Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.T. Tromp

More computations on Gauss' lattice point problem

Computer Science/Department of Algorithmics & Architecture Report CS-R9017 May

More Computations on Gauss' Lattice Point Problem

J.T. Tromp
Centre for Mathematics and Computer Science (CWI)
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands
Email: tromp@cwi.nl

May 22, 1990

Abstract

We present a new method for counting the number of lattice points within increasingly larger circles, and use it to study the difference between these numbers and the area within the circles.

1985 Mathematics Subject Classification: 11P21.
CR Categories: G.2.1.
Keywords and Phrases: Lattice points, circle, highly composite, error function.
Note: This paper will be submitted for publication elsewhere.

1 Introduction

Recently, van de Lune and Wattel [7] studied the asymptotic behaviour of the function $P : \mathbb{R}^{\geq 0} \rightarrow \mathbb{N}$, defined as

$$P(t) = |\{(x,y) \in \mathbb{Z}^2 : x^2 + y^2 \leq t\}|.$$

If we denote by $D(r)$ the closed circular disc with center $(0,0)$ and radius r, then $P(t)$ is the number of lattice points in $D(\sqrt{t})$.

Furthermore, if we denote by $L(r)$ the union of the square regions

$$R(x,y) = [x - \frac{1}{2}, x + \frac{1}{2}] \times [y - \frac{1}{2}, y + \frac{1}{2}],$$

where (x,y) ranges over all lattice points in $D(r)$, then $P(t)$ equals the area of $L(\sqrt{t})$. ($L(r)$ is what $D(r)$ might look like on the bitmapped display of a computer, the $R(x,y)$ being single pixels.)

Note that $L(\sqrt{t})$ contains the disc $D(\sqrt{t} - \sqrt{1/2})$, while being contained in the disc $D(\sqrt{t} + \sqrt{1/2})$ (see Figure 1). This yields

$$\pi(t + \frac{1}{2} - \sqrt{2t}) \leq P(t) \leq \pi(t + \frac{1}{2} + \sqrt{2t}).$$

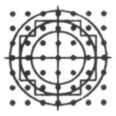

Figure 1: $L(\sqrt{7})$ and $D(\sqrt{7} + \rho)$ for $\rho = 0, \pm\sqrt{1/2}$.

Report CS-R9017
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Thus $P(t)$ can be approximated by the area of a disc with radius $r = \sqrt{t}$, which is $\pi r^2 = \pi t$. This leads us to the definition of the error function $E : \mathbb{R}^{\geq 0} \to \mathbb{R}$ as

$$E(t) = Area(L(\sqrt{t})) - Area(D(\sqrt{t})) = P(t) - \pi t.$$

In the early 1800's Gauss [1, 2] showed, by the simple geometric argument above, that

$$E(t) = O(t^{\frac{1}{2}}).$$

In general we can ask for the infimum of all α's satisfying

$$E(t) = O(t^{\alpha}).$$

Denoting this infimum by θ, we have that

$$\frac{1}{4} \leq \theta \leq \frac{7}{22}.$$

The lower bound was shown in 1915 by Landau [5], and, independently, by Hardy [3], while the upper bound was obtained in 1988 by Iwaniec and Mozzochi [6]. The value of $1/4$ cannot be achieved by any of the above α's, as in 1916, Hardy [4] showed that

$$E(t) = \Omega(t^{\frac{1}{4}}(\log t)^{\frac{1}{4}}).$$

For more historical data and further references the reader is invited to consult [7].

2 The brute force approach

The most obvious way of getting an impression of the behaviour of $P(t)$ and hence $E(t)$, is to compute these functions for large ranges of t. In the past, many of these attempts have suffered from

- lack of computing power, (the very first computations had to be done by hand)
- inefficient algorithms, and
- lack of completeness, i.e. considering only selected values of t.

In [7] van de Lune and Wattel have addressed these problems and their systematic computations (for t up to $6 \cdot 10^{10}$) led them to the conjecture that

$$E(t) = O(t^{\frac{1}{4}} \log t).$$

The aim of this paper is to extend the computations described in [7] using an improved method for counting lattice points in circles. This method can also be applied to curves other than circles.

