L'algoritmo AKS

Seminario per il corso di Elementi di Algebra Computazionale

Oscar Papini

22 luglio 2013

Come facciamo a sapere se un numero n è primo?

Definizione (Test di primalità)

Un *test di primalità* è un algoritmo che accetta in input un numero naturale e restituisce come output `PRIMO` oppure `COMPOSTO`.

Esempi di test di primalità:

Esempi di test di primalità:

1. Crivello di Eratostene [Costoso]

Esempi di test di primalità:

1. Crivello di Eratostene [Costoso]
2. Test di Fermat [Falsi positivi]

Esempi di test di primalità:

1. Crivello di Eratostene [Costoso]
2. Test di Fermat [Falsi positivi]
3. Test di Miller-Rabin [Probabilistico]

L'algoritmo AKS (Agrawal, Kayal, Saxena) è

L'algoritmo AKS (Agrawal, Kayal, Saxena) è

1. *polinomiale*: il tempo di esecuzione è polinomiale nel numero di cifre dell'input;

L'algoritmo AKS (Agrawal, Kayal, Saxena) è

1. *polinomiale*: il tempo di esecuzione è polinomiale nel numero di cifre dell'input;
2. *deterministico*: l'algoritmo restituisce `PRIMO` se e solo se l'input è un numero primo;

L'algoritmo AKS (Agrawal, Kayal, Saxena) è

1. *polinomiale*: il tempo di esecuzione è polinomiale nel numero di cifre dell'input;
2. *deterministico*: l'algoritmo restituisce `PRIMO` se e solo se l'input è un numero primo;
3. *generale*: l'algoritmo restituisce la risposta corretta su tutti gli input e non solo su una classe di numeri;

L'algoritmo AKS (Agrawal, Kayal, Saxena) è

1. *polinomiale*: il tempo di esecuzione è polinomiale nel numero di cifre dell'input;
2. *deterministico*: l'algoritmo restituisce `PRIMO` se e solo se l'input è un numero primo;
3. *generale*: l'algoritmo restituisce la risposta corretta su tutti gli input e non solo su una classe di numeri;
4. *non condizionato*: la correttezza e l'ordine di complessità non dipendono da congetture non ancora dimostrate.

Teorema

Siano $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geqslant 2$, *coprimi. Allora* n *è primo se e solo se in* $\mathbb{Z}/(n)[X]$

$$(X + a)^n = X^n + a. \tag{1}$$

Teorema

Siano $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geqslant 2$, coprimi. Allora n è primo se e solo se in $\mathbb{Z}/(n)[X]$

$$(X + a)^n = X^n + a. \tag{1}$$

Il coefficiente di X^i in $((X + a)^n - (X^n + a))$ è $\binom{n}{i} a^{n-i}$, per $0 < i < n$. Se n è primo, tutti i coefficienti binomiali sono multipli di n.

Teorema

Siano $a \in \mathbb{Z}$, $n \in \mathbb{N}$, $n \geqslant 2$, coprimi. Allora n è primo se e solo se in $\mathbb{Z}/(n)[X]$

$$(X + a)^n = X^n + a. \tag{1}$$

Il coefficiente di X^i in $((X + a)^n - (X^n + a))$ è $\binom{n}{i} a^{n-i}$, per $0 < i < n$. Se n è primo, tutti i coefficienti binomiali sono multipli di n.

Viceversa, se n è composto, sia q un suo fattore primo, e sia q^k la massima potenza di q che divide n. Allora q^k non divide $\binom{n}{q}$ ed è primo con a^{n-q}. Quindi il coefficiente di X^q è non nullo, e quindi tale polinomio non è identicamente nullo.

Quindi, per testare se n è primo, si sceglie a primo con n e si testa l'equazione (1). Il problema è che la verifica di tale equazione richiede la valutazione di n coefficienti.

Quindi, per testare se n è primo, si sceglie a primo con n e si testa l'equazione (1). Il problema è che la verifica di tale equazione richiede la valutazione di n coefficienti.

