$. A slightly more sophisticated, but much more efficient algorithm is the Graham *scan*, published in 1972 (O(*n* log *n*) time complexity). If the points are already sorted by one of the coordinates or by the angle to a fixed vector, then the algorithm takes O(*n*) time [3]. The comparison of execution divers algorithms is found in [4] and in Fig. 5. An utilization example convex hull routine from Matlab is in Fig. 6.

2.3. IMPLEMENTATION

Data structure used to determine of convex hull is an array of elements point type (see line 8), each being an article by three components: coordinates of point (false – *extreme point*, true - *inside point*), which specifies the membership of convex hull. Checking whether the point belongs to a triangle is performed by `apart()` function.

```
.   . . .
7   struct point
8    {int x,y;bool interior;} ;
9   point a[max];int n,i,j,k,l;
10  ifstream f("poligon.in");ofstream g("poligon.out");
```

```
11  float sarrus(point p1,point p2,point p3)
12  {float k=p1.x*p2.y+p2.x*p3.y+p1.y*p3.x
13   -p3.x*p2.y-p3.y*p1.x-p1.y*p2.x;return k ;}
14  bool apart(int l ,int i ,int j,int k)
15  {float k1,k2,k3;bool ok=true;
16   k1=sarrus(a[i],a[j],a[l]);k2=sarrus(a[j],a[k],a[l]);
17   k3=sarrus(a[k],a[i],a[l]);
18   if ((k1*k2<=0)||(k1*k3<=0)||(k2*k3<=0))THEN ok=false;
19   return ok;}
20  int main(){f>>n;
21   for(i=1;i<=n;i++)
22    {f>>a[i].x>>a[i].y;a[i].interior=false;}f.close();
23   for(i=1;i<=n-2;i++)
24    for(j=i+1;j<=n-1;j++)
25     for(k=j+1;k<=n;k++)
26      for(l=1;l<=n;l++)
27       if((l!=i)&&(l!=j)&&(l!=k))THEN
28      if (apart(l,i,j,k)==true)THEN a[l].interior=true;
29  // display items forming convex hull
30  for(i=1;i<=n;i++)
31   if(a[i].interior==false)
32      g<<a[i].x<<" "<<a[i].y<<endl;g.close();  return 0;}
```

Application of markings is achieved in 23-28 lines from program below. In 27 line marked with false points for that i read its coordinates. Show coordinates of points what forming convex hull is done by checking marks (31 line) (see example from Fig. 5).

```
poligon.in        poligon.out
16          11 6    1 7
1 7         10 9    3 14
3 14        5 11    8 16
8 16        8 11    13 15
13 15       13 12   16 13
16 13       6 14    15 7
15 7        9 14    8 2
5 6         8 2
8 5
```

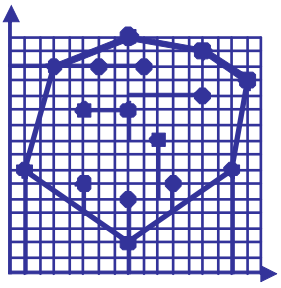
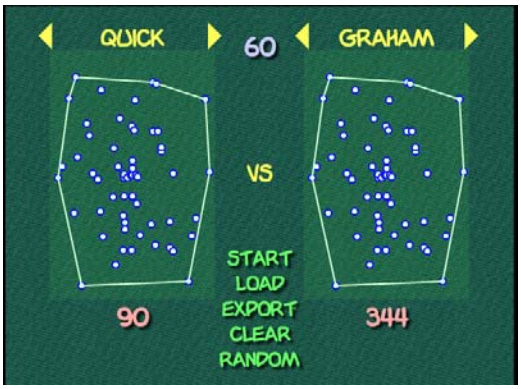


Fig. 4. Implementation of basic algorithm.

