Sul metodo ρ di Pollard

Dato un naturale $m \geq 2$, definiamo $l(m) \in \mathbb{N}$ come

$$l(m) = 2^{\lfloor \log_2(m) \rfloor} - 1.$$

In altri termini, se $2^k \leq m < 2^{k+1}$, abbiamo $l(m) = 2^k - 1$. Osserviamo che $k + 1 = \lfloor \log_2(m) \rfloor + 1$ è il numero di cifre nella scrittura binaria di m, pertanto $l(m)$ è il più grande intero che, nella scrittura binaria, ha una cifra in meno di m.

Siano $F : \mathbb{Z} \longrightarrow \mathbb{Z}$ una funzione polinomiale ed $n \in \mathbb{N}$ un intero maggiore di 0. Per ogni $a \in [0, n[$ poniamo $f(a)$ il resto della divisione di $F(a)$ per n. In questa maniera abbiamo definito una funzione $f : [0, n[\longrightarrow [0, n[$. Ogni funzione costruita in questo modo verrà detta di *tipo polinomiale*. Se r è un divisore di n, abbiamo che $a \equiv b \pmod r$ implica $f(a) \equiv f(b) \pmod r$. Infatti, dato che F è una funzione polinomiale, si ha $F(a) \equiv F(b) \pmod r$ e quindi r divide $F(a) - F(b)$. D'altra parte, per opportuni $q_1, q_2 \in \mathbb{Z}$, possiamo scrivere

$$F(a) - F(b) = q_1 n + f(a) - (q_2 n + f(b)) = (q_1 - q_2)n + (f(a) - f(b))$$

e pertanto r divide $f(a) - f(b)$. Di conseguenza $f(a) \equiv f(b) \pmod r$.

Dato un qualsiasi $x_0 \in [0, n[$, abbiamo una successione definita ricorsivamente ponendo, per ogni $i > 0$, $x_{i+1} = f(x_i)$. L'osservazione precedente ci dice che, se $x_k \equiv x_l \pmod r$, allora $x_{k+1} \equiv x_{l+1} \pmod r$. Da questo, con una semplice induzione, ricaviamo

$$x_{k+t} \equiv x_{l+t} \pmod r \qquad \text{per ogni } t \in \mathbb{N}.$$

Un altro fatto che è conveniente osservare è il seguente:

$(*)$ *se* $x_k \equiv x_l \pmod r$ *e* $j - i = k - l$, *allora* $x_j \equiv x_i \pmod r$.

Per fissare le idee possiamo pensare $j > k$ ed $i > l$. Da $j - i = k - l$ ricaviamo $t = j - k = i - l$. Allora $j = k + t$, $i = l + t$ e basta usare quanto provato in precedenza.

Veniamo ora alla descrizione della versione modificata dell'algoritmo di Pollard. In questa versione è necessario che la successione utilizzata soddisfi la proprietà $(*)$, quindi lavoreremo nell'ipotesi che la funzione f che definisce $\{x_i \mid i \in \mathbb{N}\}$ sia di tipo polinomiale. Nella nuova versione viene richiesto che, al passo k, si calcoli solamente $(x_k - x_{l(k)}, n)$. In questa maniera, al termine del passo k-esimo, sono stati calcolati k massimi comun divisori, contro i $\binom{k}{2}$ della versione originale. Assicuriamoci che, anche in questo modo, un fattore di n viene trovato.

Siano $i_0 < j_0$ tali che $(x_{j_0} - x_{i_0}, n)$ è un fattore proprio di n (diciamo un multiplo del primo p divisore di n) e scegliamo j_0 minimo rispetto a questa proprietà. Se $2^k \leq j_0 < 2^{k+1}$, poniamo $i = 2^{k+1} - 1$ e $j = i + (j_0 - i_0)$. Abbiamo

$$2^{k+1} \leq j = i + (j_0 - i_0) \leq 2^{k+1} - 1 + 2^{k+1} = 2^{k+2} - 1.$$

Allora $2^{k+1} \leq j < 2^{k+2}$ e quindi $l(j) = 2^{k+1} - 1 = i$. Se arriviamo al passo j-esimo, dobbiamo calcolare $(x_j - x_{l(j)}, n) = (x_j - x_i, n)$. Ma $j - i = j_0 - i_0$ e, dato che $x_{j_0} \equiv x_{i_0} \pmod{p}$, deduciamo $x_j \equiv x_i \pmod{p}$. Vuol dire che, al passo j-esimo, viene trovato un fattore non banale di n. Per concludere è necessario valutare quale sia il costo, in termini di tentativi, per portare a termine l'algoritmo nella nuova versione. Nella versione originale era necessario calcolare un numero di massimi comun divisori dell'ordine di ${j_0}^2$. Abbiamo

$$j = i + (j_0 - i_0) \leq 2^{k+1} - 1 + 2^{k+1} < 2^{k+2} = 4 \cdot 2^k \leq 4 \cdot j_0$$

e quindi, per trovare un fattore di n, nella nuova versione è stato necessario calcolare un numero di massimi comun divisori limitato superiormente da $4j_0$. Questo fatto prova che la nuova versione dell'algoritmo di Pollard è più veloce.