August 25, 2005

RT0622

Computer Science; Mathematics 11 pages

Research Report

A Simple Algorithm for Lattice Point Counting in Rational Polygons

Hiroki Yanagisawa

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

A Simple Algorithm for Lattice Point Counting in Rational Polygons

Hiroki Yanagisawa

IBM Tokyo Research Laboratory
yanagis@jp.ibm.com

Abstract. We propose a simple algorithm for lattice point counting in rational polygons. A rational polygon is one whose vertices have rational coordinates. The algorithm decomposes a given polygon into right trapezoids and counts the number of lattice points in the right trapezoids. Each right trapezoid can be dissected into a rectangle and a right-angled triangle in the obvious way. The number of lattice points in the rectangle is easy to determine, and we find that a short recursive function computes the number of lattice points in the right-angled triangle. We also give an algorithm for counting lattice points on line segments.

1 Introduction

Counting lattice points (i.e. points with integer coordinates) in polygons, polyhedrons, and polytopes (i.e. higher dimensional analogues of polygons and/or polyhedrons) is a fundamental mathematical problem. This problem has been studied for a long time, and the literature [8] shows many efforts to date on this problem and various applications.

The simpler problem of counting lattice points in polygons whose vertices have *integer* coordinates was successfully solved by Pick in the nineteenth century. Pick presented a formula, which is widely known as Pick's Theorem [12], and the formula has been generalized to three- and higher dimensions [11].

Counting lattice points in polytopes whose vertices have *rational* coordinates has also been studied for a long time. When the number of dimensions is fixed, Barvinok [2, 3] showed that there is a polynomial-time algorithm for counting the lattice points in rational polytopes. For two-dimensional rational polygons, Beck and Robins [4] gave an efficient algorithm which uses a triangulation algorithm of polygons and a formula for counting lattice points in right-angled triangles.

However, the existing algorithms for rational coordinates are not useful in practice. Because the algorithm by Barvinok, which is proposed in 1993, is too hard to implement, it has not been implemented until 2004. Recently De Loera *et al.* have shown a first implementation of the algorithm, named `LattE` [9]. Even the algorithm by Beck and Robins is not so easy to use either, because the triangulation algorithm involves lengthy calculations and the formula for right-angled triangles is long.

In this paper, we propose a *simple and efficient* algorithm for counting lattice points in rational polygons. Our algorithm is much simpler than Beck and

Robins's algorithm [4], while the two algorithms have the same time complexity. Simplicity of algorithms is important, because simplicity makes easy to implement the algorithms and may lead to reduce the possibility of adding bugs in programs. It is also reasonable to expect that a simple program runs faster in practice, when two programs have the same time complexity in theory.

First, our algorithm decomposes a polygon into *right trapezoids* instead of triangles as in existing algorithm, including Beck and Robins'. The decomposition is done by a similar way to the algorithm for calculating the area of polygons (see details in section 4). This makes our algorithm simpler than the algorithm by Beck and Robins, which relies on a complex $O(n)$ time triangulation algorithm by Chazel [6], where n is the number of vertices of a given polygon. Although the triangulation algorithm could be replaced by simpler algorithms such as $O(n \log\log n)$ algorithm by Kirkpatrick *et al* [10] or $O(n \log^* n)$ randomized algorithm by Seidel [13], this would increase the time complexity. Note that our algorithm takes $O(n)$ time.

Second, we introduce a simple algorithm for counting lattice points in the right-angled triangles, which is used to count the lattice points in the decomposed right trapezoids. Our algorithm consists of a short recursive function (see Fig. 4 in Section 3) which counts the number of lattice points in a right-angled triangle, while the time complexity of our algorithm is no worse than existing algorithms including the one used in Beck and Robins'. The recursive function is developed by finding a small right-angled triangle such that the difference between the number of lattice points in the small triangle and that in the original triangle is easy to compute.

In order to count the number of lattice points in polygons, another non-trivial problem is counting the number of lattice points on rational segments, namely the boundary of the rational polygons. This problem is considerably easier than counting the number of lattice points in two-dimensional regions, but it is still non-trivial and has only recently been completely solved [5, 14]. Because both papers only gives formulas for counting the number of nonnegative integer solutions to the equation $ax + by = c$ with positive integers a, b, and c, it is not directly applicable for counting the number of lattice points on segments. We give an algorithm that can be used directly for this problem, and it has the same time complexity as the existing algorithms.

