

TEST di PRIMALITÀ

PRIMO(N):

for $i = 2$ to \sqrt{N} do

if ($N \% i = 0$) then return "COMPOSTO"

else skip;

done

return "PRIMO"

Svolge al più \sqrt{N} iterazioni ciascuna di costo $O(\log^2 N)$

con $I :=$ istanza di input = N

↳ costo della

$|I| :=$ dimensione di I : $\Theta(\log N)$

divisione

$$T(|I|) = O(\sqrt{N} \cdot \log^2 N) = O(|I| \cdot 2^{|I|/2})$$

$$|I| = \Theta(\log N)$$

$$N = \Theta(2^{|I|})$$

$$\sqrt{N} = N^{1/2} = \Theta(2^{|I|/2})$$

L'algoritmo è quindi polinomiale sul valore dell'input,
esponentiale sulla dimensione.

ALGORITMI RANDOMIZZATI

CAS VEGAS : risultato sicuramente corretto in un tempo probabilmente breve

MONTE CARLO : risultato probabilmente corretto in un tempo sicuramente breve

La probabilità di errore deve essere matematicamente misurabile e arbitrariamente piccola.

TEST di PRIMALITÀ di MILLER-RABIN

N , intero di n cifre t.c. dispari ($N \geq 7$)

$$N-1 = 2^w \cdot z \quad z: \text{numero dispari}$$

$$\text{e.g. } N=45, \quad 45-1 = 2^2 \cdot 11$$

w, z si determinano in $\mathcal{O}(\log N)$ passi

Si sceglie $y \in [2; N-1] \cap \mathbb{N}$

Vale inoltre che:

N primo \Rightarrow

$P_1 :=$ 1) $\text{MCD}(N, y) = 1$ (ovvio perché N è primo)

$P_2 :=$ 2) $y^z \pmod{N} = 1 \vee \exists i \in [0; w-1] \cap \mathbb{N} \text{ t.c.}$
 $y^{2^i \cdot z} \pmod{N} = -1$

Quando valgono 1,2 probabilmente N è primo

LEMMA di MILLER-RABIN Se N è un numero composito, il numero di interi $2 \leq y \leq N-1$ che

soddisfano entrambi i predicati e' minore di $0.25 \cdot N$.

$$\#\{2 \leq y \leq N-1 \mid P_1(y) \wedge P_2(y)\} < N/4$$

\Rightarrow probabilità di scegliere un fattorino y che rende vari P_1, P_2 è minore di $N/4N = 1/4$.

L'idea è:

. Scegli a caso $y \in [2; N-1]$

\rightarrow se $\neg(P_1 \wedge P_2)$ allora N è composto

\rightarrow altrimenti

$\rightarrow N$ è composto con probabilità < 0.25

$\rightarrow N$ è primo con probabilità > 0.75

. Inferendo k volte con k scelte di fattorini casuali e

indipendenti la probabilità di errore scende a $(0.25)^k$

VERIFICA (N, y): // controlla la validità del certificato y

if (not(P_1) or not(P_2)) then

return false; // N è composto

else return true; // N potrebbe essere composto

TEST-MR (N, k):

for $i=1$ to k do

$y = \text{random}(2, N-1);$

if not Verifica (N, g) then return 0; // N composto

done

return 1; // N primo con probabilità di errore minore di $(0.25)^k$

La parte computazionalmente costosa è VERIFICA:

P_1 esegue il test di Euclide

$$T(n) = O(\log^3 n)$$

P_2 :

$$N-1 = 2^w \cdot d$$

$y^2 \bmod N$ come si calcola?

Algoritmo di esponenti-izzazione veloce

- o delle quadrature successive

obiettivo \rightarrow numero di operazioni $O(\log 2)$

$$\text{esg } x = g^{45} \bmod n$$

$$= g^{32} \cdot g^8 \cdot g^4 \cdot g^1 \bmod n$$

Si calcolano le g^{2^i} fino a g^{32}

come il quadrato dello precedente:

$$5 \text{ multiplication} \left\{ \begin{array}{l} g^1 \rightarrow g \\ g^2 \rightarrow 4 \\ g^4 = 5 \\ g^8 = 3 \\ g^{16} = 9 \end{array} \right. \quad g^{32} = 1$$

+ multiplication con $t := \lfloor \log_2(2) \rfloor$

A questo punto confrisco g^{45} :

O(numero di bit) $\rightarrow g^{32} \cdot g^8 \cdot g^4 \cdot g = 4 \cdot 3 \cdot 5 \cdot 9 = 1$

$\Rightarrow 1$ nell'esponente

scrifto in base 2)

moltiplicazione

= O(f) mult.

Algoritmo quadrature successive;

$$x = y^2 \bmod s$$

1. Si scomponere s in una

y, z, s dello stesso ordine

Somma di potenze di 2

d'grandezza

$$z = \sum_{i=0}^t 2^i \cdot k_i$$

$$k_i \in \{0, 1\}, t = \lfloor \log_2 z \rfloor =$$

2. Si calcolano tutte le

$$= O(\log z)$$

potenze

$$y^{2^i} = (y^{2^{i-1}})^2 \bmod s$$

come quadrato della precedente

$O(\log z)$ mult.

3. Si calcola $x = y^2 \bmod s$ come

$$x = (\prod_{i=k+1}^t y^{2^i} \bmod s) \bmod s$$

$\} O(\log z)$ mult

Per la seconda parte di P_i :

. Si calcola (QS) $y^8 \bmod N$

$$y^2 \bmod N$$

/ | \

$\equiv 1$ $\equiv -1$ $\not\equiv \pm 1$

ok, finito ok, finito

$P_2 = T$ $P_2 = T$

↓

$$y^{2k} \bmod N$$

/ | \

...

↓

$$y^{4k} \bmod N$$

/ | \

...

↓

$$w = \Theta(\log N)$$

GENERAZIONE di NUMERI PRIMI

Si genera un numero casuale e si esegue il test di primalita' (si ripete fino a che non lo si dichiara primo con probabilita' di errore $< (0.25)^k$)

Teorema

Il numero t numeri interi primi e minori di N tende per $N \rightarrow \infty$ a $N / \log_2(N)$.

\Rightarrow Per N sufficientemente grande, in un suo intorno di ampiezza $\log_2(N)$, c'è mediamente un numero primo.



e' proporzionale al numero di cifre di N
(dimensione dell'input).
e' polinomiale in $\log_2(N)$

PRIMO(N): // N : numero di bit del numero primo da generare

// genera un numero primo lungo N

S = sequenza casuale di $n-2$ bit;

$P = "1" + S + "1";$ // concateno 1 a destra e sinistra

// per fornire i numeri a essere d'ipari

while TEST-MR(P) = 0 do

$P += 2;$

done

return $P;$

Il costo complessivo è $O(N^4)$:

al più N volte il test di MR

che ha complessità di N^3 .

Il numero generato può avere al più da $N+1$ bit.

Il test di primalità viene dunque risolto in tempo polinomiale da un algoritmo randomizzato;

Si introduce la classe RP:

Problemi decisionali verificabili in tempo
polinomiale randomizzato

RP problemi decisionali

a istanza di input di RP

y è un certificato probabilistico di RP se

- y è di lunghezza al più polinomiale in $|x|$ con $1 \cdot 1$ dimensione di
- y è estratto casualmente da un insieme associato a x

A è un algoritmo di verifica polinomiale f.c.

$A(x, y)$ in tempo polinomiale afferma che x

non possiede la proprietà con certezza

oppure

che possiede la proprietà con probabilità > 0.5

Sì congettura

P C RP

C NP.