MA-53G Intr. a la Criptografía **Semestre Primavera, 1996**

Cátedra No. 3 : 13 de Agosto

Transcriptor: Alejandro Hevia

1 Costo Computacional Operaciones Básicas

Definición 1 *Diremos que un algoritmo que realiza cálculos que involucran enteros $n_1, \ldots, n_r$ es polinomial o eficiente si existen $d_1, \ldots, d_r$ enteros tal que el número de operaciones del algoritmo es:*

$$O(lg(n_1)^{d_1} \ldots lg(n_r)^{d_r})$$

Convención: *Salvo que se indique lo contrario, asumimos que todos los enteros se codifican en binario.*

El siguiente resultado nos será útil en el resto de la sección.

Proposición 1 *La cantidad de dígitos de la representación binaria (número de bits) de un entero n es $\lfloor lg(n) \rfloor + 1$.*

La demostración se deja como ejercicio.

Estimar el tiempo que se demora un algoritmo corresponde a estimar el número de operaciones que realiza en función del tamaño la entrada.

Ejemplo 1 *Sumar dos números de k y l bits*

$$\begin{array}{rl} 1100101 & \longleftarrow k \text{ bits} \\ \underline{10011} & \longleftarrow l \text{ bits} \\ 1111000 & \end{array}$$

Para realizar esta operación debemos recorrer ambos números de derecha a izquierda e ir sumando bit a bit cada uno de sus dígitos. En el caso general, el número de sumas bit a bit *efectuadas no será mayor que la cantidad de dígitos del mayor de ambos números*[1]*. En notación de orden, esto significa que la operación toma $O(max\{k, l\})$*

[1] Notar que la cantidad de bits de un número es una función no decreciente de él.

Ejemplo 2 *Multiplicar dos números de k y l bits*

$$\begin{array}{rl}
1100101 & \longleftarrow k \text{ bits} \\
\underline{10011} & \longleftarrow l \text{ bits} \\
1100101 & \\
1100101 & \\
\underline{1100101} & \\
11101111111 &
\end{array}$$

Esta operación toma $O(k * l)$ operaciones binarias. Por ello, la multiplicación de dos enteros n y m tomará $O((\lfloor lg(m) \rfloor + 1)(\lfloor lg(n) \rfloor + 1)) = O(lg(n)lg(m))$.

Observación 1 *En este caso sólo contamos las operaciones binarias y consideramos el peor caso (hacemos la suma sobre enteros de k bits), debido a que nos interesa encontrar una estimación general del número de operaciones, lo cual debe ser independiente de características circunstanciales de la entrada en cuestión.*

Ejemplo 3 *Número de operaciones necesarias para calcular $n!$*

El cálculo de $n!$ requiere que en el $(j-1)-$ésimo paso ($j = 2, \ldots, n$) multipliquemos $j!$ por $(j+1)$. Acotaremos el número máximo de bits requeridos en la representación binaria, de $j!$ por los bits requeridos en la representación binaria de $n!$.
Dado que el número de bits del producto de dos números de k y l bits, es a lo más kl bits, entonces el producto de n números de k bits será de a lo más nk bits. Si n posee k bits, cada entero positivo menor que n poseerá menos de k bits, luego $n!$ poseerá a lo sumo nk bits.
Si consideramos que $(j+1)$ tiene a lo más k bits, $j!$ tendrá nk bits a lo sumo, luego como esta multiplicación se repite $(n-2)$ veces y $k = \lfloor lg(n) \rfloor + 1$, el número de operaciones binarias es

$$O((k(nk))(n-2)) = O(n^2(lg(n))^2)$$

Para otras operaciones usuales es posible determinar una cota para el número de operaciones que demanda su realización. Esta cota la denotaremos por $Tiempo(\cdot)$, donde el argumento es la operación en referida.
Algunas cotas usuales se detallan a continuación:

1. $Tiempo(nm) = O(lg(n)lg(m))$
2. $Tiempo(n/m) = O(lg(n)lg(m))$
3. $Tiempo(n+m) = O(max\{lg(n), lg(m)\})$
4. $Tiempo(ab) = O(n_1n_2((lg(m))^2 + lg(min\{n_1,n_2\})))$, donde $a = \sum_{i=0}^{n_1} a_ix^i$, $b = \sum_{i=0}^{n_2} b_ix^i$ y $a_i, b_i \leq m$.
5. $Tiempo(\text{expresar n en base b}) = O((lg(n))^2)$
6. $Tiempo(\text{mcd}(a,b)) = O((lg(a))^3)$ donde $a \geq b$.
7. $Tiempo(\text{encontrar } u,v \text{ tq. mcd}(a,b) = ua+vb) = O((lg(a))^3)$ donde $a \geq b$.
8. $Tiempo(\sqrt{n}) = O((lg(n))^3)$.

Ejercicio 1 *Demostrar la estimación de tiempo para* $\sqrt{n}$.

Algunas mejoras:

1. $Tiempo(nm) = O(klg(k)(lg(lg(k)))$ con $k = max\{lg(n), lg(m)\}$ (ver [2]).
2. $Tiempo(\text{mcd}(a,b)) = O((lg(a))^2)$ (ver [3] y el siguiente ejercicio).

Ejercicio 2 (a) *Pruebe que el siguiente algoritmo encuentra* $d = \text{mcd}(a,b)$ *un número finito de pasos. Primero note que* $\text{mcd}(a,b) = \text{mcd}(|a|,|b|)$ *luego, sin pérdida de generalidad, podemos suponer que a y b son positivos. Si a y b son ambos pares, hacemos* $d = 2d'$ *con* $d' = \text{mcd}(a/2, b/2)$. *Si a es impar y b par, hacemos* $d = d'$ *con* $d' = \text{mcd}(a, b/2)$ *(analogamente con a par y b impar). Si a y b son impares distintos (con* $a > b$ *por ejemplo) hacemos* $d = d'$ *con* $d' = \text{mcd}(a-b, b)$. *Finalmente, si* $a = b$, *entonces hacemos* $d = a$. *El algoritmo se basa en repetir este procedimiento hasta llegar al último caso* ($a = b$).
(b) *Pruebe que el algoritmo de la parte* (a) *toma sólo* $O((lg(a))^2)$ *operaciones de bits (donde* $a > b$).
(c) *Por qué este algoritmo (en la forma presentada) puede no ser preferible al algoritmo de Euclides ?*

[2]D.E.Knuth *Art of Computing Programming: Seminumerical Algorithms Vol. 2, 2da. ed., Addison Wesley, Reading Massachusetts, 1981.*

[3]N. Koblitz *A Course in Number Theory and Criptography, pág. 16, 2da. ed., Springer-Verlag, 1994.*

2 Criptoanálisis

Corresponde a tratar de quebrar un criptosistema, lo cual entenderemos, por ahora, como tratar de determinar la clave k usada en la encriptación de un texto plano x, conocido el texto cifrado y.

Principio: Siempre asumiremos que el adversario Eva conoce el criptosistema (algoritmo) utilizado.

Diferenciamos entre distintos tipos de ataques a un criptosistema (en orden creciente de fortaleza):

(i) **Sólo texto cifrado** (*ciphertext only*): El adversario sólo ve el canal y el texto cifrado $y = y_1 y_2 \ldots y_n$. Es el tipo de ataque más débil y usualmente considera el supuesto que el mensaje (texto plano) está escrito en algún lenguaje conocido.

(ii) **Texto plano conocido** (*know plaintext*): El adversario conoce $x = x_1 x_2 \ldots x_n$ y $y = y_1 y_2 \ldots y_n$ (pero el adversario *no* elige x).