The rest of this paper is organized as follows. In Section 2, we propose an algorithm for counting lattice points on rational segments. In Section 3, we propose an algorithm for counting lattice points in right-angled triangles. The two algorithms proposed in Sections 2–3 are then used to construct an algorithm for counting lattice points in rational polygons in Section 4.

Notation In this paper, $\lfloor x \rfloor$ denotes the greatest integer not more than x, $\lceil x \rceil$ denotes the least integer not less than x, and $\{x\}$ denotes $x - \lfloor x \rfloor$.

2 Algorithm for Line Segments

In this section, we give an algorithm for calculating the number of lattice points on a given line segment. In order to construct the algorithm, we use the following notation and two existing lemmas.

Definition 1. *Let x_1, y_1, x_2, and y_2 be rational numbers. We define $L(x_1, y_1, x_2, y_2)$ as the number of lattice points on the line segment (including its endpoints) which connects the two points (x_1, y_1) and (x_2, y_2).*

Lemma 1. *[7] (Extended Euclidean Algorithm) Given two nonnegative integers a and b, there is an algorithm which computes their greatest common divisor (GCD(a, b)) as well as integers x and y such that $ax + by = GCD(a, b)$.*

Lemma 2. *[1] The linear Diophantine equation $ax + by = c$ has a solution if and only if c is divisible by d, where $d = GCD(a, b)$. Furthermore, if (x_0, y_0) is a solution of this equation, then the set of solutions of the equation consists of all integer pairs (x, y), where*

$$x = x_0 + \frac{b}{d}k, \quad y = y_0 - \frac{a}{d}k \qquad (k = \dots, -2, -1, 0, 1, 2, \dots).$$

We are now ready to construct our algorithm.

Theorem 1. *$L(x_1, y_1, x_2, y_2)$ can be calculated in $O(\max\{\log|x_1|, \log|y_1|, \log|x_2|, \log|y_2|\})$ steps.*

Proof. Without loss of generality, we can assume that $0 \leq x_1 \leq x_2$ and $y_1 \geq y_2 \geq 0$ (e.g. change (x_1, y_1) and (x_2, y_2) into ($-x_1$, y_1) and ($-x_2$, y_2) if $x_2 \leq x_1 < 0$ and $y_1 \geq y_2 > 0$). Because x_1, y_1, x_2, and y_2 are rational numbers, we can find nonnegative integers a, b, and c such that the line $ax + by = c$ passes through the two points (x_1, y_1) and (x_2, y_2). If $a = 0$ or $b = 0$, then $L(x_1, y_1, x_2, y_2)$ is trivially computable. Otherwise, by Lemma 1, we can compute the greatest common divisor d of a and b as well as integers p and q such that $ap + bq = d$. Because $(x, y) = ((c/d)p, (c/d)q)$ satisfies the equation $ax + by = c$, by Lemma 2, all integer solutions of the equation can be represented in the following form:

$$x = \frac{c}{d}p + \frac{b}{d}k, \quad y = \frac{c}{d}q - \frac{a}{d}k \qquad (k = \dots, -2, -1, 0, 1, 2, \dots).$$

Because $x_1 \leq x \leq x_2$, the following inequality must be satisfied: $(x_1 - (c/d)p)(d/b) \leq k \leq (x_2 - (c/d)p)(d/b)$. Hence,

$$L(x_1, y_1, x_2, y_2) = \left\lfloor \left(x_2 - \frac{c}{d}p\right)\frac{d}{b} \right\rfloor - \left\lceil \left(x_1 - \frac{c}{d}p\right)\frac{d}{b} \right\rceil + 1.$$

The most time-consuming part of the algorithm is the extended Euclidean algorithm, which takes $O(\max\{\log a, \log b\})$ steps. Therefore this lemma holds. □

3 Algorithm for Triangle

In this section, we give a short recursive function for counting the number of lattice points in a right-angled triangle. To construct the algorithm, we first show three lemmas (Lemmas 3–5).

Before showing the lemmas, we introduce the following notations.

Definition 2. *Let a, b, and c be positive integers. We define the right-angled triangle $T(a, b, c)$ as follows:*

$$T(a, b, c) = \{(x, y) \in \mathbf{R}^2 \mid ax + by \leq c, x > 0, y > 0\}.$$

Definition 3. *Let a, b, and c be positive integers. We define $N(a, b, c)$ as the number of lattice points in the triangle $T(a, b, c)$.*

By definition, it is easy to verify the following lemma.

