1.2 Addenda

A pagina 167-172 Nel paragrafo §6.4, che contiene la dimostrazione del Teorema di Agrawal-Kayal-Saxena, sono presenti vari misprint. In particolare quasi tutte le volte in cui compare $\mathbb{Z}_r$ va invece letto $\mathbb{Z}_r^*$. Inoltre negli ultimi anni sono state sviluppate delle varianti che migliorano la trattazione. In particolare si vedano gli articoli di Granville [17], Bernstein [4], Pomerance-Lenstra [37].

Riportiamo quindi in questa sede una nuova trattazione riveduta e corretta di tutta la sezione che segue da vicino la trattazione di Granville [17].

6.4 Criteri di primalità

Abbiamo già visto nel §3.4 i Teoremi di Lucas 3.4.9 e di Pocklington 3.4.10 che permettono di stabilire se un intero è primo o no, insieme a vari criteri di pseudoprimalità.

Presentiamo adesso il recente risultato di Agrawal, Kayal e Saxena [1] che fornisce prova del fatto che il problema di decisione "p è primo" appartiene alla classe di complessità P (ossia esiste un algoritmo deterministico con complessità polinomiale che risponde correttamente ad ogni istanza del problema, si veda l'Appendice A.1). La versione che descriveremo ha una dimostrazione completamente elementare e segue l'articolo originale di Agrawal, Kayal e Saxena [1] e la presentazione di Granville [17].

Negli ultimi anni sono state inoltre sviluppate delle varianti che ottimizzano la trattazione in vari punti; tra queste, oltre al già citato lavoro di Granville [17], ricordiamo anche quello di Bernstein [4].

Il fermento suscitato da questa scoperta ha portato molti esperti a lavorare sul tipo di approccio di Agrawal, Kayal e Saxena. Il risultato più importante è stato ottenuto da Lenstra e Pomerance [37] ed è stato presentato durante le "Journées Arithmétiques" tenutesi nel 2003 a Graz, in Austria. In tale articolo viene dimostrata la validità di una variante dell'algoritmo AKS avente complessità computazionale $\mathcal{O}\big(\log^{6+\epsilon} n\big)$ (usando gli algoritmi FFT, si veda l'Osservazione 6.1.12) che si ritiene essere la migliore stima ottenibile per la complessità computazionale degli algoritmi deterministici di primalità. Tale algoritmo però coinvolge aspetti matematici più avanzati di quelli che vogliamo utilizzare in questo testo mentre la versione che presenteremo, sebbene abbia una complessità computazionale maggiore (ma pur sempre polinomiale), utilizza solamente concetti algebrici che sono patrimonio del curriculum di una laurea di primo livello in Matematica e risultati elementari di Teoria dei Numeri (per la valutazione della complessità computazionale). Puntualizziamo inoltre che nella trattazione seguente gli aspetti algebrici sono tutti racchiusi nella dimostrazione del Teorema 6.4.2 e quindi, il Lettore interessato al solo algoritmo e alla sua complessità computazionale può, dopo aver letto l'enunciato del teorema sopra citato, passare direttamente allo schema dell'algoritmo.

Un'altra linea di ricerca si è concentrata sulle varianti probabilistiche del metodo AKS. I risultati più importanti sono dovuti indipendentemente a Berrizbeitia [7], Bernstein [6], Cheng [11] e a Mihăilescu e Avanzi [31]. Essi hanno trovato una variante dell'algoritmo AKS che in ogni iterazione ha probabilità $> 1/2$ di distinguere un intero primo da uno composto con complessità $\mathcal{O}\big(\log^{4+\epsilon} n\big)$ (usando gli algoritmi FFT). Ripetendone l'applicazione per k volte si ottiene quindi un algoritmo che ha probabilità di fallire $< 2^{-k}$ e che ha complessità $\mathcal{O}\big(k \log^{4+\epsilon} n\big)$. Il problema è chiaramente dato dal fatto che un tale algoritmo può non terminare e ciò dipende dal dover eseguire, in un passo dell'algoritmo stesso, una ricerca di radici dell'unità in una data estensione finita di $\mathbb{Z}_n$; si veda anche Granville [17].

Tali ricerche sono dovute al fatto che al momento non esistono implementazioni "praticamente" efficienti di AKS e quindi si è cercato, e si cerca tuttora, di migliorare il più possibile l'applicazione "pratica" di queste idee al fine di poter dimostrare la primalità di interi di "forma generale" sempre più grandi.

Infine, ricordiamo che Bernstein [5] ha recentemente scritto un esaustivo, sebbene schematico, resoconto dello sviluppo cronologico degli algoritmi di primalità.

Il risultato elementare su cui è basato l'algoritmo AKS è il

Teorema 6.4.1 *La congruenza*

$$(x+b)^n \equiv x^n + b \bmod n$$

vale per tutti i $b \in \mathbb{Z}$ se e solo se $n \geq 2$ è primo.