3 Counting lattice points

We can write

$$P(t) = \sum_{r=0}^{\lfloor t \rfloor} p(r),$$

where

$$p(r) = |\{(x, y) \in \mathbb{Z}^2 : x^2 + y^2 = r\}|.$$

Our method consists of computing the $2r + 1$ values $p(r^2), \ldots, p((r+1)^2 - 1)$ in a single "pass". A pass starts at the point $(r, 0)$ and goes up, visiting lattice points between the circles of radius

r and $r+1$, until it crosses the line $x = y$. More precisely, the pass visits the points (x, y) with $0 \leq y \leq x$ and $r^2 \leq x^2 + y^2 < (r+1)^2$, which accounts for approximately one-eighth of all points between the two circles. Using an array of $2r+1$ counters $c[0], \ldots, c[2r]$, initialized to zero, we add 8 to the counter $c[x^2 + y^2 - r^2]$ upon visiting (x, y). (In the special case that $y = 0$ (x-axis) or $x = y$ (diagonal), we only add 4.) Simple arithmetic shows that after (x, y), the next point to be visited is either $(x, y+1)$, $(x-1, y)$ or $(x-1, y+1)$. The corresponding increments of the index $d = x^2 + y^2 - r^2$ are $2y+1$, $-2x+1$ or $2y - 2x + 2$ respectively.

By also summing those y for which we have decreased x in the last step, we obtain the number of lattice points in the region below the radius-r circle and bounded by the x-axis and the vertical line $x = r/\sqrt{2}$. From this it is easy to compute the value of $P(r-1)$, the number of lattice points strictly within the circle of radius r.

This method is a big win over previous ones, in which one pass along the circle segment was made for every value of t, requiring $2r+1$ passes to compute $P(t)$ for all $r^2 \leq t < (r+1)^2$. It follows that the speedup equals the size of the diameter of the circles considered.

For the purpose of estimating the order of growth of $E(t)$, only those t for which $E(t)$ achieves a new *best* extreme (larger in absolute value than all previous extremes) are of interest. In [7] this was exploited by observing that, since $P(t)$ is nondecreasing, the slope of $E(t) = P(t) - \pi t$ is no less than $-\pi$. Hence, given a computed $E(t)$, it takes a certain number of t-steps before $E(t)$ can become negative enough to beat the previously found best infimum. E.g., if we look for the infima of E, which at some $t > 10^{12}$ is positive, then the next 2900 t's can be safely skipped because we cannot improve on the negative extreme

$$E(914697101156-) \approx -9186.57 < 2900\pi.$$

A few remarks on notation are in order here. For $n \in I\!N$, we use $E(n-)$ to denote $\lim_{t \uparrow n} E(t)$, and similarly, $E(n+)$ to denote $\lim_{t \downarrow n} E(t)$. Actually, $E(n+) = E(n)$ because we defined $P(n)$ to be right-continuous, but we prefer the symmetrical notation with a plus or minus sign, so as not to depend on the choice of continuity. Note that the infima of E are of the form $E(n-)$, while the suprema are of the form $E(n+)$.

The "skipping" technique can also be used to locate best suprema, except that the scan of a given range should proceed backwards.

It turns out that the speedup achieved by skipping, at some point, is about equal to $1/\pi$ times the best extreme up to that point, which is still significantly less than a speedup equal to the size of the diameter. In fact, the first is roughly of the order $t^{\frac{1}{4}}$, whereas the second is of the order $t^{\frac{1}{2}}$. Hence, our method improves on the skipping method as much as the latter does on plain evaluation. The only drawback of our method is the excessive use of memory. Fortunately, it is possible to exchange memory for either speed or accuracy.

If there is not enough memory to store all $2r+1$ counters, but say, only one k-th of that, then we may proceed by subdividing the range $r^2, \ldots, (r+1)^2 - 1$ into k pieces, and making k passes instead of one. In this way, we lose a factor k in speed.

