GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: NÚMEROS COPRIMOS

1. Introducción

En Teoría de Números unos de los elementos que suelen ser usados en la confección de los problemas es los números coprimos. Sobre como saber si dos números son coprimos entre sí, saber la cantidad de números comprimos con un valor N y menores que este, generar la cantidad de coprimos para todos los valores hasta N. De los anteriores aspectos trata la siguiente guía.

2. Conocimientos previos

2.1. Numero primo

Número primo es cualquier número natural mayor que 1 cuyo únicos divisores posibles son el mismo número primo y el factor natural 1 .A diferencia de los números primos , los números compuestos son naturales que pueden factorizarse. Ejemplo de números primos son el 2, 3, 5, 7, 11, 13.

2.2. Máximo común divisor

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

2.3. Teorema chino del resto

El teorema chino del resto es un resultado sobre congruencias en teoría de números y sus generalizaciones en álgebra abstracta. Fue publicado por primera vez en el siglo III por el matemático chino Sun Tzu.

2.4. Criba de Eratóstenes

Es un algoritmo para encontrar todos los números primos en un segmento $[1; n]$ usando $O(n \log \log n)$ operaciones.

3. Desarrollo

3.1. Numeros comprimos

Dos números son coprimos si su máximo común divisor es igual 1 (1 se considera coprimo de cualquier número).

3.2. Saber si dos numeros son comprimos

Basados en la propia definición no sería dificil implementar un algoritmo que permita conocer si dos números a y b son coprimos basta con verificar si el maximo común entre ello es 1 para que sean coprimos cualquier otro valor significaría que no lo son.

3.3. Calcular la cantidad de comprimos de N menores que este

Para calcular la cantidad de números menores que N que sean coprimos con N podemos realizar una iteración de desde 1 hasta N y comprobar con cada número pero esta idea tendría una complejidad de $O(N \log(\text{mín}(i, N))$.

Otra idea sería asumir que en el rango de 1 a N todos los valores son coprimos con N y por tanto vamos a buscar aquellos que no lo son con la factorización en números primos de N y viendo por cada uno de ellos cuantos multiplos tienen menores que N que serían los numeros que no son coprimos con N. De esto último se encarga la Función totient de Euler

3.3.1. Función totient de Euler

Función totient de Euler, también conocida como función phi $\phi(n)$, cuenta el número de enteros entre 1 y n inclusive, que son coprimos de n. Dos números son coprimos si su máximo común divisor es igual 1 (1 se considera coprimo de cualquier número). Aquí están los valores de $\phi(n)$ para los primeros enteros positivos:

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

Propiedades

Las siguientes propiedades de la función totient de Euler son suficientes para calcularla para cualquier número:

- Si p es un número primo, entonces $\gcd(p, q) = 1$ para todos $1 \leq < p$. Por lo tanto tenemos:

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

- Si p es un número primo y $k \geq 1$, entonces hay exactamente p^k/p números 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 relativamente primos, entonces:

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

  Esta relación no es trivial de ver. Se sigue del teorema chino del resto . El teorema chino del resto garantiza que para cada 0 lex $< a$ y cada $0 \leq y < b$, existe un único $0 \leq z < ab$ con $z \equiv x \pmod{a}$ y $z \equiv y \pmod{b}$. No es difícil demostrar que z es coprimo de ab si y solo si x es coprimo de a y y es coprimo de b. Por lo tanto, la cantidad de números enteros coprimos a ab es igual al producto de las cantidades de a y b.

- En general, para no coprime a y b , la ecuacion:

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

con $d = \gcd(a, b)$ se sostiene

Por lo tanto, usando las primeras tres propiedades, podemos calcular $\phi(n)$ a través de la factorización de n (descomposición de n en un producto de sus factores primos). Si $n = {p_1}^{a_1} \cdot {p_2}^{a_2} \cdots {p_k}^{a_k}$, dónde p_i son factores primos de n ,

$$\phi(n) = \phi({p_1}^{a_1}) \cdot \phi({p_2}^{a_2}) \cdots \phi({p_k}^{a_k}) \tag{1}$$

$$\tag{2}$$

$$= \left({p_1}^{a_1} - {p_1}^{a_1-1}\right) \cdot \left({p_2}^{a_2} - {p_2}^{a_2-1}\right) \cdots \left({p_k}^{a_k} - {p_k}^{a_k-1}\right) \tag{3}$$

$$\tag{4}$$

$$= p_1^{a_1} \cdot \left(1 - \frac{1}{p_1}\right) \cdot p_2^{a_2} \cdot \left(1 - \frac{1}{p_2}\right) \cdots p_k^{a_k} \cdot \left(1 - \frac{1}{p_k}\right) \tag{5}$$

$$\tag{6}$$

$$= n \cdot \left(1 - \frac{1}{p_1}\right) \cdot \left(1 - \frac{1}{p_2}\right) \cdots \left(1 - \frac{1}{p_k}\right) \tag{7}$$

Y con esto ya tendriamos un algoritmo mas eficiente que el anterior

3.4. Calcular la cantidad de comprimos de todos los valores hasta N

Si necesitamos todo todo el totient de todos los números entre 1 y n , luego factorizando todo n números no es eficiente. Podemos utilizar la misma idea que el Criba de Eratóstenes . Todavía se basa en la propiedad que se muestra arriba, pero en lugar de actualizar el resultado temporal de cada factor primo para cada número, buscamos todos los números primos y para cada uno actualizamos los resultados temporales de todos los números que son divisibles por ese número primo.