Lemma 3. *Let a, b, and c be positive integers. Then, $N(a, b, c) = N(b, a, c)$.*

When $a = b$, it is also easy to verify the following lemma.

Lemma 4. *Let a and c be positive integers. Then $N(a, a, c) = \lfloor c/a \rfloor (\lfloor c/a \rfloor - 1)/2$.*

By the above two lemmas, it is sufficient to consider the case $a > b$. We show the following lemma.

Lemma 5. *Let a, b, and c be positive integers with $a > b$. Let $m = \lfloor c/a \rfloor$, $h = (c - am)/b$, $k = \lfloor (a - 1)/b \rfloor$, and $c' = c - b(km + \lfloor h \rfloor)$. Then the following equation holds:*

$$N(a, b, c) = N(a - bk, b, c') + km(m - 1)/2 + m\lfloor h \rfloor.$$

Proof. First, we express the area of the triangle $T(a, b, c)$ by using $N(a, b, c)$. When we look at the unit squares (squares consisting of four lattice points with area 1) in the triangle $T(a, b, c)$, the number of unit squares in $T(a, b, c)$ is equal to $N(a, b, c)$. Then, we divide the remainder of the triangle $T(a, b, c)$ into multiple trapezoids S_is and a triangle R (see Fig. 1). For each integer $0 < i \leq m$, we define trapezoid S_i as follows:

$$S_i = \{(x, y) \in \mathbf{R}^2 \mid ax + by \leq c, i - 1 \leq x \leq i, y \geq \lfloor (c - ai)/b \rfloor\}.$$

We define triangle R as follows:

$$R = \{(x, y) \in \mathbf{R}^2 \mid ax + by \leq c, x \geq m, y \geq 0\}.$$

Then, the area of the triangle $T(a, b, c)$ is given by

$$|T(a, b, c)| = N(a, b, c) + \sum_{i=1}^{m} |S_i| + |R|.$$

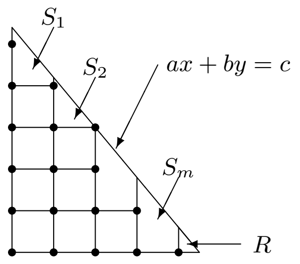

Fig. 1. Decomposition of the triangle $T(a, b, c)$

It is easy to verify that $|T(a, b, c)| = c^2/(2ab)$ and $|R| = h(c/a - m)/2$, noting that the triangle R consists of the three vertices $(m,\ h)$, $(m,\ 0)$, and $(c/a,\ 0)$. The area of the region S_i is also easily computable as follows (see Fig. 2):

$$\begin{aligned}|S_i| &= \frac{1}{2}\left(\left(\frac{c - a(i-1)}{b} - \left\lfloor \frac{c - ai}{b} \right\rfloor\right) + \left(\frac{c - ai}{b} - \left\lfloor \frac{c - ai}{b} \right\rfloor\right)\right) \\ &= \frac{1}{2}\left(\frac{a}{b} + 2\left\{\frac{c - ai}{b}\right\}\right).\end{aligned}$$

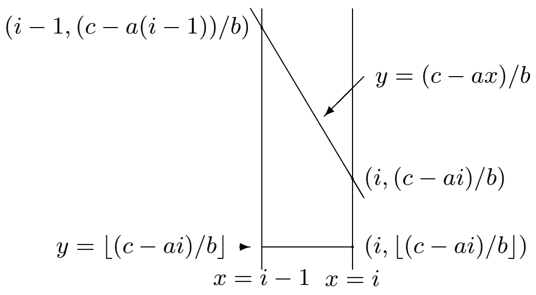

Fig. 2. The region S_i

Hence, $N(a, b, c)$ is given as follows:

$$\begin{aligned}N(a, b, c) &= |T(a, b, c)| - |R| - \sum_{i=1}^{m} |S_i| \\ &= \frac{c^2}{2ab} - \frac{h}{2}\left(\frac{c}{a} - m\right) - \frac{1}{2}\sum_{i=1}^{m}\left(\frac{a}{b} + 2\left\{\frac{c - ai}{b}\right\}\right) \\ &= \frac{cm}{2b} + \frac{h}{2}m - \frac{1}{2}\sum_{i=1}^{m}\left(\frac{a}{b} + 2\left\{\frac{c - ai}{b}\right\}\right).\end{aligned}$$

The third equality is due to the fact that $c-bh = am$, because the line $ax+by = c$ passes through the point (m, h).