Algorithm	Complexity	Discovered By
Brute Force	$O(n^4)$	
Graham Scan	$O(n \log n)$	Graham, 1972
Jarvis March	$O(nh)$	Jarvis, 1973
QuickHull	$O(nh)$	Eddy, 1977 ; Bykat, 1978
Div-and-Conq	$O(n \log n)$	Preparata & Hong, 1977
Monotone Chain	$O(n \log n)$	Andrew, 1979
Chan's algorithm	$O(n \log h)$	Timothy Chan, 1993



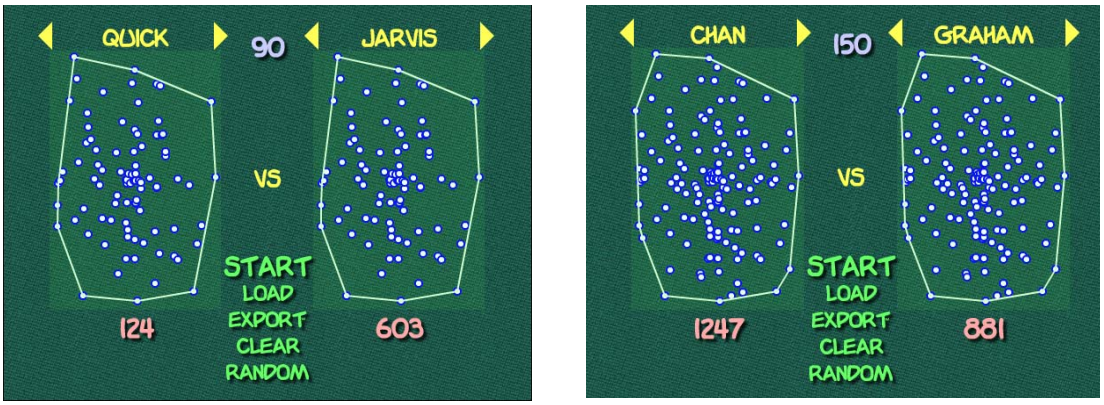


Fig. 5. The comparison of execution convex hull algorithms.

```
xx = -1:.05:1;
yy = abs(sqrt(xx));
[x,y] = pol2cart(xx,yy);
k = convhull(x,y);
plot(x(k),y(k),'r-',x,y,'b+')
```

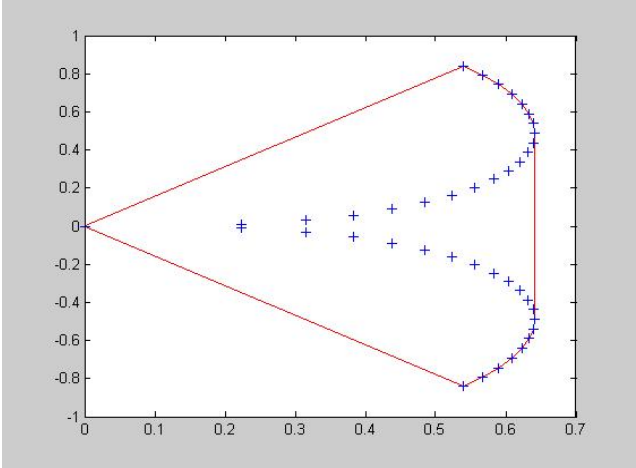


Fig. 6. Using *convhull* routine from Matlab.

3. LATTICE POINTS

From a geometric perspective when one tries to solve the diophantine equation $ax+by=c$, one has the objective of determining all the lattice points, if any, that lie on straight line described by the equation $ax+by=c$, $a,b,c \in Z$. In a given coordinate plane with coordinate axes x and y, a lattice point (x_1, y_1) is simply a point in which both coordinates x_1 and y_1 are integer.

3.1. AREA OF A POLYGON

It gives a polygon, in order specifying Cartesian coordinates of its vertices. To determine the polygon are if polygon is: a) convex; b) some.

Solution: We represent a polygon as a vector of points in plane, the mind in order vertices coordinates polygon:

```
#define Max 500
struct Point
 {float x,y;};
typedef Point Polygon[Max];
Polygon P;
int n;
```

```
void citire()
{ifstream f("poligon.dat ");f>>n;
 for(int i=1;i<=n;i++)
 f>>P[i].x>>P[i].y;f.close();
 P[0]=P[n];P[n+1]=P[1];}
```

Function `citire()` read n, number of vertices and vertice coordinates. We use two additional components `P[0]` and `P[n+1]`, which we initialize with `P[n]`, respectively `P[1]`, easy to calculate polygon area.

