[28/02] Alg. di Euclide, MCD, primi

Wednesday, February 28, 2018 11:33

La funzione $\Pi(x)$ rappresenta il numero di numeri primi inferiori o uguali a x.
Si è dimostrato che: $\Pi(x) \sim \frac{x}{\ln(x)}$ (ovviamente per $x \to \infty$).

Per ogni numero primo p ($p \neq 2$) vale:

p è primo $\Rightarrow p \equiv 1 \pmod 4 \vee p \equiv 3 \pmod 4$
p è primo $\Rightarrow p \equiv \pm 1 \pmod 6$

Un buon metodo per verificare se un numero x è primo è testare la divisione per tutti i numeri $n_k = 6k \pm 1 \leq \sqrt{x}$.

Due numeri a e b sono **primi tra loro (o primi relativi)** se non hanno divisori in comune oltre 1 (il numero uno):

$\text{MCD}(a, b) = 1 \Rightarrow \text{a} \perp \text{b}$

Vale la relazione $\mathbf{MCD}(\boldsymbol{a}, \boldsymbol{b}) = \mathbf{MCD}(\boldsymbol{a} \textbf{ mod } \boldsymbol{b}, \boldsymbol{b})$, che può facilmente essere usata per definire un algoritmo ricorsivo (o iterativo) per il calcolo del MCD di due numeri (interi positivi) qualsiasi:

Algoritmo di Euclide ricorsivo e iterativo:

```
def gcd_rc(a, b):
    if b == 0: return a
    return gcd_rc(a, b % a)

def gcd_it(a, b):
    while b:
        a, b = b, a % b
    return a
```

Cercare funzione toziente: $\Phi(n)$
Definizione di $\mathbb{Z}_n^*$

[01/03] Euclide esteso, teo cinese del resto

Thursday, March 1, 2018 10:40

Algoritmo di Euclide esteso

Questo algoritmo permette di generare i due interi $\boldsymbol{x}, \boldsymbol{y}$: $ax + by = \text{MCD}(a, b)$ oltre al MCD *tra* a e b. Ricordare che **questo algoritmo assume $\boldsymbol{a \leq b}$**.

	$x_0 = 0; x_1 = 1$	$y_0 = 1; y_1 = 0$
$b = q_1 a + r_1$	$x_2 = -q_1 x_1 + x_0$	$y_2 = -q_1 y_1 + y_0$
$a = q_2 r_1 + r_2$	$x_3 = -q_2 x_2 + x_1$	...
$r_1 = q_3 r_2 + r_3$	...	...
...	...	...
$r_{k-2} = q_k r_{k-1} + \boldsymbol{r_k}$	$\boldsymbol{x_{k+1}} = -q_k x_k + x_{k-1}$	$\boldsymbol{y_{k+1}} = -q_k y_k + y_{k-1}$
$r_{k-1} = q_{k-1} r_k + 0$	-	-

Una volta definita e calcolata questa tabella i due numeri x_{k+1} e y_{k+1} sono i due coefficenti cercati e r_k =MCD(a, b). **Inoltre $\boldsymbol{x_{k+1} = a^{-1}}$ (mod b).**

$$x_{k+1}a + y_{k+1}b = r_k = \text{MCD}(a, b)$$

Proprietà dell'operatore di congruenza

- $a \equiv 0 \pmod{n} \iff n \backslash a$ (n divide a)
- $a \equiv b \pmod{n} \equiv b + kn \pmod{n}, k \in \mathbb{Z}$
- $a \equiv b \Leftrightarrow b \equiv a$
- $a \equiv b, b \equiv c \Rightarrow a \equiv c$
- $a \equiv b$ e $c \equiv d \Rightarrow a \pm c \equiv b \pm d$ e $ac \equiv bd$
- $ab \equiv ac \pmod{n}$: se $a \perp n \Rightarrow b \equiv c$

Sfruttando queste proprietà:

$$2x + 7 \equiv 3 \pmod{17}$$
$$2x \equiv -4 \pmod{17}$$
$$x \equiv -2 \equiv 15 \pmod{17}$$

$$5x + 6 \equiv 13 \pmod{11}$$
$$5x \equiv 7 \pmod{11}$$
$$5x \equiv 7 \equiv 18 \equiv 29 \equiv \mathbf{40} \pmod{11}$$
$$5x \equiv 40 \pmod{11}$$
$$x \equiv 8 \pmod{11}$$

$5x \equiv 7 \pmod{11} \Rightarrow$ trovo 5^{-1} (mod 11) usando l'algoritmo di euclide esteso e moltiplico enrambe le parti per esso.

Funzione toziente

Tra gli elementi di $\mathbb{Z}_n$ **solo gli elementi appartenenti a $\mathbb{Z}_n^*$, ovvero gli elementi di $\mathbb{Z}_n$ che sono primi relativi con $\boldsymbol{n}$, sono invertibili all'interno di $\mathbb{Z}_n$. La funzione toziente $\boldsymbol{\phi(n)}$**

rappresenta la cardinalità di $\mathbb{Z}_n^*$: $\phi(n) = |\mathbb{Z}_n^*|$. Siccome per un numero primo p vale $|\mathbb{Z}_p| = p$ e $|\mathbb{Z}_p^*| = p - 1 \Rightarrow \boldsymbol{\phi(p) = p - 1}$.

