GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: ECUACIÓN DIOFÁNTICA LINEAL

1. Introducción

Una ecuación diofántica lineal (en dos variables) es una ecuación de la forma general:

$$ax + by = c$$

dónde a, b, c se dan números enteros, y x, y son números enteros desconocidos. En esta guía, consideramos varios problemas clásicos sobre estas ecuaciones:

- encontrar una solución
- encontrando todas las soluciones
- encontrar el número de soluciones y las soluciones mismas en un intervalo dado
- encontrar una solución con un valor mínimo de $x + y$

2. Conocimientos previos

2.1. Inverso modular

El inverso modular es un concepto importante en teoría de números y criptografía que se relaciona con la aritmética modular. En términos simples, el inverso modular de un número a módulo n es otro número b tal que $(a * b) \equiv 1 (\text{mód} n)$. En otras palabras, el inverso modular de a módulo n es el número b que, al multiplicarlo por a y tomar el residuo módulo n, da como resultado 1.

La existencia de un inverso modular para un número a módulo n está condicionada a que a sea coprimo con n, es decir, que el máximo común divisor de a y n sea igual a 1 ($gcd(a, n) = 1$). En caso de que a y n no sean coprimos, no existe un inverso modular para a módulo n.

2.2. Números coprimos

Dos números se consideran coprimos si su máximo común divisor (gcd) es igual a 1. En otras palabras, dos números son coprimos si no tienen ningún factor primo en común, excepto el 1. Por ejemplo, los números 15 y 28 son coprimos porque su único factor primo común es 1.

3. Desarrollo

3.1. El caso degenerado

Un caso degenerado que debe ser atendido es cuando $a = b = 0$. Es fácil ver que o no tenemos soluciones o tenemos infinitas soluciones, dependiendo de si $c = 0$ o no. En el resto de este artículo ignoraremos este caso.

3.2. Solución analítica

Cuando $a \neq 0$ y $b \neq 0$, la ecuación $ax + by = c$ puede ser tratado de manera equivalente como cualquiera de los siguientes:

$$ax \equiv c \pmod{b},$$
$$by \equiv c \pmod{a}.$$

Sin pérdida de generalidad, supongamos que $b \neq 0$ y considere la primera ecuación. Cuando a y b son coprimos, la solución se da como:

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

dónde a^{-1} es el inverso modular de a módulo b.

Cuando a y b no son coprimos, los valores de ax módulo b para todo entero x son divisibles por $g = \gcd(a, b)$, por lo que la solución sólo existe cuando c es divisible por g. En este caso, una de las soluciones se puede encontrar reduciendo la ecuación por g:

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

Por la definición de g, los números a/g y b/g son coprimos, por lo que la solución se da explícitamente como:

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

3.3. Solución algorítmica

Para encontrar una solución de la ecuación diofántica con 2 incógnitas, puedes utilizar el algoritmo euclidiano extendido. Primero, suponga que a y b no son negativos. Cuando aplicamos el algoritmo euclidiano extendido para a y b, podemos encontrar su máximo común divisor g y 2 números x_g y y_g tal que:

$$ax_g + by_g = g$$

Si c es divisible por $g = \gcd(a, b)$, entonces la ecuación diofántica dada tiene solución; de lo contrario, no tiene solución. La prueba es sencilla: una combinación lineal de dos números es divisible por su divisor común.

Ahora se supone que c es divisible por g, entonces nosotros tenemos:

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

Por tanto una de las soluciones de la ecuación diofántica es:

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

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

La idea anterior todavía funciona cuando a o b o ambos son negativos. Sólo necesitamos cambiar el signo de x_0 y y_0 cuando sea necesario.

Finalmente, podemos implementar esta idea de la siguiente manera (tenga en cuenta que no considera el caso $a = b = 0$).

3.4. Obteniendo todas las soluciones

De una solución (x_0, y_0), podemos obtener todas las soluciones de la ecuación dada. Dejar $g = \gcd(a, b)$ y deja x_0, y_0 ser números enteros que satisfagan lo siguiente:

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

Ahora deberíamos ver que añadiendo b/g a x_0, y al mismo tiempo restar a/g de y_0 no romperá la igualdad:

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

Obviamente, este proceso se puede repetir nuevamente, así que todos los números de la forma:

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

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

son soluciones de la ecuación diofántica dada. Además, este es el conjunto de todas las posibles soluciones de la ecuación diofántica dada.

3.5. Encontrar el número de soluciones y las soluciones en un intervalo dado

De la sección anterior, debería quedar claro que si no imponemos ninguna restricción a las soluciones, habría un número infinito de ellas. Entonces, en esta sección, agregamos algunas restricciones en el intervalo de x y y, e intentaremos contar y enumerar todas las soluciones.