If the polygon is convex, the area is easily calculated, partitioning polygon into triangles, calculating area of each triangle and summing areas obtained.

Note that to determine the area of a convex polygon function was needed to determine the area of a triangle specified by its vertices. To calculate the area of a triangle, use Heron's formula. To determine the length of the sides, i used the function `dist()`determining the Euclidean distance between two points specified as a parameter:

```
float area_convexe(Poligon P,int n)
{float area=0;
 for(i=2;i<n;i++)
area+=triangle(P[1],P[i],P[i+1]);
return area;
}
```

```
float dist(Pointt a, Pointt b)
{return fsqrt((a.x-b.x)*(a.x-b.x)+
             (a.y-b.y)*(a.y-b.y));}
float triangle(Point a, Point b, Point c)
{float p,
 l1=dist(a,b);l2=dist(b,c);l3=dist(a,c);
 p=(l1+l2+l3)/2;
 return fsqrt(p*(p-l1)*(p-l2)*(p-l3));}
```

Area of a convex polygon can be written with the help of determinants:

$$S = \frac{1}{2} abs\left(\sum_{i=1,n} \begin{vmatrix} x_1 & y_1 & 1 \\ x_i & y_i & 1 \\ x_{i+1} & y_{i+1} & 1 \end{vmatrix} \right) \tag{3.1}$$

If the polygon is not necessarily convex, area calculation is complicated. To deduce the formula for calculating the area, we draw parallel to the Oy axis by each vertices polygon. Any two consecutive parallel together with sides of two vertices corresponding which are drawn and Ox axis form a rectangular trapezoid. Calculate trapezoid rectangular area that the area to sign:

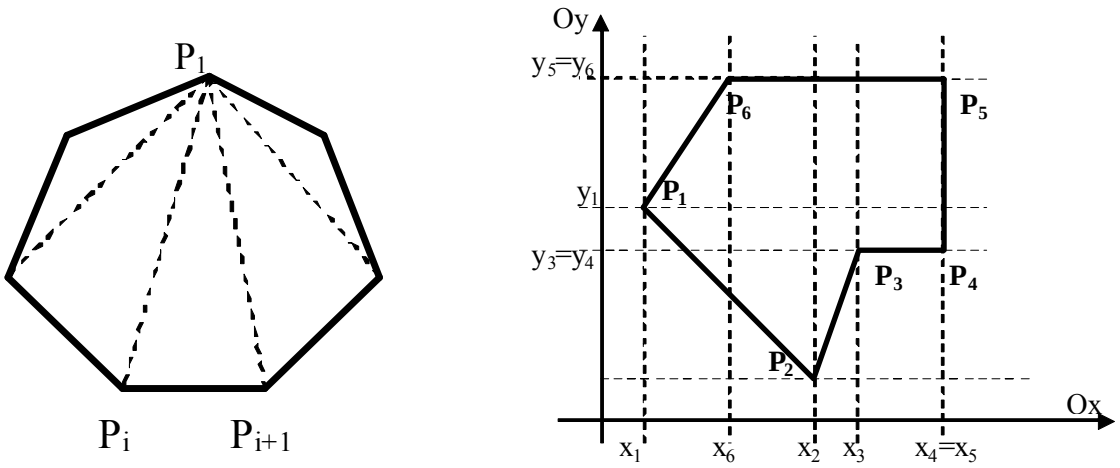
$$A_i = h_i \cdot (b_i + B_i)/2 = (x_{i+1} - x_i) \cdot (y_{i+1} + y_i)/2 \tag{3.2}$$

Calculating the sum of the sign areas of rectangular trapezoids, in absolute value, get some polygon area (see [3, 5]):

$$A = \left| \sum_{i=1,n} \left((x_{i+1} - x_i) \cdot (y_{i+1} + y_i)/2 \right) \right| \tag{3.3}$$

