Bollettino
Unione Matematica Italiana

Sezione A – La Matematica nella Società e nella Cultura

Luisella Caire, Umberto Cerruti

Numeri primi: la certezza

Bollettino dell'Unione Matematica Italiana, Serie 8, Vol. **10-A**—*La Matematica nella Società e nella Cultura* (2007), n.1, p. 85–117.

Unione Matematica Italiana

<http://www.bdim.eu/item?id=BUMI_2007_8_10A_1_85_0>

Bollettino dell'Unione Matematica Italiana, Unione Matematica Italiana, 2007.

Bollettino U. M. I.
La Matematica nella Società e nella Cultura
Serie VIII, Vol. X-A, Aprile 2007, 85-117

Numeri primi: la certezza

LUISELLA CAIRE - UMBERTO CERRUTI

Tutte le volte che puoi, conta.
Sir Francis Galton

Tutti questi numeri mi fanno stare male.
William Shakespeare, Amleto

1. – Introduzione.

In un precedente articolo ([11]) abbiamo parlato di tre tipi di algoritmi rivolti a verificare se un dato intero N è primo:

– algoritmi *deterministici*, cioè che rispondono con certezza SÌ o NO a seconda che N sia primo o meno (sono i *criteri di primalità*); fino al 2002 tutti gli algoritmi di questo tipo risultavano inefficienti (tecnicamente *non polinomiali*), in pratica erano troppo lenti per poterli utilizzare con interi grandi

– algoritmi *probabilistici* che danno la certezza quando rispondono NO (ossia dicono che N è composto) mentre nel caso di risposta SÌ assicurano soltanto un limite inferiore alla probabilità che N sia primo

– algoritmi che sono al tempo stesso deterministici ed efficienti, ma sono *condizionati*, cioè dipendono dalla congettura estesa di Riemann, tuttora non dimostrata.

In questo lavoro trattiamo di altri due metodi: ECPP e AKS.

– ECPP sta per 'Elliptic Curve Primality Proving'. Questa tecnica, che si basa sui gruppi formati dai punti delle curve ellittiche, è stata introdotta e sviluppata da Goldwasser e Kilian ([18], [19]). ECPP è deterministico e incondizionato, e polinomiale per *quasi tutti* i primi (infatti può risultare inefficiente per una -- piccolissima -- frazione di essi).

– AKS è un acronimo che sta per Agrawal - Kayal - Saxena, i nomi dei tre studiosi indiani che per primi, nel 2002, hanno scoperto finalmente un criterio di primalità deterministico, polinomiale e incondizionato ([2], [3], [4]).

Nel paragrafo 2 tratteremo della complessità computazionale, nell'ambito della quale il problema della primalità ha avuto un particolare rilievo, perché fino al 1974 non si sapeva se esso appartenesse o meno alla classe NP; il paragrafo è dedicato alla descrizione delle classi P e NP, con particolare attenzione alla complessità del problema del riconoscimento dei numeri primi e all'algoritmo di Pratt.

Nel paragrafo 3 introdurremo il gruppo delle curve ellittiche e tratteremo di ECPP, e in particolare del *certificato* di primalità che esso fornisce.

Nel quarto paragrafo parleremo del teorema AKS, della sua nascita, del suo impatto sulla comunità scientifica, e delle relazioni che ha con alcuni test di primalità introdotti in [11].

Nell'ultima parte accenneremo brevemente ad alcuni sviluppi delle idee di Agrawal, Kayal e Saxena, e in particolare ad una interessante proposta di Qi Chen sul possibile utilizzo *simultaneo* di AKS e ECPP.

2. – Le classi P e NP.

Il titolo dell'articolo di Agrawal, Kayal e Saxena è '*Primes is in* P'. Sarebbe possibile limitarsi a introdurre la classe P. Però le sue relazioni con la classe NP, proprio per quanto riguarda i numeri primi, sono così interessanti che parleremo di entrambe.

Le sigle P e NP si riferiscono a due classi di problemi decisionali, quelli Polinomiali e quelli Nondeterministicamente Polinomiali.

Un **problema decisionale** è un problema che ammette come risposta solo 'sì' oppure 'no'.

Chiameremo nel seguito P_1 e P_2 questi due classici problemi decisionali:

P_1: *Problema del riconoscimento dei numeri primi*. Dato un intero positivo n, dire se n è primo.

P_2: *Problema dello zaino*. Dati un insieme A di interi positivi ed un intero positivo σ, dire se esiste un sottoinsieme B di A tale che la somma degli elementi di B sia σ.

In generale un problema decisionale consiste di una infinità di domande possibili, che vengono dette **istanze** del problema.

Per esempio una istanza del problema P_1 può essere:

domanda 1: 79925742525202824779 è primo?

Ecco invece una istanza del problema P_2:

domanda 2: Esiste un sottoinsieme B di A$=\{2, 4, 9, 13, 17, 23, 32, 70, 123, 157\}$ tale che la somma dei suoi elementi sia 228?

Una istanza contiene dati ai quali si assegna un **peso** p, che ne caratterizza la difficoltà.

Per il problema P_1, il peso p è il numero delle cifre dell'intero n (essenzialmente il logaritmo in base 10 di n, Log n).

Per il problema P_2, il peso p è il numero $\#$A degli elementi dell'insieme A.

Per risolvere un problema decisionale si costruiscono appositi algoritmi.

Generalmente esistono sempre algoritmi che sono in grado di rispondere a qualsiasi istanza del problema: gli algoritmi di forza bruta, che esplorano tutto lo spazio di ricerca dei dati.

Per esempio alla domanda 1 si può tentare di rispondere dividendo 79925742525202824779 per tutti gli interi dispari che lo precedono. Un algoritmo molto più intelligente, ma purtroppo anch'esso inefficace per numeri grandi, consiste nel dividere n per tutti i primi non maggiori della radice quadrata di n.

Per rispondere alla domanda 2 si possono considerare tutti i 2^{10} sottoinsiemi di A. Poiché $2^{10} = 1024$, questo lavoro può essere eseguito in un istante da un computer. Però se il peso dei dati passa da 10 a 100, cioè se $\#\,\mathrm{A} = 100$, ci sono 2^{100} sottoinsiemi, e le cose si complicano un tantino . . .

Ovviamente il tempo T di esecuzione di un algoritmo che risponde ad una data istanza del problema è una funzione del peso dei dati, $\mathrm{T} = \mathrm{T}(p)$.

Facciamo un esempio concreto, per il problema P_2, sempre pensando all'algoritmo di forza bruta. Se siamo in grado di eseguire la somma di un sottoinsieme B di A in un microsecondo ($= 10^{-6}$ sec), con $p = 10$ impieghiamo circa un millisecondo a rispondere. Se $p = 100$, poiché 2^{100} vale circa 10^{30}, occorreranno, nel caso peggiore, approssimativamente 10^{24} secondi, ovvero qualcosa come 3×10^{16} anni (trenta milioni di miliardi di anni ...)

DEFINIZIONE 2.1. – *Diciamo che un algoritmo V* **risponde in tempo polinomiale** *al problema decisionale Q se esiste una costante k tale che, data una qualunque istanza i di Q, V risponde 'sì' oppure 'no' in un ammontare di tempo che è minore di p^k, dove p è il peso di i.*

Si richiede insomma che V termini in un ammontare di tempo che è $O(p^k)$.

Possiamo ora definire la classe dei problemi P.

DEFINIZIONE 2.2. – *La* **classe P** *è formata dall'insieme dei problemi decisionali Q per i quali esiste un algoritmo che risponde in tempo polinomiale.*

Si osservi che il limite temporale imposto dalla definizione deve valere per tutte le istanze. Quasi sempre si hanno algoritmi che funzionano velocemente nella gran parte dei casi, ma si rallentano enormemente in corrispondenza di istanze particolarmente difficili. I «casi peggiori» non si possono trascurare. Sono anzi proprio loro, spesso, a determinare l'appartenenza o meno di un problema alla classe P.

Come sappiamo da AKS, P_1 appartiene a P.

Siamo quindi in grado di decidere se un numero n è primo in un ammontare di tempo che è proporzionale ad una potenza del numero delle cifre di n, e non a n stesso (o alla sua radice quadrata)! La differenza è abissale, inimmaginabile.

Per quanto riguarda P_2, non sono noti algoritmi polinomiali che lo risolvano. Questo però non significa che tali algoritmi non possano esistere.

Per i problemi più importanti che vengono dalla teoria dei numeri, dalla combinatorica e dalle applicazioni concrete non sono noti algoritmi che li risolvano in tempo polinomiale.

Si è pensato dunque di introdurre una nuova classe, quella dei problemi che sono risolubili in tempo polinomiale da un algoritmo non deterministico.

