MAXimal

added: 10 Jun 2008 18:15
Edited: 26 Apr 2012 1:46

Contents [hide]

Linear Diophantine equations in two variables

Diophantine equation with two unknowns has the form:

$$a \cdot x + b \cdot y = c,$$

where a, b, c- given integers, $x$and y- unknown integers.

Below we consider some classical problems on these equations: finding any solution, obtaining all solutions, finding the number of solutions and the solutions themselves in a certain interval, to find a solution with the least amount of unknowns.

Degenerate case

A degenerate case we immediately excluded from consideration when $a = b = 0$. In this case, of course, the equation has infinite number of random or solutions, or it has no solution at all (depending on whether $c = 0$or not).

Finding the solution

Find one of the solutions of the Diophantine equation with two unknowns, you can use the Extended Euclidean algorithm . Assume first that the numbers $a$and $b$non-negative.

Advanced Euclidean algorithm to specify a non-negative numbers $a$and $b$finds their greatest common divisor g, as well as such factors $x_g$and $y_g$that:

$$a \cdot x_g + b \cdot y_g = g.$$

It is argued that if $c$divisible by $g = \gcd(a, b)$, the Diophantine equation $a \cdot x + b \cdot y = c$has a solution; otherwise Diophantine equation has no solutions. This follows from the obvious fact that a linear combination of two numbers still can be divided by a common divisor.

Suppose that $c$is divided into g, then obviously performed:

$$a \cdot x_g \cdot (c/g) + b \cdot y_g \cdot (c/g) = c,$$

ie one of the solutions of the Diophantine equation are the numbers:

$$\begin{cases} x_0 = x_g \cdot (c/g), \\ y_0 = y_g \cdot (c/g). \end{cases}$$

We have described the decision in the case where the number $a$and $b$non-negative. If one of them or both are negative, then we can proceed as follows: take their modulus and apply them Euclid's algorithm, as described above, and then found to change the sign $x_0$and $y_0$the present symbol numbers $a$and b, respectively.

Implementation (recall here, we believe that the input data $a = b = 0$are not allowed):

```
int gcd (int a, int b, int & x, int & y) {
        if (a == 0) {
                x = 0; y = 1;
                return b;
        }
        int x1, y1;
        int d = gcd (b%a, a, x1, y1);
        x = y1 - (b / a) * x1;
        y = x1;
        return d;
}

bool find_any_solution (int a, int b, int c, int & x0, int & y0, int & g) {
        g = gcd (abs(a), abs(b), x0, y0);
        if (c % g != 0)
                return false;
        x0 *= c / g;
        y0 *= c / g;
        if (a < 0)     x0 *= -1;
```

```
        if (b < 0)   y0 *= -1;
        return true;
}
```

Getting all the solutions

We show how to obtain all the other solutions (and there are an infinite number) of the Diophantine equation, knowing one of the solutions (x_0, y_0).

Thus, suppose $g = \gcd(a, b)$, and the numbers x_0, y_0satisfy the condition:

$$a \cdot x_0 + b \cdot y_0 = c.$$

Then we note that, by adding to the $x_0$number $b/g$and at the same time taking away $a/g$from y_0, we do not disturb the equality:

$$a \cdot (x_0 + b/g) + b \cdot (y_0 - a/g) = a \cdot x_0 + b \cdot y_0 + a \cdot b/g - b \cdot a/g = c.$$

Obviously, this process can be repeated any number, ie all numbers of the form:

$$\begin{cases} x = x_0 + k \cdot b/g, \\ y = y_0 - k \cdot a/g, \end{cases} \quad k \in Z$$

are solutions of the Diophantine equation.

Moreover, only the number of this type are solutions, ie we describe the set of all solutions of the Diophantine equation (it turned out to be infinite if not imposed additional conditions).

Finding the number of solutions and the solutions themselves in a given interval

Given two segments $[min_x; max_x]$and $[min_y; max_y]$, and you want to find the number of solutions $(x, y)$of the Diophantine equation lying in these segments, respectively.

Note that if one of the numbers $a, b$is zero, then the problem has at most one solution, so these cases in this section, we exclude from consideration.

First, find a solution with the minimum appropriate x, ie $x \ge min_x$. To do this, first find any solution to the Diophantine equation (see para. 1). Then get out of it the solution with the least $x \ge min_x$- for this we use the procedure described in the preceding paragraph, and will increase / decrease x, until it is $\ge min_x$, and thus minimal. This can be done , considering a coefficient with which this conversion must be applied to obtain a minimum number greater than or equal to min_x. Denote found $x$through $lx1$.

