Last update: February 23, 2024 Translated From: e-maxx.ru

Linear Diophantine Equation

A Linear Diophantine Equation (in two variables) is an equation of the general form:

$$ax + by = c$$

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

In this article, we consider several classical problems on these equations:

- finding one solution
- finding all solutions
- finding the number of solutions and the solutions themselves in a given interval
- finding a solution with minimum value of $x + y$

The degenerate case

A degenerate case that need to be taken care of is when $a = b = 0$. It is easy to see that we either have no solutions or infinitely many solutions, depending on whether $c = 0$ or not. In the rest of this article, we will ignore this case.

Analytic solution

When $a \neq 0$ and $b \neq 0$, the equation $ax + by = c$ can be equivalently treated as either of the following:

$$\begin{aligned} ax &\equiv c \pmod{b}, \\ by &\equiv c \pmod{a}. \end{aligned}$$

Without loss of generality, assume that $b \neq 0$ and consider the first equation. When a and b are co-prime, the solution to it is given as

$$x \equiv ca^{-1} \pmod{b},$$

where a^{-1} is the modular inverse of a modulo b.

When a and b are not co-prime, values of ax modulo b for all integer x are divisible by $g = \gcd(a, b)$, so the solution only exists when c is divisible by g. In this case, one of solutions can be found by reducing the equation by g:

$$(a/g)x \equiv (c/g) \pmod{b/g}.$$

By the definition of g, the numbers a/g and b/g are co-prime, so the solution is given explicitly as

$$\begin{cases} x \equiv (c/g)(a/g)^{-1} \pmod{b/g}, \\ y = \frac{c-ax}{b}. \end{cases}$$

Algorithmic solution

To find one solution of the Diophantine equation with 2 unknowns, you can use the Extended Euclidean algorithm. First, assume that a and b are non-negative. When we apply Extended Euclidean algorithm for a and b, we can find their greatest common divisor g and 2 numbers x_g and y_g such that:

$$ax_g + by_g = g$$

If c is divisible by $g = \gcd(a, b)$, then the given Diophantine equation has a solution, otherwise it does not have any solution. The proof is straight-forward: a linear combination of two numbers is divisible by their common divisor.

Now supposed that c is divisible by g, then we have:

$$a \cdot x_g \cdot \frac{c}{g} + b \cdot y_g \cdot \frac{c}{g} = c$$

Therefore one of the solutions of the Diophantine equation is:

$$x_0 = x_g \cdot \frac{c}{g},$$

$$y_0 = y_g \cdot \frac{c}{g}.$$

The above idea still works when a or b or both of them are negative. We only need to change the sign of x_0 and y_0 when necessary.

Finally, we can implement this idea as follows (note that this code does not consider the case $a = b = 0$):

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

    x0 *= c / g;
    y0 *= c / g;
    if (a < 0) x0 = -x0;
    if (b < 0) y0 = -y0;
    return true;
}
```

Getting all solutions

From one solution (x_0, y_0), we can obtain all the solutions of the given equation.

Let $g = \gcd(a, b)$ and let x_0, y_0 be integers which satisfy the following:

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

Now, we should see that adding b/g to x_0, and, at the same time subtracting a/g from y_0 will not break the equality:

$$a \cdot \left(x_0 + \frac{b}{g}\right) + b \cdot \left(y_0 - \frac{a}{g}\right) = a \cdot x_0 + b \cdot y_0 + a \cdot \frac{b}{g} - b \cdot \frac{a}{g} = c$$

Obviously, this process can be repeated again, so all the numbers of the form:

$$x = x_0 + k \cdot \frac{b}{g}$$

$$y = y_0 - k \cdot \frac{a}{g}$$

are solutions of the given Diophantine equation.

Moreover, this is the set of all possible solutions of the given Diophantine equation.

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

From previous section, it should be clear that if we don't impose any restrictions on the solutions, there would be infinite number of them. So in this section, we add some restrictions on the interval of x and y, and we will try to count and enumerate all the solutions.

Let there be two intervals: $[min_x; max_x]$ and $[min_y; max_y]$ and let's say we only want to find the solutions in these two intervals.

Note that if a or b is 0, then the problem only has one solution. We don't consider this case here.

First, we can find a solution which has minimum value of x, such that $x \geq min_x$. To do this, we first find any solution of the Diophantine equation. Then, we shift this solution to get $x \geq min_x$ (using what we know about the set of all solutions in previous section). This can be done in $O(1)$. Denote this minimum value of x by l_{x1}.

Similarly, we can find the maximum value of x which satisfies $x \leq max_x$. Denote this maximum value of x by r_{x1}.

Similarly, we can find the minimum value of y $(y \geq min_y)$ and maximum value of y $(y \leq max_y)$. Denote the corresponding values of x by l_{x2} and r_{x2}.

The final solution is all solutions with x in intersection of $[l_{x1}, r_{x1}]$ and $[l_{x2}, r_{x2}]$. Let denote this intersection by $[l_x, r_x]$.

Following is the code implementing this idea. Notice that we divide a and b at the beginning by g. Since the equation $ax + by = c$ is equivalent to the equation $\frac{a}{g}x + \frac{b}{g}y = \frac{c}{g}$, we can use this one instead and have $\gcd(\frac{a}{g}, \frac{b}{g}) = 1$, which simplifies the formulas.

```
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);

    if (lx > rx)
        return 0;
    return (rx - lx) / abs(b) + 1;
}
```

Once we have l_x and r_x, it is also simple to enumerate through all the solutions. Just need to iterate through $x = l_x + k \cdot \frac{b}{g}$ for all $k \geq 0$ until $x = r_x$, and find the corresponding y values using the equation $ax + by = c$.

Find the solution with minimum value of $x + y$

Here, x and y also need to be given some restriction, otherwise, the answer may become negative infinity.

The idea is similar to previous section: We find any solution of the Diophantine equation, and then shift the solution to satisfy some conditions.

Finally, use the knowledge of the set of all solutions to find the minimum:

$$x' = x + k \cdot \frac{b}{g},$$

$$y' = y - k \cdot \frac{a}{g}.$$

Note that $x + y$ change as follows:

$$x' + y' = x + y + k \cdot \left(\frac{b}{g} - \frac{a}{g} \right) = x + y + k \cdot \frac{b - a}{g}$$

If $a < b$, we need to select smallest possible value of k. If $a > b$, we need to select the largest possible value of k. If $a = b$, all solution will have the same sum $x + y$.

Practice Problems

- Spoj - Crucial Equation
- SGU 106
- Codeforces - Ebony and Ivory
- Codechef - Get AC in one go
- LightOj - Solutions to an equation
- Atcoder - F - S = 1

Contributors:
Trung, Nguyen (40.81%) Jakob Kogler (36.77%) Oleksandr Kulkov (14.799999999999999%)
Mariia Mykhailova (3.59%) Bhuvnesh Jain (1.79%) boxlesscat (1.35%) traxex2 (0.45%) Manuel Pineda (0.45%)