Dim. Se n è primo, per il Lemma 3.2.2 tutti i coefficienti binomiali $\binom{n}{k}$ con $0 < k < n$ sono divisibili per n e quindi qualunque sia l'intero b si ha

$$(x+b)^n \equiv x^n + b^n \bmod n.$$

La tesi segue dal Teorema di Fermat 3.2.1 nella forma (3.2.1).

Se n non è primo, si consideri un suo fattore primo p e la più alta potenza p^α di p che divide n: in altre parole, $p^\alpha \mid n$ ma $p^{\alpha+1} \nmid n$. Indicheremo questa relazione con la notazione $p^\alpha \parallel n$. Osserviamo che $p < n$ per ipotesi. Dunque il coefficiente binomiale

$$\binom{n}{p} = \frac{n(n-1)\cdots(n-p+1)}{p(p-1)\cdots 1} > 1$$

non è divisibile per p^α (e quindi non è divisibile neppure per n) poiché il numeratore è divisibile per p^α, ma c'è anche un fattore p a denominatore. In entrambi i prodotti, solo il primo fattore è divisibile per p. □

Il Teorema precedente può essere usato come test di primalità, ma la sua complessità computazionale è esponenziale perché vanno valutati n coefficienti del polinomio di sinistra. Si cerca allora di ridurre il numero di verifiche da fare valutando entrambi i membri del Teorema 6.4.1 modulo il polinomio $x^r - 1$, dove r è un intero opportunamente piccolo. Si ottiene il seguente

Teorema 6.4.2 (Agrawal, Kayal, Saxena [1]) *Sia $n \geq 4$ un intero e sia $r < n$ un intero positivo tale che n abbia ordine $d > \log_2^2 n$ in $\mathbb{Z}_r^*$. Allora n è primo se e solo se*

(i) n non è una potenza perfetta;

(ii) n non ha fattori primi $\leq r$;

(iii) $(x+b)^n \equiv x^n + b \pmod{x^r - 1, n}$ per ogni b intero, $1 \leq b \leq \sqrt{r}\log_2 n$.

Dim. Dal Teorema 6.4.1 sappiamo che se $n \geq 2$ è primo allora valgono le condizioni *(i)-(iii)*. Supponiamo adesso che le condizioni *(i)-(iii)* siano valide e assumiamo che $n \geq 2$ sia composto. Procederemo per assurdo: mostreremo che l'ipotesi che $n \geq 2$ sia composto conduce ad una contraddizione e quindi n dovrà necessariamente essere primo.

Sia $p \mid n$ un fattore primo di n e sia $A = \sqrt{r}\log_2 n$. Per l'ipotesi (iii) abbiamo immediatamente che

$$(x+b)^n \equiv x^n + b \bmod (x^r - 1, p) \tag{6.4.1}$$

per ogni intero b, $1 \leq b \leq \lfloor A \rfloor$. Fattorizziamo ora il polinomio $x^r - 1$ in prodotto di polinomi irriducibili di $\mathbb{Z}[x]$, ossia scriviamo $x^r - 1 = \prod_{d|r} \Phi_d(x)$ dove $\Phi_d(x)$ è il d-esimo polinomio ciclotomico (ricordiamo che le sue radici sono le radici primitive d-esime dell'unità). Ogni $\Phi_r(x)$ è irriducibile in $\mathbb{Z}[x]$, ma non è detto che lo sia in $\mathbb{Z}_p[x]$; sia quindi $h(x)$ un polinomio irriducibile in $\mathbb{Z}_p[x]$ che divide l'r-esimo polinomio ciclotomico $\Phi_r(x) \bmod p$. Abbiamo che

$$(x+b)^n \equiv x^n + b \bmod (h(x), p)$$

per ogni intero b, $1 \leq b \leq \lfloor A \rfloor$, perché $(h(x), p) \mid (x^r - 1, p)$. Osserviamo incidentalmente che le classi di congruenza $\bmod (h(x), p)$ appartengono al campo $\mathbb{F} := \mathbb{Z}[x]/(h(x), p)$ che è isomorfo al campo con p^m elementi, dove m indica il grado di $h(x)$. In particolare gli elementi non nulli di $\mathbb{F}$ formano un gruppo di ordine $p^m - 1$ e tra essi vi è anche l'elemento x. Consideriamo ora l'insieme G costituito da tutti i possibili prodotti delle quantità:

$$\Big\{ x + b : b \in \{0, 1, \ldots, \lfloor A \rfloor\} \Big\}. \tag{6.4.2}$$

Tutti gli elementi di G devono essere non nulli in $\mathbb{F}$ perché altrimenti da $x + b = 0$ in $\mathbb{F}$ seguirebbe $x^n + b = (x+b)^n = 0$ per l'ipotesi (iii). Quindi $x^n = -b = x$ in $\mathbb{F}$ da cui si otterrebbe $n \equiv 1 \bmod r$. Pertanto $d = 1$ e ciò contraddirebbe $d > \log_2^2 n$. Abbiamo quindi dimostrato che tutti gli elementi di G devono essere non nulli in $\mathbb{F}$.

