Prop: Si $\lim_{x\to\infty} \frac{f(x)}{g(x)}$ existe (y es finito), entonces

$$f(x) = \Theta(g(x))$$

Ej: La proposición es útil, pero se puede hacer $f(x) \doteq \Theta(g(x))$ ~~funciones~~ sin usar la proposición:

$$f(x) = (x+2)\cos^2 x \quad = O(x).$$

$$\lim_{x\to\infty} \frac{(x+2)\cos^2 x}{x} = \lim_{x\to\infty} \underbrace{\cos^2 x}_{\text{oscila}} + \frac{2\cos^2 x}{x}$$

Pero $(x+2)\cos^2 x \leq x+2 \leq 2x \quad \forall x \geq 2.$

Clases de problemas:

Def: Supongamos que hay una constante $A \geq 0$, independiente del tamaño del input, t.q si el input tiene $O(k)$ bits, el algoritmo toma $O(k^A)$ pasos para resolver el problema. Se dice que el problema es de tiempo polinomial (en bits)

$A=1$ tiempo lineal

$A=2$ tiempo cuadrático.

Si existe una constante $c>0$ t.q para inputs de tamaño $O(k)$, hay un algoritmo que resuelve el problema en $\Theta(e^{ck})$ pasos, se dice que el problema es de tiempo <u>exponencial</u>.

Si $\forall \varepsilon>0$ se puede resolver el problema con inputs de tamaño $O(k)$ en $\Theta_\varepsilon(e^{\varepsilon k})$ pasos, se dice que el problema es de tiempo subexponencial (el Θ_ε dice que los c, C ~~dep~~ pueden depender de ε).

<u>Obs</u>: polinomial ← rápido

exponencial ← lento.

subexponencial ← entre los dos.

<u>Análisis del DLP</u> (clásico: $g^x = \overbrace{g \cdot g \cdots g}^{x \text{ veces}} \pmod p$)

$$G = \mathbb{F}_p^\times, \quad 2^k < p < 2^{k+1} \Rightarrow g, h \text{ y } p \text{ requieren } k\text{-bits}$$

Osea, el input al ~~p~~ DLP es de $O(k)$ bits

Algoritmo 1: fuerza bruta:

Orden de $g \mid \# \mathbb{F}_p^{\times} = \cancel{\ldots} = p-1 = \mathcal{O}(2^k)$

Entonces $g^1, g^2, \ldots, g^{p-1}$ toma $\mathcal{O}(2^k)$ pasos.

exponencial

$\mathcal{O}(2^k) = \mathcal{O}(e^{k \log 2})$.

Algoritmo 2: Si $(p-1)$ factoriza en un producto de primos pequeños.

$\cancel{(p-1) = \prod p_i^{e_i}}$ Pohlig Hellman ~~de~~ $\cancel{\mathcal{O}(\sum e_i (\log n + \sqrt{}}$

~~tiene~~ es de tiempo $\mathcal{O}(\log^2(p)) = \mathcal{O}((\log_2 2^k)^2)$

$= \mathcal{O}(k^2)$

polinomial

Algoritmo 3: Colisión: $\mathcal{O}(\sqrt{p} \log p)$, ~~no~~ exponencial pero más rápido que $\mathcal{O}(p)$.

Algoritmo 4: Cálculo de índices

$\mathcal{O}\left(e^{c \sqrt{(\log p)(\log \log p)}}\right)$, subexponencial.

Análisis de DLP (simétrico : $g^x = g + g + \cdots + g$)

O sea ~~queremo~~ tenemos $g, h, p \approx \Theta(2^k)$ y queremos hayar x t.q. $x \cdot g \equiv h \pmod{p}$

~~Para resolver~~

Fuerza bruta: $\Theta(2^k)$

Euclides: $\Theta(\log p) = \Theta(k)$. lineal.

Análisis de DLP (para curvas elípticas): (ECDLP)

Si el grupo de la curva elíptica tiene N ~~puntos~~ elementos, el mejor algoritmo conocido resuelve el DLP en $\Theta(\sqrt{N})$. exponencial.

Ahora lo hacemos un poco más formal:

Prop (cota trivial) G grupo, $g \in G$ $ord(g) = N$. Entonces el DLP $g^x \equiv h$ ~~(mod p)~~ se puede resolver en $\Theta(N)$ pasos donde cada caso consiste de multiplicar por g.

