Metodi per la fattorizzazione di numeri interi

Filippo Bergamasco

Dipartimento di Informatica, Universita' Ca'Foscari di Venezia

28 gennaio 2009

Fattorizzazione

Nella teoria dei numeri, per **fattorizzazione** si intende la scomposizione di un numero intero in un insieme divisori non banali che, moltiplicati fra loro, diano il numero di partenza.

In particolare scomporre un numero $n \in \mathbb{N}$ in **fattori primi** significa trovare quell'insieme di numeri primi $p_1, p_2, ...p_k$ tali che

$$\prod_{i=1}^{k} p_i = n$$

Fattorizzazione

La scomposizione esiste sempre ed e'unica (a meno di ri-ordinare i fattori).

Teorema fondamentale dell'aritmetica

Ogni numero naturale diverso da 1 o e' un numero primo o si puo' esprimere come prodotto di numeri primi. Tale rappresentazione e' unica se si prescinde dall'ordine in cui compaiono i fattori.

Semplificazione

Nel nostro caso siamo interessati a trovare, dato $n \in \mathbb{N}$, un suo divisore non banale (non necessariamente primo).

Problema

Dato $n \in \mathbb{N}$ trovare $a \in \mathbb{N}$ tale che:

- $a \mid N$
- $1 < a < n$

Semplificazione

Con un semplice algoritmo ricorsivo e' possibile, se necessario, procedere alla scomposizione completa in fattori primi del numero n.

Algorithm 1 primeFactorization(n)

if $isprime(n)$ **then**
  return n
else
  $a \leftarrow getFactor(n)$
  $b \leftarrow \frac{n}{a}$
  return $primeFactorization(a), primeFactorization(b)$
end if

Fattorizzare e' un problema complesso

Trovare un divisore non banale di un numero naturale e' un problema per cui non e'conosciuto un algoritmo risolutivo con complessita' polinomiale.

- Non e' stato dimostrato che questo algoritmo esista.
- Non e' stato nemmeno dimostrato che non possa esistere!

E' un problema ancora aperto della teoria dei numeri.

Perche' fattorizzare

Molte applicazioni moderne (specialmente quelle legate alla crittografia) si basano sul presupposto che non esista soluzione polinomiale.

- RSA e relative applicazioni:
  - SSL
  - Firma digitale
  - etc.
- Generatori di numeri pseudo-casuali
  - Blum Blum Shub

Perche' fattorizzare

Se l'algoritmo esistesse renderebbe possibile la risoluzione di altri problemi considerati ora complessi

- Calcolo rapido di funzioni moltiplicative
- Compressione di un insieme di numeri primi
  - Es: Arecibo Message

Perche' fattorizzare

RSA factoring challenge

RSA-2048

25195908475657893494027183240048398571429282126204032027777137836043662020

70759555626401852588078440691829064124951508218929855914917618450280848912

00728449926873928072877767359714183472702618963750149718246911650776133798

59095700097330459748808428401797429100642458691817195118746121515172654632

28221686998754918242243363725908514186546204357679842338718477444792073993

42365848238242811981638150106748104516603773060562016196762561338441436038

33904414952634432190114657544454178424020924616515723350778707749817125772

46796292638635637328991215483143816789988504044536402352738195137863656439

121201039712282212072035 7

Premio: 200000 USD

Una possibile classificazione

Gli algoritmi esistenti possono essere divisi in due grandi categorie:

Deterministici Forniscono sempre una soluzione corretta. Es:

- trial-division
- fermat

Probabilistici Piu' veloci, ma non e' garantito che producano un output in tempo finito. Es:

- Pollard p-1
- Pollard rho
- ECM (puo' anche esser reso deterministico)

Un'altra possibile classificazione

Special-Purpose La velocita' di fattorizzazione dipende dalle proprieta' dei suoi fattori. Es:

- trial-division
- Pollard rho
- fermat
- ...

General-Purpose La velocita' dipende solo dal numero da fattorizzare. Es:

- General Number Field Sieve
- Dixon's algorithm
- ...

Upper-Bound

Fattorizzare non e' poi cosi' complesso:

Teorema 1

Sia $n \in \mathbb{N}$. Siano $s, t \in \mathbb{N}$ divisori non banali di n tali che $st = n$ e $s \leq t$. Allora $s \leq \sqrt{n}$

In sostanza, provando per forza bruta tutti i possibili divisori di n, la complessita' e' nell'ordine di $O(\sqrt{n})$

Il piu' veloce algoritmo di fattorizzazione conosciuto

L'algoritmo piu' veloce conosciuto ad oggi e' chiamato **General Number Field Sieve** , con una complessita' nell'ordine di: $O(e^{(\frac{64}{9}b)^{\frac{1}{3}}(logb)^{\frac{2}{3}}})$ Dove b e' la dimensione (in bit) del numero da fattorizzare.