Un *algoritmo non deterministico* si può pensare come una macchina capace di parallelismo illimitato.

Deve essere ben chiaro che un tale algoritmo non esiste in realtà. Si tratta di una *finzione* che serve soltanto per tentare di comprendere meglio la complessità computazionale di alcuni problemi.

Essenzialmente la possibilità di usare una macchina non deterministica rende trascurabile quello che è il principale ostacolo per un algoritmo reale: la grandezza esponenziale dello spazio di ricerca.

Consideriamo il Problema dello Zaino con $n = 100$, ed una certa somma assegnata σ. Il programma di forza bruta esaminerà, in un certo ordine, i 2^{100} sottoinsiemi B di A, calcolando per ognuno di essi la somma. Se esiste un solo sottoinsieme la cui somma è σ e se si trova casualmente al fondo della lista, il programma dovrà eseguire 2^{100} somme, e abbiamo già osservato che non lo vedremo mai terminare.

Invece un programma non deterministico può creare 2^{100} copie di se stesso, una per ogni sottoinsieme B di A. Ognuna di queste copie esegue una somma sola!

Il tempo di esecuzione si è ora ridotto al tempo necessario per la verifica del fatto che un elemento dello spazio di ricerca sia una soluzione.

DEFINIZIONE 2.3. – *Diciamo che un problema Q appartiene alla* **classe NP** *se siamo in grado di verificare in tempo polinomiale la correttezza* DI OGNI RISPOSTA 'SÌ'.

Possiamo, equivalentemente, pensare che esista un *Oracolo*, il quale, per ogni risposta 'sì', ci doni un certificato di garanzia, *controllabile in tempo polinomiale*.

Si noti che la grandezza dello spazio di ricerca è sparita, nella classe NP!

Dato questo incredibile vantaggio quasi sempre è immediato provare che un problema Q appartiene alla classe NP.

Dobbiamo infatti soltanto dimostrare che ogni istanza 'sì' di Q possiede un certificato controllabile in tempo polinomiale.

Vediamo un esempio significativo.

P_3: Problema del riconoscimento dei numeri composti. Dato un intero positivo n, dire se n è composto.

TEOREMA 2.4. – *P_3 appartiene alla classe* NP.

Dimostrazione. Il certificato di n è semplicemente un *divisore* di n. Il costo computazionale della verifica è una singola divisione. □

In alcuni, rari, casi non è facile trovare quale certificato possa essere al tempo stesso *succinto* (cioè verificabile in tempo polinomiale) e *convincente*.

Nel 1974 Vaughan R. Pratt ideò, per la prima volta nella storia, un certificato di primalità verificabile in tempo polinomiale [29], basato sul Criterio di Lucas ([11], 2.3). Si tratta in realtà di un insieme di certificati.

2.5. – *Certificato di Pratt (1974).*

Supponiamo di conoscere i primi fino ad un limite L. Dato N primo, l'oracolo deve certificare la primalità di N.

L'oracolo fornisce

(i) l'insieme D dei divisori primi di $N - 1$

(ii) un intero a che soddisfa le due condizioni del teorema di Lucas per ogni q in D.

La procedura è ricorsiva. Per ogni primo q in D maggiore di L l'oracolo fornisce un certificato, sino a quando tutti i primi coinvolti sono minori di L.

ESEMPIO 2.6 – *Esempio di certificazione di un primo con il metodo di Pratt.*

Supponiamo di conoscere l'elenco dei primi fino a 10.000.

Sia dato il numero $N = 79925742525202824779$.

L'oracolo fornisce l'insieme dei divisori primi di $N-1 : D = \{2, 11, 263, 53279, 259269950887\}$.

L'oracolo fornisce la base a $= 2$.

L'utente U, con divisioni successive, verifica che $N - 1 = 2 \cdot 11 \cdot 263 \cdot 53279 \cdot 259269950887$.

U calcola:

$2^{N-1} (\text{mod } N) = 1$
$2^{(N-1)/2} (\text{mod } N) = 79925742525202824778$
$2^{(N-1)/11} (\text{mod } N) = 76721605684067720274$
$2^{(N-1)/263} (\text{mod } N) = 71522329290127897097$
$2^{(N-1)/53279} (\text{mod } N) = 26384095619722806544$
$2^{(N-1)/259269950887} (\text{mod } N) = 2959614048108645507$

A questo punto U sa che, se gli elementi di D sono davvero primi, allora, per il Teorema di Lucas, N è primo.

Vista la sua limitata conoscenza dei primi, U sospetta di 53279 e di 259269950887.

L'oracolo fornisce un certificato per 53279.

Offre la fattorizzazione di 53278 ($53278 = 2 \cdot 17 \cdot 1567$) e una base che funziona: a $= 2$. U verifica la fattorizzazione.

U calcola:

$2^{53278} (\text{mod } 53279) = 1$
$2^{53278/2} (\text{mod } 53279) = 37052$
$2^{53278/17} (\text{mod } 53279) = 31344$
$2^{53278/1567} (\text{mod } 53279) = 5138$

Poiché U sa che 2, 17 e 1567 sono primi, 53279 è certamente primo.

Ora l'oracolo fornisce un certificato per 259269950887: dà la fattorizzazione $259269950886 = 2 \cdot 3 \cdot 29 \cdot 2441 \cdot 610429$, e una base che va bene: a $= 3$ (in questo caso 2 non funziona).

U verifica la fattorizzazione.

U calcola:

$3^{259269950886} (\text{mod } 259269950887) = 1$

$3^{259269950886/2} (\text{mod } 259269950887) = 259269950886$
$3^{259269950886/3} (\text{mod } 259269950887) = 176218747612$
$3^{259269950886/29} (\text{mod } 259269950887) = 39029987097$
$3^{259269950886/2441} (\text{mod } 259269950887) = 223915085356$
$3^{259269950886/610429} (\text{mod } 259269950887) = 53534810325$

U sa che 2, 3, 29 e 2441 sono primi, ma sospetta di 610429.

L'oracolo fornisce un certificato per 610429.

Fattorizzazione per $610428 : 610428 = 2^2 \cdot 3 \cdot 7 \cdot 13 \cdot 43$; base adatta: $a = 2$.

U verifica la fattorizzazione.

U calcola:

$2^{610428} (\text{mod } 610429) = 1$
$2^{610428/2} (\text{mod } 610429) = 610428$
$2^{610428/3} (\text{mod } 610429) = 255876$
$2^{610428/7} (\text{mod } 610429) = 583168$
$2^{610428/13} (\text{mod } 610429) = 461598$
$2^{610428/43} (\text{mod } 610429) = 174519$

U sa che 2, 3, 7, 13 e 43 sono primi. Pertanto è sicuro che anche 610429 è primo.

Non occorrono più certificati. Ora U è certo che $N = 79925742525202824779$ è primo!

Dalla definizione 2.3 della classe NP, se ne trae immediatamente un'altra, complementare, quella della classe co-NP:

DEFINIZIONE 2.7. – *Diciamo che un problema Q appartiene alla* **classe co-NP** *se siamo in grado di verificare in tempo polinomiale la correttezza* DI OGNI RISPOSTA 'NO'.

Le classi NP e co-NP sono intuitivamente assai diverse.

Consideriamo P_2, il Problema dello Zaino. È chiaro che P_2 appartiene a NP. Se i è una istanza che ha risposta sì , il certificato consiste semplicemente nell'esibire un sottoinsieme B di A la cui somma sia quella richiesta. La verifica è immediata: eseguire la somma!

È difficile però credere che P_2 stia in co-NP. Se ci sono 2^{100} sottoinsiemi, come certificare in modo succinto che *nessuno* di essi ha somma σ? Si ricordi che deve essere possibile produrre il certificato in *tutti* i casi.

Vi assicuriamo che diventerebbe famoso chiunque ci riuscisse!

Il problema P_1 (decidere se un intero è primo) è ovviamente il complementare di P_3 (decidere se un intero è composto). Pertanto da 2.4 segue immediatamente che:

TEOREMA 2.8. – *Il problema P_1 del riconoscimento dei numeri primi appartiene alla classe* co-NP. □

Fino al 1974 (l'anno del certificato di Pratt) non si sapeva se PRIMES (il problema P_1) stesse in NP, mentre risultava banalmente essere in co-NP. Una situazione singolare.

Ma c'è di più.

Le ipotesi che vengono ritenute vere (anche se rimangono pure congetture) sui rapporti tra le classi P, NP e co-NP sono le seguenti:

2.9. – *Congetture.*

1) $\mathrm{NP} \neq \text{co-NP}$
2) $\mathrm{P} = \mathrm{NP} \cap \text{co-NP}$
3) $\mathrm{P} \neq \mathrm{NP}$

Tra queste la più importante è la 2.9 3).

