Informatica: scienza (teorico/pratica) che si occupa di come rendere "automatica" la gestione di "informazioni" (in formato prevalentemente "digitale", grazie soprattutto all'avvento del computer).
Teoria dei numeri: la più antica delle teorie matematiche, la "regina" della matematica (K.F.Gauss). Considerata matematica "pura", dalla seconda metà del XX secolo ha cominciato ad avere numerose applicazioni (grazie soprattutto all'avvento del computer).
Chiediamoci:
a) dove non è possibile un'interazione tra le due discipline?
b) quali interazioni sono possibili, ma richiedono enormi risorse?
c) quali interazioni sono possibili con "poche" risorse?
e tentiamo di dare una risposta sensata a queste tre domande.
a) l'informatica non è di nessun aiuto alla dimostrazione della verità di teoremi (di teoria dei numeri) che hanno validità in infiniti casi. Per esempio, si prenda l'ultimo teorema di Fermat ($x^n+y^n=z^n$ non ha soluzioni intere in x, y, z, tutte diverse da 0 per ogni $n>2$ intero).
b) l'informatica (da "sola") è di scarso aiuto alla risoluzione di problemi di "tipo NP", per i quali non è noto se esiste un algoritmo risolutivo in "tempo polinomiale", dove il numero di "operazioni" elementari per giungere alla soluzione (a partire dai dati) non è maggiorabile da una potenza intera avente esponente fissato e per base il numero dei dati in ingresso (es: commesso viaggiatore). In casi come questi, però, esiste uno stimolo a ricercare **algoritmi** più efficienti e, in qualche circostanza, la ricerca è arrivata a produrre algoritmi "polinomiali" (es: algoritmo AKS, verifica di primalità di un intero). Se si riesce a raggiungere un siffatto obiettivo, il passo successivo può consistere in un miglioramento dell'algoritmo (una riduzione dell'esponente fissato).
c) È il caso di cui mi occupo principalmente. L'avere a disposizione poche risorse stimola a trovare soluzioni più efficienti. Gli esempi possibili sono innumerevoli: dai classici problemi di tipo puramente "informatico" (ordinamento e ricerca) alla verifica della verità (o della falsità) di qualche congettura. Nel 1976 è stato "dimostrato", ad esempio, il teorema relativo alla colorazione di una generica mappa su una superficie di genere 0 (es: il piano. **4 colori** sono necessari e sufficienti perché 2 regioni confinanti abbiano colori diversi), ma la dimostrazione ha richiesto l'utilizzo di un computer (e di un programma apposito messo a punto da 2 programmatori). Viceversa, una congettura di Eulero si è dimostrata falsa (nel 1966), grazie sempre all'utilizzo di un computer (è bastato un **controesempio**). È il caso di considerare come un'analisi accurata del problema da affrontare permette di ridurre il numero totale di possibilità da esaminare per giungere alla soluzione, senza basarsi sulla semplice "forza bruta" dei calcoli (tanto li fa il computer).
Una delle interazioni più feconde tra Informatica e Teoria dei Numeri riguarda la questione relativa all'**efficienza** e alla **sicurezza** delle trasmissioni a distanza di informazioni (digitali). È noto che oggi la maggioranza delle informazioni "viaggia" in forma digitale, per cui, se si vogliono ridurre i costi e aumentare l'efficienza di tali trasmissioni, è utile avere dei buoni algoritmi di "compressione" (ci ha già pensato Huffmann), tali da "condensare" l'informazione nel minimo numero possibile di "cifre" in modo da ridurre il carico delle linee di trasmissione, senza che l'informazione perda qualcosa a livello di contenuto (mai sentito parlare di Winzip, Rar,…?).
Purtroppo, risolto il problema dell'efficienza, resta quello della sicurezza (i due problemi si possono ragionevolmente considerare indipendenti). In campo informatico, inoltre, il termine sicurezza possiede (almeno) due significati: **integrità** e **riservatezza** (anche questi si possono considerare indipendenti).
Integrità: devo poter garantire che i dati arrivino tutti a destinazione e qui sia possibile verificare che coincidano in tutto e per tutto con i dati trasmessi. Se è virtualmente impossibile prevenire un qualsiasi disturbo su un canale di trasmissione, può essere indispensabile trasmettere delle informazioni (cifre) aggiuntive in quantità non eccessiva e che possano permettermi di scoprire se è avvenuto un errore sul canale ed eventualmente di individuarlo (e correggerlo: ci ha già pensato Hamming).