We can also trade memory for accuracy by merging k consecutive counters $c[ik], \ldots, c[(i+1)k-1]$ into one new counter $C[i]$. Instead of adding 8 to $c[d]$, we add 8 to $C[d \text{ div } k]$. The resulting inaccuracy is made explicit in the equation

$$\sum_{i=0}^{\lfloor \frac{t-k+1}{k} \rfloor} C[i] \leq P(t) \leq \sum_{i=0}^{\lfloor \frac{t}{k} \rfloor} C[i].$$

For the extremes of $E(t)$ this may lead to an inaccuracy of $k-1$ in the value of t where the extreme occurs, and an additional inaccuracy of $(k-1)\pi$ in the value of the extreme. We may use a memory friendly version of the procedure to locate *likely* best extremes (which occur with a very low frequency) and filter these through an exact version.

We close this section by considering the merits of our method for more general curves $f(x, y) = t$, with $f(\mathbb{Z}^2) \subseteq I\!N$. We can visit all lattice points in the region between two curves $f(x, y) = n$ and

$f(x,y) = n'$ to obtain values of the function

$$F(t) = |\{(x,y) \in \mathbb{Z}^2 : f(x,y) \leq t\}|$$

for all $n \leq t < n'$. The efficiency now depends on the ratio between $n' - n$ and the area between the two curves. The circle is a prime candidate for our method since it combines a small constant ratio ($2t : 2t\pi/8 = 1 : \pi/8$) with a very simple scheme for visiting lattice points.

4 The program

Appendix A gives a program in the language C embodying the procedure outlined above. Since the program is required to handle very large values and still be efficient, the sizes of the variables have been chosen very carefully. In order not to hurt performance, floating point arithmetic is avoided whenever possible. Fortunately all the computations on $P(t)$ can be done with integers. Furthermore, since no variable needs to take on negative values, we declare them as `unsigned`.

Most integers are declared of type `long`, which should be at least 32 bits wide, giving a range of about $0, \ldots, 4.3 \cdot 10^9$. This suffices for the radius, the coordinates, as well as for the index into the array of counters. However, 32 bits do not suffice for counting the lattice points below the circle-segment that we visit. We choose to represent this in two separate `long`'s, the most significant part `sumh` and the least significant part `suml`. Of course we have to do the carries ourselves (see the comment). A single variable `sum` can be used on machines supporting integers of at least 48 bits.

Because the array of counts can be very large, we need to make a single count as small as possible. To this end we will analyze for which values of t we can compute $P(t)$ using 16-bit counters (with range $0, \ldots, 65535$). It is well known that

$$p(n) = |\{(x,y) \in \mathbb{Z}^2 : x^2 + y^2 = n\}| = 4(d_1(n) - d_3(n)),$$

where $d_1(n)$ and $d_3(n)$ are the number of divisors of n of the form $4k+1$ and $4k+3$, respectively. If the prime factorization of n is $p_1^{e_1} \cdots p_k^{e_k}$, we thus have

$$p(n) \leq 4d_1(n) \leq 4(e_1+1) \cdots (e_k+1).$$

Call a number *highly d_1-composite* if it has more divisors of the form $4k+1$ than all of its predecessors (see also [8]). Table 1 shows a computer-generated list of all highly d_1-composite numbers with $n \leq 10^{16}$. From the last entry we infer that a counter with a range of $0, \ldots, 4 \cdot 1536$ suffices for all these n. The largest highly d_1-composite number below $2.7 \cdot 10^{22}$ is

$$5^4 13^2 17 \cdot 29 \cdot 37 \cdot 41 \cdot 53 \cdot 61 \cdot 73 \cdot 89 \cdot 97 \cdot 101,$$

with

$$p(n) = 4(4+1)(2+1)2^{10} = 61440 < 2^{16}.$$

Hence, 16 bits suffice for all practical purposes.