Velocita' degli algoritmi conosciuti

1. General Number Field Sieve
2. Quadratic Sieve
3. ECM
4. Pollard rho
5. Pollard p-1
6. ...
7. Fermat
8. Trial-Division

Algoritmo Trial-division

L'algoritmo piu' semplice per trovare un fattore non banale di n e' testare la divisibilita' fra n e ciascun numero compreso fra $[2...\sqrt{n}]$

Algorithm 2 trial-division(n)

for $s \leftarrow 2 to \sqrt{n}$ **do**
  if $s \mid n$ **then**
    return s
  end if
end for

Complessita': $O(\sqrt{n})$

Algoritmo di fattorizzazione di Fermat

L'algoritmo fu' ideato dal matematico francese Pierre de Fermat nel 1600.
E' basato sul fatto che ciascun numero dispari non primo maggiore di 2 puo' essere espresso come differenza di due quadrati:

Teorema

Sia $n > 2 \in \mathbb{N}$ un numero non primo dispari. Allora $\exists x, y \in \mathbb{N}$ tali che $n = x^2 - y^2$.

Dal teorema deriva immediatamente, trovando x, y una fattorizzazione di n:
$N = (x + y)(x - y)$

Algoritmo di fattorizzazione di Fermat

E' possibile dimostrare facilmente che:

- $\sqrt{n} \leq x \leq n$.
- Quando $n = x^2 - y^2 \Rightarrow y = \sqrt{x^2 - n}$.

Algorithm 3 fermat(n)

for $x \leftarrow \sqrt{n}$ to n **do**
  $ysqr \leftarrow (x^2 - n)$
  if $isPerfectSquare(ysqr)$ **then**
    $y \leftarrow \sqrt{ysqr}$
    if $1 < (x - y) < n$ **then**
      return $(x - y)$
    end if
  end if
end for

Algoritmo Pollard rho

Pollard's rho e' un metodo probabilistico di fattorizzazione pubblicato da J.M. Pollard nel 1975.
Si basa su un idea molto semplice:

Sia p il piu' piccolo divisore primo di n. Se $\exists x, x' \in \mathbb{Z}_n$ tali che $x \neq x'$ e $x \equiv x' \ mod \ p \Rightarrow p \leq GCD(x - x', n) \leq n$

Si puo' notare che:

- Presi casualmente $x, x' \in \mathbb{Z}_n$, e' possibile calcolare $GCD(x - x', n)$ senza conoscere p
- $1 \leq p \leq GCD(x - x', n)$ se x, x' "collidono" in $\mathbb{Z}_p$

Algoritmo Pollard rho

- Con che probabilita', presi casualmente $x, x' \in \mathbb{Z}_n$, si avra' una collisione in $\mathbb{Z}_p$?
  La situazione e' analoga a quella descritta nel **birthday paradox**.
- Sia $X \subseteq \mathbb{Z}_n$. Se $|X| \approx 1.17\sqrt{p}$, vi e' 50% di probabilita' che due elementi di X collidano in $\mathbb{Z}_p$

Problema:

Non possiamo, scelto $X \subseteq \mathbb{Z}_n$, calcolare $x \ mod \ p \ \forall x \in X$ poiche' non conosciamo p!
Il numero dei confronti da effettuare sale quindi a

$$\binom{|X|}{2} > \frac{p}{2}$$

Algoritmo Pollard rho

Soluzione:

Si usa uno stratagemma per risparmiare memoria e tempo computazionale.

- Sia f un polinomio a coefficienti in $\mathbb{Z}_n$ (in genere $f(x) = x^2 + 1$).
- Si consideri la successione:
  $x_0 \equiv 2 \ mod \ n$
  $x_{n+1} \equiv f(x_n) \equiv x_n^2 + 1 \ mod \ n$
- Statisticamente il numero di elementi dopo i quali la successione diventa periodica e la lunghezza del periodo sono proporzionali a $\sqrt{n}$.
- Si puo' dimostrare che, se $\exists x_i, x_j$ tali che $x_i \equiv x_j \ mod \ p$ ma $x_i \not\equiv x_j \ mod \ n \Rightarrow GCD(x_i - x_j, n)$ fattore non banale di n.

Algoritmo Pollard rho

Come trovo x_i e x_j?