Trovare un problema che stia nella classe NP e *non* nella classe P è considerato uno dei problemi aperti più importanti della matematica: il Clay Mathematics Institute lo inserisce nell'elenco dei sette problemi matematici per la soluzione di ognuno dei quali offre un milione di dollari di premio!

Come dice Paolo Serafini [32]:

«Sarebbe un fatto davvero sorprendente se la risoluzione di un problema fosse altrettanto facile quanto la sua verifica, e il buon senso tende quindi a scartare la possibilità di avere $\mathrm{P} = \mathrm{NP}$».

In ogni modo nel 1974 si venne a sapere che PRIMES sta nella intersezione di NP con co-NP: ma *non si sapeva se PRIMES fosse in* P.

Quale sfida migliore alla congettura 2.9 2?

Ci sono voluti quasi trent'anni (e i nostri geniali amici Agrawal, Kayal e Saxena!) per provare che

PRIMES is in P

Agrawal, Kayal e Saxena hanno scoperto un algoritmo che, ricevuto un intero n in input, risponde 'sì' o 'no' alla domanda 'n è primo?' in un tempo che è $O(\log^k n)$.

3. – Curve ellittiche e numeri primi.

3.1. – *Richiami.*

Sia $\mathbb{K}$ un qualunque campo. Consideriamo nel piano proiettivo $\mathcal{P}^2(\mathbb{K})$ una **cubica non singolare**, cioè una curva algebrica proiettiva luogo degli zeri di un polinomio omogeneo di terzo grado $F_3(X,Y,Z) \in \mathbb{K}[X,Y,Z]$ che non abbia punti singolari, cioè non esista nessun punto P per cui si abbia contemporaneamente

$$\begin{cases} F_3(X,Y,Z)_P = 0 \\ \left(\dfrac{\partial F_3}{\partial X}\right)_P = 0 \\ \left(\dfrac{\partial F_3}{\partial Y}\right)_P = 0 \\ \left(\dfrac{\partial F_3}{\partial X}\right)_P = 0 \end{cases} .$$

DEFINIZIONE 3.1. – *Diciamo* **cubica ellittica** $E(\mathbb{K})$ *sul campo* $\mathbb{K}$ *una cubica non singolare* $E : F_3(X,Y,Z=0)$ *che abbia almeno un punto* $\mathcal{O}$.

Possiamo identificare una cubica ellittica con la coppia $(E,\mathcal{O})$, *dove* $\mathcal{O} \in E(\mathbb{K})$.

Sappiamo, dal Teorema di Weierstrass, che, se $\mathrm{char}(\mathbb{K}) \neq 2,3$, esiste una trasformazione birazionale per cui l'equazione di una cubica

ellittica diventa

$$E_{\mathbb{K}}(a,b): \quad y^2 = x^3 + ax + b$$

e il punto $\mathcal{O}$ diventa (0, 1, 0).

Se $\mathbb{K}$ è il campo finito $GF(q)$, scriveremo $E_q(a,b)$.

Ricordiamo che

PROPRIETÀ 3.2. – *La cubica $E_{\mathbb{K}}(a,b)$ è non singolare se e solo se $\Delta := 4a^3 + 27b^2 \neq 0$.* □

3.2. – *Legge di gruppo sulla cubica.*

Siano $E = E_{\mathbb{K}}(a,b)$ una cubica ellittica e $L = L_{\mathbb{K}}$ una retta. Qualunque sia il campo $\mathbb{K}$ si può provare che:

PROPRIETÀ 3.3. – *Se E ed L hanno due punti in comune (contati opportunamente) ne hanno necessariamente un terzo.* □

La proprietà precedente giustifica la seguente legge:

DEFINIZIONE 3.4. – **Legge secante-tangente**.

Sia E una cubica ellittica. Siano P e Q due punti distinti appartenenti a E e sia L la retta passante per P e Q. Si definisce $P * Q$ il terzo punto di intersezione di E con L. Se $P = Q$ si definisce $P * P$ il punto in cui la tangente a E in P interseca nuovamente E.

Si può ora definire la

DEFINIZIONE 3.5. – **Legge di addizione sulla cubica**

Siano P e Q due punti di una cubica ellittica $E = (E(\mathbb{K}), \mathcal{O})$. Si definisce la loro **somma** *in questo modo*

$$P + Q := \mathcal{O} * (P * Q).$$

Vale il seguente

TEOREMA 3.6. – **Teorema di Poincaré**
Sia $\mathbb{K}$ un qualunque campo e sia $(E(\mathbb{K}), \mathcal{O})$ una cubica ellittica; l'operazione $P + Q = \mathcal{O} * (P * Q)$ definisce una struttura di gruppo commutativo su E di cui $\mathcal{O}$ è l'elemento neutro. □

3.3. – *Calcoli per la legge di gruppo sulla cubica.*

Dato un qualunque campo $\mathbb{K}$, consideriamo la cubica $(E(\mathbb{K}), \mathcal{O})$ dove

$$E_{\mathbb{K}}(a, b) : y^2 = x^3 + ax + b, \quad \mathcal{O} = Y_\infty = (0, 1, 0).$$

Dati $P = (x_1, y_1)$, $Q = (x_2, y_2) \in E$, calcoliamo $-P, P + Q, P + P = 2P$.
1) $-P = Y_\infty * P$ è il simmetrico di P rispetto all'asse delle x. Dunque

3.7 $$-P = (x_1, -y_1)$$

2) Siano $P = (x_1, y_1)$, $Q = (x_2, y_2) \in E$; distinguiamo due casi:

a) $x_1 \neq x_2$. Se m è il coefficiente angolare della retta passante per P e Q, con facili calcoli si ottiene:

3.8 $$P + Q = (x_3, y_3), \text{ dove } \begin{cases} m = \dfrac{y_2 - y_1}{x_2 - x_1} \\ x_3 = m^2 - x_1 - x_2 \\ y_3 = m(x_1 - x_3) - y_1 \end{cases}.$$

b) $x_1 = x_2$.
Poiché $P, Q \in E$, si deve avere $y_1^2 = y_2^2$. Pertanto $y_2 = \pm y_1$ e dunque o $Q = -P$, da cui $P + Q = \mathcal{O}$, oppure $Q = P$, da cui $P + Q = 2P$, calcolo che effettueremo qui sotto.
3) Il calcolo per trovare $2P$ è analogo a quello effettuato per trovare $P + Q$, solo che la retta PQ diventa la tangente a E in P.
Tale retta ha coefficiente angolare $m = -\dfrac{\left(\dfrac{\partial F}{\partial x}\right)_P}{\left(\dfrac{\partial F}{\partial y}\right)_P}$, dove $F(x, y) = x^3 + ax + b - y^2$.

Pertanto $m = \dfrac{3x_1^2 + a}{2y_1}$. Infine:

$$3.9 \qquad 2P = (x_3, y_3), \text{ dove } \begin{cases} m = \dfrac{3x_1^2 + a}{2y_1} \\ x_3 = m^2 - 2x_1 \\ y_3 = m(x_1 - x_3) - y_1 \end{cases}.$$

Poiché E, con l'operazione data, è un gruppo additivo, possiamo definire nel modo consueto i multipli di un punto $P \in E$. Dato $m \in \mathbb{Z}$, denotiamo con $[m]\,P$ il corrispondente multiplo:

Se $m = 0$ poniamo $[0]\,P = \mathcal{O}$.
Se $m > 0$ poniamo $[m]\,P = P + P + \ldots + P$, m volte.
Se $m < 0$ poniamo $[m]\,P = (-P) + (-P) + \ldots + (-P)$, $|m|$ volte.

ESEMPIO.
Sia $\mathbb{K} = \mathbb{Z}_{23} = GF(23)$, consideriamo la cubica $E_{23}(1,1) : y^2 = x^3 + x + 1$. Cerco due punti P e Q su E.
Pongo $x = 3$ e vedo se esiste y tale che $y^2 = 3^3 + 3 + 1 = 8$.
8 è un quadrato (mod 23) perché $\left(\dfrac{8}{23}\right) = \left(\dfrac{2}{23}\right)$, e 2 è un quadrato (mod 23). Infatti $8 = 10^2$. Dunque $y = \pm 10$. Scelgo $y = 10$.
Ripetendo ora lo stesso procedimento a partire da $x = 9$, trovo $y = \pm 7$ e scelgo $y = 7$. Ho trovato allora su E i punti $P = (3, 10)$ e $Q = (9, 7)$; calcolo $P + Q$ e $2P$.
a) $P + Q = (x_3, y_3)$:

$$\begin{cases} m = \dfrac{y_2 - y_1}{x_2 - x_1} = \dfrac{7 - 10}{9 - 3} = 11 \\ x_3 = 11^2 - 3 - 9 = 17 \\ y_3 = 11(3 - 17) - 10 = 20 \end{cases}.$$

Dunque $P + Q = (17, 20)$.

b) $2P = (x_3, y_3)$:

$$\begin{cases} m = \dfrac{3x_1^2 + 1}{2y_1} = \dfrac{28}{20} = 6 \\ x_3 = 36 - 2 \cdot 3 = 30 = 7 \\ y_3 = 6(3 - 7) - 10 = 12 \end{cases} .$$

Quindi $2P = (7, 12)$.