Riservatezza: devo poter garantire che nessun altro, al di fuori del legittimo destinatario (fosse anche più di 1) possa intercettare i dati in transito e utilizzarli. Se è virtualmente impossibile prevenire una qualsiasi intercettazione, può essere indispensabile nascondere i dati in modo che chiunque li intercetti non possa utilizzarli (o debba impiegare troppo tempo per farlo). Esiste una disciplina in proposito (la crittologia=crittografia+crittoanalisi). A proposito di integrità: non è detto che i dati si alterano solo a causa della grande distanza a cui spesso vengono trasmessi. Capita spesso (più di quanto si pensi) che i dati vengano alterati già in fase di elaborazione! Un guasto può verificarsi nel "cuore" di un computer o nelle memorie (interne o esterne). È possibile (e doveroso) prevenire casi del genere e, in proposito, la Teoria dei Numeri ci è di grande aiuto. Non esiste collaudo migliore di un computer che una lunga sequenza di calcoli che coinvolga tutti i dispositivi da controllare e fornisca un determinato risultato (noto a priori). È di fatto impossibile il verificarsi di errori tali da compensarsi quando si ottiene il risultato corretto. Questo spiega in parte perché, ad es., è stata verificata la primalità di numeri di diverse migliaia di cifre (anche più di un milione) e sono state calcolate miliardi di cifre significative di alcune notevoli costanti matematiche (e, π, …) quando la conoscenza di poche cifre (un centinaio) è più che sufficiente per scopi pratici ed è già stata calcolata a mano da diversi decenni.
Sempre a proposito di integrità: esistono due diversi approcci al problema. Uno semplicemente di tipo diagnostico (so che si è avuto un errore, ma non so dove e, quindi, non so rimediare) e uno di tipo autocorrettore (so esattamente dove si è verificato un errore, quindi posso intervenire e ricostruire la cifra esatta in fase di ricezione). Il secondo approccio è più lungo, complesso e costoso del primo (anche perché necessita di un maggior numero di cifre aggiuntive).
Il primo approccio fornisce comunque la base di partenza per tutti gli altri, indipendentemente dal numero di errori rilevabili. Se, ad esempio, sono sicuro che, al massimo, si verifica un errore, per sapere se si è realmente verificato, posso aggiungere un solo bit in trasmissione, in modo che i bit = 1 siano complessivamente in numero pari. In ricezione so che ho un errore se i bit = 1 sono in numero dispari (controllo di parità o parity check).
Più complesso risulta correggere un solo errore. Vediamo come:
Supponiamo di dover trasmettere un blocco di dati. Dividiamo il blocco in questione in "blocchetti" di 4 bit l'uno (le configurazioni possibili sono 2^4=16). Ad ogni blocchetto aggiungiamo 3 bit (sono apparentemente ridondanti, ma in realtà servono a smascherare l'eventuale errore). Ogni blocchetto è costituito pertanto da 7 bit che possono anche essere tutti uguali (0000 000 oppure 1111 111). Escluso questo caso (banale), considero una particolare sequenza di bit (0001 011) e la sua complementare a 1 (1110 100). Non è difficile (semmai è noioso), verificare che da tali sequenze posso ottenere, tramite shift (per esempio a sinistra: 0010 110, 1101 001, …) tutte le possibili configurazioni dei primi 4 bit (purché non tutti uguali). A questo punto (scritta la tabella) è sufficiente che ad ogni blocchetto io aggiunga i rimanenti 3 e possa confrontare quanto ricevuto con una delle 16 (sole) possibili configurazioni di 7 bit. Mi si possono presentare i casi seguenti:
a) Quanto ricevuto coincide con una delle 16 sequenze ammesse o differisce al più per uno dei 3 bit più a destra. In questo caso i primi 4 bit (a sinistra) vengono considerati corretti (scarto quelli di destra, perché ridondanti, distanza di Hamming <= 1).
b) Quanto ricevuto differisce da una delle 16 sequenze ammesse per uno solo dei 4 bit a sinistra (distanza = 1). I 4 bit originali trasmessi si possono ottenere da quelli ricevuti modificando l'unico bit supposto errato (tramite complemento a 1).
c) Quanto ricevuto differisce (considerando tutti i 7 bit) da una delle 16 sequenze ammesse per più di 1 bit (distanza > 1). Non ho alcuna possibilità di ricostruire con certezza il pacchetto originario.
Tutto ciò significa, tecnicamente, che il codice di Hamming appena visto è di tipo **autocorrettore** se ho al più un errore e (si può dimostrare) **diagnostico** se ho al più 2 errori. Ulteriori sofisticazioni (aggiunta di bit di ridondanza opportunamente calcolati) permette di ottenere codici di Hamming autocorrettori per n errori e diagnostici per almeno 2n errori (con n intero e >1).

Occupiamoci ora del problema della riservatezza.