Per ridurre tale numero, si testa l'equazione (1) non solo modulo n, ma anche modulo $X^r - 1$ per un appropriato r. È sempre vero che se n è primo l'equazione (1) resta vera per ogni a ed r, tuttavia ci sono numeri composti che la soddisfano per alcuni valori di a ed r.

Quindi, per testare se n è primo, si sceglie a primo con n e si testa l'equazione (1). Il problema è che la verifica di tale equazione richiede la valutazione di n coefficienti.

Per ridurre tale numero, si testa l'equazione (1) non solo modulo n, ma anche modulo $X^r - 1$ per un appropriato r. È sempre vero che se n è primo l'equazione (1) resta vera per ogni a ed r, tuttavia ci sono numeri composti che la soddisfano per alcuni valori di a ed r.

Vedremo che, scelto opportunamente r, basta testare l'equazione per un certo numero di a per essere certi che n sia una potenza di un primo. Sia la scelta di r che il numero di a sono limitati da un polinomio in $\log n$.

Input: $n \in \mathbb{N}$, $n \geqslant 2$

1: Se $n = a^b$ per qualche $a \in \mathbb{N}$ e $b > 1$, **return** `COMPOSTO`
2: Trovo il più piccolo r tale che $o_r(n) > (\log n)^2$
3: Se $1 < (a, n) < n$ per qualche $a \leqslant r$, **return** `COMPOSTO`
4: Se $n \leqslant r$, **return** `PRIMO`
5: **for** $a = 1$ **to** $\lfloor \sqrt{\varphi(r)} \log n \rfloor$ **do**
6: Se $(X + a)^n \neq X^n + a \pmod{X^r - 1, n}$ **return** `COMPOSTO`
7: **end for**
8: **return** `PRIMO`

Lemma

Se n è primo, l'algoritmo restituisce `PRIMO`.

Infatti le condizioni alle righe 1, 3 e 6, che terminano l'algoritmo con output `COMPOSTO`, non possono mai essere soddisfatte nel caso in cui n sia primo.

Lemma

Se n è primo, l'algoritmo restituisce `PRIMO`.

Infatti le condizioni alle righe 1, 3 e 6, che terminano l'algoritmo con output `COMPOSTO`, non possono mai essere soddisfatte nel caso in cui n sia primo.

Supponiamo allora che l'output sia `PRIMO`. Se ciò è dovuto alla riga 4, allora n è sicuramente primo, perché altrimenti alla riga 3 avremmo trovato un fattore non banale di n. Per il resto della dimostrazione, dunque, supporremo di aver ottenuto `PRIMO` alla riga 8.

Iniziamo con il limitare il valore che può assumere r.

Lemma

Esiste $r \leqslant \max\{3, \lceil (\log n)^5 \rceil\}$ *tale che* $o_r(n) > (\log n)^2$.

Iniziamo con il limitare il valore che può assumere r.

Lemma

Esiste $r \leqslant \max\{3, \lceil (\log n)^5 \rceil\}$ *tale che* $o_r(n) > (\log n)^2$.

Per $n = 2$, $r = 3$ soddisfa la condizione. Supponiamo dunque $n > 2$, cosicché $\lceil (\log n)^5 \rceil > 10$ e si può applicare il seguente lemma.

Lemma

Sia $\mathrm{mcm}(m)$ *il minimo comune multiplo dei primi* m *naturali. Allora, se* $m \geqslant 7$, *si ha*

$$\mathrm{mcm}(m) \geqslant 2^m.$$

Consideriamo il più piccolo r che *non* divide il prodotto

$$n^{\lfloor \log B \rfloor} \cdot \prod_{i=1}^{\lfloor (\log n)^2 \rfloor} (n^i - 1)$$

con $B = \lceil (\log n)^5 \rceil$. Innanzitutto mostriamo che un tale r esiste, ed è minore di B. Supponiamo che tutti gli $s \leqslant B$ dividano il prodotto; allora anche il loro minimo comune multiplo lo divide. Ma