Il gruppo $E_{23}(1, 1)$ ha 28 elementi, è ciclico ed è generato da P. Questa è la lista dei multipli di P, da $[1]P$ a $[28]P$:

$(3, 10)$, $(7, 12)$, $(19, 5)$, $(17, 3)$, $(9, 16)$, $(12, 4)$, $(11, 3)$, $(13, 16)$, $(0, 1)$, $(6, 4)$, $(18, 20)$, $(5, 4)$, $(1, 7)$, $(4, 0)$, $(1, 16)$, $(5, 19)$, $(18, 3)$, $(6, 19)$, $(0, 22)$, $(13, 7)$, $(11, 20)$, $(12, 19)$, $(9, 7)$, $(17, 20)$, $(19, 18)$, $(7, 11)$, $(3, 13)$, $\mathcal{O}$.

Nell'esempio fatto il gruppo risultava ciclico. Oltre a questa, esiste solo un'altra possibilità. Vale infatti il seguente:

TEOREMA 3.10. – **Teorema di Cassels**.
Se $\mathbb{K} = GF(q)$ allora $E(\mathbb{K})$ o è ciclico o è isomorfo al prodotto dei due gruppi ciclici $\mathbb{Z}_{d_1}$, $\mathbb{Z}_{d_2}$, dove $d_1 | d_2$ e $d_1 | q - 1$. □

3.4. – *Numero dei punti di una cubica ellittica.*

Sia $\mathbb{K} = GF(q)$ il campo finito di ordine $q = p^n$; si consideri la cubica $E = E_q(a, b)$. Sia $N = \# E$ il numero dei punti di E. Poiché per ogni $x \in \mathbb{K}$ ci sono al più due valori di y corrispondenti, tenendo anche conto del punto $\mathcal{O}$, si ha $N \leq 2q + 1$.

Precisamente si avranno due punti y corrispondenti a un certo x se la quantità $u = x^3 + ax + b$ è un quadrato in $GF(q)$, non ci sarà nessun valore di y se u non è un quadrato in $GF(q)$, e ci sarà un solo punto se $u = 0$.

Definiamo la seguente funzione:

DEFINIZIONE 3.11. – *Si chiama* **funzione carattere quadratico** *la funzione*

$$\chi : GF(q)^* \to \{-1, 0, 1\}$$

$$\chi(u) := \begin{cases} 1 & \text{se } u \text{ è un quadrato in } GF(q)^* \\ -1 & \text{se } u \text{ non è un quadrato in } GF(q)^* \\ 0 & \text{se } u = 0 \end{cases}$$

La funzione χ è un morfismo. Se $q = p$, coincide con il simbolo di Legendre.

Le soluzioni dell'equazione $y^2 = u$ sono $1 + \chi(u)$, e dunque $N = 1 + \sum_{x \in GF(q)} [1 + \chi(x^3 + ax + b)]$.

Infine:

$$N = 1 + q + \sum_{x \in GF(q)} \chi(x^3 + ax + b). \tag{3.12}$$

Facciamo un esempio in cui è facile calcolare $\chi(x^3 + ax + b)$.

ESEMPIO.

Siano $\mathbb{K} = \mathbb{Z}_{71}$ e $E = E_{71}(0, -1) : y^2 = x^3 - x$.

Siano $x \in \mathbb{K}$ e $u = x^3 - x$; considero $-x$ e $u' = (-x)^3 - (-x) = -(x^3 - x) = -u$. Si ha $\chi(u') = \chi(-u) = \chi(-1)\chi(u) = -\chi(u)$, in quanto $\chi(-1) = \left(\frac{-1}{71}\right) = -1$ (perché $71 \equiv 3 \bmod 4$).

Pertanto per ogni coppia $x, -x$, si ha $\sum_{x,-x} \chi(x^3 - x) = 0$. Poiché $\chi(0) = 0$, risulta $\sum_{x \in \mathbb{K}} \chi(x^3 - x) = 0$, e dunque $N = 1 + 71 = 72$.

Il teorema di Hasse fornisce una importante limitazione all'ordine del gruppo $E_q(a, b)$.

TEOREMA 3.13. – **Teorema di Hasse**

$$|N - 1 - q| \leq 2\sqrt{q}$$

□

Riscriviamo il teorema di Hasse in questo modo:

$$q + 1 - 2\sqrt{q} = (\sqrt{q} - 1)^2 \leq N \leq q + 1 + 2\sqrt{q} = (\sqrt{q} + 1)^2$$

e chiamiamo

$$I_q = [(\sqrt{q} - 1)^2, (\sqrt{q} + 1)^2].$$

Se $q = p$ è un primo dispari, vale il seguente

TEOREMA 3.14. – **Teorema di Deuring**
$\forall N \in I_p$, esiste una cubica ellittica non singolare $E_p(a, b)$ tale che $\#E = N$. □

Naturalmente se q è grande la formula 3.12 non è efficace per il calcolo dell'ordine di E. Per fortuna esiste un algoritmo, dovuto a René Schoof [31], che calcola $\#E_q(a, b)$ in tempo polinomiale.

3.5. – *Cubiche ellittiche e primalità.*

Dato $n \in \mathbb{N}$, $n > 2$, consideriamo l'insieme

3.15 $$E_n(a, b) = \{(x, y) \in \mathbb{Z}_n \times \mathbb{Z}_n : y^2 = x^3 + ax + b (\text{mod } n)\} \cup \{\mathcal{O}\}.$$

Sappiamo che, se n è primo, tale insieme è un gruppo finito con l'operazione 3.5.

Se invece n è composto, si ottiene una struttura che chiamiamo **pseudogruppo**.

I calcoli vengono eseguiti anche in questo caso mediante le 3.8 e 3.9. L'unica difficoltà nasce dal fatto che, se n non è primo, non sempre si può calcolare il coefficiente angolare m, perché $x_2 - x_1$ o y_1 possono non essere coprimi con n (e quindi non invertibili). Pertanto, quando n è composto, nell'insieme $E_n(a, b)$ l'operazione è solo *parzialmente* definita.

Ovunque essa è definita in $E_n(a, b)$, gode delle stesse proprietà dell'operazione di gruppo (in particolare è associativa e commutativa).

È importante osservare che, 'al di sotto' dello pseudogruppo $E_n(a,b)$, esistono i gruppi $E_p(a,b)$, per ogni primo p che divide n: se $P=(x,y)\in E_n(a,b)$, è ben definito il punto *proiezione* di P, $P'=\pi_p(P)=(x \bmod p, y \bmod p)$, che appartiene a $E_p(a,b)$.

Per comprendere il funzionamento dell'algoritmo di Goldwasser-Killian che segue, è essenziale ricordare che le operazioni algebriche svolte (quando possibile) nello pseudogruppo $E_n(a,b)$ sono trasportate da π_p su ogni gruppo $E_p(a,b)$ sottostante.

Per esempio, dato $P\in E_n(a,b)$, i suoi multipli $[k]P$ esistono in tutti gli $E_p(a,b)$ ma non necessariamente in $E_n(a,b)$.

Possiamo ora descrivere il metodo di Goldwasser-Killian per stabilire se un numero è primo. Esso utilizza in modo essenziale il seguente teorema:

TEOREMA 3.16. – **Teorema di Goldwasser-Killian**

Sia n un intero maggiore di 1 e coprimo con 6. Siano dati $E_n(a,b)$, un intero positivo m e un punto P su $E_n(a,b)$ tali che sia possibile calcolare il multiplo $[m]P$ e si abbia $[m]P=0$. Supponiamo che esista un divisore q di m, $q>(n^{1/4}+1)^2$, tale che $\left[\frac{m}{q}\right]P\neq 0$.

Allora se q è primo, n è primo.

DIMOSTRAZIONE. – Siano:

- p un divisore primo di n
- N l'ordine di $E_p(a,b)$
- v il periodo di P in $E_p(a,b)$
- q un divisore primo di m, $q>(n^{1/4}+1)^2$.

Poiché $[m]P=0$ ma $\left[\frac{m}{q}\right]P\neq 0$, v divide m ma v non divide $\frac{m}{q}$, quindi $q|v$. Del resto, poiché il periodo di un elemento divide l'ordine del gruppo, $v|N$.

