Teoría de Números

Jaime Sebastian Chavarria Fuertes

May 11, 2025

1 Aritmética Modular

1.1 Teoría de Congruencias

Una congruencia es una relación de equivalencia entre enteros que se basa en sus restos al dividirse por un número dado.

Definición 1.1 (Congruencia)**.** *Decimos que dos números enteros* a *y* b *son congruentes módulo* n*, y escribimos* $a \equiv b \pmod{n}$*, que significa que:*

$$a\%n == b\%n$$

Tambien podemos decir que si $a \equiv b \pmod{n}$ *entonces se puede decir que:*

$$n|(a-b)$$

Es decir que n *divide a:* $(a-b)$

Ejemplo 1.2. $17 \equiv 2 \pmod{5}$*, ya que* $17 - 2 = 15$ *es divisible por 5.*

Propiedades:

– Si $a \equiv b \pmod{n}$, entonces $a + x \equiv b + x \pmod{n}$.

– Si $a \equiv b \pmod{n}$, entonces $a \cdot x \equiv b \cdot x \pmod{n}$.

– Si $a \equiv b \pmod{n}$ y $x \equiv y \pmod{n}$, entonces $a + x \equiv b + y \pmod{n}$.

– $a^n \equiv b^n \pmod{m}\ \forall n \in \mathbb{N}$

– Dispuesto a completar y/o agregar cosas

2 Fibonacci

La secuencia tiene un monton de propiedades interesantes. Algunas de ellas son:

Teorema 2.1 (Identidad de Cassini)**.**

$$F_{n-1}F_{n+1} - F_n^2 = (-1)^n$$

Teorema 2.2 (Regla de adicion)**.**

$$F_{n+k} = F_k F_{n+1} + F_{k-1} F_n$$

Teorema 2.3 (Aplicando la regla previa a $k = n$ conseguimos:)**.**

$$F_{2n} = F_n(F_{n+1} + F_{n-1})$$

Teorema 2.4. *Con la definicion anterior se puede probar que para cualquier entero positivo k, F_{nk} es multiplo de F_n*

Teorema 2.5. *El inverso del anterior tambien es verdad, si F_m es multiplo de F_n entonces m es multiplo de n*

Teorema 2.6 (Identidad del GCD)**.**

$$GCD(F_m, F_n) = F_{GCD(m,n)}$$

- Solo para saber, los numeros de Fibonacci son el peor caso en la entrada del gcd extendido

2.1 Formula de Binet

$$F_n = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}}$$

Podemos deducir la formula siguiente, redondeando hacia el entero mas proximo:

$$F_n = \frac{(\frac{1+\sqrt{5}}{2})^n}{\sqrt{5}}$$

Como estas dos formulas requieren alta precicion casi no tienen uso practico.

2.2 Calculando el F en $O(\log n)$

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$$

El codigo que lo implementa esta en la pagina xxx

Expandiendo la matriz para $n = 2k$

$$\begin{pmatrix} F_{2k+1} & F_{2k} \\ F_{2k} & F_{2k-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{2k} = \begin{pmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{pmatrix}^2$$

Encontramos estas ecuaciones simples:

$$F_{2k+1} = F_{k+1}^2 + F_k^2 \quad (1)$$

$$F_{2k} = F_k(F_{k+1} + F_{k-1}) = F_k(2F_{k+1} - F_k) \quad (2)$$

Gracias a estas ecuaciones tenemos el codigo en la pagina xx para calcular el fibonacci en tiempo logaritmico

2.3 Suma de $F_0 + F_1 + ... + F_n$

La suma resulta en $F_{n+2} - 1$

2.4 Es Fibonacci?

Un numero x pertenece a la secuencia de fibonacci si y solo si alguna (o ambas) expresiones siguientes son cuadrado(s) perfecto(s):

$$5x^2 + 4$$

$$5x^2 - 4$$

3 Funciones Aritméticas

Las funciones aritméticas son aquellas que se definen en los números enteros y tienen aplicaciones importantes en teoría de números.

3.1 Función ϕ de Euler

La función $\phi(n)$ cuenta el número de enteros positivos menores o iguales que n que son coprimos con n.

Dos numeros son "coprimos" cuando el $gcd(a, b) = 1$

La funcion *phi* hasta 21:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
$\phi(n)$	1	1	2	2	4	2	6	4	6	4	10	4	12	6	8	8	16	6	18	8	12

3.1.1 Propiedades

- Si $p_1, p_2, \ldots, p_k$ son los factores primos distintos de n, entonces:

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\ldots\left(1 - \frac{1}{p_k}\right)$$

- Si p es un numero primo, entonces el $gcd(p, q) = 1$ para todo $1 \leq q < p$ Entonces tenemos:

$$\phi(p) = p - 1.$$

- Si p es un numero primo y $k \geq 1$ entonces hay exactamente $\frac{p^k}{p}$ entre 1 y p^k que son divisibles por p Lo que nos da:

$$\phi(p^k) = p^k - p^{k-1}.$$

- Si a y b son coprimos, entonces:

$$\phi(ab) = \phi(a) \cdot \phi(b).$$

- De la propiedad de arriba se extiende que, si a es un número con una factorización prima:

$$a = p_1^{e_1} \cdot p_2^{e_2} \cdot \ldots \cdot p_k^{e_k}$$

  donde $p_1, p_2, \ldots, p_k$ son primos distintos y $e_1, e_2, \ldots, e_k$ son exponentes enteros positivos, entonces la función $\phi(a)$ se calcula como:

$$\phi(a) = \phi(p_1)^{e_1} \cdot \phi(p_2)^{e_2} \cdot \ldots \cdot \phi(p_k)^{e_k}$$

  Ojo, si ya tienes la criba con los primos, facil puedes precalcular la funcion phi de un numero, entonces con la factorizacion del numero en $\emptyset \log(n)$ podemos facil tener el resultado del la funcion ϕ

- Esta propiedad interesante fue obtenida por Gauss:

$$\sum_{d|n} \phi(d) = n$$

  Es decir la suma de todos los divisores de n.