Possiamo allora concludere che è G il sottogruppo ciclico di $\mathbb{F}^*$ generato moltiplicativamente dagli elementi elencati nella (6.4.2). Sia inoltre H il sottoinsieme di $\big(\mathbb{Z}[x]/ \bmod (x^r - 1, p)\big)$ generato moltiplicativamente da $\{x + b : b \in \{0, 1, \ldots, \lfloor A \rfloor\}$. In altre parole: G è la riduzione di $H \bmod (h(x), p)$.

Esaminiamo adesso alcune proprietà di H. Il generico elemento $g \in H$ può essere scritto come $g(x) = \prod_{0 \le b \le \lfloor A \rfloor} (x+b)^{e_b}$ e, per (iii) e (6.4.1), verifica

$$g(x)^n = \prod_{0 \le b \le \lfloor A \rfloor} (x+b)^{ne_b} \equiv \prod_{0 \le b \le \lfloor A \rfloor} (x^n+b)^{e_b} = g(x^n) \bmod (x^r-1, p).$$

Definiamo ora

$$S = \{k \in \mathbb{Z},\ k > 0 \text{ tale che } g(x)^k \equiv g(x^k) \bmod (x^r-1, p) \text{ per ogni } g \in H\}.$$

Osserviamo che $n \in S$ per l'ipotesi (iii) e che $p \in S$ per il Teorema 6.4.1. Stimiamo ora la cardinalità di G e per fare ciò ci serviremo dei seguenti lemmi su S.

Lemma 6.4.3 *Se $a, b \in S$ allora $ab \in S$.*

Dim. del Lemma 6.4.3. Se $g(x) \in H$ allora $g(x)^b \equiv g(x^b) \bmod (x^r-1, p)$. Rimpiazzando adesso x con x^a otteniamo $g(x^a)^b \equiv g(x^{ab}) \bmod (x^{ar}-1, p)$ e, *a fortiori*, $\bmod (x^r-1, p)$ visto che x^r-1 divide $x^{ar}-1$. Pertanto

$$g(x)^{ab} = (g(x)^a)^b \equiv g(x^a)^b \equiv g((x^a)^b) = g(x^{ab}) \bmod (x^r-1, p)$$

ossia $ab \in S$. □

Lemma 6.4.4 *Se $a, b \in S$ e $a \equiv b \bmod r$ allora $a \equiv b \bmod \mathrm{card}(G)$.*

Dim. del Lemma 6.4.4. Se $g(x) \in \mathbb{Z}[x]$ e $u, v \in \mathbb{Z}$ allora è immediato osservare che $u - v$ divide $g(u) - g(v)$. Poiché $a \equiv b \bmod r$ abbiamo che $x^r - 1$ divide $x^{a-b} - 1$ che d'altra parte divide $x^a - x^b$. Per l'osservazione precedente concludiamo che $x^a - x^b$ divide $g(x^a) - g(x^b)$ per $g(x) \in \mathbb{Z}[x]$. Se $g(x) \in H$, abbiamo anche che $g(x)^a \equiv g(x^a) \equiv g(x^b) \equiv g(x)^b \bmod (x^r-1, p)$. Supponendo inoltre che $g(x) \in G$, otteniamo $g(x)^{a-b} \equiv 1$ in $\mathbb{F}$. Ricordando che G è ciclico, possiamo concludere che la proprietà precedente debba valere anche quando g è un generatore di G. Allora $\mathrm{card}(G) \mid a - b$. □

Lemma 6.4.5 *Abbiamo che $n/p \in S$.*

Dim. del Lemma 6.4.5. Siano $a \in S$ e $b \equiv a \bmod (n^d - 1)$, dove $d = \mathrm{ord}_{\mathbb{Z}_r^*} n$. Vogliamo dimostrare che $b \in S$. Poiché $n^d \equiv 1 \bmod r$, abbiamo che $x^{n^d} \equiv x \mod (x^r - 1)$. Quindi $x^r - 1$ divide $x^{n^d} - x$ che a sua volta divide $x^b - x^a$. Sia ora $g(x) \in \mathbb{Z}[x]$; per quanto visto nella dimostrazione del Lemma 6.4.4, $x^b - x^a$ divide $g(x^b) - g(x^a)$. Segue che $g(x^{n^d}) \equiv g(x) \bmod (x^{n^d} - x, p)$ e quindi $g(x^{n^d}) \equiv g(x) \bmod (x^r - 1, p)$ visto che $(x^r - 1) \mid (x^{n^d} - x)$. D'altra parte dal fatto che $n \in S$ si ottiene $n^d \in S$ e, per il Lemma 6.4.3, si ha che $g(x)^{n^d} \equiv g(x^{n^d}) \mod (x^r - 1, p)$ per ogni $g(x) \in H$. Pertanto $g(x)^{n^d} \equiv g(x) \mod (x^r - 1, p)$ per ogni $g(x) \in H$. Ma $(n^d - 1) \mid (b - a)$ e quindi $g(x)^b \equiv g(x)^a \mod (x^r - 1, p)$. In conclusione, sfruttando l'ipotesi $a \in S$, abbiamo che