Ne consegue che $q|N$, e in particolare $q\leq N$.

Per 3.13, $N\leq(p^{1/2}+1)^2$.

Pertanto $(n^{1/4}+1)^2<q\leq(p^{1/2}+1)^2$ e dunque $p>n^{1/2}$. Ne segue che n è primo. □

Il Teorema 3.16 permette di creare un algoritmo completamente nuovo, chiamato **ECPP**, che certifica la primalità di un intero n in modo ricorsivo.

Questo algoritmo viene applicato ad un intero n coprimo con 6, abbastanza grande (diciamo $n > 2^{32}$, altrimenti ci sono altri strumenti) e fortemente sospettato di essere primo (per esempio n ha passato il test di Miller [11], 4.13). La funzione dell'algoritmo è ridurre il problema dalla primalità di n a quella di un numero q minore di n.

Quando termina, l'algoritmo risponde — con certezza — 'n è composto', oppure 'se q è primo allora n è primo'.

Lo scopo è quello di arrivare, dopo un certo numero di iterazioni, a 'n è composto', oppure far diminuire q fino a quando si possa affermare con certezza che è primo.

Questi sono i passi dell'algoritmo:

3.17. – **Algoritmo ECPP**

1. Scelta dei parametri
   Si prendono $0 \leq a, b \leq n-1$ *tali che* $MCD(4a^3 + 27b^2, n) = 1$

2. Calcolo del numero dei punti della curva
   Mediante l'algoritmo di Schoof [31] *si calcola l'ordine m che avrebbe* $E_n(a, b)$, *se n fosse primo.*
   L'algoritmo può fallire, se n non è primo, per diversi motivi.
   Per esempio m potrebbe non cadere nell'intervallo $((\sqrt{n}-1)^2, (\sqrt{n}+1)^2)$ *(vedi* 3.13*)*
   In caso di fallimento, si vada al passo 6.

3. Ricerca di q
   Si tenta di trovare un fattore q di m dove q ha passato un test di primalità serio (per esempio il test di Miller), e $q > (n^{1/4}+1)^2$.
   Se non si trova q in un tempo massimo prestabilito, si ritorni al passo 1.

4. Scelta di un punto P su $E_n(a, b)$
   Si cerca x tale che $0 \leq x \leq n-1$ *e, posto* $u = x^3 + ax + b \bmod n$, $\left(\frac{u}{n}\right) = 1$.

Se durante i tentativi si trova $\left(\frac{u}{n}\right) = 0$, *si vada al passo* 6.
Si calcola y *soluzione di* $y^2 \equiv u \pmod{n}$, *utilizzando i metodi, efficaci, per il calcolo della radice quadrata di* u, *come se* n *fosse primo.*
Si verifica che in effetti $y^2 \equiv u \pmod{n}$. *In caso contrario* n *è certamente composto: si vada allora al passo* 6. *Altrimenti si ponga* $P = (x, y)$.

5. Calcolo di un opportuno multiplo di P
Si calcola $U = \left[\frac{m}{q}\right] P$. *Se il calcolo fallisce si vada al passo* 6.
Se $U = \mathcal{O}$, *si torni al passo* 4.
Si calcola ora $V = [q]U$. *Se il calcolo fallisce si vada al passo* 6.
Se $V \neq \mathcal{O}$, *si vada al passo* 6.
Si vada al passo 7.

6. Uscita nel caso in cui n risulta essere composto
Si termina il programma e si risponde 'n *è composto'.*

7. Uscita nel caso di successo
Si termina il programma e si risponde 'se q *è primo allora* n *è primo'.*

L'algoritmo 3.17 è un algoritmo 'probabilistico'. L'incertezza è però di tipo diverso da quella degli algoritmi di tipo Montecarlo [11], 4.11 e [11], 4.15. Essi terminano sempre in tempo polinomiale ma danno una risposta certa solo da un lato (quando rispondono 'n è composto'). Nell'altra direzione si ha soltanto una probabilità.

In 3.17 l'incertezza riguarda il tempo di esecuzione, ed è dovuta essenzialmente al tentativo di trovare, al passo 3, un fattore q di m che sia probabilmente primo e abbastanza grande.

Shafi Goldwasser e Joe Killian ([18], [19]) dimostrarono che 3.17 termina in tempo atteso polinomiale tranne al più per una piccolissima frazione di primi, tendente a zero con velocità esponenziale. Riuscirono a provare che il loro algoritmo termina in tempo atteso polinomiale per

ogni input se vale una famosa congettura di Cramer sui gap tra primi consecutivi:

$$\pi(x + \log^2 x) - \pi(x) > 0.$$

Adleman e Huang in [1] ovviarono a questa difficoltà costruendo un algoritmo che termina in tempo atteso polinomiale per ogni intero, senza alcuna ipotesi non dimostrata. È interessante osservare che inizialmente il metodo di Adleman e Huang crea primi q più grandi di n. Questo procedimento, in apparenza paradossale, serve a sostituire il numero n iniziale con un altro che sia abbastanza casuale da far funzionare l'algoritmo 3.17.

Nella pratica l'algoritmo di Schoof per il calcolo dell'ordine della curva, sebbene polinomiale, è piuttosto lento quando n è grande.

Nel 1993 Atkin e Morain [5] trovarono un nuovo approccio al problema: invece di partire dalla curva E, si cerca un opportuno $m = kq$ dove q è un probabile primo maggiore di $(n^{1/4} + 1)^2$, e *dopo* si costruisce una curva E di ordine m. In tal modo si evita il calcolo di $\#E$.

Il metodo di Adleman e Huang sembra non essere mai stato effettivamente implementato, al contrario di quelli di Goldwasser - Killian e Aitkin - Morain. Tutti e tre forniscono un *certificato di primalità*. L'idea di base è la stessa del certificato di Pratt 2.5; si tratta di una procedura ricorsiva:

n è primo se q_0 primo, q_0 è primo se q_1 è primo ... sino ad arrivare ad un q_r che sappiamo essere primo.

Il certificato contiene l'elenco dei q_i e dei parametri che via via si determinano per costruire le curve.

L'algoritmo di Aitkin e Morain è, nella pratica, uno dei metodi più efficaci e veloci per provare la primalità di un intero generico. L'analisi della complessità dell'algoritmo è molto più difficile che nel caso di Goldwasser-Killian. Vi sono però risultati nuovi assai promettenti, si veda il recente articolo di Morain [25] .

I gruppi delle curve ellittiche sono utilizzati, oltre che per test di primalità, anche per la fattorizzazione di interi grandi e in crittografia.

In questi casi i metodi che si basano sulle curve ellittiche hanno un vantaggio rilevante rispetto a quelli che lavorano in $\mathbb{Z}_p^*$ o nelle

estensioni finite di $\mathbb{Z}_p$, perché l'ordine del gruppo che viene utilizzato dall'algoritmo varia in un intervallo molto ampio.

Infatti, modificando i parametri a e b, come sappiamo da 3.14, l'ordine del gruppo può assumere ogni valore compreso nell'intervallo I_p.

4. – Il Teorema AKS.

Dal 24 al 28 marzo del 2003 si tenne presso l'American Institute of Mathematics, a Palo Alto, un workshop dal titolo *Future directions in algorithmic number theory* [22]. Tra gli organizzatori spiccavano i nomi di Hendrik Lenstra e Carl Pomerance. Notevole è anche l'incipit della presentazione:

> *This workshop, sponsored by AIM and the NSF, is occasioned by the breakthrough result of Agrawal, Kayal and Saxena devising an unconditional, deterministic, polynomial-time algorithm for distinguishing prime numbers from composite numbers. The solution of one of the basic problems in the discipline ushers in a new era. The main objective of the workshop is to consolidate the breakthrough and explore ramifications for other fundamental algorithmic problems in number theory and finite fields. In addition the workshop will look to the future of the subject and chart directions in which developments might occur.*

Non erano soltanto gli addetti ai lavori a sentire nell'aria qualcosa di veramente nuovo: nei mesi precedenti molti prestigiosi quotidiani internazionali si erano occupati della vicenda.

Giovedì 8 agosto del 2002 il New York Times usciva con un articolo di Sara Robinson dal titolo *New Method Said to Solve Key Problem in Math*, del quale riportiamo l'inizio:

> *Three Indian computer scientists have solved a longstanding mathematics problem by devising a way for a computer to tell quickly and definitively whether a number is prime - that is, whether it is evenly divisible only by itself and* 1.
>
> *Prime numbers play a crucial role in cryptography, so devising fast ways to identify them is important. Current computer recipes, or algorithms, are fast, but have a small chance of giving either a wrong answer or no answer at all.*

The new algorithm - by Manindra Agrawal, Neeraj Kayal and Nitin Saxena of the Indian Institute of Technology in Kanpur guarantees a correct and timely answer.