It is easy to verify that $a-bk > 0$, since $k = \lfloor (a-1)/b \rfloor$. While the line $ax + by = c$ passes through the two points $(0,\, c/b)$ and $(m,\, h)$, the line $(a-bk)x+by = c'$ passes through the two points $(0,\, c'/b)$ and $(m,\, \{h\})$ (see Fig. 3). Therefore, by a similar argument to that used for $N(a, b, c)$, we have $N(a-bk, b, c')$ as follows:

$$N(a-bk, b, c') = \frac{c'm}{2b} + \frac{\{h\}}{2}m - \frac{1}{2}\sum_{i=1}^{m}\left(\frac{a-bk}{b} + 2\left\{\frac{c'-(a-bk)i}{b}\right\}\right).$$

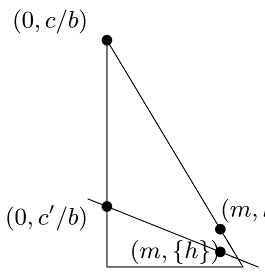

Fig. 3. The lines $ax + by = c$ and $(a - bk)x + by = c'$

We then have

$$\begin{aligned}
&N(a,b,c) - N(a-bk,b,c') \\
&= \frac{cm}{2b} - \frac{c'm}{2b} + \frac{h}{2}m - \frac{\{h\}}{2}m \\
&\quad -\frac{1}{2}\sum_{j=1}^{m}\left(\left(\frac{a}{b} + 2\left\{\frac{c-aj}{b}\right\}\right) - \left(\frac{a-bk}{b} + 2\left\{\frac{c'-(a-bk)j}{b}\right\}\right)\right) \\
&= \frac{cm}{2b} - \frac{(c-b(km+\lfloor h \rfloor))m}{2b} + \frac{1}{2}m\lfloor h \rfloor \\
&\quad -\frac{1}{2}\sum_{j=1}^{m}\left(\left(\frac{a}{b} + 2\left\{\frac{c-aj}{b}\right\}\right) - \left(\frac{a}{b} - k + 2\left\{\frac{c-aj}{b}\right\}\right)\right) \\
&= \frac{(km+\lfloor h \rfloor)m}{2} + \frac{1}{2}m\lfloor h \rfloor - \frac{1}{2}\sum_{j=1}^{m}k \\
&= \frac{k}{2}m(m-1) + m\lfloor h \rfloor.
\end{aligned}$$

Hence, we have proven the lemma. □

Now we can construct our algorithm.

Theorem 2. *Let a, b, and c be positive integers. Then $N(a,b,c)$ can be calculated in $O(\max\{\log a, \log b\})$ steps.*

Proof. Let $k = \lfloor (a-1)/b \rfloor$. It is easy to verify that $a - bk$ equals b when a is divisible by b and that $a - bk$ equals $a \bmod b$ when a is *not* divisible by b. Hence $0 < a - bk \leq b$ holds.

By Lemma 3, without loss of generality, we can assume that $a \geq b$ in the following arguments. By Lemma 4, Lemma 5, and the fact that $0 < a - bk \leq b$, we have the following algorithm "calcN" for calculating $N(a,b,c)$.

```
**Algorithm** calcN($a$, $b$, $c$)
    $a$, $b$, and $c$ are positive integers such that $a \geq b$
**begin**
    $m := \lfloor c/a \rfloor$
    **if** $a = b$ **then**
        **return** $m(m-1)/2$;
    **else**
        $k := \lfloor (a-1)/b \rfloor$; $h' := \lfloor (c - am)/b \rfloor$;
        **return** calcN$(b, a - bk, c - b(km + h')) + km(m-1)/2 + mh'$;
    **endif**
**end**
```

Fig. 4. Algorithm for counting lattice points in the right-angled triangle

Let us consider the running time of the computation of "calcN". You see that it is proportional to the number of recursive calls which the algorithm "calcN" makes. When a is *not* divisible by b, it makes a recursive call changing the parameters (a, b) into (b, $a \bmod b$). When a is divisible by b, it makes a (final) recursive call changing the parameters (a, b) into (b, b). Hence the number of recursive calls in the algorithm "calcN" is equal to the number of recursive calls in the Euclidean algorithm. Therefore, our algorithm takes $O(\log a)$ steps. $\square$

Note that the algorithm "calcN" deals with only integers. It also contributes simplicity of our algorithm.

4 Algorithm for Polygons

In this section, we give an algorithm for counting the number of lattice points in a given polygon. First we describe main ideas of our algorithm and then we show details of our algorithm.