$$g(x^b) \equiv g(x^a) \equiv g(x)^a \equiv g(x)^b \mod (x^r - 1, p)$$

per ogni $g(x) \in H$, ossia $b \in S$. Scegliamo ora a e b in modo opportuno da permettere di concludere che $n/p \in S$. Dato che $p, n \in S$, $a = np^{\varphi(n^d-1)-1} \in S$ per il Lemma 6.4.3. Sia ora $b = n/p$. Osservando, per il Teorema di Eulero 3.2.8, che $p^{\varphi(n^d-1)} \equiv 1 \mod (n^d - 1)$ si ottiene che che $b \equiv a \bmod (n^d - 1)$. Dato che $a \in S$, per quanto visto sopra abbiamo finalmente che $n/p \in S$. □

Procediamo adesso a stimare dall'alto $\mathrm{card}(G)$. Sia R il sottogruppo di $\mathbb{Z}_r^*$ generato da n/p e da p, cioè $R =< n/p, p > \mathbb{Z}_r^*$. Siccome n non è né un primo né una potenza di p, anche n/p non è una potenza di p e quindi gli interi $(n/p)^i p^j$ al variare di $0 \le i, j$ sono distinti tra loro. Siccome esistono più di $\mathrm{card}(R)$ interi di questo tipo per $0 \le i, j \le \sqrt{\mathrm{card}(R)}$, per il principio dei cassetti (si veda l'Appendice A.2), devono esistere almeno due coppie distinte (i, j), (k, l) tali che

$$\left(\frac{n}{p}\right)^i p^j \equiv \left(\frac{n}{p}\right)^k p^l \bmod r.$$

Per i Lemmi 6.4.5 e 6.4.3 entrambe le coppie sopra scritte devono essere elementi di S. Inoltre per il Lemma 6.4.4, $\mathrm{card}(G)$ divide $(n/p)^i p^j - (n/p)^k p^l$ e quindi, ricordando che la distanza massima tra due elementi compresi tra 1 e $((n/p)p)^{\sqrt{\mathrm{card}(R)}} = n^{\sqrt{\mathrm{card}(R)}}$ è al massimo $n^{\sqrt{\mathrm{card}(R)}} - 1$, si ottiene

$$\mathrm{card}(G) \le \left| \left(\frac{n}{p}\right)^i p^j - \left(\frac{n}{p}\right)^k p^l \right| \le n^{\sqrt{\mathrm{card}(R)}} - 1. \tag{6.4.3}$$

Per ottenere la contraddizione che stiamo cercando esaminiamo ora stime dal basso per $\mathrm{card}(G)$. Abbiamo bisogno di un lemma che, data una congruenza tra polinomi di G, permetta di ottenere che tali polinomi sono congrui anche in $\mathbb{Z}_p[x]$.

Lemma 6.4.6 *Siano $f(x), g(x) \in \mathbb{Z}[x]$ tali che $f(x) \equiv g(x) \bmod (h(x), p)$ e che le riduzioni di f e g in $\mathbb{F}$ siano elementi di G. Se $\mathrm{grado}(f)$, $\mathrm{grado}(g) < \mathrm{card}(R)$ allora $f(x) \equiv g(x) \bmod p$.*

Dim. del Lemma 6.4.6. Sia $\Delta(y) = f(y) - g(y) \in \mathbb{Z}[y]$. Considereremo la riduzione di Δ in $\mathbb{F}$. Se $k \in S$ abbiamo che

$$\Delta(x^k) = f(x^k) - g(x^k) \equiv f(x)^k - g(x)^k \equiv 0 \bmod (h(x), p).$$

Siccome x ha ordine r in $\mathbb{F}$ allora l'insieme $\{x^k : k \in R\}$ è formato da radici distinte di $\Delta(y) \bmod (h(x), p)$. Poiché sia $f(x)$ che $g(x)$ hanno grado minore di $\mathrm{card}(R)$ anche $\Delta(y)$ ha grado minore di $\mathrm{card}(R)$. D'altra parte $\Delta(y)$ ha un numero di radici $\ge \mathrm{card}(R)$ modulo $\bmod(h(x), p)$ ossia nel campo $\mathbb{F}$. Ma ciò non può accadere per il Teorema 2.3.15 a meno che Δ non sia identicamente nullo in $\mathbb{F}$. Quindi $\Delta(y) \equiv 0 \bmod (h(x), p)$. Da ciò segue immediatamente che $\Delta(y) \equiv 0 \bmod p$ visto che i coefficienti di $\Delta(y)$ non dipendono da x. □

