IIC2283 — Diseño y Análisis de Algoritmos
Ayudantía 12

En esta ayudantía, N se referirá a un número natural impar, con al menos 2 factores primos distintos.

Problema 1: Factorización entera 1

Parte 1 Describa como determinar si un entero es un cuadrado perfecto.

Idea de solución Se puede llamar a TieneRaizEntera$(n, 2, 1, n)$.

Parte 2 Demuestre que si x es un entero e $y = x^2 - N$ es un cuadrado, entonces si $c = (x - \sqrt{y})$ está entre 1 y N se cumple que $c|N$.

Idea de solución Si definimos z tal que $y = z^2$, entonces $x^2 - z^2 = N$, y por lo tanto $N = (x + z)(x - z)$, con $z = \sqrt{y}$.

Parte 3 Describa y analice un algoritmo aleatorizado, que dado un entero N mayor o igual a 3, encuentre un divisor propio no trivial. El algoritmo puede fallar, pero en caso de entregar una respuesta, esta debe ser correcta.

Idea de solución Generar aleatoriamente un número x, y luego verificar si $t = x^2 - N$ es un cuadrado. En ese caso, se puede generar el número $x - \sqrt{t}$, que divide a N por la parte 2, y retornarlo en caso que sea distinto de 1 y de N.

Para generar un número aleatoriamente se necesita definir un rango, por lo que se debe acotar el valor de x. Como $N = x^2 - y^2$, entonces x debe ser mayor que $\sqrt{N}$. Por otra parte, si se define $a = (x + y)$ y $b = (x - y)$, entonces $N = ab$, $x = (a + b)/2$, y por lo tanto $a = N/b$. Luego, se tiene que $x = (N/b + b)/2$, de adonde se concluye que $x \leq (N + 1)/2$.

Calcular t requiere una cantidad constante de operaciones entre enteros, y verificar si es un cuadrado es de tiempo $O(\log(N))$.

La probabilidad de falla se puede acotar considerando la cantidad de divisores de N. Como x está entre $\sqrt{N}$ y $(N + 1)/2$, entonces existen $(N + 1)/2 - \sqrt{N} + 1$ posibilidades para x, y por cada una de ellas, si $t = x^2 - N$ es un cuadrado, se genera el divisor $x - y$, con $t = y^2$. Por otra parte, cada divisor de N solo puede ser generado por un valor de x: si existen dos, entonces $(x_1 + y_1)(x_1 - y_1) = (x_2 + y_2)(x_2 - y_2)$, y también $x_1 - y_1 = x_2 - y_2$, lo que implica que $x_1 = x_2$. Luego, si el número tiene $d(N)$ divisores propios distintos de 1, entonces la probabilidad de encontrar uno de ellos es a lo más:

$$\frac{d(N)}{(N + 1)/2 - \sqrt{N} + 1}.$$

La probabilidad real puede ser menor por que no necesariamente se generan todos los divisores.

Problema 2: Factorización entera 2

En este problema se define el conjunto B_k, como el conjunto formado por los primeros k números primos. También se define BS_k, como el conjunto de todos los enteros cuya factorización en primos solo utiliza elementos de B_k. Por ejemplo, $B_4 = \{2, 3, 5, 7\}$, y $525 = 3 \cdot 5^2 \cdot 7 \in BS_4$.

Parte 1 Demuestre que existen enteros x, y tales que $\text{MCD}(x, N) = \text{MCD}(y, N) = 1$, $x^2 \equiv y^2 \mod N$, pero $x \not\equiv \pm y \mod N$.

Para esto, puede ser útil considerar las raíces de $x^2 - 1$ módulo N.

Idea de solución Si consideramos un valor c tal que $c^2 - 1 \equiv 0 \mod N$, entonces se tiene que

$$N|(c+1)(c-1).$$

Además, como N tiene al menos dos factores primos mayores que 2, entonces existen n_1, n_2 tales que $\text{MCD}(n_1, n_2) = 1$, $n_1, n_2 \geq 3$ y $N = n_1 \cdot n_2$. Luego, por el teorema chino del resto, existe C tal que:

$$C \equiv 1 \mod n_1$$
$$C \equiv -1 \mod n_2$$

y por lo tanto que $C \not\equiv 1 \mod N$ y $C \not\equiv -1 \mod N$. Por definición de módulo, se cumple que $n_1|(C-1)$ y que $n_2|(C+1)$, por lo que $N|C^2 - 1$, y entonces $C^2 \equiv 1^1 \mod N$.