But it is known that P[0]= P[n] and P[n+1]= P[1]. Easier to calculate this value, we regroup terms, removing the factors x_i:

$$A = \left| \sum_{i=1,n} x_i \cdot (y_{i+1} - y_{i-1}) \right| / 2 \tag{3.4}$$



a) convexe polygon **b) same polygon**

Fig. 7. Area of a polygon.

```
float area_polygon(Polygon P, int n)
{float area=0;
 for(int i=1;i<=n;i++)
area+=P[i].x*(P[i+1].y-P[i-1].y);
return fabs(arie/2);}
```

In 1899, *Georg Alexander Pick* published one of his most beautiful theorems [1]. This theorem provided a formula for easily calculating the area of a *planar polygon P* whose vertices have integer coordinates. Such a polygon is called lattice polygon, since the points in plane with integer coordinates are sometimes called lattice points. In fact, Pick's theorem state that if *I* is the number of lattice points in interior of *P* and *B* is the number of lattice points on its boundary, then the area *A* of *P* is given by

$$A = I + \frac{B}{2} - 1 \tag{3.5}$$

The beauty of this formula stems from its simplicity and depth. It has been successfully explained to twelve-year-olds, and yet mathematicians are still researching some of its consequences today. Pick's formula (3.5) first to popular attention in 1969 (seventy years after Pick published it) in Steinhauss's book Mathematical Snaptshots [2].

3.2. APPLICATION 1

To determine the number of points which integer coordinates by passing segment determined by points $(0,0)$ and (N,M). For example $M = 9$ and $N = 15$ on the segment will be four points of integer coordinates.

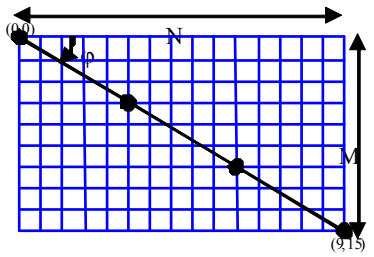


Fig. 8. Integer coordinates by passing segment determined by points $(0,0)$ and (N,M).

```
#include<iostream>
using namespace std;
int main()
{int N,M;
 cin>>N>>M;
 while(N!=M)
 {if(N>M)N=N-M;
  if(N<M)M=M-N;
 }
 cout<<M+1;
 return 0;}
```

Solution. We consider only cases where $0 \le N \le M$. If $N = 0$ then we obviously $M+1$ points on segment, so we have to resolve only if $1 \le N \le M$. Obviously any point (x, y) for our segment must satisfy equation $y = \frac{M}{N} \cdot x$. This equation is a natural number solution if Mx is divisible by N, it follows that if $D = gcd(N, M)$, then x is divisible by $\frac{N}{D}$, so x form is $k \cdot \frac{N}{D} \Rightarrow y = k \cdot \frac{M}{D}$. To that point (x, y) belong to segment, be as $0 \le x \le N$ and $0 \le y \le M$, so $0 \le k \le D$. So the number of points on a segment from $(0,0)$ to (N, M) is equals with the *greatest common divisor* of the numbers M and N to add 1. Complexity of the best algorithm is $O(\log(M+N))$.

3.3. APPLICATION 2

It gives a triangle with integer coordinates vertices. Required to determine the number of points of integer coordinates that are within the triangle or on its sides.

For example a triangle with coordinates vertices $(1, 5), (5, 1)$ and $(6, 6)$ has 16 points inside and on the sides.

Solution. For a rectangle with sides parallel to coordinate axes of length M and width N we have that number of points strictly inside the rectangle is $(M-1)\times(N-1)$ and the number of points on the sides is $2M+2N$ (see figure below, a). If we note with B number of points on the sides and I number of points strictly inside a rectangle and with A area of the rectangle, note that (3.5 - *Pick's theorem*):

$$I + \frac{B}{2} - 1 = (M-1)\times(N-1) + M + N - 1 = M \times N = A \tag{3.6}$$

If rectangular triangles why have two catheti we see that this form is kept, obviously we have that

$$A = \frac{M \times N}{2} \tag{3.7}$$

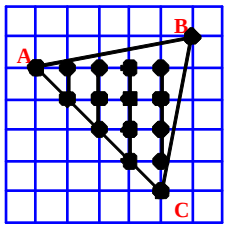


Fig. 9. The some triangle.