Sean dos intervalos: $[min_x; max_x]$ y $[min_y; max_y]$ y digamos que solo queremos encontrar las soluciones en estos dos intervalos.

Tenga en cuenta que si a o b es 0, entonces el problema sólo tiene una solución. No consideramos este caso aquí.

Primero, podemos encontrar una solución que tenga un valor mínimo de x, tal que $x \geq min_x$. Para hacer esto, primero encontramos cualquier solución de la ecuación diofántica. Luego, cambiamos esta solución para obtener $x \geq min_x$ (usando lo que sabemos sobre el conjunto de todas las soluciones en la sección anterior). Esto se puede hacer en $O(1)$. Denota este valor mínimo de x por l_{x1}.

De manera similar, podemos encontrar el valor máximo de x que satisface $x \leq max_x$. Denota este valor máximo de x por r_{x1}.

De manera similar, podemos encontrar el valor mínimo de y, $(y \geq min_y)$ y valor máximo de y $(y \leq max_y)$. Denota los valores correspondientes de x por l_{x2} y r_{x2}.

La solución final son todas las soluciones con x en la intersección de $[l_{x1}, r_{x1}]$ y $[l_{x2}, r_{x2}]$. Denotemos esta intersección por $[l_x, r_x]$.

A continuación en la sección de implementación se muestra el código que implementa esta idea. Note que dividimos a y b al principio por g. Desde la ecuación $ax + por = c$ es equivalente a la ecuación $\frac{a}{g}x + \frac{b}{g}y = \frac{c}{g}$, podemos usar este en su lugar y tener $\gcd(\frac{a}{g}, \frac{b}{g}) = 1$, lo que simplifica las fórmulas.

Una vez que tengamos l_x y r_x, también es sencillo enumerar todas las soluciones. Sólo necesito iterar $x = l_x + k \cdot \frac{b}{g}$ para todos $k \geq 0$ hasta $x = r_x$ y encuentre el correspondiente y valores usando la ecuación $ax + por = c$.

3.6. Encuentre la solución con el valor mínimo de $x + y$

Aquí, x y y también es necesario darle alguna restricción; de lo contrario, la respuesta puede convertirse en infinito negativo.

La idea es similar a la sección anterior: encontramos cualquier solución de la ecuación diofántica y luego cambiamos la solución para satisfacer algunas condiciones.

Finalmente, utilice el conocimiento del conjunto de todas las soluciones para encontrar el mínimo:

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

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

Tenga en cuenta que $x + y$ cambiar de la siguiente manera:

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

Si $a < b$, necesitamos seleccionar el valor más pequeño posible de k. Si $a > b$, necesitamos seleccionar el mayor valor posible de k. Si $a = b$, todas las soluciones tendrán la misma suma $x + y$.

4. Implementación

4.1. C++

4.1.1. Encontrar una solución

```
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;
}
```

4.1.2. Encontrar el número de soluciones y las soluciones en un intervalo dado

```
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;
}
```

4.2. Java

4.2.1. Encontrar una solución

```
// retorna { gcd(a,b), x, y } tal que gcd(a,b) = a*x + b*y
public static long[] gcd_recursive(long a, long b) {
    if (b == 0) return a > 0 ? new long[]{a, 1, 0} : new long[]{-a, -1, 0};
    long[] r = gcd_recursive(b, a % b);
    return new long[]{r[0], r[2], r[1] - a / b * r[2]};
}

//En el ArrayList answer retorna {gcd(a,b),x0,y0}
public static boolean find_any_solution(long a, long b, long c,ArrayList<Long>
     answer) {
    long [] r = gcd_recursive(Math.abs(a), Math.abs(b));
    long g= r[0];
    long x0=r[1];
    long y0=r[2];
    if (c % g!=0) { return false;}
    x0 *= c / g;
    y0 *= c / g;
    if (a < 0) x0 = -x0;
    if (b < 0) y0 = -y0;
    answer = new ArrayList<>();
```

```
    answer.add(g); answer.add(x0);answer.add(y0);
    return true;
}
```