Abbiamo ora tutti gli ingredienti per stimare dal basso $\mathrm{card}(G)$. Ricordiamo che R contiene tutti gli elementi generati da $n \bmod r$ e quindi $\mathrm{card}(R) \ge d = \mathrm{ord}_{\mathbb{Z}_r^*}(n) > \log_2^2 n$ per ipotesi. Quindi, posto $B = \lfloor \sqrt{\mathrm{card}(R)} \log_2 n \rfloor$, si ha $A = \sqrt{r} \log_2 n > B$ (perché $\mathrm{card}(R) < r$) ed inoltre $\mathrm{card}(R) > B$. Osserviamo che, per ogni sottoinsieme proprio T di $\{0, 1, \dots, B\}$, i prodotti $\prod_{b \in T}(x + b)$ forniscono elementi distinti di G. Infatti, se esistessero due sottoinsiemi propri distinti T_1, T_2 di $\{0, 1, \dots, B\}$ per cui $\prod_{\alpha \in T_1}(x + \alpha) = \prod_{\beta \in T_2}(x + \beta)$ in G, allora, grazie al Lemma 6.4.6, i due prodotti in questione sarebbero identici anche in $\mathbb{Z}_p[x]$ e quindi esisterebbe almeno una coppia $\alpha \ne \beta$ tale che $p \mid \alpha - \beta$. Dato che $\alpha, \beta \le B < \sqrt{r} \log_2 n$, abbiamo che $p < \sqrt{r} \log_2 n$. Ma, per l'ipotesi (ii), sappiamo che $p > r$ da cui segue $r < \log_2^2 n$ che è in contraddizione con $\mathrm{ord}_{\mathbb{Z}_r^*}(n) > \log_2^2 n$ (perché $r > \mathrm{ord}_{\mathbb{Z}_r^*}(n)$). Pertanto, dato che il numero di sottoinsiemi propri di un insieme finito U è $2^{\mathrm{card}(U)} - 1$ e che $n = 2^{\log_2 n}$, si ha che

$$\mathrm{card}(G) \ge 2^{B+1} - 1 = 2^{\lfloor \sqrt{\mathrm{card}(R)} \log_2 n \rfloor + 1} - 1 > 2^{\sqrt{\mathrm{card}(R)} \log_2 n} - 1 = n^{\sqrt{\mathrm{card}(R)}} - 1 \tag{6.4.4}$$

che è in contraddizione con l'equazione (6.4.3).

Allora l'ipotesi che n fosse composto deve essere falsa e quindi n è primo. □

Schema dell'algoritmo AKS

Una possibile schematizzazione dell'algoritmo di Agrawal-Kayal-Saxena è:

1) se ($n = \alpha^\beta$ per $\alpha, \beta \in \mathbb{N}$ e $\beta > 1$) allora si dia in output COMPOSTO e l'esecuzione termini;

2) determinare il minimo r per cui n abbia ordine $d \ge \lceil \log_2^2 n \rceil$ in $\mathbb{Z}_r^*$;

3) se $1 < (b, n) < n$ per qualche $b \le r$, allora si dia in output COMPOSTO e l'esecuzione termini;

4) se $n \le r$, allora si dia in output PRIMO e l'esecuzione termini;

5) per ogni b intero, $1 \le b \le \sqrt{r} \log_2 n$, si calcoli se:

$$(x + b)^n \not\equiv x^n + b \pmod{x^r - 1, n};$$

in tal caso si dia in output COMPOSTO e l'esecuzione termini;

6) si dia in output PRIMO e l'esecuzione termini.

Esaminiamo adesso la correttezza dell'algoritmo sopra esposto.

Teorema 6.4.7 (Correttezza di AKS) *L'algoritmo AKS risponde correttamente al problema della primalità.*

Dim. Supponiamo che n sia primo. Allora l'algoritmo non può terminare nei passi 1) e 3). Per il Teorema 6.4.1 non può terminare neanche nel passo 5). Quindi l'algoritmo termina nel passo 4) o nel passo 6) determinando correttamente la risposta.

Viceversa, nel caso l'algoritmo termini dichiarando PRIMO nel passo 4) allora n deve essere primo perché altrimenti nel passo 3) sarebbe stato determinato un suo fattore. Alternativamente se l'algoritmo termina dichiarando PRIMO nel passo 6) sappiamo che sono verificate le ipotesi *(i)-(iii)* del Teorema 6.4.2. Infatti nel passo 1) si è controllato che $n \neq p^\alpha$, $\alpha \geq 2$ ossia che n non è una potenza perfetta e nel passo 3) si è provato che n non ha fattori primi $\leq r$. Nel passo 5) si è studiata la correttezza delle congruenze polinomiali previste in (iii) e nel passo 2) si è verificata l'ipotesi sull'ordine di n mod r. Allora n è effettivamente primo e la risposta dell'algoritmo è corretta. □

Complessità computazionale di AKS

L'algoritmo AKS è basato sul fatto di poter determinare almeno un r intero in modo che valgano le ipotesi del Teorema 6.4.2 e che la verifica dell'ipotesi (iii) abbia complessità polinomiale (perché ciò sia vero si noti che è sufficiente dimostrare che tale r sia minore o uguale di un polinomio in $\log n$). Ricordiamo che $\log_2 n$ è il logaritmo in base 2 di n. Ci servono alcuni risultati preliminari.