$$\begin{aligned} n^{\lfloor \log B \rfloor} \cdot \prod_{i=1}^{\lfloor (\log n)^2 \rfloor} (n^i - 1) &\leqslant n^{\lfloor \log B \rfloor + 1 + \cdots + \lfloor (\log n)^2 \rfloor} \\ &= n^{\lfloor \log B \rfloor + \lfloor (\log n)^2 \rfloor \left(\lfloor (\log n)^2 \rfloor + 1 \right)/2} \\ &< n^{\lfloor (\log n)^4 \rfloor} < 2^B \end{aligned}$$

e quindi $\mathrm{mcm}(B) < 2^B$, in contraddizione con il lemma.

Notiamo che, in generale, il massimo k per cui si può avere $m^k \leqslant B$, con $m \geqslant 2$, è $\lfloor \log B \rfloor$. In particolare, poiché $r \leqslant B$, si ha che per ogni fattore primo q di r

$$\max\{\alpha \in \mathbb{N} \mid q^\alpha \text{ divide } r\} \leqslant \lfloor \log B \rfloor$$

e quindi $r \mid \prod_{q|r} q^{\lfloor \log B \rfloor}$. Di conseguenza non tutti i fattori primi di r possono dividere n, perché altrimenti si avrebbe $r \mid n^{\lfloor \log B \rfloor}$.

Allora, posto $s := \frac{r}{(n,r)}$, si ha $s \neq 1$, $s \leqslant r$, e tuttavia s non divide il prodotto. Infatti, se lo dividesse, scrivendo $r = \prod p^{\alpha_p}$, si avrebbe che

- per ogni p che divide r ma che non divide n, $p^{\alpha_p} \mid s$ e quindi $p^{\alpha_p} \mid \prod(n^i - 1)$;
- per ogni p che divide sia r che n, $p^{\alpha_p} \mid n^{\lfloor \log B \rfloor}$ (ricordiamo che $\alpha_p \leqslant \lfloor \log B \rfloor$).

Quindi r divide il prodotto, contro la definizione di r.

Di conseguenza dev'essere $r = s$, e quindi r è primo con n. Infine, dato che r non divide il prodotto, necessariamente r non divide $\prod(n^i - 1)$ e quindi $o_r(n) > (\log n)^2$.

Di conseguenza dev'essere $r = s$, e quindi r è primo con n. Infine, dato che r non divide il prodotto, necessariamente r non divide $\prod(n^i - 1)$ e quindi $o_r(n) > (\log n)^2$.

Dato che $o_r(n) > 1$, esiste p primo che divide n tale che $o_r(p) > 1$. Notiamo che $p > r$, e che $p, n \in \mathbb{Z}_r^*$ (altrimenti l'algoritmo si sarebbe fermato prima). D'ora in avanti supporremo p, r fissati.

Per le ipotesi fatte, posto $\ell := \lfloor \sqrt{\varphi(r)} \log n \rfloor$, per ogni $0 \leqslant a \leqslant \ell$ si ha

$$(X + a)^n = X^n + a \pmod{X^r - 1, n}$$

Per le ipotesi fatte, posto $\ell := \lfloor \sqrt{\varphi(r)} \log n \rfloor$, per ogni $0 \leqslant a \leqslant \ell$ si ha

$$(X + a)^n = X^n + a \pmod{X^r - 1, n}$$

e quindi

$$(X + a)^n = X^n + a \pmod{X^r - 1, p};$$

Per le ipotesi fatte, posto $\ell := \lfloor \sqrt{\varphi(r)} \log n \rfloor$, per ogni $0 \leqslant a \leqslant \ell$ si ha

$$(X + a)^n = X^n + a \pmod{X^r - 1, n}$$

e quindi

$$(X + a)^n = X^n + a \pmod{X^r - 1, p};$$

d'altra parte, p è primo, quindi

$$(X + a)^p = X^p + a \pmod{X^r - 1, p};$$

Per le ipotesi fatte, posto $\ell := \lfloor \sqrt{\varphi(r)} \log n \rfloor$, per ogni $0 \leqslant a \leqslant \ell$ si ha