Sia M il messaggio da trasmettere. È possibile trovare una funzione opportuna (f) che lo trasformi in una sequenza diversa di cifre (M'). Vediamo le caratteristiche salienti di tale funzione:
a) essendo M, M' due numeri naturali tale funzione ha dominio e codominio in un sottoinsieme finito dell'insieme N.
b) deve essere facilmente invertibile (ma solo da parte del legittimo destinatario).
c) deve rendere incomprensibile M (M' deve essere incomprensibile rispetto ad M).
d) deve avere "periodo molto lungo". Non posso accettare che, partendo da M, poche applicazioni di f mi riportino ancora ad M.
Cerchiamo una possibile soluzione.
Vediamo, passo passo, in cosa consistono questi requisiti.
a) la f non è una funzione "continua" essendo dominio e codominio sottoinsiemi finiti di N (non possiedono punti di accumulazione).
b) devo assicurare che l'inversa di f, oltre a esistere, sia unica (cosa non del tutto scontata).
c) devo far sì che sia difficile (o enormemente costoso in termini di tempo e risorse richieste) trovare tale inversa (decifrare M' per avere M da parte di chiunque, escluso il legittimo destinatario).
C'è poi da aggiungere una ulteriore considerazione relativa al punto d, ma occorre prima fare qualche considerazione preliminare.
- La funzione f di solito è nota (sono incogniti solo i parametri o chiavi). È quindi necessario "proteggere" solo le chiavi. Esistono due approcci distinti:
Uno a chiave "privata" (cifratura simmetrica, es. DES), molto efficiente e veloce, in cui mittente e destinatario concordano in precedenza una chiave (da cambiare spesso), uguale per entrambi e segreta a tutti gli altri. La decifratura coincide con la cifratura.
Uno a chiave "pubblica" (cifratura asimmetrica, es. RSA), più lento, ma molto più sicuro (utilizzato, spesso, per trasmettersi la chiave "privata"). Qui, essendo le chiavi di cifratura e decifratura diverse, la prima può essere tranquillamente pubblicata, purché da questa sia molto costoso (e lungo) riuscire a risalire alla seconda. È in questo secondo caso che è richiesta la caratteristica d) vista prima, altrimenti può darsi che pochi e ripetuti tentativi di cifratura possano dar luogo, di fatto, ad una decifratura.
Vediamo se è possibile usare la teoria dei numeri per trovare una strada percorribile nel secondo caso.
Stando a quanto accennato in precedenza, esistono algoritmi veloci che permettono di stabilire in tempi rapidi la **primalità** di un intero anche abbastanza grande (diciamo di qualche centinaio di cifre), ma non esistono, a tutt'oggi, algoritmi altrettanto veloci ("polinomiali") in grado di **scomporre** un numero nei suoi fattori primi, anche se il più piccolo di questi avesse "solo" un centinaio di cifre. È possibile sfruttare questo aspetto nel modo seguente (algoritmo RSA).
- scelgo due primi (grandi) **p**, **q** e calcolo **N=pq**, **K=(p-1)(q-1)**.
- scelgo un numero **s** tale che **MCD(s,K)=1**.
- calcolo **t** in modo che **st** diviso K dia resto **1**.
- pubblico **N** e **t** e tengo segreto **s** (posso scambiare i ruoli di s e t).
È possibile dimostrare che, dato un M<N tale che MCD(M,N)=1, se definisco M' come il resto di M^t diviso N, risulta anche che M è il resto di $(M')^s$ diviso N.
Esempio numerico (si prega di non utilizzarlo per scambiare messaggi con chiave pubblica se si vuole che restino segreti!):
sia **p=11**, **q=17**, **s=23**. Ho **N**=pq=**187**, **K**=(p-1)(q-1)=**160**, t=**7**. I numeri pubblicati sono **187** e **7**, quello "segreto" è **23**.
Sia ora **M=13**. Trovo **M'=106** (numero da trasmettere: elevo 13 alla potenza 7, divido per 187 e prendo il resto). Supponiamo che M' venga ricevuto correttamente. Il ricevente eleva 106 alla potenza 23, divide per 187 e trova come resto… **13**. Magia? Niente affatto!

Esiste un teorema (di **Eulero**) che recita così:
se N è un intero positivo e K(<N) è il numero di interi positivi minori di N che non hanno divisori comuni con N (relativamente primi con N), allora, preso un qualsiasi M(<N), relativamente primo con N, M^K diviso per N dà sempre resto 1.
Nell'esempio appena proposto è come se avessimo elevato M alla potenza 161 (abbiamo elevato prima alla potenza 7, poi alla potenza 23 essendo 7*23=161). Ora, poiché il resto del prodotto è uguale al prodotto dei resti (v. proprietà delle classi, o insiemi, di resti) è come se avessimo elevato alla potenza 160 (guarda caso K), trovato il resto (Eulero ci dice che è 1) e moltiplicato per M (e ciò dovrebbe essere sufficiente a spiegare senza sorprese il risultato).