A polygon is formally defined as follows. Let P be a two-dimensional rational polygon with n vertices $p_0, p_1, \ldots, p_n$ such that $p_n = p_0$, and its sides are the line segments connecting p_i and p_{i+1} (for $0 \leq i < n$). Each vertex p_i has coordinates $(x_i,\ y_i)$ where x_i and y_i are positive rational numbers. We assume that the vertices are given by clockwise ordering and that the polygon is not degenerate, namely, two sides can intersect only at a common endpoint and no point will occur as a vertex more than once. Note that the polygon is not necessarily convex.

We present main ideas of our algorithm here. For simplicity, suppose here that there are no lattice points on the sides of the polygon P and none of the vertices of P has integer coordinates. We count the number of lattice points by using a formula similar to that of the area of the given polygons. The area of the polygon P is computable by using the following formula:

$$\mathrm{area}(P) = \left| \sum_{i=1}^{n} \frac{(x_i - x_{i-1})(y_i + y_{i-1})}{2} \right|.$$

Let region D_i be the trapezoid whose vertices are $(x_{i-1},\ y_{i-1})$, $(x_{i-1},\ 0)$, $(x_i,\ 0)$, and $(x_i,\ y_i)$. Then the area of the polygon P can be rewritten as follows: $\mathrm{area}(P) = |\sum_{i=1}^{n} \mathrm{sgn}(x_i - x_{i-1})|D_i||$, where $\mathrm{sgn}(x)$ equals 1 if $x > 0$ otherwise -1. The number of lattice points in the polygon P is given as follows:

$$\mathrm{num}(P) = \left| \sum_{i=1}^{n} \mathrm{sgn}(x_i - x_{i-1})(\#\{(x, y) \in \mathbf{Z}^2 \mid (x, y) \in D_i\}) \right|.$$

The remaining problem is how to calculate the number of lattice points in the trapezoid D_i. All we have to do is to divide the trapezoid D_i into a rectangle and a right-angled triangle, because counting lattice points in rectangles are trivial and the number of lattice points in right-angled triangles can be calculated by the algorithm “calcN” in the previous section.

Based on the above ideas, we show our algorithm. If there are lattice points on the sides of the polygon P or some of the vertices of P has integer coordinates, we have to modify the above algorithm.

Before showing our algorithm, we introduce the following notation:

Definition 4. *Let x_{i-1}, y_{i-1}, x_i, and y_i be positive rational numbers such that $x_{i-1} \neq x_i$. We define $N(x_{i-1}, y_{i-1}, x_i, y_i)$ as the number of lattice points in the trapezoid D_i (including its boundary).*

First we show our algorithm for a *convex* polygon. The number of lattice points in the polygon P can be calculated by the following formula:

$$\mathrm{num}(P) = \left| \sum_{i=1}^{n} (N(D_i) - u(P, i)) \right|,$$

where

$$N(D_i) = \begin{cases} N(x_{i-1}, y_{i-1}, x_i, y_i) & \text{if } x_{i-1} < x_i, \\ -(N(x_i, y_i, x_{i-1}, y_{i-1}) - L(x_i, y_i, x_{i-1}, y_{i-1})) & \text{if } x_{i-1} > x_i, \\ 0 & \text{if } x_{i-1} = x_i. \end{cases}$$

and

$$u(P,i) = \begin{cases} \lfloor y_i \rfloor + 1 & \text{if } x_i \text{ is integer and } x_{i-1} < x_i < x_{i+1}, \\ -\lceil y_i \rceil & \text{if } x_i \text{ is integer and } x_{i-1} > x_i > x_{i+1}. \end{cases}$$

We can verify that the above formula correctly counts the number of lattice points in the polygon P. Note that the $u(P,i)$ is incorporated to avoid double counting. If $u(P,i)$ did not exist, the formula would count some points twice (e.g. open circles in Fig. 5).