(iii) **Texto plano elegido** (*chosen plaintext*): El adversario tiene acceso al mecanismo de encripción, donde puede elegir $x = x_1 x_2 \ldots x_n$ y calcular $y = y_1 y_2 \ldots y_n$. El poder de este ataque se basa en que es posible elegir el texto plano de manera de disminuir ciertos efectos de la encripción y así obtener información acerca de la clave.

(iv) **Texto cifrado elegido** (*chosen ciphertext*): El adversario tiene acceso al mecanismo de decripción y puede elegir $y = y_1 y_2 \ldots y_n$ y calcular $x = x_1 x_2 \ldots x_n$. Este tipo de ataque es especialmente relevante en los criptosistemas de llave pública.

Ilustraciones:
1. Ataque de texto cifrado en el criptosistema Afín.

Teníamos $\mathcal{P} = \mathcal{C} = \mathcal{Z}_m$, donde

$$\mathcal{K} \;=\; \{(a,b) \in \mathcal{Z}_m \times \mathcal{Z}_m \,|\, \mathrm{mcd}(a,m) = 1\}\,.$$

Con las funciones de encripción y decripción

$$\begin{aligned} e_K(x) &= (a\,x + b) \bmod m \\ d_K(y) &= a^{-1}\,(y - b) \bmod m \end{aligned}$$

Si asumimos que conocemos

$$\Pi\colon\ \mathcal{Z}_m \to [0,1]\ tq\ \sum_{i \in \mathcal{Z}_m} \Pi(i) = 1$$

donde $\Pi(i)$ es la frecuencia relativa con que $i \in \mathcal{Z}_m$ aparece en un mensaje típico, o en el lenguaje que está escrito el mensaje[4].

Conocido $y = y_1 y_2 \ldots y_n$ determinamos

$$\Pi_y\colon\ \mathcal{Z}_m \to [0,1]\ tq\ \sum_{i \in \mathcal{Z}_m} \Pi_y(i) = 1$$

donde $\Pi_y(i)$ es la frecuencia relativa con que $i \in \mathcal{Z}_m$ aparece en el texto cifrado y. Elegimos $s, t \in \mathcal{Z}_m$ tales que

$$\Pi(s) \geq \Pi(t) \geq max_{i \in \mathcal{Z}_m \setminus \{s,t\}}\, \Pi_{x \in \mathcal{Z}_m}(x)$$

i.e. los dos valores que maximizan Π. Luego hacemos lo mismo para elegir $s', t' \in \mathcal{Z}_m$ con respecto a Π_y. Con esto, resolvemos

$$\begin{aligned} s' &= (a\,s + b) \bmod m \\ t' &= (a\,t + b) \bmod m \end{aligned}$$

obteniendo,

$$s' - t' \;=\; a\,(s - t) \bmod m$$

De aquí existen tres casos:

[4]Usualmente la frecuencia de aparición de letras, pares de letras o trios de letras.

(i) Existe un único valor para a tal que $\mathrm{mcd}(a, m) = 1$ y con el determinamos b. Basta verificar que el mensaje decriptado con esta llave (a,b) tenga sentido. Si así es, hemos quebrado el criptosistema.

(ii) Existen varios valores posibles para a. Luego, falló la suposición de que s' es la codificación de s. Elegimos otro valor para s (o bien para t) dentro de los elementos de maximizan Π sobre $\mathcal{Z}_m \backslash S$, donde S contiene a los elementos de $\mathcal{Z}_m$ ya probados. Luego, iteramos.

(iii) $s - t$ no es invertible en $\mathcal{Z}_m$, lo que indica que falló alguna suposición (acerca de s o t). Luego elegimos según lo señalado en el punto anterior e iteramos.

