GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: MÁXIMO COMÚN DIVISOR Y MÍNIMO COMÚN MULTIPLO (*GCD Y LCM*)

---

1. Introducción

Dados dos números enteros a y b podemos hallar de ambos el máximo común divisor (comunmente llamado gcd del inglés *greatest common divisor*) o el mínimo común multiplo (nombrado como lcm del inglés *least common multiple*). Del significado de estos dos conceptos y como hallarlos trata la presente guía.

2. Conocimientos previos

2.1. Números coprimos

En matemáticas, los números coprimos (números primos entre sí o primos relativos) son dos números enteros a y b que no tienen ningún factor primo en común. Dicho de otra manera, si no tienen otro divisor común más que 1 y -1. Equivalentemente son coprimos, si y solo si, su máximo común divisor (GCD) es igual a 1.

2.2. Descomposición en factores primos

Por el teorema fundamental de la aritmética, cada entero positivo tiene una única descomposición en números primos (factores primos).

3. Desarrollo

3.1. Máximo común divisor (*GCD*)

En matemática, se define el máximo común divisor (GCD) de dos o más números enteros al mayor número entero que los divide sin dejar resto.

Si a y b son números enteros distintos y si el número c es de modo que $c|a$ y a su vez $c|b$ donde $|$ es el símbolo de divisibilidad y para el primer caso se lee como que c divide a mientras en el segundo se interpreta como c divide a b , a este número c se denomina divisor común de los números a y b. Obsérvese que dos números enteros cualesquiera tienen divisores comunes. Cuando existen, únicamente, como divisores comunes 1 y -1 de los números a y b, estos se llaman primos entre sí o comprimos.

Un número entero c se llama máximo común divisor de los números a y b cuando:

1. c es divisor común de los números a y b
2. c es divisible por cualquier otro divisor común de los números a y b.

Lo anterior se define matemáticamente como :

$$\gcd(a,b) = \max_{k=1\ldots\infty\,:\,k|a\,\wedge k\,|b} k.$$

Autor: Luis Andrés Valido Fajardo
Email: luis.valido1989@gmail.com
Teléfono: 53694742

Cuando uno de los números es cero mientras el otro no lo es, el máximo común divisor por definción es de los dos números aquel que su valor sea distinto de cero. Cuando ambos son ceros el máximo común divisor no está definido (pudiera ser cualquier valor incluso un número arbitrariamente grande), pero nosotros podemos definir que es cero lo cual no da una simple regla: Cuando uno de los números es cero el maximo común es divisor es el otro número.

Algunas de las propiedades del gcd son:

1. Si $gcd(a, b) = d$ entonces $gcd(\frac{a}{d}, \frac{b}{d}) = 1$
2. Si m es un entero, $gcd(m * a, m * b) = |m| * gcd(a, b)$
3. Si p es un número primo, entonces $gcd(p, m) = p$ o bien $gcd(p, m) = 1$
4. Si $m = n * q + r$, entonces $gcd(m, n) = gcd(n, r)$

Los métodos más utilizados para el cálculo del máximo común divisor de dos números son:

1. **Usando el mínimo común múltiplo:** El máximo común divisor puede ser calculado usando el mínimo común múltiplo. Si a y b son distintos de cero, entonces el máximo común divisor de a y b se obtiene mediante la siguiente fórmula, que involucra el mínimo común múltiplo (lcm) de a y b.

$$gcd(a, b) = \frac{a * b}{lcm(a, b)}$$

   Esta variante tiene que conocer de antemano la forma de hallar el mínimo común multiplo de a y b que hasta este momento desconocemos por tanto esta vía queda descartada por el momento.

2. **Por descomposición en factores primos:** El máximo común divisor de dos números puede calcularse determinando la descomposición en factores primos de los dos números y tomando los factores comunes elevados a la menor potencia, el producto de los cuales será el GCD. En la práctica, este método solo es operativo para números pequeños tomando en general demasiado tiempo calcular la descomposición en factores primos de dos números cualquiera, esto último hace que esta vía sea también descartada.

3. **Usando el algoritmo de Euclides:** Un método más eficiente es el algoritmo de Euclides, que utiliza el algoritmo de la división junto al hecho que el GCD de dos números también divide al resto obtenido de dividir el mayor entre el más pequeño.

4. **GCD binario:** El algoritmo GCD binario es una optimización del algoritmo euclidiano normal. La parte lenta del algoritmo normal son las operaciones de módulo. Operaciones de módulo, aunque las vemos como $O(1)$, son mucho más lentos que las operaciones más simples como la suma, la resta o las operaciones bit a bit. Así que sería mejor evitarlos.

3.1.1. Algoritmo de Euclides

El algoritmo se describió por primera vez en los *Elementos de Euclides* (alrededor del 300 a.c), pero es posible que el algoritmo tenga orígenes incluso anteriores.