- **Floyd's cycle-finding algorithm:** Percorro la successione con due puntatori, x_i e x_j, di cui uno si "muove" a velocita' doppia rispetto all'altro ($j = 2i$).
- Controllo se $1 < GCD(x_i - x_j, n) < n$. Se la condizione si verifica ho trovato un divisore non banale di n.

Perche' funziona?

- La successione in $\mathbb{Z}_p$ cicla molto prima di quella in $\mathbb{Z}_n$
- E' probabile che i due puntatori finiscano per "scontrarsi" nel ciclo su $\mathbb{Z}_p$ prima che nel ciclo su $\mathbb{Z}_n$ ($GCD(x_i - x_j, n) = n$)

Algoritmo Pollard rho

Algorithm 4 pollard-rho(n)

$x \leftarrow 2$
$x' \leftarrow f(x) \ mod \ n$
$p \leftarrow GCD(x - x', n)$
while $p = 1$ **do**
  $x \leftarrow f(x) \ mod \ n$
  $x' \leftarrow f(f(x')) \ mod \ n$
  $p \leftarrow GCD(x - x', n)$
end while
if $p = n$ **then**
  return Fail.
else
  return p
end if

Algoritmo Pollard p-1

Pollard's p-1 e' un metodo probabilistico di fattorizzazione pubblicato da J.M. Pollard nel 1974.
E' basato su due concetti:

Fermat's little theorem

Sia p un numero primo e a coprimo con p.
$\Rightarrow a^{(p-1)} \equiv 1 \ mod \ p$

Numero B-smooth

Sia $a, B \in \mathbb{N}$.
a e' B-smooth $\Leftrightarrow$ tutti i suoi fattori primi sono minori di B.

Algoritmo Pollard p-1

Idea:

Sia $B \in \mathbb{N}$ un numero, detto **bound.**
Sia $p > 2$ un fattore primo di n.
Se $(p-1)$ e' B-smooth $\Rightarrow GCD(\ 2^{B!} - 1 \ mod \ n \ , n) > 1$

Dimostrazione...

Algoritmo Pollard p-1

Algorithm 5 pollard-p-1(n)

for $B \leftarrow 2$ to n **do**
  $x \leftarrow (2^{B!} - 1)$
  if $1 < d = GCD(x, n) < n$ **then**
    return d
  end if
  if $GCD(x, n) = n$ **then**
    return Fail.
  end if
end for

Pollard p-1

Considerazione:

Se $p-1$ e' composto da fattori piccoli, l'algoritmo termina con successo molto rapidamente.
Sfortunatamente e' facile generare un numero n che non contenga un fattore primo p con questa caratteristica.

Idea:

Pollard p-1 e' basato sul fatto che l'ordine del gruppo $\mathbb{Z}_p$ rispetto al prodotto e' $p-1$. Se potessimo lavorare, invece che in $\mathbb{Z}_p$, su una struttura algebrica "simile" ma di ordine inferiore, potremo avere piu' possibilita' di successo.

Introduzione alle curve ellittiche

Curva Ellittica

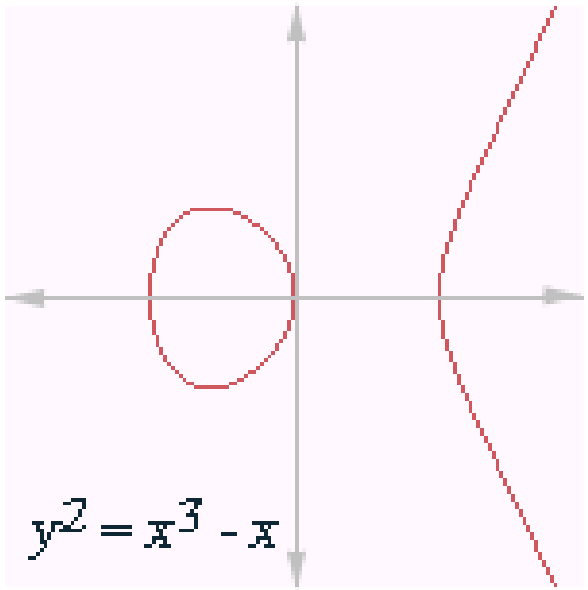
Una curva ellittica E su un campo K $(E(K))$ e' definita dall'insieme dei punti:

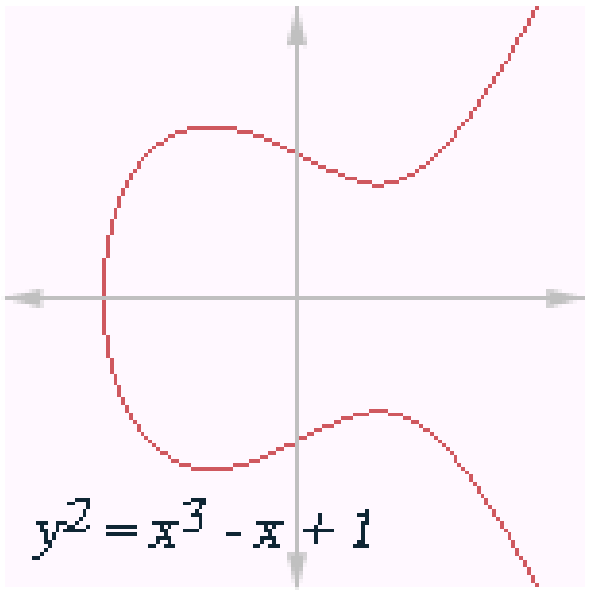
$$\{(x, y) \mid y^2 = x^2 + ax + b\}$$

con $a, b \in K$ tali che la curva non sia singolare:

$$-16(4a^3 + 27b^2) \neq 0$$

Introduzione alle curve ellittiche





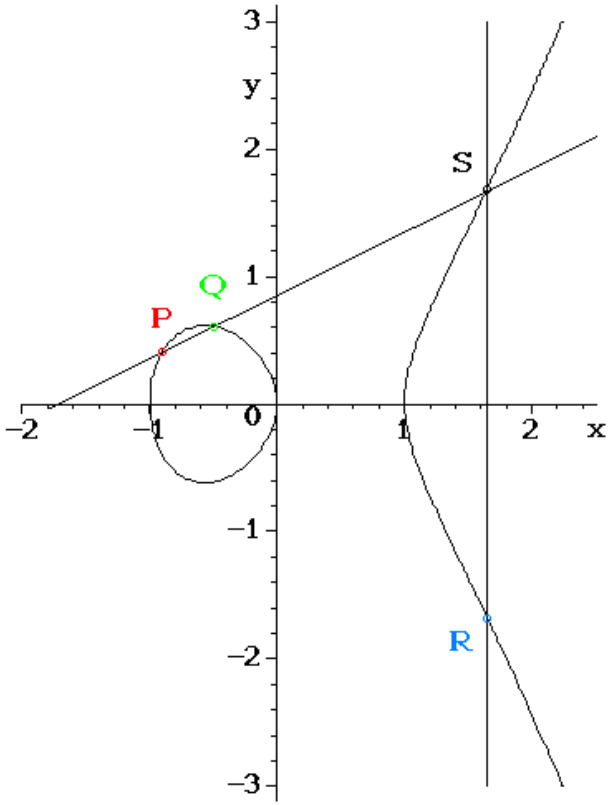
Introduzione alle curve ellittiche

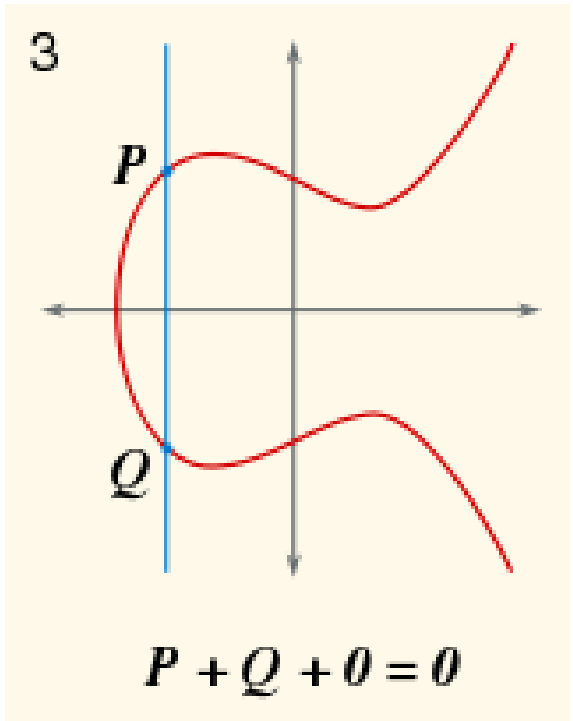
Se all'insieme dei punti di $E(K)$ aggiungiamo un punto O, detto "punto all'infinito", e' possibile definire un operazione "$+$" per cui $E(K) \cup \{O\}$ e' un gruppo abeliano.