The main part of the progam, where the counts are computed, has been commented. Note that the variables `x` and `y` do not hold the coordinates themselves. Letting `x` hold $2x+1$ and `y` hold $2y-1$ proved to be the most convenient for making changes to `d` and `sum`. The visit ends when after decrementing x we find that (x,x) is below the radius-r circle. This will happen when $x = \lfloor (r-1)/\sqrt{2} \rfloor$. If we have visited the points $(x+1, x+1)$ and/or $(x+1, x+2)$, then the corresponding counters are properly adjusted.

Having computed the array of counts, the program proceeds to find new extremes of the error function. This part of the program requires the use of floating point arithmetic since the computations involve π.

$p(n)/4$	n	
1	1	$= 1$
2	5	$= 5^1$
3	25	$= 5^2$
4	65	$= 5^1 13^1$
6	325	$= 5^2 13^1$
8	1105	$= 5^1 13^1 17^1$
9	4225	$= 5^2 13^2$
12	5525	$= 5^2 13^1 17^1$
16	27625	$= 5^3 13^1 17^1$
18	71825	$= 5^2 13^2 17^1$
20	138125	$= 5^4 13^1 17^1$
24	160225	$= 5^2 13^1 17^1 29^1$
32	801125	$= 5^3 13^1 17^1 29^1$
36	2082925	$= 5^2 13^2 17^1 29^1$
40	4005625	$= 5^4 13^1 17^1 29^1$
48	5928325	$= 5^2 13^1 17^1 29^1 37^1$
64	29641625	$= 5^3 13^1 17^1 29^1 37^1$
72	77068225	$= 5^2 13^2 17^1 29^1 37^1$
80	148208125	$= 5^4 13^1 17^1 29^1 37^1$
96	243061325	$= 5^2 13^1 17^1 29^1 37^1 41^1$
128	1215306625	$= 5^3 13^1 17^1 29^1 37^1 41^1$
144	3159797225	$= 5^2 13^2 17^1 29^1 37^1 41^1$
160	6076533125	$= 5^4 13^1 17^1 29^1 37^1 41^1$
192	12882250225	$= 5^2 13^1 17^1 29^1 37^1 41^1 53^1$
216	53716552825	$= 5^2 13^2 17^2 29^1 37^1 41^1$
256	64411251125	$= 5^3 13^1 17^1 29^1 37^1 41^1 53^1$
288	167469252925	$= 5^2 13^2 17^1 29^1 37^1 41^1 53^1$
320	322056255625	$= 5^4 13^1 17^1 29^1 37^1 41^1 53^1$
384	785817263725	$= 5^2 13^1 17^1 29^1 37^1 41^1 53^1 61^1$
432	2846977299725	$= 5^2 13^2 17^2 29^1 37^1 41^1 53^1$
512	3929086318625	$= 5^3 13^1 17^1 29^1 37^1 41^1 53^1 61^1$
576	10215624428425	$= 5^2 13^2 17^1 29^1 37^1 41^1 53^1 61^1$
640	19645431593125	$= 5^4 13^1 17^1 29^1 37^1 41^1 53^1 61^1$
768	51078122142125	$= 5^3 13^2 17^1 29^1 37^1 41^1 53^1 61^1$
864	173665615283225	$= 5^2 13^2 17^2 29^1 37^1 41^1 53^1 61^1$
960	255390610710625	$= 5^4 13^2 17^1 29^1 37^1 41^1 53^1 61^1$
1024	286823301259625	$= 5^3 13^1 17^1 29^1 37^1 41^1 53^1 61^1 73^1$
1152	745740583275025	$= 5^2 13^2 17^1 29^1 37^1 41^1 53^1 61^1 73^1$
1280	1434116506298125	$= 5^4 13^1 17^1 29^1 37^1 41^1 53^1 61^1 73^1$
1536	3728702916375125	$= 5^3 13^2 17^1 29^1 37^1 41^1 53^1 61^1 73^1$
1728	12677589915675425	$= 5^2 13^2 17^2 29^1 37^1 41^1 53^1 61^1 73^1$

Table 1: Highly d_1-composite numbers.