Lemma 6.4.8 (Nair [34]) *Sia $m \in \mathbb{N}$ e sia $m \geq 7$. Allora* $\operatorname{lcm}\{1, \ldots, m\} \geq 2^m$.

Una dimostrazione di un risultato equivalente al Lemma 6.4.8 è data nel Lemma 8.1.1, ma, per completezza, ne includiamo un'altra qui.

Dim. Indichiamo con d_m la quantità $\operatorname{lcm}\{1, \ldots, m\}$. Sia inoltre n un intero, $1 \leq n \leq m$, e

$$I(n,m) = \int_0^1 t^{n-1}(1-t)^{m-n}\mathrm{d}t.$$

Calcoliamo $I(m,n)$ in due modi diversi. Il primo è usando l'espansione binomiale di $(1-t)^{m-n}$:

$$(1-t)^{m-n} = \sum_{j=0}^{m-n} \binom{m-n}{j}(-t)^j.$$

In tal modo abbiamo che

$$I(n,m) = \sum_{j=0}^{m-n} \binom{m-n}{j}\frac{(-1)^j}{n+j}$$

che è chiaramente un numero razionale. È anche immediato verificare che il suo denominatore divide d_m. Pertanto abbiamo che

$$d_m I(n,m) \in \mathbb{N}. \tag{6.4.5}$$

Usiamo adesso l'integrazione per parti su $I(n,m)$ usando $(1-t)^{m-n}$ come fattore differenziale. Otteniamo che

$$\begin{aligned} I(n,m) &= -\frac{t^{n-1}(1-t)^{m-n+1}}{m-n+1}\Bigg|_0^1 + \frac{n-1}{m-n+1}\int_0^1 t^{n-2}(1-t)^{m-n+1}\mathrm{d}t \\ &= \frac{n-1}{m-n+1}\int_0^1 t^{n-2}(1-t)^{m-n+1}\mathrm{d}t = \frac{n-1}{m-n+1}I(n-1,m). \end{aligned}$$

Ripetendo lo stesso calcolo $n-1$ volte abbiamo

$$\begin{aligned} I(n,m) &= \frac{(n-1)(n-2)\cdots 2\cdot 1}{(m-n+1)(m-n+2)\cdots(m-1)}\, I(1,m) \\ &= \frac{(n-1)(n-2)\cdots 2\cdot 1}{(m-n+1)(m-n+2)\cdots(m-1)} \int_0^1 (1-t)^{m-1}\mathrm{d}t \\ &= \frac{(n-1)(n-2)\cdots 2\cdot 1}{(m-n+1)(m-n+2)\cdots(m-1)m} = \frac{1}{n\binom{m}{n}}. \end{aligned} \tag{6.4.6}$$

Confrontando (6.4.5) e (6.4.6), è chiaro che $n\binom{m}{n} \mid d_m$ per ogni n intero, $1 \le n \le m$.

Pertanto, come caso particolare, abbiamo che $m\binom{2m}{m} \mid d_{2m}$. Osservando che vale l'identità

$$(2m+1)\binom{2m}{m} = (m+1)\binom{2m+1}{m+1},$$

possiamo scrivere che $(2m+1)\binom{2m}{m} \mid d_{2m+1}$. Dato che $(m, 2m+1) = 1$ e che $d_{2m} \mid d_{2m+1}$, concludiamo

$$m(2m+1)\binom{2m}{m} \mid d_{2m+1}.$$

Ma, considerando lo sviluppo binomiale di $(1+1)^{2m}$ e osservando che $\binom{2m}{m}$ è il massimo coefficiente presente, abbiamo immediatamente che $(2m+1)\binom{2m}{m} \ge 2^{2m}$. Quindi

$$d_{2m+1} \ge m2^{2m}. \tag{6.4.7}$$

Applicando la (6.4.7) abbiamo che

$$d_{2m+1} \ge 2\cdot 2^{2m} = 2^{2m+1} \quad \text{per ogni} \quad m \ge 2$$

e che

$$d_{2m+2} \ge d_{2m+1} \ge 4\cdot 2^{2m} = 2^{2m+2} \quad \text{per ogni} \quad m \ge 4.$$