Operazione $+$

Sia $P, Q \in E(K)$. Si tracci la retta passante per i punti P e Q. Sia $R = (x_r, y_r)$ il terzo punto (se esiste) in cui la retta interseca la curva.
$P + Q = (x_r, -y_r)$

Introduzione al gruppo delle Curve Ellittiche





Introduzione alle curve ellittiche

Dal punto di vista algebrico, possiamo definire $P = (x_p, y_p) + Q = (x_q, x_q)$ come:

$$P + Q = R = ((\frac{y_q - y_p}{x_q - x_p})^2 - x_p - x_q, \frac{(y_q - y_p)(x_p - x_r)}{x_q - x_p} - y_p)$$

Nota:

Non e' detto che $x_q - x_p$ sia invertibile in K. Se non e' invertibile allora $P + Q = O$.

Introduzione alle curve ellittiche

Una curva ellittica puo' essere definita su un qualsiasi campo K, anche finito!
es: $\mathbb{Z}_p$

L'ordine della curva (la cardinalita' dei suoi elementi) non dipende solo dal campo ma dalla curva stessa.

In generale e' difficile, data una curva su un campo finito, stabilirne l'ordine.

Teorema di Hasse

Data una curva ellittica su $\mathbb{Z}_p$:

$$p + 1 - 2\sqrt{p} \leq |E(\mathbb{Z}_p)| \leq p + 1 + 2\sqrt{2}$$

ECM (Elliptic Curve Method)

ECM e' un algoritmo sviluppato da Lenstra nel 1993 come "evoluzione" dei concetti alla base dell'algoritmo p-1 di Pollard.

Conseguenza del teorema di Lagrange

Sia G un gruppo abeliano finito.Sia $a \in G$

$$\Rightarrow a^{|G|} = 1_G$$

Applicato alle curve ellitiche...

Sia $E(\mathbb{Z}_p)$ una curva ellittica sul campo $\mathbb{Z}_p$. Sia $P \in E(\mathbb{Z}_p)$ un punto della curva.

$$\Rightarrow \overbrace{P + P + \cdots + P}^{|E(\mathbb{Z}_p)| \text{ volte}} = O$$

ECM

Domanda: Cosa significa *algebricamente* che $P + P + \cdots + P = O$?
Risposta: Alla $|E(\mathbb{Z}_p)|$-esima somma non saremo in grado di invertire $x_q - x_p$ in $\mathbb{Z}_p$.

Se per cercare l'inverso di $x_q - x_p$ in $\mathbb{Z}_p$ utilizziamo l'**algoritmo di euclide esteso** otteniamo, nel caso non esista l'inverso, il $GCD(x_q - x_p, p)$ che per definizione sara' diverso da 1.

ECM

Problema:

Quando fattorizziamo non conosciamo p (dobbiamo trovarlo!), possiamo soltanto lavorare in Z_n.
Inoltre... non conosciamo nemmeno l'ordine di $E(\mathbb{Z}_p)$.

Se $p \mid n$ e a non e' invertibile in $\mathbb{Z}_p$ ($GCD(a, p) \neq 1$) a maggior ragione a non e' invertibile in $\mathbb{Z}_n$ ($GCD(a, n) \neq 1$).

ECM - Algoritmo

1. Si genera in modo casuale una curva ellittica su $\mathbb{Z}_n$, $E(\mathbb{Z}_n)$ e un punto $P \in E(\mathbb{Z}_n)$.
2. Si continua iterativamente a sommare P per se stesso sperando di non riuscire piu' ad invertire $x_q - x_p$ in $\mathbb{Z}_n$.
3. Se non si riesce ad invertire allora e' stato trovato un fattore di n diverso da 1.
4. Se il fattore e' diverso da n l'algoritmo ha avuto successo, altrimenti si ritorna al passo 1.

ECM - Algoritmo

Quante somme dobbiamo effettuare?

Non possiamo dirlo con certezza ma:
Se p e' il piu' piccolo divisore primo di n e k e' l'ordine della curva ellittica scelta ma costruita su $\mathbb{Z}_p$, $\Rightarrow$ dopo k somme ottieniamo un $GCD \neq 1$.

Considerazioni

- L'ordine di $E(\mathbb{Z}_p)$ puo' essere, in genere, molto piu' piccolo di p-1
- Se l'algoritmo fallisce possiamo tentare con un altra curva ellittica

Bibliografia

- D.R.Stinson, *Cryptography, Theory and Practice*, Third Ed., CRC Press, 2005.
- William Stein, *An Explicit Approach to Elementary Number Theory*, Math124 Lectures, Harvard University, 2001.
- Connelly Barnes, *Integer Factorization Algorithms*, Oregon State University, December 2004.
- Lenstra Jr., H. W. *Factoring integers with elliptic curves.*, Annals of Mathematics 2 126 (1987), 649-673. MR 89g:11125.
- From Wikipedia, an online encyclopedia, *Elliptic Curve*, Jan. 2009, `http://en.wikipedia.org/wiki/Elliptic_curve`.