Obs: en $G = (\mathbb{F}_p)^\times$ cada multiplicación require $\Theta((\log p)^2)$ pasos ~~para~~, y el algoritmo de fuerza bruta requiere ~~$\Theta(N \cdot$~~ en general $\Theta(N (\log p)^k)$ pasos. (donde k depende del algoritmo de multiplicación y de la computadora). Pero

$(\log p)^k$ es muy pequeño comparado con N. Entonces escribimos $\mathcal{O}(N)$ (a veces se escribe $\mathcal{O}(N^{1+\epsilon})$)

Algoritmo de colisión: hacer dos listas y encontrar un elemento en común. El elemento en común nos da la solución que buscamos.

Prop: (Shanks's Baby step - Giantstep algorithm (BSGS)). Sea G un grupo, $g \in G$ de orden N. El siguiente algoritmo resuelve el DLP $g^x = h$ en $\mathcal{O}(\sqrt{N} \log N)$ pasos.

(1) Sea $n = 1 + \lfloor \sqrt{N} \rfloor$ $\quad (n \geq \sqrt{N})$

(2) Crear dos listas

Lista 1: $e, g, g^2, \dots, g^n$

Lista 2: $h, h \cdot g^{-n}, h \cdot g^{-2n}, \dots, h g^{-n^2}$

(3) encontrar el mismo elemento en cada lista: $g^i = h g^{-jn}$

(4) entonces $x = i + jn$ es la solución

Dem:

Empezamos probando que hay un match: sea x la solución incógnita. Escribimos $x = nq + r \quad 0 \leq r < n$.

Como $1 \leq x < N$,

$$q = \frac{x-r}{n} < \frac{N}{n} < n$$

$n > \sqrt{N}$

Entonces $g^x = h \iff \underbrace{g^r}_{\text{en lista 1}} = \underbrace{h \cdot g^{-qn}}_{\text{en lista 2}}$ con $0 \leq r < n$, $0 \leq q < n$

Y ahora el orden:

- Lista 1 se arma por baby steps (multiplicar por g): toma n multiplicaciones.
- Se calcula g^{-n} a lo peor toma n multiplicaciones
- Lista 2 se arma por giant steps (multiplicar por g^{-n}) toma n multiplicaciones
- Se "sort"ea las dos listas: $O(n \log n)$ pasos
- Se encuentra el "match": $O(\log n)$.

En total toma $O(3n + n\log n + \log n) = O(n \log n)$ pasos. Pero $n \approx \sqrt{N}$, y, entonces

$$O(n\log n) = O(\sqrt{N} \log N). \quad \square$$

Teoría chino de los restos: Sea $m_1, \dots, m_k$ una colección de enteros t.q $\text{mcd}(m_i, m_j) = 1$ $\forall i \neq j$

Sean $a_1, \dots, a_k \in \mathbb{Z}$ arbitrarios. Entonces el sistema de ecuaciones

$$x \equiv a_1 \pmod{m_1}$$
$$x \equiv a_2 \pmod{m_2}$$
$$\vdots$$
$$x \equiv a_k \pmod{m_k}$$

tiene una solución $x = c$. Además, si $x = c$ y $x = c'$ son dos soluciones $c \equiv c' \pmod{m_1 m_2 \cdots m_k}$.

Dem: ~~Inducción sobre ... $i \le k$~~

~~Si $k=1$, tenemos una solución. Ahora supongamos~~

Supongamos que par valor de c hemos encontrado una solución a los primeros i ecuaciones

$$x \equiv a_1 \pmod{m_1}, \dots, x \equiv a_i \pmod{m_i}. \quad (*)$$

Si $i=1$, $c_1 = a_1$ sería la solución. Usando $(*)$ vamos a mostrar como hallar una solución a

$$x \equiv a_1 \pmod{m_1}, \dots, x \equiv a_i \pmod{m_i}, x \equiv a_{i+1} \pmod{m_{i+1}} \quad (**)$$

La idea es buscar una solución de la forma

$$x = c_i + m_1 \cdots m_i y.$$

Eligiendo y t.q satisfaga $x \equiv a_{i+1} \pmod{m_{i+1}}$ nos da la solución a $(**)$.