The hypotenuse of the triangle we have D points($D = gcd\,(M,N)+1$,but this time we are not interested in) so the number of points on the sides of triangle is

$$B = M + N + D - 1 \tag{3.8}$$

Number of points I inside triangle is the number of points inside the rectangle $(M-1)\times(N-1)$ minus the interior points of the diagonal $D-2$, result is then divided in two, here we have:

$$I = \frac{((M-1)\times(N-1)-D+2)}{2}$$

Make check

$$A = I + \frac{B}{2} - 1 = \frac{(M-1)\times(N-1)}{2} - \frac{D}{2} + 1 + \frac{M}{2} + \frac{N}{2} + \frac{D}{2} + \frac{1}{2} = \frac{M\times N}{2} \tag{3.9}$$

So, check and make such triangles. Now we look at the smallest rectangle containing a certain triangle, this rectangle can be divided into four triangles of which one is the initial and three rectangular triangles which catheti go along lattice lines. We see that our formula check and for certain triangles.

Denote with I_d the number of points inside the rectangle, A_d area of the rectangle and B_d the number of points on the sides of the rectangle, with I_1, I_2, I_3 the number of points within the three rectangular triangles, B_1, B_2, B_3 the number of points on the sides of three triangles, and A_1, A_2, A_3 will be areas of three triangles. Hence we have that:

$$I + \frac{B}{2} - 1 = I_d - I_1 - I_2 - I_3 - \frac{3B_1}{2} - \frac{3B_2}{2} - \frac{3B_3}{2} + \frac{3B_d}{2} + 3 - 1$$

Convenient grouping, we have:

$$\left(I_d - \frac{B_d}{2} - 1\right) - \left(I_1 + \frac{B_1}{2} - 1\right) - \left(I_2 - \frac{B_2}{2} - 1\right) - \left(I_3 - \frac{B_3}{2} - 1\right) = A_d - A_1 - A_2 - A_3 = A \tag{3.10}$$

So equality is valid for some triangles.

Now the triangle area with vertices $(x_1,y_1),(x_2,y_2),(x_3,y_3)$ one can easily find using *Heron's formula* or using the relationship with determinant developed after *Sarrus' rule*. Number of points on the sides of the triangle can be easily determined using the formula shown in the previous issue.

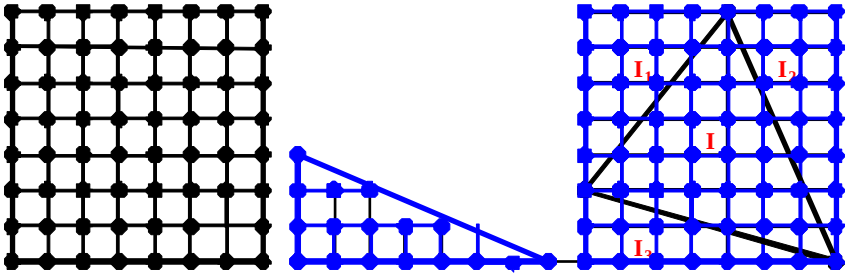


Fig. 10. The area of some triangle.

Now using the relationship $I+\frac{B}{2}-1=A$ we find the number I and problem is solved by turning number $I+B$. Solving complexity is $O(\log N)$ where the maximum number of points is N.

```
using namespace std;
int gcd(int x,int y){. . .}
int main()
{int xa,ya,xb,yb,xc,yc,N,M,B,B1,B2,B3,I,DoiA;
cout<<" coordinates of vertices A,B,C:";
cin>>xa>>ya>>xb>>yb>>xc>>yc;
DoiA=abs(xa*yb+xb*yc+xc*ya-xc*yb-xb*ya-xa*yc);
cout<<"aria= "<<DoiA<<endl;
N=abs(xa-xb);M=abs(ya-yb);B1=gcd(N,M)+1;
N=abs(xb-xc);M=abs(yb-yc);B2=gcd(N,M)+1;
N=abs(xc-xa);M=abs(yc-ya);B3=gcd(N,M)+1;
B=B1+B2+B3-3;I=(DoiA-B+2)/2;
cout<<"Result= "<<I+B;return 0;}
```