$$(X + a)^n = X^n + a \pmod{X^r - 1, n}$$

e quindi

$$(X + a)^n = X^n + a \pmod{X^r - 1, p};$$

d'altra parte, p è primo, quindi

$$(X + a)^p = X^p + a \pmod{X^r - 1, p};$$

È facile vedere che, combinando le ultime due, si ha infine

$$(X + a)^{n/p} = X^{n/p} + a \pmod{X^r - 1, p}.$$

Per le ipotesi fatte, posto $\ell := \lfloor \sqrt{\varphi(r)} \log n \rfloor$, per ogni $0 \leqslant a \leqslant \ell$ si ha

$$(X + a)^n = X^n + a \pmod{X^r - 1, n}$$

e quindi

$$(X + a)^n = X^n + a \pmod{X^r - 1, p};$$

d'altra parte, p è primo, quindi

$$(X + a)^p = X^p + a \pmod{X^r - 1, p};$$

È facile vedere che, combinando le ultime due, si ha infine

$$(X + a)^{n/p} = X^{n/p} + a \pmod{X^r - 1, p}.$$

Definizione

Diciamo che $m \in \mathbb{N}$ è *introspettivo* per $f(X) \in \mathbb{Z}[X]$ se

$$(f(X))^m = f(X^m) \pmod{X^r - 1, p}.$$

Lemma

Se m, m' sono introspettivi per $f(X)$, allora anche mm' lo è.

Lemma

Se m è introspettivo per $f(X)$ e $g(X)$, allora lo è anche per $f(X)g(X)$.

Lemma

Se m, m' sono introspettivi per $f(X)$, allora anche mm' lo è.

Lemma

Se m è introspettivo per $f(X)$ e $g(X)$, allora lo è anche per $f(X)g(X)$.

Quindi ogni numero dell'insieme

$$I := \left\{ \left(\frac{n}{p}\right)^i p^j \;\middle|\; i, j \geqslant 0 \right\}$$

è introspettivo per ogni polinomio dell'insieme

$$P := \left\{ \prod_{a=0}^{\ell} (X+a)^{e_a} \;\middle|\; e_a \geqslant 0 \right\}.$$

Definiamo ora

$$G := \{m \pmod r \mid m \in I\}.$$

Definiamo ora

$$G := \{m \pmod r \mid m \in I\}.$$

G è generato da n e p (modulo r), ed è quindi un sottogruppo di $(\mathbb{Z}_r)^*$. Posto $t := \#(G)$, si ha che $t > (\log n)^2$ (perché $o_r(n) > (\log n)^2$).

Definiamo ora

$$G := \{m \pmod r \mid m \in I\}.$$

G è generato da n e p (modulo r), ed è quindi un sottogruppo di $(\mathbb{Z}_r)^*$. Posto $t := \#(G)$, si ha che $t > (\log n)^2$ (perché $o_r(n) > (\log n)^2$).

Sia inoltre

$$\mathcal{G} := \{f \pmod{h(X), p} \mid f \in P\}$$

dove $h(X)$ è un fattore irriducibile dell'r-esimo polinomio ciclotomico su $\mathbb{F}_p$, che ha grado $o_r(p)$. $\mathcal{G}$ è generato da $X, X+1, \ldots, X+\ell$ nel campo $\mathbb{K} := {}^{\mathbb{F}_p[X]}/_{(h(X))}$, ed è quindi un sottogruppo di $\mathbb{K}^*$.

Lemma

$$\#(\mathcal{G}) \geqslant \binom{t+\ell}{t-1}.$$

Lemma

$$\#(\mathcal{G}) \geqslant \binom{t+\ell}{t-1}.$$

Intanto mostriamo che due polinomi in P di grado minore di t corrispondono a due elementi distinti di $\mathcal{G}$.