Soluzioni di equazioni frazionarie in modulo

Nel caso generale $ax \equiv b$ (mod n) e $\text{MCD}(a, n) = d > 1$:

- se d non divide $b \Rightarrow$ non esiste soluzione
- se $d\backslash \text{b} \Rightarrow \frac{a}{d}x \equiv \frac{b}{d} \left(\text{mod} \frac{n}{d}\right)$

ES:

$$12x \equiv 21 \text{ (mod 39)}$$
$$\text{MCD}(12, 39) = 3 > 1$$

$$\frac{12}{3}x \equiv \frac{21}{3}\left(\text{mod} \frac{39}{3}\right)$$

Equazione ridotta: $4x \equiv 7 \text{ (mod 13)} \Rightarrow \text{x}_0 = 5$
Soluzioni dell'equazione originale: x_0, $x_0 + 13$, $x_0 + 13 \cdot 2$ ovvero $5, 18, 31$.
In generale saranno $x_k = x_0 + 13k$ per tutti i $k \geq 0$ "validi", ovvero tali che $x_k < 39$.

Teorema cinese del resto

Sappiamo che se $x \equiv k$ (mod nm) $\Rightarrow x \equiv a$ (mod n) $\equiv b$ (mod m), ma non il contrario. Il teorema cinese del resto afferma che **nei casi in cui $\boldsymbol{n \perp n}$ l'implicazione è doppia**:

$$n \perp m \Rightarrow \begin{cases} x \equiv a \text{ (mod n)} \\ x \equiv b \text{ (mod m)} \end{cases} \Leftrightarrow x \equiv k \text{ (mod } nm)$$

Ed avremo quindi:

$b + mk \equiv a$ (mod n) $\Rightarrow b - a \equiv mk$ (mod n) $\Rightarrow \boldsymbol{k = (a - b)m^{-1}}$ **(mod $\boldsymbol{n}$)**
(NB: $n \perp m \Rightarrow \exists\, m^{-1} \in \mathbb{Z}_n$)

ES:

$$\begin{cases} x \equiv 3 \bmod 7 \\ x \equiv 5 \bmod 15 \end{cases}$$

$k \equiv (3 - 5)15^{-1} \bmod 7$
$15 = 1 \text{ (mod 7)} \Rightarrow 1^{-1} = 1 \text{ (mod 7)}$
$\Rightarrow k = (3 - 5) \cdot 1 \text{ (mod 7)} \Rightarrow k \equiv -2 \equiv 5 \text{ (mod 7)}$

Algoritmo square & multiply

Questo algoritmo è utile per calcolare moduli di potenze molto grandi che sono difficili da ridurre e calcolare, e.g. $9^{616} \bmod 17$.

L'algoritmo S&M scompone l'esponente in somma di potenze di 2 per semplificare il calcolo scomponendo il numero in una produttoria con esponenti minori.

ES:

Calcoliamo $7^{11} \pmod{26}$:

11 (bin)	**Scomposizione di 7^{11}**
1 (LSB)	$7^1 \equiv \mathbf{7} \pmod{26}$
1	$7^2 \equiv \mathbf{23} \pmod{26}$
0	$7^4 \equiv 23^2 \equiv 9 \pmod{26}$
1 (MSB)	$7^8 \equiv 9^2 \equiv \mathbf{3} \pmod{26}$

$$\Rightarrow \mathbf{7^{11} \equiv 7 \cdot 23 \cdot 3 \pmod{26} \equiv 15}$$

Ovviamente 7^{11} non è un buon esempio per dimostrare il vantaggio dell'algoritmo S&M, dato che l'esponente è molto piccolo.

Piccolo teorema di Fermat

Dato un numero primo p e una base a tale che $\boldsymbol{a \perp p \Rightarrow a^{p-1} \equiv 1 \pmod{p}}$.

Questa relazione può essere usata come test di **probabile** primalità, dato che **la relazione è vera per qualsiasi numero primo, ma anche per alcuni numeri composti** (non primi).

Teorema di Eulero

Generalizzazione del piccolo teorema di Fermat per qualsiasi numero (anche composto):

$$a \perp n \Rightarrow a^{\phi(n)} = 1 \pmod{n}$$

La $\phi(n)$ può essere calcolata facilmente considerando che:

$$n = \prod_{i=1}^{n} p_i^{r_i} \text{ (produttoria dei fattori primi elevati ai loro coefficenti)}$$

Perciò:

$$\phi(n) = n \prod_{\mathrm{p_i \backslash n}} \left(1 - \frac{1}{p_i}\right)$$

ES:

$$\phi(1000) = (2^3 - 2^2)(5^3 - 5^2) = 4 \cdot 100 = 400$$

Calcolo semplificato per esponenti molto grandi tramite $\phi(n)$

Per il calcolo del modulo di un numero elevato ad un esponente molto grande posso sfruttare $\phi(n)$ ed eseguire il calcolo portando l'esponente in modulo $\phi(n)$:

$$\boldsymbol{x^y \bmod n = x^{y \bmod \phi(n)} \bmod n}$$

ES:

$7^{803} \bmod 1000$

$= 7^{803 \ mod \ \phi(1000)} mod \ 1000$
$= 7^{803 \ mod \ 400} \ mod \ 1000$
$= 7^{3} \ mod \ 1000$
$= 343$