3.4. APPLICATION 3

It gives some polygonal surface; whose vertices are the positive integer coordinates in a cartesian system of axes. Is required to determine the number of points of integer coordinates, which are strictly within the given polygon. For example the polygon given by vertices $(0,4),(0,5),(2,7),(2,4),(6,6)$ and $(4,0)$ our results in the figure below (Fig. 11).

Solution. Formula $I+\frac{B}{2}-1=A$ is valid for some polygons. This is true for any polygon that has an internal diagonal, this diagonal is divided into two distinct polygons, if I_1 and I_2 denote the interior points of two polygons , B_1 and B_2 points on sides the two polygons, A_1 and A_2 areas of two polygons and D number of points of integer coordinates of the diagonal where equality would be satisfied if the two polygons have $B_1+B_2=B+2D-2$, $I=I_1+I_2+D-2$.

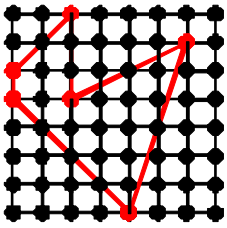


Fig. 11.

date.in
```
6
0 4
0 5
2 7
2 4
6 6
4 0
```

date.out
```
Area= 18
Border points = 14
Interior points = 12
```

Hence we have

$$A = A_1 + A_2 = I_1 + \frac{B_1}{2} - 1 + I_2 + \frac{B_2}{2} - 1 = I_1 + I_2 + \frac{B}{2} + D - 1 - 2 = I + \frac{B}{2} - 1.$$

Thus we can prove by induction that the relation is satisfied for any polygon. Area of a polygon can be calculated easily using (3.4) relation,where index points $N+1$ is the first point. Thus we get an algorithm of complexity $O(N \cdot log(MaxX + MaxY))$

```
int main()
{int n,k,A=0,B=0,I,U,V;
 struct Point{int x,y;};Point P[max];
 ifstream f("date.in");
 ofstream g("date.out");f>>n;
 for(k=1;k<=n;k++)f>>P[k].x>>P[k].y;
 f.close();P[0]=P[n];P[n+1]=P[1];
 for(k=1;k<=n;k++)
   A+=P[k].x*(P[k+1].y-P[k-1].y);
 A=abs(A)/2; g<<"Area= "<<A<<endl;
 for(k=1;k<=n;k++)
{U=abs(P[k].x-P[k+1].x);
 V=abs(P[k].y-P[k+1].y);
 B+=gcd(U,V)+1;} B-=n;
 g<<"Border points = "<<B<<endl;
 I=(2*(A+1)-B)/2;
 g<<"Interior points = "<<I<<endl;return 0;}
```

REFERENCES

[1] Pick, G. A., *Sitzungsberichte des Deutschen Naturwissenschaftlich-Medicinischen Vereins für Böhmen "Lotos" in Prag*., **19,** 311, 1899.

[2] Steinhaus, H., *Mathematical Snapshots*, Oxford University Press, 1969.

[3] Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., *Introduction to Algorithms*, Second Edition, MIT Press and McGraw-Hill, 947-957, 2001.

[4] Yanagisawa, H., *A Simple Algorithm for Lattice Point Counting in Rational Polygons*, Research Report, IBM Tokyo Research Laboratory, 2005.

[5] Iaglom, I. M., Iaglom, A. M., *Probleme neeelementare tratate elementar*, Ed. Tehnică, Bucureşti, 1962.

[6] Buşneag, D., Maftei, I., *Teme pentru cercurile şi concursurile de matematică ale elevilor*, Ed. Scrisul Românesc, Craiova, 1983.