Similarly, we can find an appropriate solution with a maximum $x = rx1$, ie .

Then move on to the satisfaction of restrictions y, ie consideration of the segment $[min_y; max_y]$. The method described above will find a solution with the minimum $y \ge min_y$and maximum solution $y \le max_y$. Denote $x$the coefficients of these solutions through $lx2$and , respectively.

Cross the line segments $[lx1; rx1]$and $[lx2; rx2]$; denote the resulting cut through $[lx; rx]$. It is argued that any decision which $x$the coefficient is in $[lx; rx]$- any such decision is appropriate. (This is true in virtue of the construction of this segment: we first met separately restrictions $x$and $y$getting two segments, and then crossed them, having an area in which both conditions are satisfied.)

Thus, the number of solutions will be equal to the length of this interval, divided by (since $x$the coefficient may be changed only) plus one.

We give implementation (it is difficult to obtain because it requires carefully consider the cases of positive and negative coefficients $a$and b)

```
void shift_solution (int & x, int & y, int a, int b, int cnt) {
        x += cnt * b;
        y -= cnt * a;
}

int find_all_solutions (int a, int b, int c, int minx, int maxx, int miny, int maxy) {
        int x, y, g;
        if (! find_any_solution (a, b, c, x, y, g))
                return 0;
        a /= g;  b /= g;

        int sign_a = a>0 ? +1 : -1;
        int sign_b = b>0 ? +1 : -1;

        shift_solution (x, y, a, b, (minx - x) / b);
        if (x < minx)
                shift_solution (x, y, a, b, sign_b);
```

```
        if (x > maxx)
                return 0;
        int lx1 = x;

        shift_solution (x, y, a, b, (maxx - x) / b);
        if (x > maxx)
                shift_solution (x, y, a, b, -sign_b);
        int rx1 = x;

        shift_solution (x, y, a, b, - (miny - y) / a);
        if (y < miny)
                shift_solution (x, y, a, b, -sign_a);
        if (y > maxy)
                return 0;
        int lx2 = x;

        shift_solution (x, y, a, b, - (maxy - y) / a);
        if (y > maxy)
                shift_solution (x, y, a, b, sign_a);
        int rx2 = x;

        if (lx2 > rx2)
                swap (lx2, rx2);
        int lx = max (lx1, lx2);
        int rx = min (rx1, rx2);

        return (rx - lx) / abs(b) + 1;
}
```

Also it is easy to add to this realization the withdrawal of all the solutions found: it is enough to enumerate $x$in a segment $[lx; rx]$ with a step by finding for each of them corresponding $y$directly from Eq $ax + by = c$.

Finding solutions in a given interval with the least amount of x + y

Here on $x$and $y$should also be imposed any restrictions, otherwise the answer will almost always be negative infinity.

The idea of the solution is the same as in the previous paragraph: first find any solution to the Diophantine equation, and then apply this procedure in the previous section, we arrive at the best solution.

Indeed, we have the right to do the following transformation (see. Previous paragraph):

$$\begin{cases} x' = x + k \cdot (b/g), \\ y' = y - k \cdot (a/g), \end{cases} \quad k \in Z.$$

Note that the sum of changes as follows:

$$x' + y' = x + y + k \cdot (b/g - a/g) = x + y + k \cdot (b - a)/g.$$

Ie if $a < b$it is necessary to choose the smallest possible value k, if $a > b$it is necessary to choose the largest possible value k.

If $a = b$we can not improve the solution - all solutions will have the same amount.

Problem in online judges

List of tasks that can be taken on the subject of Diophantine equations with two unknowns:

- SGU # 106 **"The Equation"** [Difficulty: Medium]

3 Комментариев e-maxx Войти

Лучшее вначале Поделиться Избранный

Присоединиться к обсуждению...

gavriil9999@mail.ru • 2 года назад

, а здесь gcd это что вообще не понятны выводы с переменными ,лучше бы на конкретных примерах
вот например берем 3x+5y=7 и как тут найти без перебора одну пару целых чисел x и y

1 • Ответить • Поделиться ›

Rima → gavriil9999@mail.ru • 11 месяцев назад

dumayu ne mojesh "без перебора".

• Ответить • Поделиться ›

Ivan • 2 года назад

А не нужно еще добавить
if(lx > rx)
return 0;

• Ответить • Поделиться ›

Подписаться Добавь Disqus на свой сайт