Though their paper has not been published yet, they have distributed it to leading mathematicians, who expressed excitement at the finding.

'This was one of the big unsolved problems in theoretical computer science and computational number theory ', said Shafi Goldwasser, a professor of computer science at the Massachusetts Institute of Technology and the Weizmann Institute of Science in Israel, 'it's the best result I've heard in over 10 years'.

Effettivamente qualche giorno prima, domenica 4 agosto, i tre autori avevano inviato il loro preprint ad alcuni esperti. La sera stessa cominciarono ad arrivare entusiastici messaggi di congratulazioni.

Uno dei massimi leader del campo, Carl Pomerance, verificò il risultato, organizzò sul momento un seminario e informò il New York Times.

L'articolo '*Primes is in* P' [2] è rimasto in rete dal 6 agosto 2002, a disposizione dell'intera comunità scientifica. È stato via via modificato fino ad arrivare alla versione numero 6 [4] .

Diversi matematici, tra i quali ricordiamo Daniel J. Bernstein [7][8], Pedro Berrizbeitia [10], Hendrick Lenstra Jr. e Carl Pomerance [23], hanno suggerito miglioramenti e trovato scorciatoie.

Nel 2004 il lavoro è stato pubblicato sul numero **160** degli Annals of Mathematics [3]. L'idea di base di AKS affonda le radici nel seguente

TEOREMA 4.1. – *Sia a un qualsiasi intero coprimo con n. Allora n è primo se e solo se*

$$(x+a)^n = x^n + a$$

nell'anello dei polinomi $\mathbb{Z}_n[x]$. □

La dimostrazione di 4.1 non è difficile e utilizza due fatti ben noti. Il primo riguarda i coefficienti binomiali:

n è primo se e solo se n divide $\binom{n}{k}$, per ogni k, $0 < k < n$.

Il secondo è il Piccolo Teorema di Fermat [11], 4.3, scritto nella forma

$$a^n \equiv a(\text{mod } n), \text{ per ogni } a \in \mathbb{Z}_n.$$

Il Teorema 4.1 è in effetti un *criterio* di primalità deterministico. Si tratta però di un algoritmo non efficace. Il problema non sta tanto nel numero dei passi che occorrono per elevare $x + a$ a n (mediante successive quadrature sono sufficienti $O(\log n)$ passi) quanto nella *esplosione* del numero dei termini coinvolti nel calcolo. Infatti l'espressione $(x + a)^n$ contiene $n + 1$ monomi: si tratta di una stringa di $n + 1$ elementi, e se n ha 100 cifre non basterebbero gli atomi dell'universo intero per contarli tutti...

AKS hanno allora pensato di *fissare* la lunghezza della stringa, calcolando tutto non soltanto modulo n, ma anche modulo $x^r - 1$ dove l'esponente r sia *piccolo* rispetto a n.

In questo modo si lavora in un ambiente diverso, l'anello

$$R_{n,r} = \mathbb{Z}_n[x]/(x^r - 1).$$

È possibile che non tutti abbiano familiarità con le operazioni di $R_{n,r}$, per cui le esplicitiamo, e facciamo un esempio.

Un elemento $a \in R_{n,r}$ è una stringa di lunghezza r, $a = (a_0, a_1, \ldots, a_{r-1})$, $a_i \in \mathbb{Z}_n$, che rappresenta il polinomio $A(x) = a_0 + a_1 x + \ldots + a_{r-1} x^{r-1}$.

La somma di due stringhe è definita nel modo usuale.

Il prodotto tra due stringhe, α e β, si effettua pensando ai corrispondenti polinomi $A(x)$ e $B(x)$. Si ottiene così un polinomio $C(x)$ di grado $\leq 2r - 2$. Ora $C(x)$ va ridotto modulo $x^r - 1$. La riduzione modulo $x^r - 1$ si effettua riducendo gli *esponenti* modulo r e raccogliendo poi i monomi dello stesso grado. Infine bisogna ridurre tutti i coefficienti modulo n: a questo punto si ha una stringa γ che sta in $R_{n,r}$ ed è il risultato del prodotto.

Si tratta insomma di *due* diverse riduzioni, una dei coefficienti (piano terra) modulo n, e una degli esponenti (primo piano) modulo r. Per il resto si fanno somme e prodotti come al solito tra polinomi.

ESEMPIO
Poniamo $n = 101$, $r = 5$, $a = (1, 5, 8, 0, 87)$ e $\beta = (70, 0, 91, 6, 0)$.
I corrispondenti polinomi sono

$$A(x) = 1 + 5x + 8x^2 + 87x^4, \quad B(x) = 70 + 91x^2 + 6x^3.$$

Il loro prodotto in $\mathbb{Z}[x]$ è

$$C(x) = 70 + 350x + 651x^2 + 461x^3 + 6848x^4 + 48x^5 + 7917x^6 + 522x^7.$$

Riducendo gli esponenti modulo 5 e raccogliendo otteniamo

$$C(x) = 118 + 8267x + 1173x^2 + 461x^3 + 6848x^4.$$

Riduciamo adesso i coefficienti modulo 101, e troviamo

$$17 + 86x + 62x^2 + 57x^3 + 81x^4.$$

Infine, il prodotto di a per β in $R_{101,5}$ è:

$$\gamma = (17, 86, 62, 57, 81).$$

Naturalmente, passando da $\mathbb{Z}_n$ a $R_{n,r}$ cessa l'esplosione della lunghezza delle stringhe, ma si paga per questo un costo notevole: il Teorema 4.1 funziona solo più da un lato, diventando

TEOREMA 4.2. – *Se n è primo, allora, per ogni $a \in \mathbb{Z}$ e per ogni intero positivo r si ha*:

$$(x + a)^n = x^n + a \text{ in } R_{n,r}.$$

□

Il Teorema 4.2 eprime una condizione *necessaria* affinché un intero sia primo. Si presta dunque ad un test di primalità, come abbiamo visto nel paragrafo 4 di [11].
Kayal e Saxena hanno studiato a fondo questo test in [20].

DEFINIZIONE 4.3. – *Diciamo che un intero n* **passa il test di Kayal e Saxena sulla base** *r se*

$$(x - 1)^n = x^n - 1 \text{ in } R_{n,r}.$$

Se n è composto e passa il test viene detto **KS-pseudoprimo** *sulla base r.*

In [20] Kayal e Saxena hanno dimostrato che, se n è un KS-pseudoprimo sulla base r, allora n è anche uno pseudoprimo di Eulero su r ([11], 4.8).

Da questo segue immediatamente che:

1) non esistono interi composti che passano il test KS su tutte le basi.

2) se si ammette la ERH, il test KS diventa un criterio di primalità deterministico e polinomiale.

Hanno inoltre provato che, se un intero n composto passa il test KS sulla base 5, allora n è uno pseudoprimo forte di Fibonacci ([11], 4.16).

Pomerance ha congetturato che un intero n composto può passare sia il test forte di Fibonacci che il test di Fermat solo se soddisfa l'equazione $n^2 \equiv 1 (\mathrm{mod}\ 5)$.

Una forma generalizzata di questa congettura è stata presentata in [28]:

4.4. – *Congettura di Pandey - Bhattacharjee.*

Siano n un numero composto ed r un primo che non divide n. Se

$$(x+1)^n = x^n + 1 \quad in \quad \mathbb{Z}_n[x]/(x^r+1)$$

allora $n^2 \equiv 1 (\mathrm{mod}\ r)$.

Sempre in [20] KS hanno verificato la congettura 4.4 per $r = 5$ fino a $n = 10^{11}$.

Si tratta evidentemente di una strada molto promettente.

Se guardiamo al Teorema 4.2 osserviamo subito che in 4.3 e in 4.4 si utilizza soltanto $a = \pm 1$. AKS hanno constatato che la possibilità di *variare* a aumenta *essenzialmente* la quantità di informazione. Si tratta di una sorta di TAC del numero. Per ricostruire la struttura interna del numero occorre un ammontare minimo di dati. Occorre cioè far variare a in un intervallo sufficientemente ampio.

Ricordiamo ora la definizione di periodo e precisiamo la notazione.

DEFINIZIONE 4.5. – *Se r è un intero coprimo con n, diciamo* **periodo di n modulo r**, *e lo denotiamo con $o_r(n)$, il minimo intero positivo t tale che $n^t \equiv 1 (\mathrm{mod}\ r)$.*

Possiamo finalmente enunciare il teorema di Agrawal, Kajal e Saxena ([2], [3], [4]). Nel seguito utilizzeremo la loro notazione e indicheremo con log il logaritmo in base 2.