Originalmente, el algoritmo de Euclides se formuló de la siguiente manera: resta el número más pequeño del más grande hasta que uno de los números sea cero. De hecho, si g divide a a y b, también divide a $a-b$. Por otro lado, si g divide a $a-b$ y b, entonces también divide a $a = b+(a-b)$, lo que significa que los conjuntos de los divisores comunes de $\{a, b\}$ y $\{b, a-b\}$ coinciden.

Tenga en cuenta que a sigue siendo el número más grande hasta que b se le resta al menos $\left\lfloor \frac{a}{b} \right\rfloor$ veces. Por lo tanto, para acelerar las cosas, $a - b$ se sustituye por $a - \left\lfloor \frac{a}{b} \right\rfloor b = a \bmod b$. Entonces el algoritmo se formula de una manera extremadamente simple:

$$\gcd(a, b) = \begin{cases} a, & \text{si } b = 0 \\ \gcd(b, a \bmod b), & \text{en otro caso.} \end{cases}$$

3.1.2. GCD binario

Resulta que puede diseñar un algoritmo GCD rápido que evite las operaciones de módulo. Se basa en algunas propiedades:

- Si ambos números son pares, entonces podemos factorizar un dos de ambos y calcular el GCD de los números restantes: $\gcd(2a, 2b) = 2\gcd(a, b)$.
- Si uno de los números es par y el otro es impar, entonces podemos quitarle el factor 2 al par: $\gcd(2a, b) = \gcd(a, b)$ si b es impar.
- Si ambos números son impares, restar un número del otro no cambiará el GCD: $\gcd(a, b) = \gcd(b, ab)$

3.2. Mínimo común múltiplo (*LCM*)

En matemáticas, el mínimo común múltiplo (abreviado lcm), de dos o más números naturales es el menor número natural que es múltiplo común de todos ellos (o el ínfimo del conjunto de los múltiplos comunes). Este concepto ha estado ligado históricamente con números naturales, pero se puede usar para enteros negativos o enteros gaussianos.

Partiendo de dos o más números y por descomposición en factores primos, expresados como producto de factores primos, su mínimo común múltiplo será el resultado de multiplicar todos los factores comunes y no comunes elevados a la mayor potencia, por ejemplo el lcm de 72 y 50 será:

$72 = 2^3 3^2$

$50 = 25^2$

Tomando los factores con su mayor exponente, tenemos que:

$lcm(72, 50) = 2^3 3^2 5^2 = 1800$

Conociendo el máximo común divisor de dos números, se puede calcular el mínimo común múltiplo de ellos, que será el producto de ambos dividido entre su máximo común divisor:

$$\text{lcm}(a,b) = \frac{a \cdot b}{\text{gcd}(a,b)}$$

Por lo tanto, LCM se puede calcular utilizando el algoritmo euclidiano o binario y tendrá la misma complejidad de tiempo.

Propiedades básicas:

- Si a es un entero, entonces $lcm(a,a) = a$.
- Cuando a y b son enteros, $lcm(a,b) = b$ si, sólo si b es múltiplo de a.
- $gcd(a,b) = lcm(a,b)$ si son iguales u opuestos.
- $lcm(a,b) = ab$ si, sólo si $gcd(a,b) = 1$.
- $lcm(a/d, b/d)$ = $lcm(m/a, m/b)$ donde $m = lcm$ y $d = gcd$.
- $lcm(ma,b) = m * lcm(a,b)$ si $gcd(lcm(a,b)/a, m) = 1$
- $lcm(a,b,c) = lcm(lcm(a,b), lcm(b,c))$
- $lcm(a,b,c)|abc$, donde $abc \neq 0$
- $lcm(a,b,c) = abcgcd(a,b,c)/gcd(a,b)gcd(b,c)gcd(c,d)$
- Si el producto de dos números lo dividimos por su máximo común divisor dicho cociente es el mínimo común múltiplo. A y B que descompuestos en números primos será $A = (p_1p_2)p_3p_4$ y $B = (p_1p_2)p_5p_6$ donde si gcd. es (p_1p_2) y el producto de $AB = (p_1p_2)p_3p_4(p_1p_2)p5p6$ donde vemos que p_1p_2 está repetido dos veces, luego si dividimos ese total por (p_1p_2) tendremos el total menor que contiene a A y B siendo su lcm.
- El mínimo común múltiplo de dos números, donde el menor divide al mayor, será el mayor. Es lógico ya que un múltiplo de ambos inferior al mayor sería imposible ya que no sería múltiplo del mayor.
- El mínimo común múltiplo de dos números primos es el total de su multiplicación. Esto es lógico ya que su máximo común divisor es 1.
- El mínimo común múltiplo de dos números compuestos será igual al cociente entre su producto y el m.c.d de ellos. Es evidente según la propiedad 1 de este tema.
- El máximo común divisor de varios números es un divisor del mínimo común múltiplo de tales números.
- Sea mZ el conjunto de los múltiplos del entero m, nZ el del entero n. Entonces el conjunto $nZ \bigcap mZ$ está formado por los múltiplos comunes de m y n; en otra notación es el conjunto $[m,n]Z$.