Le ultime due disequazioni implicano che $d_m \ge 2^m$ per ogni $m \ge 9$. Il Lemma segue osservando che $d_8 = 840 > 2^8$ e $d_7 = 420 > 2^7$. □

Lemma 6.4.9 *Sia $n \ge 4$. Allora esiste almeno un $r \le \lceil \log_2^5 n \rceil$ tale che $d = \mathrm{ord}_{\mathbb{Z}_r^*}(n) > \log_2^2 n$.*

Dim. Siccome $n \ge 4$, si ha che $\lceil \log_2^5 n \rceil \ge 32$ e si può applicare il Lemma 6.4.8. Siano ora $V = \lceil \log_2^5 n \rceil$,

$$\Pi = n^{\lfloor \log_2 V \rfloor} \prod_{i=1}^{\lfloor \log_2^2 n \rfloor} (n^i - 1) \quad \text{e} \quad \mathcal{V} = \Big\{ s \in \{1,\dots,V\} : s \nmid \Pi \Big\}.$$

Supponiamo per assurdo che $\mathcal{V} = \emptyset$. Allora abbiamo che ogni elemento di $\{1,\dots,V\}$ divide Π. Pertanto $\mathrm{lcm}\{1,\dots,V\}$ divide

$$\Pi \le n^{\lfloor \log_2 V \rfloor + \sum_{i=1}^{\lfloor \log_2^2 n \rfloor} i} = n^{\lfloor \log_2 V \rfloor + (1/2)\lfloor \log_2^2 n \rfloor (\lfloor \log_2^2 n \rfloor + 1)} < n^{\lfloor \log_2^4 n \rfloor} < 2^V,$$

perché $n \ge 4$ e $n = 2^{\log_2 n}$. D'altra parte, per il Lemma 6.4.8, si ha che il minimo comune multiplo dei primi V interi è maggiore o uguale a 2^V e ciò contraddice quanto sopra.

Quindi $\mathcal{V} \ne \emptyset$. Sia $r = \min \mathcal{V}$ e sia q un divisore primo di r. Dato che $r \le V$, si ha che $\max\{\alpha \in \mathbb{N} : q^\alpha | r\} \le \lfloor \log_2 V \rfloor$ e quindi $r \mid \prod_{q|r} q^{\lfloor \log_2 V \rfloor}$. Osserviamo inoltre che, se ogni divisore primo di r dividesse anche n, si avrebbe che

$$r \mid \prod_{q|r} q^{\lfloor \log_2 V \rfloor} \mid n^{\lfloor \log_2 V \rfloor}$$

e quindi $r \notin \mathcal{V}$. Ciò contraddirebbe $r = \min \mathcal{V}$. È chiaro quindi che non tutti i divisori primi di r dividono anche n. Di conseguenza $r/(r,n) \in \mathcal{V}$: infatti, se cosí non fosse, posto $r = \prod_{p|r} p^{\alpha_r}$, si avrebbe che per

ogni $p \mid r$ e $p \nmid n$ avremmo che p^{α_r} dividerebbe $\prod_{i=1}^{\lfloor \log_2^2 n \rfloor}(n^i - 1)$ mentre, se $p \mid r$ e $p \mid n$, avremmo, dato che $\alpha_r \leq \lfloor \log_2 V \rfloor$, che $p^{\alpha_r} \mid n^{\lfloor \log_2 V \rfloor}$ e quindi, in conclusione $r \mid \Pi$ che è in contraddizione con $r \in \mathcal{V}$. Dato che $r/(r,n) \leq r$ e $r = \min \mathcal{V}$, otteniamo che $(r,n) = 1$. Pertanto $r \nmid n$ e quindi $r \nmid \prod_{i=1}^{\lfloor \log_2^2 n \rfloor}(n^i - 1)$ da cui segue immediatamente che $\mathrm{ord}_{\mathbb{Z}_r^*}(n) > \log_2^2 n$. □

Passiamo adesso a stimare la complessità computazionale dell'algoritmo AKS.

Consideriamo la complessità del passo 1). Osserviamo che è sufficiente calcolare, con l'aritmetica reale, $n^{1/2}$, $n^{1/3}$, ecc. e controllare se gli interi vicini a tali risultati, quando elevati alla potenza opportuna, forniscono n. Si noti che questo controllo richiede al più $\lfloor \log_2 n \rfloor$ passi (si calcolano tutte le radici $n^{1/k}$ fino a $k = \lfloor \log_2 n \rfloor$). Ognuno di questi passi comprende una radice reale ed una potenza. Il calcolo della potenza ha complessità $\mathcal{O}\big(\log^3 n\big)$, mentre il calcolo della radice può essere svolto mediante il metodo di Newton di approssimazioni successive (si vedano Crandall e Pomerance [13, §9.2] e l'Esercizio 9.15) con complessità $\mathcal{O}\big(\log^{2+\epsilon} n\big)$. Pertanto la complessità del passo 1) è $\mathcal{O}\big(\log^{4+\epsilon} n\big)$.