TEOREMA 4.6. – **Teorema AKS**
Sia n un intero ≥ 2.
Sia r un intero positivo coprimo con n tale $o_r(n) > \log^2 n$.
Allora n è primo se e solo se
1) *n non è una potenza perfetta*
2) *n non possiede fattori primi $\leq r$*
3) $(x+a)^n \equiv x^n + a$, *in* $R_{n,r}$, *per ogni* a, $1 \leq a \leq \sqrt{\varphi(r)} \log n$. □

Dunque, per provare che n è primo, è sufficiente, *nelle ipotesi fatte su r*, eseguire il test per al più $\sqrt{\varphi(r)} \log n$ interi a.

L'algoritmo descritto nel Teorema AKS 4.6 è *polinomiale*?

Il punto 1) non costituisce un problema. Esistono algoritmi polinomiali molto veloci ([6]) che determinano se un intero n è della forma b^e, con $e > 1$.

Un esame dei punti 2) e 3) di 4.6 evidenzia che tutto dipende dalla grandezza di r. Si arriva pertanto alla seguente conclusione:

4.7. – *L'algoritmo descritto nel Teorema AKS è polinomiale, se e solo se esiste k indipendente da n tale che r è $O(\log^k n)$.*

Rimane infine solo una domanda:

4.8. – ***Quanto** deve essere grande il minimo intero positivo r tale che $o_r(n) > \log^2 n$?*

È proprio a questo punto che i nostri AKS si arresero, momentaneamente, dopo avere dimostrato il teorema 4.6. Non avevano la minima idea di come limitare superiormente, in modo

efficace, questo r, che nella versione iniziale doveva essere un numero primo.

Scoprirono poi un risultato di E. Fouvry ([17]), sufficiente allo scopo. Si tratta di una proprietà assai riposta, che si basa su sofisticati argomenti di teoria analitica dei numeri, pubblicata nel 1985, 17 anni prima di AKS .

In seguito, come si è accennato nell'introduzione, molte cose si sono semplificate, e si è visto che è possibile utilizzare risultati che richiedono soltanto tecniche elementari.

Nella versione pubblicata sugli Annals [3] utilizzarono questo teorema, la cui dimostrazione appare in un lavoro di M. Nair del 1982 [27]:

TEOREMA 4.9. – *Denotiamo con $G(m)$ il minimo comune multiplo dei primi m interi. Per $m > 6$ si ha che $G(m) \geq 2^m$.* □

Si può ora rispondere alla domanda 4.8.

TEOREMA 4.10. – *Esiste un $r < \max(3, \lceil \log^5 \rceil)$ coprimo con n tale che $o_r(n) > \log^2 n$.*

DIMOSTRAZIONE. – Il teorema è banalmente vero se $n = 2$: infatti $r = 3$ soddisfa la condizione. Supponiamo quindi $n > 2$.

Poniamo $B = \lceil \log^5 n \rceil$ e consideriamo il prodotto

$$P = n^{\lfloor \log B \rfloor} \prod_{j=1}^{\lfloor \log^2 n \rfloor} (n^j - 1).$$

Si vede facilmente che valgono le seguenti diseguaglianze:

$$P = n^{\lfloor \log B \rfloor} \prod_{j=1}^{\lfloor \log^2 n \rfloor} (n^j - 1) < n^{\lfloor \log B \rfloor + \frac{1}{2}\log^2 n(\log^2 n - 1)} \leq n^{\log^4 n} \leq 2^{\log^5 n} \leq 2^B.$$

Se tutti gli interi compresi tra 1 e B dividessero P avremmo che $G(B)|P$ e, di conseguenza, $G(B) \leq P < 2^B$, mentre per 4.9 si deve avere $G(B) \geq 2^B$.

Pertanto esiste almeno un

$$r \leq \lceil \log^5 n \rceil$$

che non divide P.

Sia $r = p_1^{e_1} p_2^{e_2} \dots p_h^{e_h}$ la fattorizzazione di r. Da $r \leq B$ segue subito che $e_j \leq \lfloor \log B \rfloor$, per ogni j (si ricordi che log è in base 2).

Ne consegue che, se tutti i primi che dividono r dividessero anche n, allora r dividerebbe $n^{\lfloor \log B \rfloor}$, cosa che non può essere perché $r \nmid P$.

Pertanto esiste almeno un primo p che divide r e non n. Dunque anche $\dfrac{r}{(r,n)}$ *non divide* il prodotto P.

Si ottiene da questo che il *minimo* r che non divide P è *coprimo* con n.

Ricordiamo che $o_r(n) \leq \log^2 n$ se e solo se r divide $n^j - 1$ per un $j \leq \log^2 n$. Poiché r non divide $n^j - 1$ per ogni $j \leq \lfloor \log^2 n \rfloor$ abbiamo infine che $o_r(n) > \lfloor \log^2 n \rfloor$. □

Questo conclude il nostro itinerario. L'algoritmo AKS è effettivamente deterministico, incondizionato e polinomiale!

5. – Sinergie.

Spesso accade che un risultato importante e inatteso susciti inizialmente stupore e attenzione, ma finisca presto tra le cose fatte, gli argomenti conclusi. Non è stata questa la sorte del teorema AKS.

Dal 2002 sono stati pubblicati molti articoli che ne migliorano i tempi di esecuzione, ne danno nuove e illuminanti dimostrazioni, lo confrontano con altri metodi.

In *Primalité théorique et primalité pratique, ou AKS vs ECPP* ([24] (2002)) Morain confronta direttamente AKS e ECPP.

Bernstein fu uno dei primi a studiare l'impatto dell'AKS nel campo, da lungo tempo consolidato, dei test di primalità, e a indicare nuove direzioni (si vedano *Proving primality after Agrawal-Kayal-Saxena* ([7] (2003)) e *Proving primality in essential quartic random time* ([8] (2003)). Bernstein fece il punto della situazione nel 2004

nell'articolo *Distinguishing prime numbers from composite numbers: the state of the art in 2004* ([9]).

Nella sua tesi *An Efficient Implementation of the AKS Polynomial-Time Primality Proving Algorithm*, Rotella ([30] (2005)) compie una interessante analisi dei tempi di esecuzione dell'algoritmo AKS.

In *Primality testing with Gaussian Periods*, preliminary version, July 2005 ([23]) Lenstra e Pomerance mostrano che è possibile utilizzare in AKS, al posto di $x^r - 1$, particolari polinomi interi $f(x)$ che si comportano «come se fossero irriducibili» in $\mathbb{Z}_n[x]$.

Berrizbeitia in *Sharpening* Primes is in P *for a large family of numbers* ([10] (2005)), propone una variante di AKS che consente di provare la primalità con *un solo test* per una classe particolare di interi. Precisamente dimostra che:

TEOREMA 5.1. – *Sia dato un intero n, non potenza perfetta, $n \equiv 1 (\mathrm{mod}\ 4)$. Poniamo $s = \lceil 2 \log \log n \rceil$. Supponiamo che $2^k \| n-1$ e $k \geq s$. Se esiste un intero a tale che $\left(\frac{a}{n}\right) = -1$ e $a^{\frac{n-1}{2}} \equiv -1 (\mathrm{mod}\ n)$ allora*

$$(1+x)^n \equiv 1 + x^n (\mathrm{mod}\ n, x^{2^s} - a)$$

se e solo se n è primo. □

(Ricordiamo che $2^k \| a$ significa che $2^k | a$ e $2^{k+1} \nmid a$.)

Cheng in *Primality Proving via One Round in ECPP and One Iteration in AKS* ([13], 2005) estende la 5.1 ad una classe più vasta di interi e fa una osservazione, a nostro parere, molto interessante:

AKS e ECPP possono agire *insieme*. Se l'intero n in ingresso non è della particolare forma desiderata si può ridurre, con l'algoritmo ECPP (3.17) il problema della primalità di n a quello della primalità di un intero q che sia del tipo voluto.

Sembra proprio che siamo all'inizio di qualcosa di veramente nuovo!

6. – Conclusioni.

L'impegno scientifico ed economico che viene oggi rivolto alla ricerca veloce di primi grandi, in termini di intelligenze, strutture e investimenti, è veramente notevole.

Al convegno di Palo Alto del marzo 2003, di cui abbiamo parlato all'inizio del quarto paragrafo, erano presenti certamente inviati della CIA, della NSA e di altre agenzie mondiali di Intelligence.

Come è noto molti protocolli crittografici attuali utilizzano, per la costruzione delle chiavi, numeri primi di centinaia o migliaia di bit, a seconda dei livelli di sicurezza (si veda [21]).

Persino i grandi quotidiani sono diventati sensibili alle scoperte sui numeri primi!

