Il metodo $p-1$ di Pollard è un algoritmo di fattorizzazione mirato alla individuazione di fattori primi p con la proprietà che $p-1$ è B-smooth. Per il fatto di cercare fattori cosí speciali è di utilità pratica limitata. Ma l'idea di questo metodo è alla base del metodo di fattorizzazione con le curve ellittiche di Lenstra.

Metodo $p-1$ di Pollard.

Sia n un numero intero composto. Un modo per trovare un fattore di n è quello di prendere interi x a caso e calcolare $gcd(x,n)$, sperando di azzeccare prima o poi un intero x con $1 < gcd(x,n) < n$. Il metodo $p-1$ di Pollard, cerca x in modo "più sistematico".

Sia p un divisore primo di n. Dal Piccolo Teorema di Fermat segue che se k è un multiplo intero di $p-1$, vale

$$a^k \equiv 1 \mod p, \quad \forall\, a \in \mathbb{Z}_p^*.$$

Ciò equivale a $p \mid a^k - 1$ e dunque $p \mid \gcd(a^k-1, n)$. In particolare risulta

$$\gcd(a^k - 1, n) > 1.$$

Sfruttando l'ordine del gruppo moltiplicativo $\mathbb{Z}_p^*$, si ottiene un elemento x con $gcd(x,n) > 1$.
Il problema è costruire un elemento x con $gcd(x,n) > 1$, senza conoscere p.

Il metodo $p-1$ di Pollard lo fa per fattori p molto particolari: quelli con la proprietà che $p-1$ è B-smooth. †
Sia B un intero fissato.
Sia $k = \Pi p_i^{\alpha_i}$ il prodotto di tutti i numeri primi $p_i \le B$ tali che $p_i^{\alpha_i} \le B$, con $\alpha_i > 0$ massimo.
Se p è un divisore primo di n e $p-1$ è B-smooth allora $p-1$ (probabilmente) divide k e vale

$$a^k \equiv 1 \mod p.$$

In particolare

$$\gcd(a^k - 1, n) > 1, \qquad \forall\, a \in \mathbb{Z}_n^*.$$

L'algoritmo.

Sia n il numero da fattorizzare.
Fissati B, e k come sopra, sia a un numero a caso $1 < a < n$.

SI CALCOLANO

$$a^k - 1 \mod n \qquad \text{e} \qquad \gcd(a^k - 1, n).$$

Se $\gcd(a^k-1, n) = 1$ oppure $\gcd(a^k-1, n) = n$ (improbabile), l'algoritmo ha fallito;

Se $n > \gcd(a^k-1, n) > 1$ abbiamo un fattore non banale di n.

Per quanto detto sopra, calcolando $\gcd(a^k-1, n)$ *per* $a \in \mathbb{Z}_n^*$*, (probabilmente) si determinano in un colpo solo tutti i fattori primi p di n per cui $p-1$ è B-smooth.*

† Per definizione, un intero m si dice *B-smooth* se è prodotto di fattori primi $p_i \le B$

Cerchiamo adesso di valutare le probabilità di successo di trovare un fattore con questo algoritmo.

• *Se troviamo un fattore primo p di n, che tipo di fattore è ?*

Sappiamo che con questo algoritmo si individuano i fattori primi p di n per cui $p-1$ è B-smooth. Viceversa *un fattore primo p indiduato con questo metodo è probabilmente un primo con la proprietà che $p-1$ è B-smooth*:
Sia p un primo che divide $\gcd(a^k-1, n)$.
Si ha che p divide a^k-1 se e solo se

$$a^k \equiv 1 \bmod p$$

e ciò avviene se e solo se l'ordine di a in $\mathbb{Z}_p^*$ divide k.
Poiché k è prodotto di primi $p_i \leq B$, l'ordine di a è B-smooth.
Ne segue che probabilmente anche $p-1$ è B-smooth. †

• *Che probabilità c'è che n sia divisibile per un primo p, di una certa consistenza, tale $p-1$ sia B-smooth?*

Fissato B, i numeri B-smooth si diradano sempre di più a mano a mano che crescono (vedi Esercizi 2). A maggior ragione si diradano i primi p, per cui $p-1$ è B-smooth. Proprio per il fatto di cercare fattori cosí speciali, questo algoritmo raramente ha successo.

• *Se l'algoritmo fallisce, cosa possiamo fare per avere qualche probabilità di successo?*

Sia p un fattore di n. Se $\gcd(a^k-1, n) = 1$ allora

$$a^k \not\equiv 1 \bmod p.$$

Si hanno due possibilità: l'ordine di a in $\mathbb{Z}_p$ ha un fattore p_i^β, con $\beta > \alpha_i$, oppure ha un fattore grosso $Q > B$. In entrambi i casi l'ordine di a in $\mathbb{Z}_p$ è grosso.
Un altro a preso a caso avrebbe probabilmente ordine grosso anche lui. Quindi è poco probabile avere successo variando a.
Per avere qualche probabilità di successo è necessario accrescere B.

• *Qual è la complessità di questo algoritmo?*

Assegnato B, calcolare k ha un costo computazionale dell'ordine di $\log k$.
Assegnato a, calcolare

$$a^k \bmod n$$

ha complessità $\mathcal{O}(\log k \log^2 n)$ (vedi esercizio 10, in Esercizi 1).
Calcolare

$$\gcd(a^k-1, n)$$

ha complessità $\mathcal{O}(\log^3 n)$ (vedi esercizio 8, in Esercizi 1).
In totale

$$\mathcal{O}(\log k \log^2 n + \log^3 n).$$

OSSERVAZIONE: Segue dal Teorema dei Numeri Primi che $\log k$ ha grosso modo l'ordine di grandezza di B. Dunque la complessità è lineare in B e quindi esponenziale nel numero di cifre di B. Aumentare B è dunque molto "costoso" dal punto di vista computazionale.

Esempio. Sia $n = p \cdot q$ con p, q primi dell'ordine di $\sqrt{n}$. Supponiamo anche che $p-1$, $q-1$ siano "quasi" primi, cioè prodotto di un fattore piccolo e un fattore dell'ordine di $\sqrt{n}$. Affinché l'algoritmo $p-1$ di Pollard riesca a individuare p e q, anche B deve essere dell'ordine di $\sqrt{n}$...

† Se $p-1$ fosse il prodotto di primi $p_i \leq B$ e di un numero grosso Q, la classe di a avrebbe ordine piccolo in Z_p^*, il che è improbabile. (Vedi Nota sulle radici primitive).