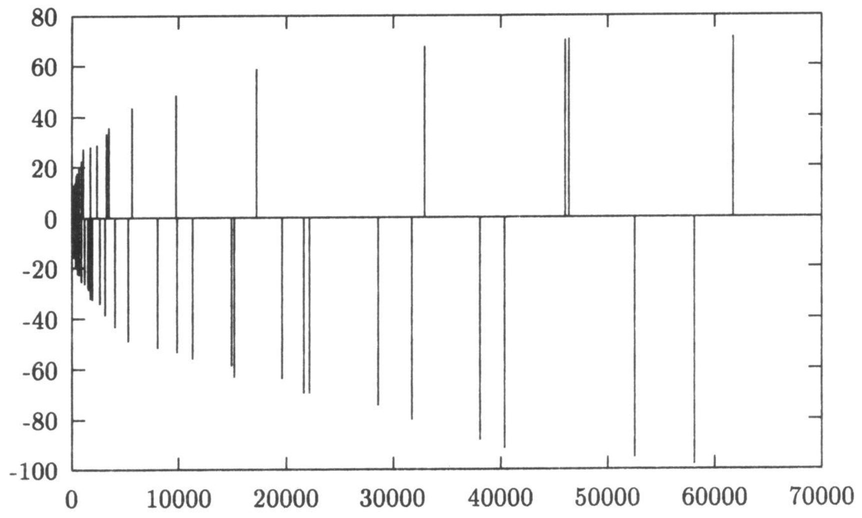

Figure 2: the first 64 best extremes of $E(t)$.

5 Computational results

We have run our program on two workstations (a DecStation 3100 and a SparcStation 1, yielding similar performance) over a period of some two months (mostly idle time). This resulted in a table of all best extremes in the range $0 \leq t \leq 2.29 \cdot 10^{12}$. Of the 1391 extremes in the table, 577 are positive, while the remaining 816 are negative. We represent this data in the following plots.

The first 64 best extremes are shown in Figure 2, which is a close up of Figure 3, showing all 1391 we found. Notice the similarity in shape, i.e. the behaviour of the function $E(t)$ close to the origin $(0, \ldots, 6 \cdot 10^4)$ is similar to that in a much wider range $(0, \ldots, 2 \cdot 10^{12})$. By plotting our data relative to the function $t^{\frac{1}{4}} \log t$, we can more clearly examine the order of growth of $E(t)$. As Figure 4 shows, this function is quite a sharp bound on $E(t)$ for the given range.

We finally examine the non-uniformity in the distribution of fractional radii at which the best extremes occur, as observed in [7]. To this end, we have plotted the points with x-coordinates as $t - \lfloor t \rfloor$ (Figure 5). We see positive extremes occurring roughly between .2 and .5, while the negative extremes are preponderantly concentrated between .9 and .0. These phenomena, if persistent, have yet to be accounted for.

6 Conclusion

The vast amounts of memory present on computers nowadays can be exploited to significantly speed up the counting of lattice points within circles (and other curves). Its application to Gauss' lattice point problem has not shown any deviations from the results put forward in van de Lune and Wattel [7]. Our computations thus provide additional support for their conjecture that

$$E(t) = O(t^{\frac{1}{4}} \log t).$$

Furthermore, no exceptions were found to their observations regarding the distribution of the fractional parts of radii at which best extremes occur.

