Apuntes OIFem II Nivel 3

Problemas de matemáticas: exponenciación rápida, álgebra y teoría de números

Exponenciación rápida

En las clases del año pasado aprendimos un algoritmo que tarda $O(\log n)$ en calcular una potencia con n como exponente. Repasamos el código en la clase de hoy, además de demostrar por qué es correcto.

```
int exponenciacion (int base, int potencia) {
    if (potencia == 0) return 1;
    int x = exponenciacion(base, potencia>>1);
    if (potencia & 1) return x*x*base;
    else return x*x;
}
```

Álgebra

Hay muchos problemas (la mayoría de productos o sumatorios) que se simplifican si antes de ponernos a programar la operación que nos piden, nos paramos a escribirla de forma algebraica y a manipularla, de forma que reescribamos la expresión de una forma más sencilla de calcular a ordenador o que ocupe menos memoria.

Por ejemplo, dada una lista de números $a_1, ..., a_n$, podemos ver el siguiente problema:

Problema 1: Calcular $\sum_{i=1}^{n} \prod_{j=i}^{n} a_j$

Esto tardaría $O(n^2)$ si lo hiciéramos a fuerza bruta. Pero podemos simplificar bastante:

$\sum_{i=1}^{n} \prod_{j=i}^{n} a_j$

$= (a_1 \times a_2 \times ... \times a_n) + (a_2 \times a_3 \times ... \times a_n) + ... + (a_{n-1} \times a_n) + a_n$

$= a_1 \times (a_2 \times a_3 \times ... \times a_n) + a_2 \times (a_3 \times a_4 \times ... \times a_n) + ... + a_{n-1} \times a_n + a_n$

obteniendo una solución sencilla que podemos implementar en $O(n)$ usando programación dinámica.

Algoritmo de Euclides (MCD)

Para encontrar el MCD de dos enteros positivos a y b, podemos usar la siguiente función:

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

Recomiendo leer además la demostración de que es correcto y la descripción del algoritmo extendido.

Cómo calcular el MCM rápidamente

Para calcular el mínimo común múltiplo de dos enteros positivos, podemos multiplicarlos y dividir el resultado entre su máximo común divisor, es decir, $lcm(a, b) = ab/gcd(a, b)$.

Extra: ¿sabrías demostrar esto?

Problemas de teoría de números

En las Olimpiadas de Informática, en teoría no hace falta saber más teoría de números que poder descomponer un número en sus factores primos, calcular el MCD y el MCM y el algoritmo de Euclides extendido. Sin embargo, suelen aplicarse estas propiedades básicas de formas muy enrevesadas. Lo mejor para poder puntuar en estos problemas es practicar con problemas algo aplicados de Olimpiadas Matemáticas y libros con problemas de ese tipo. Lo importante es ir seleccionando problemas que sean similares a los que podrían aparecer en Olimpiadas de Informática.

Un truco importante es tantear si no se nos ocurre una idea para resolver el problema o empezar una demostración. Podemos usar como ejemplo el problema de la OIFem I *Copiando de la pizarra*. Implementando una solución a fuerza bruta y probándola para los números 1 a 30, podríamos haber observado que se daba el caso positivo únicamente cuando $N + 1$ era primo. Partiendo de esta hipótesis, hacer la demostración habría sido mucho más sencillo que tratar de pensar esta idea desde cero. Este truco podemos aprovecharlo para algoritmos voraces también.

Problema 2: Los valores de una secuencia de enteros $a_0, a_1, ...$ pueden calcularse con la siguiente fórmula: $a_0 = 0, a_1 = 1, a_{n+2} = 2a_{n+1} + a_n$. Recibirás q queries con dos enteros no negativos n y k y deberás responder la siguiente pregunta: ¿es a_n un múltiplo de 2^k?

Fuente: adaptación de IMO Shortlist 1988

Si hacemos unos bucles anidados iterando sobre $k = 0$ hasta $k = 10$ y $n = 1$ hasta $n = 10$, podremos observar que a_n es múltiplo de 2^k siempre y cuando n lo sea también. Dejamos como ejercicio de deberes el demostrar esto matemáticamente (2 puntos).

Materiales de apoyo y extensión de Matemáticas: Mathematics for Computer Science capítulos 1, 5 y 8

Materiales de extensión de Big O: Master Theorem