2. Ataque de texto plano conocido en el criptosistema de chorro

Teníamos $\mathcal{P} = \mathcal{C} = \mathcal{L} = \mathcal{Z}_2$, con

$$\mathcal{K} \;=\; (\mathcal{Z}_2)^m \setminus \{(0, \ldots, 0)\}$$

donde definimos

$$z_{i+m} \;=\; (\sum_{j=0}^{m-1} c_j \, z_{i+j}) \bmod 2$$

y $c_0, c_1, \ldots, c_{m-1} \in \mathcal{Z}_2, c_0 \neq 0$ para las funciones de encripción y decripción

$$\begin{aligned} e_z(x) &= (x + z) \bmod m \,, \\ d_z(y) &= (y + z) \bmod m \,. \end{aligned}$$

Supongamos que Eva conoce $x_1, x_2, \ldots, x_n$ y $y_1, y_2, \ldots, y_n$. Entonces ella puede calcular:

$$z_i \;=\; (x_i + y_i) \bmod 2 \,, \qquad i = \, 1, \ldots, n \,.$$

Si asumimos que m es conocido por el adversario, y se tiene que $n > 2m$, entonces

$$(z_{m+1} \, \ldots z_{2m}) = \underbrace{(c_0 \, \ldots c_{m-1})}_{C} \underbrace{\begin{pmatrix} z_1 & z_2 & \ldots & z_m \\ z_2 & z_3 & \ldots & z_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_m & z_{m+1} & \ldots & z_{2m-1} \end{pmatrix}}_{Z}$$

donde la aritmética es en $\mathcal{Z}_2$.
Es posible demostrar que Z es invertible. Con ello, podemos obtener

$$C = (z_{m+1} \ldots z_{2m})\, Z^{-1}\,.$$

Con lo cual recuperamos, los coeficientes $c_0 \ldots c_{m-1} \in \mathcal{Z}_2$ que generan la clave z del criptosistema.

Ejercicio 3 *Demuestre que la matriz $Z \in M_m(\mathcal{Z}_2)$ es invertible. Para ello, demuestre que las filas de esta matriz son vectores linearmente independientes sobre $\mathcal{Z}_2$.*

(a) *Sea* $\quad v_i = (z_i, \ldots, z_{i+m-1})\,, \qquad i \geq 1\,.$

Pruebe que, para todo $i \geq 1$,

$$v_{m+i} = (\sum_{j=0}^{m-1} c_j\, v_{i+j}) \bmod 2$$

(b) *Elija h de manera que sea el mínimo entero tal que existe una combinación lineal no trivial de los vectores $v_1, \ldots, v_h$ que suma el vector nulo (mod 2). Entonces*

$$v_h = (\sum_{j=0}^{h-2} \alpha_j\, v_{j+1}) \bmod 2$$

donde los α_j no son todos cero. Notar que $h \leq m+1$ pues cualesquiera $m+1$ vectores en un espacio $m-$dimensional son linealmente dependientes.

(c) *Pruebe que los $z's$ de la clave deben satisfacer la recurrencia lineal*

$$z_{h-1+i} = (\sum_{j=0}^{h-2} \alpha_j\, z_{j+i}) \bmod 2$$

para todo $i \geq 1$.

(d) *Observe que si $h \leq m$ los $z's$ de la clave satisfacen una recurrencia lineal de grado menor que m, una contradicción. Luego, $h = m+1$, y la matriz debe ser invertible.*

3 Seguridad Incondicional

Existen dos tipos de seguridad que estudiaremos:

Definición 2 *Seguridad Computacional*
Tiene que ver con los requisitos computacionales necesarios para quebrar un criptosistema. Diremos que un criptosistema es computacionalmente seguro *si el mejor algoritmo para quebrarlo requiere a lo menos N operaciones, donde N es un número específico muy grande.*
No sabemos como probar que un criptosistema tiene este nivel de seguridad por lo que usualmente sólo se prueba que quebrar un criptosistema es equivalente a resolver un problema bien conocido que se piensa es difícil.

Definición 3 *Seguridad Incondicional*
Un criptosistema es seguro incondicionalmente *si es imposible de quebrar aunque se utilice una cantidad infinita de recursos computacionales.*