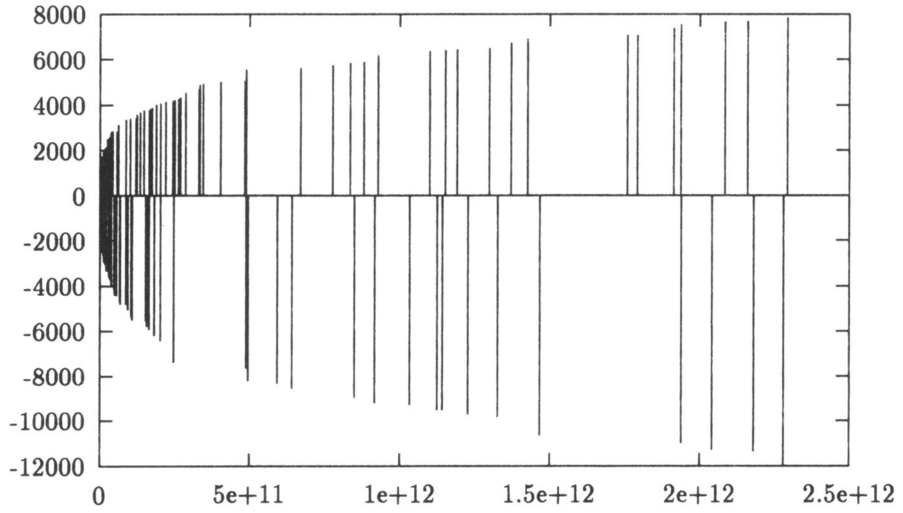

Figure 3: $E(t)$.

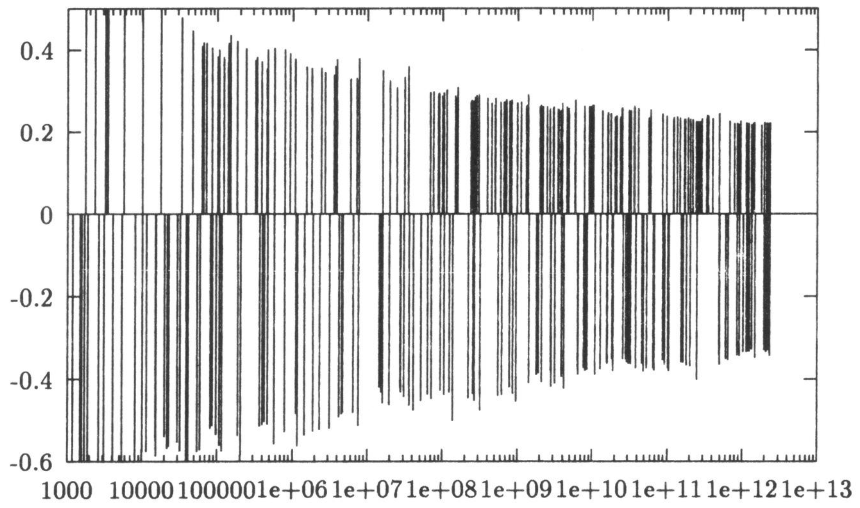

Figure 4: $E(t)/(1+t^{\frac{1}{4}}\log t)$.

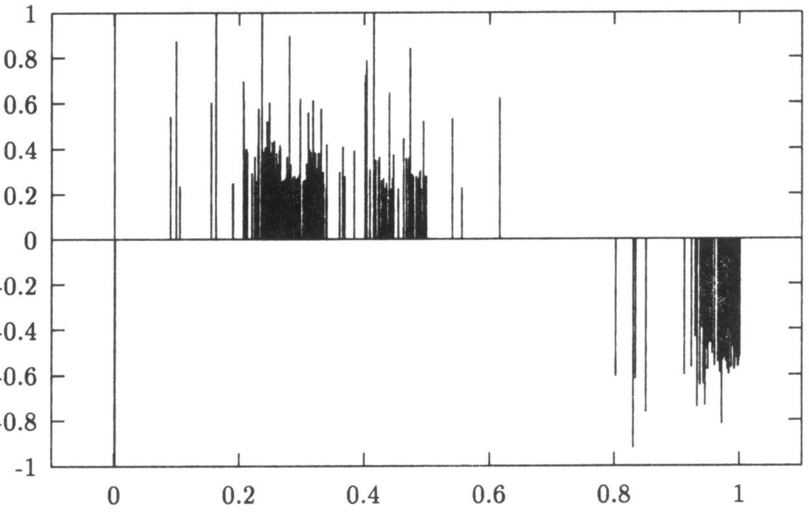

Figure 5: $E(t)/(1+t^{\frac{1}{4}}\log t))$ as a function of $t-\lfloor t\rfloor$.

