内容：

- 退化的情况
- 找到一组解
- 找到所有解
- 给定区间内解的个数和所有解
- 找到 $x+y$ 值最小的一组解
- 练习题

一个线性不定方程（两个变量）的通用形式为：

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

其中 a, b, c 是已知整数，x, y 是未知整数

在这篇文章中我们将解决如下关于不定方程的常见问题：

- 找到一组解
- 找到所有解
- 给定区间内解的个数和所有解
- 找到 $x+y$ 值最小的一组解

退化的情况

一种需要注意的退化情况是 $a = b = 0$，此时如果 $c = 0$ 那么不定方程有无数组解，否则方程无解。在下文中我们将不再考虑这种退化情况

找到一组解

要找到有两个变量的不定方程的一组解，可以使用扩展欧几里得算法。首先，假设 a, b 非负，当我们使用扩展欧几里得算法时可以得到它们的最大公约数 g 和两个整数 x_g, y_g 使得：

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

如果 c 能被 $g = gcd(a, b)$ 整除，那么该不定方程有解，否则无解。对应的证明很直接，两个数线性组合的结果一定能被它们的最大公约数整除。

所以现在我们考虑 c 能被 g 整除的情况，那么我们有：

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

因此原不定方程有一组解：

$$x_0 = x_g \cdot \frac{c}{g}$$
$$y_0 = y_g \cdot \frac{c}{g}$$

上面的解法对 a, b 中有负数（不管1个还是2个）的情况仍然奏效，只需要对应改变 x_0, y_0 的符号即可

上述解法对应的实现如下，注意我们并没有考虑 a, b 同时为0的情况

```
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) {
        return false;
    }

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

找到所有解

从一组解 (x_0, y_0) 出发，我们可以找到不定方程的所有解

已知 $g = gcd(a, b)$ 和一组整数 x_0, y_0 使得：

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

我们可以看到在给 x_0 加上 $\frac{b}{g}$ 的同时，如果给 y_0 加上 $\frac{a}{g}$，等式依然成立：

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

显然，这个操作可以重复无数次，所以所有满足如下形式的 x_0, y_0 都是原不定方程的解：

$$x = x_0 + k \cdot \frac{b}{g}$$
$$y = y_0 - k \cdot \frac{a}{g}$$

事实上，这是原不定方程的所有解

给定区间内解的个数和所有解

从前一节可以看到，如果我们对解没有任何限制，那么解的个数是无穷的。所以在这一节中，我们对 x, y 的可行范围作出限制，并尝试给出限制情况下解的个数和所有解

假设我们要求 $x \in [min_x, max_x]$，$y \in [min_y, max_y]$。注意到如果 a 或 b 为0时方程有唯一解，下面的讨论将忽略这种情况

首先我们可以找到满足 $x \geq min_x$ 的解中 x 最小的一组解。为了找这组解，我们可以从任意一组解出发，在 $O(1)$ 的时间里得到大于等于 min_x 的最小 x，记作 l_{x_1}。类似的，我们还可以找到满足 $x \leq max_x$ 的最大 x 值，记作 r_{x_1}。同样我们可以找到最小和最大的 y 值，分别记作对应的 x 值为 l_{x_2} 和 r_{x_2}。最终的所有解的 x 应在区间 $[l_{x_1}, r_{x_1}]$ 和$[l_{x_2}, r_{x_2}]$ 的相交区间内，记作 $[l_x, r_x]$

下面是代码实现。注意到一开始我们将 a, b 都除去了最大公约数 g，因为方程 $ax + by = c$ 和 $\frac{a}{g}x + \frac{b}{g}y = \frac{c}{g}$ 是等价的，后者可以简化方程

```
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;
}
```

一旦我们有了 l_x 和 r_x，我们只需要从 $k=0$ 开始遍历 $x=l_x+k\cdot\frac{b}{g}$，直到 $x=r_x$ 就得到了区间内所有解的 x 值，进而求出每一组 y

找到 $x+y$ 值最小的一组解

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:

这里 x,y 同样要受到特定限制，否则答案会是无穷小。思路和上一节类似，我们先找到不定方程在区间内的解的形式：

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

写出 $x + y$：

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

当 $a < b$ 时，我们要找到最小的 k，当 $a > b$ 时，要找最大的 k，如果 $a = b$，那么 $x + y$ 的值是固定的

练习题

- ✅ Spoj - Crucial Equation
- ✅ SGU 106
- ✅ Codeforces - Ebony and Ivory
- ✅ Codechef - Get AC in one go
- ✅ LightOj - Solutions to an equation