4. Implementación

4.1. C++

4.1.1. Saber si dos numeros son comprimos

```
// En versiones modernas de C++ puede utilizar la funcion __gcd(a,b)
int gcd(int a,int b){
   while (b > 0){
      a=a%b; a^=b;
      b^=a;  a^=b;
```

```
    }
    return a;
}

bool isCoprime(int a,int b){
    return 1==gcd(a,b)
}
```

4.1.2. Calcular la cantidad de comprimos de N menores que este

```
int phi(int n) {
    int result = n;
    for (int i = 2; i * i <= n; i++) {
        if(n % i == 0) {
            while (n % i == 0) n /= i;
            result -= result / i;
        }
    }
    if (n > 1) result -= result / n;
    return result;
}
```

4.1.3. Calcular la cantidad de comprimos de todos los valores hasta N

```
// Idea de la criba de Eratostenes
vector<int> phi_1_to_n(int n) {
    vector<int> phi(n + 1);
    for (int i = 0; i <= n; i++) phi[i] = i;
    for (int i = 2; i <= n; i++) {
        if (phi[i] == i) {
            for (int j = i; j <= n; j += i)
                phi[j] -= phi[j] / i;
        }
    }
    return phi;
}

// Idea suma del divisor
vector<int> phi_1_to_n(int n) {
    vector<int> phi(n + 1);
    phi[0] = 0;
    phi[1] = 1;
    for (int i = 2; i <= n; i++) phi[i] = i - 1;
    for (int i = 2; i <= n; i++)
        for (int j = 2 * i; j <= n; j += i)
            phi[j] -= phi[i];
```

```
    return phi
}
```

4.2. Java

4.2.1. Saber si dos numeros son comprimos

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

public boolean isCoprime(int a,int b){
    return 1==gcd(a,b)
}
```

4.2.2. Calcular la cantidad de comprimos de N menores que este

```
public int phi(int n) {
    int result = n;
    for (int i = 2; i * i <= n; i++) {
        if(n % i == 0) {
            while (n % i == 0) n /= i;
            result -= result / i;
        }
    }
    if (n > 1) result -= result / n;
    return result;
}
```

4.2.3. Calcular la cantidad de comprimos de todos los valores hasta N

```
// Idea de la criba de Eratostenes
public int [] phi_1_to_n(int n) {
    int [] phi =new int [n + 1];
    for (int i = 0; i <= n; i++) phi[i] = i;
    for (int i = 2; i <= n; i++) {
        if (phi[i] == i) {
            for (int j = i; j <= n; j += i)
                phi[j] -= phi[j] / i;
        }
    }
```

```
    return phi;
}

// Idea suma del divisor
public int [] phi_1_to_n(int n) {
    int [] phi =new int [n + 1];
    phi[0] = 0;
    phi[1] = 1;
    for (int i = 2; i <= n; i++) phi[i] = i - 1;
    for (int i = 2; i <= n; i++)
        for (int j = 2 * i; j <= n; j += i)
            phi[j] -= phi[i];
    return phi
}
```

5. Complejidad

Es evidente que la complejidad de saber si dos números son comprimos va depender de la complejidad del calculo del máximo común divisor la cual es $O(\log N)$ siendo $N = \text{mín}(a, b)$

La utilización de la factorización de N para calcular los coprimos de este menores que él provoca que el algoritmo tenga una complejidad de $O(\sqrt{N})$

En cuanto al cálculo de todos los coprimos para los todos los números de 1 a n el primer enfoque es básicamente idéntico al criba de Eratóstenes, la complejidad también será la misma: $O(n \log \log n)$ mientras para el segundo enfoque utilizando la propiedad de la suma del divisor la complejidad es $O(n \log n)$

6. Aplicaciones

Los números coprimos son utilizados en diferentes tipos de problemas o ejercicios, en algunos son la parte fundamental de a solución porque el problema trata propiamente de ellos, en otros son las parte de la base o sirven de apoyo para elaborar una solución.

7. Ejercicios propuestos

A continuación una lista de ejercicios que se resuelven utilizando aplicando los conocimientos abordados en esta guía:

- DMOJ - Cuban Roulette
- UVA 10179 - Irreducable Basic Fractions
- UVA 10299 - Relatives