Infatti, supponiamo che $f(X), g(X) \in P$ con $\deg(f) < t$, $\deg(g) < t$ siano tali che $f(X) = g(X)$ in $\mathbb{K}$, e sia $m \in I$. Si ha ovviamente $f(X)^m = g(X)^m$ in $\mathbb{K}$, e quindi

$$f(X^m) = g(X^m) \text{ in } \mathbb{K}$$

(m è introspettivo per f e g, e $h(X)$ divide $X^r - 1$). Quindi X^m è radice di $Q(X) := f(X) - g(X)$ per ogni $m \in G$.

Ora, X è una radice primitiva r-esima dell'unità in $\mathbb{K}$, e $(m, r) = 1$ (perché $G < (\mathbb{Z}_r)^*$), quindi ogni X^m è radice primitiva r-esima dell'unità in $\mathbb{K}$.

Ora, X è una radice primitiva r-esima dell'unità in $\mathbb{K}$, e $(m, r) = 1$ (perché $G < (\mathbb{Z}_r)^*$), quindi ogni X^m è radice primitiva r-esima dell'unità in $\mathbb{K}$.

Ma allora $Q(X)$ ha t radici distinte in $\mathbb{K}$, tuttavia $\deg(Q) < t$ per le ipotesi su f e g. Ne segue allora che $f(X) \neq g(X)$.

A questo punto notiamo che se $i \neq j$, con $1 \leqslant i, j \leqslant \ell$, allora $i \neq j$ anche in $\mathbb{F}_p$, dato che

$$\ell = \lfloor \sqrt{\varphi(r)} \log n \rfloor < \sqrt{r} \log n \overset{(\star)}{<} r < p$$

(la disuguaglianza $(\star)$ discende dal fatto che $o_r(n) > (\log n)^2$, e quindi $r > (\log n)^2$).

Quindi $X, X+1, \ldots, X+\ell$ sono elementi distinti di $\mathbb{K}$, e nessuno di questi è nullo perché $\deg(h) > 1$.

A questo punto notiamo che se $i \neq j$, con $1 \leqslant i, j \leqslant \ell$, allora $i \neq j$ anche in $\mathbb{F}_p$, dato che

$$\ell = \lfloor \sqrt{\varphi(r)} \log n \rfloor < \sqrt{r} \log n \overset{(\star)}{<} r < p$$

(la disuguaglianza $(\star)$ discende dal fatto che $o_r(n) > (\log n)^2$, e quindi $r > (\log n)^2$).

Quindi $X, X+1, \ldots, X+\ell$ sono elementi distinti di $\mathbb{K}$, e nessuno di questi è nullo perché $\deg(h) > 1$.

Dunque ci sono almeno $\ell + 1$ polinomi distinti di grado 1 in $\mathcal{G}$, e un po' di combinatoria mostra che ci sono almeno $\binom{t+\ell}{t-1}$ polinomi distinti di grado minore di t in $\mathcal{G}$.

Lemma

Se n non è potenza di p, allora $\#(\mathcal{G}) \leqslant n^{\sqrt{t}}$.

Lemma

Se n non è potenza di p, allora $\#(\mathcal{G}) \leqslant n^{\sqrt{t}}$.

Consideriamo il sottoinsieme $J \subseteq I$ definito da

$$J := \left\{ \left(\frac{n}{p}\right)^i p^j \;\middle|\; 0 \leqslant i,j \leqslant \lfloor \sqrt{t} \rfloor \right\}.$$

Dato che n non è potenza di p, $\#(J) = (\lfloor \sqrt{t} \rfloor + 1)^2 > t$ elementi (distinti). Poiché $\#(G) = t$ per definizione, almeno due numeri distinti in J devono essere uguali modulo r.

Siano m_1, m_2 tali numeri (supponiamo $m_1 > m_2$). Dunque $X^{m_1} = X^{m_2} \pmod{X^r - 1}$. Questo implica che, se $f(X) \in P$, $f(X)^{m_1} = f(X)^{m_2}$ in $\mathbb{K}$, e quindi $f(X)$ è una radice (in $\mathbb{K}$) del polinomio $V(Y) := Y^{m_1} - Y^{m_2}$.