Pero como $\text{mcd}(m_{i+1}, m_1 \cdots m_i) = 1$ hay solución y a

$$c_i + m_1 \cdots m_i y_0 \equiv a_{i+1} \pmod{m_{i+1}}. \quad \square$$ (17)

Esto se convierte en algoritmo para hallar las soluciones.

Ej: Encontrar x t.q. $x^2 \equiv 197 \pmod{\underbrace{437}_{19 \cdot 23}}$.

~~Dem~~: Trabajamos mod 19 y módulo 23 y, usando CRT, encontramos la solución:

Prop: Sea $p \equiv 3 \pmod 4$ un primo. ~~Supongan~~ Sea $a \in \mathbb{Z}$ t.q. $x^2 \equiv a \pmod p$ tiene ~~una~~ solución. Entonces

$$b \equiv a^{\left(\frac{p+1}{4}\right) \in \mathbb{Z}} \pmod p$$

es una solución.

Dem: Sea g una raíz primitiva módulo p. A tiene raíz cuadrada $\iff \exists k \in \mathbb{Z}$ $g^{2k} \equiv a \pmod p$

Ahora

$$\begin{aligned}
b^2 &\equiv \left(a^{\frac{p+1}{4}}\right)^2 \\
&\equiv a^{\frac{p+1}{2}} \\
&\equiv \left(g^{2k}\right)^{\left(\frac{p+1}{2}\right)} \\
&= g^{k(p+1)} \\
&= g^{2k + k(p-1)} \\
&\equiv g^{k(p-1)} \\
&\equiv a\, g^{k(p-1)} \equiv a \pmod p
\end{aligned}$$

porque pequeño tma de Fermat

$x^2 \equiv 197 \pmod{19 \cdot 23}$

$3 \pmod 4$.

Empezamos

$y^2 \equiv 197 \equiv 7 \pmod{19}$ y $z^2 \equiv 197 \equiv 13 \pmod{23}$

$$y \equiv (7)^{\frac{19+1}{4}} \pmod{19}$$
$$\equiv \pm 8 \pmod{19}$$

$$z \equiv (13)^{\frac{(23+1)}{4}} \equiv \pm 6 \pmod{23}$$

Ahora usando CRT $x \equiv 8 \pmod{19}$

$x \equiv 6 \pmod{23}$

nos da $x \equiv 236 \pmod{437}$ como una solución.

Obs: La solución no es única: ej $-236 \equiv 201 \pmod{437}$ también es una solución

En particular se puede ver que hay 4 soluciones.

El algoritmo de Pohlig-Hellman:

El CRT es más que un teorema, es una filosofía.

DLP: $g^x \equiv h \pmod p$: CRT no tiene nada que ver. Pero les recuerdo que x es definido mód $p-1$. Y entonces, el CRT se aplica con respecto a la factorización de $p-1$.

(19)

<u>Teorema</u>: (Pohlig-Hellman) G un grupo. y supongamos que tenemos un algoritmo que resuelve el DLP en G para cualquier elemento con orden ~~pe~~ una potencia de un primo: supongamos

$g \in G$ tiene orden $q^e \Rightarrow$ se puede resolver el DLP en $\mathcal{O}(S_{q^e})$ pasos.

Sea g un elemento de orden $N = q_1^{e_1} q_2^{e_2} \cdots q_t^{e_t}$.

Entonces $g^x = h$ se puede resolver en

$$\mathcal{O}\left(\log N + \sum_{i=1}^{t} S_{q_i^{e_i}} \right) \text{ pasos}$$

usando el siguiente algoritmo:

(1) Para cada $1 \leq i \leq t$ sea

$$g_i = g^{N/q_i^{e_i}} \quad y \quad h_i = h^{N/q_i^{e_i}}.$$

Ahora cada g_i tiene orden $q_i^{e_i}$ y usando el algoritmo dado para resolver $g_i^{y} = h_i$ (*). Sea y_i una solución a (*)

(2) Usar CRT para resolver

$$x \equiv y_1 \pmod{q_1^{e_1}}$$
$$\vdots$$
$$x \equiv y_t \pmod{q_t^{e_t}}.$$