  Por ejemplo, los divisores de 10 son $1, 2, 5, 10$. Entonces: $\phi(1)+\phi(2)+\phi(5)+\phi(10) = 1 + 1 + 4 + 4 = 10$

- Para $x \geq 3$ el valor de $\phi(x)$ siempre sera par, para $x = 1$ y $x = 2$ el valor de $\phi(x)$ es 1

- Por consecuencia para $x \geq 3$ el valor de $\phi(x)$ no sera primo. **(Comprobado solo hasta $x \leq 10^9$)**

3.1.2 Aplicacion

La propiedad mas famosa e importante es expresada en **el teorema de Euler:**

$$a^{\phi(m)} \equiv 1 \pmod m \quad \text{Si } a \text{ y } m \text{ son coprimos.}$$

Cuando m es primo, el teorema de Euler se convierte en el **pequeno teorema de Fermat**

$$a^{m-1} \equiv 1 \pmod m$$

El teorema de euler tiene aplicaciones practicas como **calcular el inverso multiplicativo modular**

Como inmediata consecuencia de esto sabemos que:

$$a^n \equiv a^{n \bmod \phi(m)} \pmod m$$

3.1.3 Exponenciacion x^y para un y demasaido grande

$$x^n \equiv x^{\phi(m)+[n \bmod \phi(m)]} \mod m$$

3.1.4 Teoría de Grupos

$\phi(n)$ es el orden del grupo multiplicativo módulo n, denotado como $(\mathbb{Z}/n\mathbb{Z})^\times$. Este es el grupo de unidades (elementos con inversos multiplicativos). Los elementos con inversos multiplicativos son precisamente aquellos coprimos con n.

****El orden multiplicativo de un elemento a módulo n, denotado como $\text{ord}_n(a)$, es el menor entero positivo $k > 0$ tal que:****

$$a^k \equiv 1 \pmod{n}.$$

$\text{ord}_n(a)$ es el tamaño del subgrupo generado por a. Por el Teorema de Lagrange, el orden multiplicativo de cualquier a debe dividir a $\phi(n)$.

Si el orden multiplicativo de a es exactamente $\phi(n)$ (el máximo posible), entonces a es una raíz primitiva y el grupo es cíclico por definición.

3.2 Suma y cantidad de divisores

3.2.1 Cantidad de divisores

Si la factorizacion de un numero n es $p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$ donde p_i son primos distintos, entonces el numero de divisores es:

$$d(n) = (e_1 + 1) \cdot (e_2 + 1) \cdots (e_k + 1)$$

4 Bitmasks

4.1 Bitwise Identities

- $x + y = (x \oplus y) + 2 \cdot (x \& y)$
- $x \oplus y = (x \mid y) - (x \& y)$
- $\sim x = -x - 1$ (complemento a dos)
- $(x \oplus y) \oplus y = x$
- $x \oplus x = 0$
- $x \& (x - 1)$ borra el bit más bajo prendido de x
- $x \& (-x)$ aísla el bit más bajo prendido
- x es potencia de 2 $\iff$ $x \& (x - 1) = 0$
- `__builtin_popcount(x)` cuenta los bits prendidos en x
- $x \% 2^k = x \& (2^k - 1)$

4.2 Identidades básicas:

- $x + y = (x \oplus y) + 2 \cdot (x\&y)$
- $x \oplus y = (x \mid y) - (x\&y)$
- $\sim x = -x - 1$
- $(x \oplus y) \oplus y = x$
- $x \oplus x = 0$
- $x \oplus 0 = x$

4.3 Propiedades de XOR

- XOR es asociativo: $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
- XOR es conmutativo: $a \oplus b = b \oplus a$
- $a \oplus a = 0 \quad \text{y} \quad a \oplus 0 = a$
- Si $a \oplus b = c$ entonces $a = b \oplus c$

4.4 Bits útiles para manipulación

- $x\&(x - 1)$: borra el bit más bajo prendido
- $x\& - x$: aísla el bit más bajo prendido
- $x \mid (x - 1)$: prende todos los bits a la derecha del más bajo prendido
- x es potencia de dos $\iff x\&(x - 1) = 0$
- $x\%2^k = x\&(2^k - 1)$
- $x \uparrow 1$ (o $x \ll 1$): multiplica por 2
- $x \downarrow 1$ (o $x \gg 1$): divide por 2 (sin signo)

4.5 Conteo de bits y máscaras

- `__builtin_popcount(x)`: número de bits prendidos
- `__builtin_ctz(x)`: cantidad de ceros a la derecha (trailing zeros)
- `__builtin_clz(x)`: cantidad de ceros a la izquierda (leading zeros)
- Total de subconjuntos de una máscara x: $2^{\texttt{popcount}(x)}$
- Recorrer todos los subconjuntos de x:

```
for (int sub = x; sub; sub = (sub - 1) & x)
    // usar sub
```

4.6 Extra (interesante para DP o teoría de bits)

- $(x \oplus y) + 2 \cdot (x\&y) = x + y$
- $x + y = x \mid y + (x\&y)$ (no exacto, pero útil en razonamiento)
- Para todo x, $x \oplus (x - 1)$ tiene todos los bits a la derecha del más alto en 1