Siano m_1, m_2 tali numeri (supponiamo $m_1 > m_2$). Dunque $X^{m_1} = X^{m_2} \pmod{X^r - 1}$. Questo implica che, se $f(X) \in P$, $f(X)^{m_1} = f(X)^{m_2}$ in $\mathbb{K}$, e quindi $f(X)$ è una radice (in $\mathbb{K}$) del polinomio $V(Y) := Y^{m_1} - Y^{m_2}$.

Quindi V ha almeno $\#(\mathcal{G})$ radici distinte in $\mathbb{K}$, e il suo grado è

$$m_1 \leqslant \left(\frac{n}{p}p\right)^{\lfloor\sqrt{t}\rfloor} \leqslant n^{\sqrt{t}}$$

da cui $\#(\mathcal{G}) \leqslant n^{\sqrt{t}}$.

Tirando le somme, supponiamo che l'algoritmo abbia restituito `PRIMO` alla riga 8. Allora

$$\#(\mathcal{G}) \geqslant \binom{t+\ell}{t-1}$$

Tirando le somme, supponiamo che l'algoritmo abbia restituito `PRIMO` alla riga 8. Allora

$$\begin{aligned}\#(\mathcal{G}) &\geqslant \binom{t+\ell}{t-1} \\ &\geqslant \binom{\ell+1+\lfloor\sqrt{t}\log n\rfloor}{\lfloor\sqrt{t}\log n\rfloor}\end{aligned}$$

Tirando le somme, supponiamo che l'algoritmo abbia restituito `PRIMO` alla riga 8. Allora

$$\begin{aligned}
\#(\mathcal{G}) &\geqslant \binom{t+\ell}{t-1} \\
&\geqslant \binom{\ell+1+\lfloor\sqrt{t}\log n\rfloor}{\lfloor\sqrt{t}\log n\rfloor} \\
&\geqslant \binom{2\lfloor\sqrt{t}\log n\rfloor+1}{\lfloor\sqrt{t}\log n\rfloor}
\end{aligned}$$

Tirando le somme, supponiamo che l'algoritmo abbia restituito `PRIMO` alla riga 8. Allora

$$\begin{aligned}
\#(\mathcal{G}) &\geqslant \binom{t+\ell}{t-1} \\
&\geqslant \binom{\ell+1+\lfloor\sqrt{t}\log n\rfloor}{\lfloor\sqrt{t}\log n\rfloor} \\
&\geqslant \binom{2\lfloor\sqrt{t}\log n\rfloor+1}{\lfloor\sqrt{t}\log n\rfloor} \\
&> 2^{\lfloor\sqrt{t}\log n\rfloor+1} \geqslant n^{\sqrt{t}}.
\end{aligned}$$

Tirando le somme, supponiamo che l'algoritmo abbia restituito `PRIMO` alla riga 8. Allora

$$\begin{aligned}
\#(\mathcal{G}) &\geqslant \binom{t+\ell}{t-1} \\
&\geqslant \binom{\ell+1+\lfloor\sqrt{t}\log n\rfloor}{\lfloor\sqrt{t}\log n\rfloor} \\
&\geqslant \binom{2\lfloor\sqrt{t}\log n\rfloor+1}{\lfloor\sqrt{t}\log n\rfloor} \\
&> 2^{\lfloor\sqrt{t}\log n\rfloor+1} \geqslant n^{\sqrt{t}}.
\end{aligned}$$

Allora n è necessariamente potenza di p. Ma se $n = p^k$ con $k > 1$, la riga 1 lo avrebbe identificato come composto, quindi $k = 1$ e $n = p$.

Indicheremo con $\mathcal{O}^{\sim}(g(n)) = \mathcal{O}(g(n) \cdot \text{poly}(\log g(n)))$.
Analizziamo passo-passo l'algoritmo.