4. Implementación

4.1. C++

4.1.1. GCD

```
int gcdRecursiveEcluid (int a, int b) {
   if (b == 0) return a;
   else return gcdRecursiveEcluid (b, a % b);
}

int gcdTernaryEcluid (int a, int b) {
   return b ? gcdTernaryEcluid (b, a % b) : a;
}

int gcdNotRecursiveEcluid (int a, int b) {
   while (b) { a %= b; int t = a; a = b; b = t; }
   return a;
}

int gcdNotRecursiveEcluidOptimed(int a,int b) {
   while (b > 0){a = a % b;a ^= b; b ^= a; a ^= b; }
   return a;
}

int gcdBinary(int a, int b) {
   if (!a || !b) return a | b;
   unsigned shift = __builtin_ctz(a | b);
   a >>= __builtin_ctz(a);
   do{
      b >>= __builtin_ctz(b);
      if (a > b){int t = a; a = b; b = t;}
      b -= a;
   }while (b);
   return a << shift;
}
```

Tenga en cuenta que la optimización de gcd binaria generalmente no es necesaria, y la mayoría de los lenguajes de programación ya tienen una función GCD en sus bibliotecas estándar. Por ejemplo, C++17 tiene una función de este tipo $std :: gcd$ en el $numeric$ encabezado.

4.1.2. LCM

```
int lcm (int a, int b) { return a / gcd(a, b) * b; }
```

Tenga en cuenta que la implemantación generalmente no es necesaria, y la mayoría de los lenguajes de programación ya tienen una función LCM en sus bibliotecas estándar. Por ejemplo, C++17 tiene una función de este tipo $std :: lcm$ en el $numeric$ encabezado.

4.2. Java

4.2.1. GCD

```
public int gcdRecursiveEcluid(int a, int b) {
   if (b == 0) return a;
   else return gcdRecursiveEcluid(b, a % b);
}

public int gcdTernaryEcluid(int a, int b) {
   return b != 0 ? gcdTernaryEcluid(b, a % b) : a;
}

public int gcdNotRecursiveEcluid(int a, int b) {
   while (b != 0) {
      a %= b; int t = a; a = b; b = t;
   }
   return a;
}

public int gcdNotRecursiveEcluidOptimed(int a, int b) {
   while (b > 0) {
      a = a % b; a ^= b; b ^= a; a ^= b;
   }
   return a;
}
```

4.2.2. LCM

```
public int lcm (int a, int b) { return a / gcd(a, b) * b; }
```

5. Aplicaciones

El GCD se utiliza para simplificar fracciones, junto con el algoritmo de Euclides se emplea en la resolución de ecuaciones diofánticas lineales con dos incógnitas. El algoritmo de Euclides se emplea en el desarrollo de un número racional en fracción continuada.

El LCM se puede emplear para sumar o restar fracciones de distinto denominador, tomando el lcm de los denominadores de las fracciones, y convirtiéndolas en fracciones equivalentes que puedan ser sumadas.

6. Complejidad

El tiempo de ejecución del algoritmo de Euclides del GCD se estima mediante el teorema de Lamé, que establece una conexión sorprendente entre el algoritmo de Euclides y la sucesión de

Fibonacci:

Si $a > b \geq 1$ y $b < F_n$ para algunos n, el algoritmo de Euclides realiza como máximo $n - 2$ llamadas recursivas.

Además, es posible demostrar que el límite superior de este teorema es óptimo. Cuando $a = F_n$ y $b = F_{n-1}$, $gcd(a, b)$ funcionará exactamente $n - 2$ llamadas recursivas. En otras palabras, los números de Fibonacci consecutivos son la entrada del peor de los casos para el algoritmo de Euclides.

Dado que los números de Fibonacci crecen exponencialmente, obtenemos que el algoritmo de Euclides funciona en $O(\log \text{mín}(a, b))$.

Otra forma de estimar la complejidad es notar que $a \bmod b$ para el caso $a \geq b$ Por lo menos 2 veces menor que a, por lo que el número mayor se reduce al menos a la mitad en cada iteración del algoritmo.

7. Ejercicios propuestos

A continuación una lista de ejercicios que se pueden resolver apoyados en estos conceptos abordados en la presente guía:

- DMOJ - GCD en el Arreglo
- Codechef - GCD and LCM