Parte 2 Describa un algoritmo que dado un número z, determine si $z^2 \in BS_k$, y además encuentre su factorización.

Idea de solución Algoritmo greedy. Se ordena B_k de mayor a menor, y se divide todas las veces posibles por cada elemento en orden. Si al terminar se obtiene 1, entonces se retorna SI, y en otro caso se retorna NO.

El algoritmo es correcto por que la multiplicación es conmutativa, y realiza a lo más $\log_2(z)$ divisiones. Cada una de ellas se realiza en tiempo $O(\log_2^2(n))$ con el algoritmo usual, por lo que el algoritmo completo funciona en tiempo $O(\log_2^3(n))$, que es polinomial en el tamaño de la entrada.

Parte 3 Desarrolle y analice un algoritmo, que dados números $Z = \{z_1, \ldots, z_\ell\}$ con $z_i^2 \in BS_k$ para todo i, determine un subconjunto no vacío $Z' \subseteq Z$, tal que

$$\prod_{z \in Z'} z^2 \equiv \prod_{p_i \in B_k} p_i^{e_i} \mod N$$
$$e_i \equiv 0 \mod 2$$

Idea de solución Cada elemento de BS_k se puede considerar como un vector en $\mathbb{Z}^k$, que en la posición i, contiene el exponente de p_i. Bajo esta representación, $z_1 \cdot z_2$ corresponde a $\vec{z_1} + \vec{z_2}$. De esta forma, el problema se transforma en encontrar una combinación lineal de los vectores de Z, tal que sea congruente al vector 0 módulo 2, es decir resolver:

$$\begin{bmatrix} \vec{z_1} & \vec{z_2} & \cdots & \vec{z_\ell} \end{bmatrix} \vec{x} \equiv 0 \mod 2$$

para lo cual se puede usar eliminación gaussiana, en tiempo $O(k^3)$.

Parte 4 Muestre que si $a^2 \equiv b^2 \mod N$, con $a \not\equiv \pm b$, entonces $\text{MCD}(a+b, N) > 1$, y $\text{MCD}(a+b, N) < N$.

Idea de solución Si $a^2 \equiv b^2 \mod N$, entonces $N|(a+b)(a-b)$. Además, si $a \not\equiv -b$, entonces $N \not|(a+b)$, y como $a \not\equiv b$, entonces $N \not|(a-b)$.

Finalmente, de $N \not|(a+b)$ se obtiene que $\text{MCD}(a+b, N) \neq N$, y de $N \not|(a-b)$ que $\text{MCD}(a+b, N) \neq 1$, concluyendo que $1 < \text{MCD}(a+b, N) < N$.

Parte 5 Describa un algoritmo aleatorizado que con input N y B_k, encuentre un divisor propio y no trivial de N.

Idea de solución

- Generar números aleatorios e independientes entre 1 y $N-1$. La cantidad es un parámetro, pero debe ser mayor o igual a k.
- Eliminar los elementos que no pertenezcan a BS_k. Si se obtienen menos de k elementos, entonces el algoritmo falla.
- Obtener un subconjunto del anterior, tal que la suma de los exponentes sea par. Con esto se obtienen dos elementos cuyos cuadrados son congruentes módulo N: $a = \prod_{z \in Z'} z$ y $b = \prod_{p_i \in B_k} p_i^{e_i/2}$
- Revisar que $a \not\equiv \pm b$. Si son equivalentes, entonces el algoritmo falla.
- Retornar $\text{MCD}(a+b, N)$.

El primer paso es de tiempo $O(k)$, el segundo de tiempo $O(\log^3(N))$, el tercer paso es de tiempo $O(k^3)$, el cuarto de tiempo $O(\log(N))$, y el quinto de tiempo $O(\log(N))$, por lo que el algoritmo funciona en tiempo $O(\log^3(N) + k^3)$.