Added: Jun 10 2008 18:15
edited 26 Apr 2012 1:46

Linear Diophantine equations with two variables

Content [hide]

Diophantine equation with two unknowns has the form:

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

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

Below we consider several classical problems for these equations: finding any solution, obtaining all the solutions, finding the number of solutions and the solutions themselves in a certain segment, finding the solution with the smallest sum of unknowns.

Degenerate case

We will immediately exclude one degenerate case from consideration: when $a = b = 0$. In this case, it is clear that the equation has either infinitely many arbitrary solutions, or it has no solutions at all (depending on whether $c = 0$ or not).

Finding one solution

To find one of the solutions of the Diophantine equation with two unknowns, you can use the Extended Euclidean algorithm . Suppose first that the numbers a and b nonnegative.

Advanced Euclidean algorithm from the given 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 $g = \gcd(a, b)$, then the Diophantine equation $a \cdot x + b \cdot y = c$ has a solution; otherwise, the Diophantine equation has no solutions. The proof follows from the obvious fact that a linear combination of two numbers must still be divided by their common divisor.

Suppose it c is divided by g, then, obviously, it is executed:

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

those. One solution to 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 solution in the case when the numbers a and b non-negative. If one of them or both of them are negative, then you can do this: take them modulo and apply the Euclidean algorithm to them, as described above, and then change the sign of the numbers found x_0 and y_0 in accordance with this sign a and b respectively.

Implementation (recall, here we believe that the input data is $a = b = 0$ invalid):

```
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

Let us show how to obtain all the other solutions (and their infinite set) of the Diophantine equation, knowing one of the solutions (x_0, y_0).

So, let $g = \gcd(a, b)$, 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 simultaneously taking away $a/g$from y_0, we will not violate 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 as long as desired, i.e. 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 Diophantine equations.

Moreover, only numbers of this type are solutions, i.e. we described the set of all solutions of the Diophantine equation (it turned out to be infinite, if no additional conditions were imposed).

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

Let two segments $[min_x; max_x]$and be given $[min_y; max_y]$, and it is required to find the number of solutions of the (x, y)Diophantine equation lying in these segments, respectively.

Note that if one of the numbers $a, b$is zero, then the problem has no more than one solution; therefore, we exclude these cases in this section from consideration.

First we find a solution with the minimum appropriate x, i.e. $x \geq min_x$. To do this, we first find any solution to the Diophantine equation (see point 1). Then we get the solution with the smallest from it $x \geq min_x$- for this we use the procedure described in the previous paragraph, and we will reduce / increase $x$until it turns out $\geq min_x$, and at the same time be minimal. This can be done for $O(1)$, considering the coefficient with which to apply this transformation to get the minimum number, greater or equal min_x. Denote found $x$by $lx1$.

Similarly, you can find a solution with the maximum appropriate $x = rx1$, i.e. $x \leq max_x$.

Next, we proceed to the satisfaction of the restrictions on y, i. to the consideration of the segment $[min_y; max_y]$. In the manner described above, we find a solution with a minimum $y \geq min_y$, as well as a solution with a maximum $y \leq max_y$. We denote the $x$coefficients of these solutions by $lx2$and $rx2$respectively.

We intersect the segments $[lx1; rx1]$and $[lx2; rx2]$; denote the resulting segment by $[lx; rx]$. It is argued that any solution for which the $x$coefficient lies in $[lx; rx]$is any such solution is appropriate. (This is true due to the construction of this segment: first, we separately satisfied the constraints on $x$and y, having received two segments, and then crossed them, having obtained an area in which both conditions are satisfied.)

Thus, the number of solutions will be equal to the length of this segment, divided by $|b|$(since the $x$coefficient can change only by $\pm b$), and plus one.

Here is the implementation (it turned out to be quite complicated, since it is necessary to 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;
}
```

It is also easy to add to this implementation the derivation of all the solutions found: it is enough to go $x$through the interval $[lx;rx]$with a step $|b|$, finding for each of them the corresponding $y$directly from the equation $ax+by=c$.

Finding a solution in a given segment with the smallest sum x + y

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

The idea of the solution is the same as in the previous paragraph: first we find any solution of the Diophantine equation, and then, applying the procedure described in the previous paragraph, we come to the best solution.

Indeed, we have the right to perform the following conversion (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 in this case the amount $x+y$changes as follows:

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

Those. if $a<b$, then you need to choose the lowest possible value k; if $a>b$, then you need to choose the largest possible value k.

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

Tasks in online judges

The list of tasks that you can pass on the topic of Diophantine equations with two unknowns:

- SGU # 106 **"The Equation"** [difficulty: medium]