Indicheremo con $\mathcal{O}^{\sim}(g(n)) = \mathcal{O}(g(n) \cdot \text{poly}(\log g(n)))$.
Analizziamo passo-passo l'algoritmo.

Riga 1: il test può essere fatto in $\mathcal{O}^{\sim}((\log n)^3)$.

Indicheremo con $\mathcal{O}^{\sim}(g(n)) = \mathcal{O}(g(n) \cdot \text{poly}(\log g(n)))$.
Analizziamo passo-passo l'algoritmo.

Riga 1: il test può essere fatto in $\mathcal{O}^{\sim}((\log n)^3)$.

Riga 2: per trovare r, il metodo più semplice è verificare che $n^k \neq 1 \pmod{r}$, per $k = 1, \ldots, (\log n)^2$. Questo richiede $\mathcal{O}((\log n)^2)$ moltiplicazioni modulo r per ogni r, per un totale di $\mathcal{O}^{\sim}((\log n)^7)$.

Indicheremo con $\mathcal{O}^{\sim}(g(n)) = \mathcal{O}(g(n) \cdot \text{poly}(\log g(n)))$.
Analizziamo passo-passo l'algoritmo.

Riga 1: il test può essere fatto in $\mathcal{O}^{\sim}((\log n)^3)$.

Riga 2: per trovare r, il metodo più semplice è verificare che $n^k \neq 1 \pmod{r}$, per $k = 1, \ldots, (\log n)^2$. Questo richiede $\mathcal{O}((\log n)^2)$ moltiplicazioni modulo r per ogni r, per un totale di $\mathcal{O}^{\sim}((\log n)^7)$.

Riga 3: ogni GCD richiede $\mathcal{O}^{\sim}((\log n)^2)$ operazioni (con l'algoritmo euclideo), per un totale di $\mathcal{O}^{\sim}((\log n)^7)$ operazioni.

Il grosso del costo viene dalle **righe 5–7:** dobbiamo testare $\lfloor \sqrt{\varphi(r)} \log n \rfloor$ equazioni. Ciascuna richiede $\mathcal{O}(\log n)$ moltiplicazioni tra polinomi di grado al più r, con coefficienti scritti modulo n (e quindi di taglia $\mathcal{O}(\log n)$), per un totale di $\mathcal{O}^{\sim}(r(\log n)^2) = \mathcal{O}^{\sim}((\log n)^7)$. Quindi il costo totale è

$$\mathcal{O}^{\sim}(\sqrt{\varphi(r)}(\log n)(\log n)^7) = \mathcal{O}^{\sim}((\log n)^{10.5})$$

che è anche il costo totale dell'algoritmo.

Il grosso del costo viene dalle **righe 5–7:** dobbiamo testare $\lfloor \sqrt{\varphi(r)} \log n \rfloor$ equazioni. Ciascuna richiede $\mathcal{O}(\log n)$ moltiplicazioni tra polinomi di grado al più r, con coefficienti scritti modulo n (e quindi di taglia $\mathcal{O}(\log n)$), per un totale di $\mathcal{O}^{\sim}(r(\log n)^2) = \mathcal{O}^{\sim}((\log n)^7)$. Quindi il costo totale è

$$\mathcal{O}^{\sim}(\sqrt{\varphi(r)}(\log n)(\log n)^7) = \mathcal{O}^{\sim}((\log n)^{10.5})$$

che è anche il costo totale dell'algoritmo.

Il costo computazionale può essere abbassato migliorando la stima su r. Ci sono alcune congetture sulla densità dei numeri primi che suggeriscono la possibilità di trovare $r = \mathcal{O}((\log n)^2)$, portando il costo totale a $\mathcal{O}^{\sim}((\log n)^6)$. Nel tentativo di dimostrare queste congetture, sono stati provati lemmi che comunque permettono di avere $r = \mathcal{O}((\log n)^3)$, ottenendo un costo di $\mathcal{O}^{\sim}((\log n)^{7.5})$.