Nel Settembre del 2004 si sparse la voce che Louis de Branges de Bourcia aveva dimostrato la **RH**: i mass media diffusero immediatamente la preoccupante notizia che questo avrebbe potuto compromettere la sicurezza dei sistemi elettronici di transazione commerciale ([12]) ([1]).

Ricordiamo che la RH è tra i problemi aperti più importanti della matematica, ed è uno dei famosi 23 problemi posti da Hilbert ai matematici riuniti nel Congresso Internazionale di Parigi nell'agosto 1900. Inoltre è uno dei 'magnifici sette' per la cui soluzione il Clay Mathematics Institute offre un milione di dollari!!

Questa congettura implica (anche se non è affatto evidente) che i numeri primi sono distribuiti nel modo 'più uniforme possibile'; ad esempio, nella seguente versione equivalente alla RH i primi sono direttamente coinvolti:

$$(\mathbf{RH1}) \qquad p_{n+1} - p_n = O(\sqrt{p_n} \log p_n)$$

dove p_n indica l'n-esimo numero primo.

([1]) Dopo qualche tempo venne messa in evidenza una possibile lacuna nella dimostrazione pubblicata. In seguito Louis de Branges produsse altre versioni, sino all'ultima del Dicembre 2005 - Gennaio 2006 [15], [16]. Mentre egli è assolutamente certo del suo risultato (come si può leggere nella sua 'Apology for the Proof of the Riemann Hypothesis' [14]), la comunità matematica non ha ancora preso una posizione certa sulla dimostrazione.

Come diceva Atle Selberg:

'L'ipotesi condurrebbe alla distribuzione più naturale dei numeri primi: tendi a pensare che almeno qualche cosa debba andare diritta in questo universo!'

La 'grande paura' era dovuta non alle conseguenze della RH sulla regolarità della distribuzione dei primi, ma a temuti 'effetti collaterali' riguardanti immaginarie applicazioni della dimostrazione di de Branges alla fattorizzazione degli interi.

Molti tra i più usati sistemi crittografici (citiamo per tutti il ben noto RSA [21]) si basano sul fatto che è facile trovare primi grandi ma è per ora impossibile (in tempi umani) fattorizzare numeri interi grandi.

I tentativi in questa direzione hanno originato algoritmi di grandissimo interesse matematico, ma tutti di complessità subesponenziale. Insomma rimane molto da fare.

Come diceva Lenstra scherzando, se scrivi due numeri primi di 100 cifre su un foglio, e il loro prodotto su un secondo foglio, e poi la donna delle pulizie ti butta via il primo, la tua unica speranza di ritrovare i due primi iniziali è di frugare nella spazzatura ....

RIFERIMENTI BIBLIOGRAFICI

[1] L. M. Adlemam - M. A. Huang, *Primality Testing and Abelian Varieties Over Finite Fields*, Lecture Notes in Mathematics, 1512, Springer-Verlag 1992.

[2] M. Agraval - N. Kayal - N. Saxena, *PRIMES is in P*, 2002.

[3] M. Agraval - N. Kayal - N. Saxena, *PRIMES is in P*, Annals of Mathematics, **160** (2004), 781-793.

[4] M. Agraval - N. Kayal - N. Saxena, *PRIMES is in P*, URL: http://www.cse.iitk.ac.in/users/manindra/primality v6.pdf, 2005.08.11

[5] A.O.L. Atkin - F. Morain, *Elliptic curves and primality proving*, Mathematics of Computation, **61** (1993), 29-68.

[6] D. J. Bernstein, *Detecting perfect powers in essentially linear time*, Mathematics of Computation, **67** (1998), 1253-1283.

[7] D. J. Bernstein, *Proving primality after Agrawal-Kayal-Saxena*, URL: http://cr.yp.to/papers.html#aks, 2003.01.25

[8] D. J. Bernstein, *Proving primality in essential quartic random time*, URL: http://cr.yp.to/papers.html#quartic, 2003.01.28, in corso di pubblicazione su Mathematics of Computation.

[9] D. J. BERNSTEIN, *Distinguishing prime numbers from composite numbers: the state of the art in 2004*, URL: http://cr.yp.to/papers.html#prime2004, 2004.12.23
[10] P. BERRIZBEITIA, *Sharpening Primes is in P for a large family of numbers*, Mathematics of Computation, **74** (2005), 2043-2059.
[11] L. CAIRE - U. CERRUTI, *Questo numero è primo? Forse sì , dipende...*, Bollettino U.M.I. sez. A, la Matematica nella Società e nella Cultura, Serie VIII, Vol. IX-A, Dicembre 2006/1, 449-481.
[12] U. CERRUTI, *Congettura di Riemann e sicurezza mondiale* URL:http: // alpha01.dm.unito.it/ personalpages/cerruti/luglio04-gennaio28.html#riemann
[13] QI CHENG, *Primality Proving via One Round in ECPP and One Iteration in AKS*, Lecture Notes in Mathematics, 2729, Springer-Verlag 2003.
[14] L. DE BRANGES DE BOURCIA, *Apology for the Proof of the Riemann Hypothesis*, 2006.04.25, http://www.math.purdue.edu/ branges/apology.pdf
[15] L. DE BRANGES DE BOURCIA, *A proof of the Riemann Hypothesis I*, 2005.12.29, http://www.math.purdue.edu/ branges/riemannzeta.pdf
[16] L. DE BRANGES DE BOURCIA, *A proof of the Riemann Hypothesis II*, 2006.01.11, http://www.math.purdue.edu/ branges/riemannII.pdf
[17] E. FOUVRY, *Théorème de Brun-Tichmarsh; application au théorème de Fermat*, Invent. Math., **79** (1985), 383-407.
[18] S. GOLDWASSER - J. KILIAN, *Almost all primes can be quickly certified*, Proceedings of the 18-th Annual ACM Symposium on Theory of Computing, New York, 1986, 316-329.
[19] S. GOLDWASSER - J. KILIAN, *Primality Testing using Elliptic Curves*, Journal of the ACM, **46** (1999), 450-472.
[20] N. KAYAL - N. SAXENA, *Toward a deterministic polynomial time primality tests*, URL: http://www.cse.iitk.ac.in/research/btp2002/primality.html, 2002
[21] A. LANGUISCO - A. ZACCAGNINI, *Introduzione alla Crittografia*, Hoepli Editore, Milano, 2004.
[22] H. LENSTRA - J. PILA - C. POMERANCE (organizers), *Future directions in algorithmic number theory*, Workshop, March 24 -28, 2003, Palo Alto, California, http: //www. aimath.org/ARCC/workshops/primesinp.html
[23] H. W. LENSTRA, JR. - C. POMERANCE, *Primality testing with Gaussian Periods*, preliminary version, July 2005, URL: http://www.math.dartmouth.edu/ carlp/ PDF/complexity072805.pdf
[24] F. MORAIN, *Primalité théorique et primalité pratique, ou AKS vs ECPP*, 2002, URL: http://www.lix.polytechnique.fr/Labo/Francois.Morain/aks-f.pdf, 2002.10.04
[25] F. MORAIN, *Implementing the asymptotically fast version of the Elliptic Curve Primality Proving Algorithm*, 2005, URL: http: //www.lix.polytechnique. fr/ Labo/Francois. Morain/Articles/fastecpp-final.pdf
[26] S. MÜLLER, *On the combined Fermat/Lucas probable prime test, in Crypto & Coding '99*, Lecture Notes in Computer Science 1746, Springer-Verlag 1999, 222-235.

[27] M. NAIR, *On Chebyshev-type inequalities for primes*, The American Mathematical Monthly, **89** (1982), 126-129.
[28] P. PANDEY - R. BHATTACHARJEE, *Primality testing, Thesis*, April 2001, URL: http://www.cse.iitk.ac.in/research/btp2001/primality.ps.gz, 2001
[29] V. R. PRATT, *Every prime has a succinct certificate*, SIAM Journal on Computing, **4** (1975), 214-220.
[30] C. ROTELLA, *An Efficient Implementation of the AKS Polynomial-Time Primality Proving Algorithm*, SCS Undergraduate Thesis (Advisor: Klaus Sutner), Carnegie Mellon, May 2005.
[31] R. SCHOOF, *Elliptic curves over finite fields and the computation of square roots mod p*, Math. Computation, **44** (1985), 483-494.
[32] P. SERAFINI, *Introduzione alla complessità computazionale*, URL:http:// www.dimi.uniud.it/ serafini/complcomp.pdf

Luisella Caire: Dipartimento di Matematica, Politecnico di Torino,
Corso Duca degli Abruzzi 24, 10129 Torino, Italy,
e-mail: luisella.caire@polito. it

Umberto Cerruti: Dipartimento di Matematica, Università di Torino,
Via Carlo Alberto 10, 10121 Torino, Italy,
e-mail: umberto.cerruti@unito.it