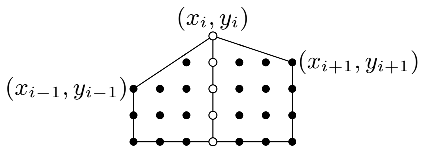

Fig. 5. Double counting on sides

For *nonconvex* polygons, we have to incorporate further modifications to the above formula. It is because some vertices of the polygon P may mistakenly counted as interior points (not vertices) of the polygon. The open circle in Fig. 6 is an example of this type of double counting. To avoid this, let $v(P,i)$ be defined as follows:

$$v(P,i) = \begin{cases} 1 & \text{if } (x_i,y_i) \in \mathbf{Z}^2, x_{i-1}, x_{i+1} < x_i, \text{ and } \begin{vmatrix} x_i - x_{i-1} & x_{i+1} - x_i \\ y_i - y_{i-1} & y_{i+1} - y_i \end{vmatrix} > 0, \\ \\ -1 & \text{if } (x_i,y_i) \in \mathbf{Z}^2, x_{i-1}, x_{i+1} > x_i, \text{ and } \begin{vmatrix} x_i - x_{i-1} & x_{i+1} - x_i \\ y_i - y_{i-1} & y_{i+1} - y_i \end{vmatrix} < 0; \end{cases}$$

where $|A|$ is the determinant of the matrix A, and let

$$\mathrm{num}(P) = \left| \sum_{i=1}^{n} (N(D_i) - u(P,i) - v(P,i)) \right|.$$

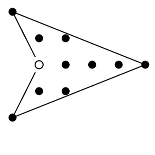

Fig. 6. Double counting on vertices

Thus we have a linear time (in the number of vertices of polygons) algorithm for counting the lattice points in polygons.

While our algorithm assumes that the vertices of the polygons are given by a clockwise ordering, you can also see that our algorithm calculates the number of lattice points in the interior of polygons (i.e. excluding the lattice points on its boundary) if the vertices of polygons are given by a counterclockwise ordering.

5 Concluding Remarks

We have shown that our simple algorithm calculates the number of lattice points in a given polygon. Because our algorithm does not need to store all of the vertices of the given polygon at a time, our algorithm uses smaller space than the counting algorithms which uses a triangulation algorithm. This also makes our algorithm easy to implement.

Our future work is to generalize our algorithm to three and higher dimensions. We hope that generalizing our algorithm leads to reduce the time complexity of existing algorithms.

Acknowledgement

We thank Takayuki Osogami for his useful comments on this paper.

References

1. G.E. Andrews, *Number Theory*, Dover Publications, Inc. New York, 1994.
2. A.I. Barvinok, A polynomial-time algorithm for counting integral points in polyhedra when the dimension is fixed, in *Proceedings of 34th Symposium on the Foundations of Computer Science (FOCS '93)*, pp. 566–572, IEEE Computer Society Press, New York, 1993.
3. A.I. Barvinok, Computing the Ehrhart polynomial of a convex lattice polytope, *Discrete and Computational Geometry*, Vol. 12, pp. 35–48, 1994.
4. M. Beck and S. Robins, Explicit and efficient formulas for the lattice point count inside rational polygons using Dedekind - Rademacher sums, *Discrete and Computational Geometry*, Vol. 27, No. 4, pp. 443–459, 2002.
5. M. Beck and S. Robins, A formula related to the Frobenius problem in two dimensions, Number Theory: New York Seminar, pp. 17–23, 2003.
6. B. Chazelle, Triangulating a simple polygon in linear time *Discrete and Computational Geometry*, Vol. 6, No. 5, pp. 485–524, 1991.
7. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, Mit press, 1990.
8. J.A. De Loera, The many aspects of counting lattice points in polytopes, Mathematische Semesterberichte manuscript (http://www.math.ucdavis.edu/˜ deloera /RECENT_WORK/semesterberichte.pdf).
9. J.A. De Loera, R. Hemmecke, J. Tauzer, and R. Yoshida, Effective lattice point counting in rational convex polytopes, *The Journal of Symbolic Computation*, Vol. 38, No. 4, pp. 1273–1302, 2004.
10. D.G. Kirkpatrick, M.M. Klawe, and R.E. Tarjan, Polygon triangulation in $O(n \log\log n)$ time with simple data-structures, in *Proceedings of the Sixth Annual Symposium on Computational Geometry*, pp. 34–43, 1990.

11. K. Kolodziejczyk, Hadwiger - Wills-type higher dimensional generalizations of Pick's theorem, *Discrete and Computational Geometry*, Vol. 24, No. 2-3, pp. 355–364, 2000.
12. G. Pick, Geometrisches zur Zahlenlehre *Sitzungber. Lotos, Naturwissen Zeitschrift* Prague, Vol. 19, pp. 311–319, 1899.
13. R. Seidel, A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons, *Computational Geometry: Theory and Applications*, Vol. 1, No. 1, pp. 51–64, 1991.
14. A. Tripathi, The number of solutions to $ax + by = n$, *The Fibonacci Quaterly*, Vol. 38, No. 4, pp. 290–293, 2000.