Per stimare la complessità del passo 2) sappiamo, per il Lemma 6.4.9, che è sufficiente ricercare tra tutti gli interi $\leq \lceil \log_2^5 n \rceil$ un r per cui $d = \mathrm{ord}_{\mathbb{Z}_r^*}(n) > \log_2^2 n$. Per un fissato r vanno calcolate le congruenze $n^k \not\equiv 1 \bmod r$ per ogni $k \leq \log_2^2 n$. Ossia vanno eseguite $\mathcal{O}\big(\log^2 n\big)$ potenze modulo r. Ognuna di esse richiede $\mathcal{O}(\log k)$ quadrati modulo r, si veda il §6.9.2, al costo di $\mathcal{O}\big(\log^2 r\big)$ operazioni elementari per ogni quadrato. Quindi sono necessarie $\mathcal{O}\big(\log^2 n \log k \log^2 r\big) = \mathcal{O}\big(\log^{2+\epsilon} n\big)$ operazioni elementari per una verifica su un dato r. Siccome la verifica va fatta per tutti gli $r \leq \lceil \log_2^5 n \rceil$ abbiamo che il passo 2) dell'algoritmo ha una complessità computazionale $\mathcal{O}\big(\log^{7+\epsilon} n\big)$ operazioni elementari.

Il passo 3) dell'algoritmo richiede il calcolo di un massimo comun divisore da ripetere per r volte. Visto che l'algoritmo di Euclide, si veda la §6.2, consente il calcolo del massimo comun divisore (b,n), $b \leq r$, con $\mathcal{O}\big(\log^2 \max(r;n)\big)$ operazioni elementari si ha che il passo 3) ha un costo computazionale totale pari a $\mathcal{O}\big(r \log^2 \max(r;n)\big)$. Tale stima, per n sufficientemente grande[1] e per $r \leq \lceil \log_2^5 n \rceil$, diviene $\mathcal{O}\big(\log^7 n\big)$ operazioni elementari.

Nel passo 5) la relazione (iii) va verificata su ogni elemento b, $1 \leq b \leq \sqrt{r} \log_2 n$. Poiché lavoriamo modulo $x^r - 1$, ogni polinomio ha al più grado $r-1$ e, siccome i suoi coefficienti sono minori di n, per ogni b fissato il calcolo del termine di sinistra della (iii) ha, per quanto esposto nel §6.9.3, complessità $\mathcal{O}\big(r^2 \log^3 n\big)$ operazioni elementari. Il termine di destra della (iii) è molto più semplice da calcolare: infatti, se $n = qr + \ell$, dove $q, \ell \in \mathbb{N}^*$, $\ell < r$, abbiamo che $x^n = (x^r - 1)(x^{n-r} + x^{n-2r} + \ldots + x^{n-qr}) + x^\ell \equiv x^\ell \bmod (x^r - 1)$ e pertanto è sufficiente calcolare una sola divisione euclidea per determinare $x^n \bmod (x^r - 1)$. Va poi calcolato $b \bmod n$ per ogni b fissato, $1 \leq b \leq \sqrt{r} \log_2 n$. Il costo di questo passo è pari a quello di una divisione euclidea. Per ogni b fissato, il costo totale del calcolo del lato destro della (iii) è pertanto dominato dal costo di calcolo del termine di sinistra.

Allora la verifica complessiva delle congruenze polinomiali del passo 5) ha una complessità $\mathcal{O}\big(r^{5/2} \log^4 n\big)$ operazioni elementari. Usando il fatto che $r \leq \lceil \log_2^5 n \rceil$, si ha che la complessità del passo 5) è $\mathcal{O}\Big(\log^{33/2} n\Big)$ operazioni elementari.

Osservazione 6.4.10 *Si noti che se viene impiegata FFT allora il passo precedente ha complessità $\mathcal{O}\big(r \log^{2+\epsilon} n\big)$; quindi la verifica su tutti i polinomi ha complessità $\mathcal{O}\big(r^{3/2} \log^{3+\epsilon} n\big)$. In definitiva, utilizzando tale tecnica, la complessità della verifica sui polinomi è $\mathcal{O}\Big(\log^{21/2+\epsilon} n\Big)$ operazioni elementari.*

Da quanto abbiamo visto sopra, possiamo allora concludere che la complessità computazionale dell'algoritmo AKS è $\mathcal{O}\Big(\log^{33/2} n\Big)$ operazioni elementari e, per l'osservazione alla fine del paragrafo precedente, sappiamo che l'algoritmo risponde PRIMO se e solo se n è primo. Quindi abbiamo provato il

Teorema 6.4.11 (Agrawal-Kayal-Saxena) *Il problema di decidere se un intero è primo o meno appartiene alla classe di complessità P.*

[1] È sufficiente avere $n \geq \lceil \log_2^5 n \rceil$ ossia $n \geq 5690034$.