4.2.2. Encontrar el número de soluciones y las soluciones en un intervalo dado

```
public static long [] shift_solution(long x, long y, long a, long b, long cnt)
     {
    x += cnt * b;
    y -= cnt * a;
    return new long[]{x,y};
}

public static long find_all_solutions(long a, long b, long c, long minx, long
    maxx, long miny, long maxy) {
    long x, y, g;
    ArrayList<Long> t = new ArrayList<>();
    if (!find_any_solution(a, b, c, t)) return 0L;
    g = t.get(0); x = t.get(1); y = t.get(1);
    a /= g;
    b /= g;
    long sign_a = a > 0 ? +1 : -1;
    long sign_b = b > 0 ? +1 : -1;
    long [] r =shift_solution(x, y, a, b, (minx - x) / b);
    x = r[0]; y = r[1];
    if (x < minx){
       r= shift_solution(x, y, a, b, sign_b);
       x = r[0]; y = r[1];
    }
    if (x > maxx) return 0;
    long lx1 = x;
    r = shift_solution(x, y, a, b, (maxx - x) / b);
    x = r[0]; y = r[1];
    if (x > maxx){
       r=  shift_solution(x, y, a, b, -sign_b);
       x = r[0]; y = r[1];
    }
    long rx1 = x;
    r = shift_solution(x, y, a, b, -(miny - y) / a);
    x = r[0]; y = r[1];
    if (y < miny){
       r=  shift_solution(x, y, a, b, -sign_a);
       x = r[0]; y = r[1];
    }
    if (y > maxy) return 0;
    long lx2 = x;
    r = shift_solution(x, y, a, b, -(maxy - y) / a);
    x = r[0]; y = r[1];
```

```
    if (y > maxy){
        r=  shift_solution(x, y, a, b, sign_a);
        x = r[0]; y = r[1];
    }
    long rx2 = x;
    if (lx2 > rx2){
        long tmp = lx2; lx2 = rx2; rx2 = tmp;
    }
    long lx = Math.max(lx1, lx2);
    long rx = Math.min(rx1, rx2);
    if (lx > rx) return 0;
    return (rx - lx) / Math.abs(b) + 1;
}
```

5. Aplicaciones

Algunas de las aplicaciones más comunes de las ecuaciones lineales diofánticas son:

1. **Criptografía:** En criptografía, las ecuaciones diofánticas se utilizan en la construcción de algoritmos de cifrado y descifrado. Por ejemplo, en el algoritmo RSA, se utilizan ecuaciones diofánticas para encontrar los coeficientes x e y que satisfacen la relación de congruencia $ax \equiv 1 (\text{mód} b)$, donde a y b son enteros relacionados con las claves pública y privada.
2. **Teoría de números:** Las ecuaciones diofánticas son fundamentales en la teoría de números para estudiar propiedades de los números enteros. Por ejemplo, el teorema de Bezout establece que si a y b son enteros no nulos, entonces existen enteros x e y tales que $ax + by = \gcd(a, b)$, lo cual se puede resolver mediante una ecuación diofántica.
3. **Optimización:** En problemas de optimización combinatoria, las ecuaciones diofánticas pueden utilizarse para modelar restricciones que involucran números enteros. Por ejemplo, en el problema del transporte, donde se busca minimizar el costo total de transporte entre diferentes ubicaciones, las ecuaciones diofánticas pueden utilizarse para modelar las restricciones de oferta y demanda.
4. **Física y ciencias de la computación:** En física y ciencias de la computación, las ecuaciones diofánticas pueden utilizarse para modelar sistemas físicos o algoritmos computacionales que involucran cantidades discretas o enteras. Por ejemplo, en la teoría de grafos, las ecuaciones diofánticas pueden utilizarse para resolver problemas de flujo máximo o corte mínimo en redes.

En resumen, las ecuaciones lineales diofánticas tienen una amplia variedad de aplicaciones en diversos campos de las matemáticas y en disciplinas como la criptografía, la teoría de números, la optimización, la física y las ciencias de la computación. Su estudio y resolución son fundamentales para abordar problemas complejos que involucran cantidades enteras o discretas.

6. Complejidad

La complejidad temporal de resolver una ecuación diofántica depende en gran medida de la naturaleza de la ecuación y de los métodos utilizados para encontrar sus soluciones. Las ecuaciones diofánticas son ecuaciones polinómicas en las que se buscan soluciones enteras, es decir, soluciones donde todas las variables involucradas son números enteros.

En el caso de las implementaciones realizadas en esta guía la complejidad por la complejidad temporal del algoritmo extendido de Euclides para encontrar el máximo común divisor (gcd) de dos números a y b es O($\log(\text{mín}(a, b))$), donde $\log$ representa el logaritmo en base 2 y $\text{mín}(a, b)$ es el menor de los dos números a y b.

7. Ejercicios

A continuación una lista de ejercicios que se puede resolver aplicando los aspectos abordados en la presente guía:

- DMOJ - Resolviendo ecuaciones
- DMOJ - David's expeditions
- SPOJ - Crucial Equation
- SGU 106
- Codeforces - Ebony and Ivory
- Codechef - Get AC in one go
- LightOj - Solutions to an equation