A The program

```
#define PI      3.14159265358979323846
#define TWO32   4294967296.0
#define SQR(x)  ((double)x * (double)x)

typedef unsigned short ushort;
typedef unsigned long ulong;

char *calloc();

void
main(argc,argv)
int argc;
char *argv[];
{
  register ulong d,x,y,sumh,suml,r2;
  register ushort *count;
  ulong r,radius,stop;
  double inf,max,error,sqrr;

  if (argc < 4 || argc > 5) {
    printf("usage: %s radius inf max [range]\n",argv[0]);
    exit(0);
  }
  radius = atol(argv[1]);
  stop = radius + (argc == 5 ? atol(argv[4]) : 1);
  count = (ushort *)calloc(2 * stop - 1, sizeof(ushort));
  if (count == 0) {
    printf("conserve memory\n");
    exit(0);
  }
  inf = atof(argv[2]);
  max = atof(argv[3]);

  for (r = radius; r < stop; r++) {
    sumh = suml = 0;
    count[0] = 4;                     /* visit (r,0) */
    x = (r2 = r * 2) + 1;             /* At coordinates (X,Y) we have */
                                      /* x = 2*X + 1 and y = 2*Y - 1 */
    y = d = 1;                        /* d(r,1) = 1 */
    for (;;) {                        /* visit circle segment */
      do count[d] += 8;
      while ((d += (y += 2)) <= r2);  /* go up */
      if ((d -= (x -= 2)) >= y) {     /* go left */
        d -= y;
        y -= 2;                       /* and possibly down */
      }
      if (y >= x)
        break;                        /* we crossed the diagonal */
      sumh += (suml += y) < y;        /* DIY carry */
    }
```

```
      d += x;                                 /* go right */
      if (y > x) {                            /* if above diagonal */
        if (d <= r2)                          /* and in range */
          count[d] -= 8;                      /* then undo last count */
        d -= y;                               /* go down */
      }
                                              /* now we are on diagonal */
      if (d <= r2)                            /* if in range */
        count[d] -= 4;                        /* count only 4 times */

    sqrr = SQR(r);
    error = SQR(x) + 4 * (TWO32 * sumh + suml) - PI * sqrr;

    for (d = 0; d <= r2; count[d++] = 0) {
      if (x = count[d]) {
        if (error < inf)
          printf("%12.0lf- %20lf\n", sqrr + d, inf = error);
        if ((error += (double)x) > max)
          printf("%12.0lf+ %20lf\n", sqrr + d, max = error);
      }
      error -= PI;
    }
  }
  printf("%ld %lf %lf\n", stop, inf, max);
}
```

References

[1] C.F. Gauss, *De nexu inter multitudinem classium, in quas formae binariae secundi gradus distribuuntur, earumque determinantem*, Werke (1863), Vol. 2, pp. 269–291.

[2] C.F. Gauss, *Disquisitiones arithmeticae*, (German edition by H. Maser), 1886, p. 657.

[3] G.H. Hardy, *On the expression of a number as the sum of two squares*, Quart. J. Math., Oxford Ser. 46 (1915) pp. 263–283. Collected Papers, Vol. 2 (1967) pp. 243–283.

[4] G.H. Hardy, *On Dirichlet's divisor problem*, Proc. London Math. Soc. 15 (1916) pp. 1–25.

[5] E. Landau, *Über die Gitterpunkte in einem Kreise (II)*, Göttinger Nachr. (1915) pp. 161–171.

[6] H. Iwaniec, C.J. Mozzochi, *On the divisor and circle problems*, J. of Number Th. 29 (1988) pp. 60–93.

[7] J. van de Lune, E. Wattel, *Systematic computations on Gauss' lattice point problem*, CWI technical report AM-R90-08, 1990.

[8] S. Ramanujan, *Highly Composite Numbers*, Proc. London. Math. Soc. 2, XIV (1915), pp. 347–409.