Un eventuale "hacker", per trovare il numero segreto (23), avrebbe dovuto scomporre N=187 in fattori primi (11.17), sottrarre 1 ad entrambi, moltiplicare i risultati ottenuti (trovo K=160), sommare 1 ad un opportuno multiplo di K (K stesso) ottenendo 161 e dividere per l'esponente "pubblico" (7), ottenendo effettivamente 161/7=23 (non devo, ovviamente, avere resto!). Dal punto di vista del calcolo automatico la "scomposizione" in fattori primi è il problema più lungo (gli altri calcoli prevedono tutti algoritmi di complessità polinomiale e sono molto rapidi).
Ed ora, per i curiosi, un po' di… crittoanalisi, della serie… cosa può fare il "povero hacker" davanti ad un N di 200 cifre (circa), prodotto di 2 primi p, q di 100 cifre (circa) l'uno e, girando la domanda, come è possibile, al contrario, trovare rapidamente i due numeri p e q (di circa 100 cifre l'uno) ed essere sicuri che siano realmente primi?
Comincio a rispondere alla seconda domanda. Se N è primo, usando il criterio di Eulero (semplificato da Fermat), ho, riciclando gli stessi nomi utilizzati in precedenza, K=N-1, 1<M<N. Esistono strategie varie (alcune anche molto efficienti, ma basate sull'ipotesi di Riemann, tuttora indimostrata) che permettono, scelti pochi M, di stabilire che, se per tutti gli M scelti, M^K dà resto 1 se diviso per N, allora N è primo (vale il cenno iniziale all'algoritmo AKS).
Resta più difficile rispondere alla prima domanda in modo efficiente e, in proposito, è possibile, nell'era del computer, tentare due diversi approcci:
Approccio deterministico: Una volta stabilito che il numero dato N è composto, posso cercare i suoi fattori primi (2 per ipotesi) usando delle varianti dell'algoritmo di Fermat (tramite frazioni continue) o l'algoritmo di Pomerance (crivello quadratico), preferibile perché più efficiente e implementabile in parallelo su più elaboratori.
Approccio probabilistico (metodo Montecarlo): Una volta stabilito che il numero dato N è composto, posso cercare i suoi fattori primi (2 per ipotesi) generando una sequenza di numeri pseudocasuali (il primo è preso "a caso", gli altri sono generati per ricorrenza) ed estraendo da questi (per tentativi) due numeri la cui differenza (o la cui somma) abbia un divisore "non banale" in comune con N. Vari esempi sono stati proposti da Pollard, Brent, Lenstra (metodo delle curve ellittiche, preferibile perché più efficiente e implementabile in parallelo su più elaboratori).
Vediamo qualche applicazione pratica (solo per illustrare qualche tipo di approccio).
Algoritmo di Fermat: Scelto N è possibile considerare qualche x tale che $x^2>N$. Se trovo che $x^2-N=y^2$ allora N=(x-y)(x+y). Esempio: sia N=901. Deve essere x>30. Il primo valore utilizzabile è x=35, che fornisce y=18 ed N=17*53.
Algoritmo di Pomerance: Sia N=111. Considero ancora qualche x tale che $x^2>N$ e scompongo il resto R di x^2 diviso N. Es: x=11, R=10=2*5, x=14, R=85=5*17, x=16, R=34=2*17. Nessun R è un quadrato perfetto, ma lo è il loro prodotto. Moltiplicando gli x trovo 11*14*16=2464 (che diviso per N=111 dà resto 22). Moltiplicando gli R trovo $(2*5*17)^2=170^2$ (170 diviso 111 dà resto 59). I divisori di 111 si possono trovare calcolando MCD(111,59±22) che danno 111=3*37.
Nota: per utilizzare al meglio l'algoritmo di Pomerance, oltre ad avere numeri grandi da scomporre (altrimenti non ne vale la pena) si può utilizzare qualche ulteriore accorgimento. Vediamo quali:

- Scegliere quali divisori primi considerare in numero sufficiente rispetto alla grandezza di N. Si scartano tutti quei resti che hanno almeno un fattore primo diverso rispetto a quelli scelti.
- Una volta trovato un x che fornisce un certo resto R divisibile per un divisore scelto d, provare con numeri del tipo $x \pm kd$ con k intero.
- Volendo implementare un calcolo parallelo, è possibile scegliere (da elevare al quadrato) il numero ax+b (invece del semplice x), con a, b interi.

Il vantaggio del metodo di Pomerance, rispetto a quello di Fermat, sta nello scomporre, invece di un intero "grande", tanti numeri interi più "piccoli" e nel poter condurre i calcoli in parallelo in modo che ognuno può dare contributi significativi (e non solo il "fortunato" che trova la y richiesta da Fermat).
Presentiamo ora un esempio di applicazione di un metodo di tipo Montecarlo (algoritmo di Pollard). Sia N=221, $f(X)=X^2-2$, X=194. Applico f ad X, divido per N e considero il resto R, pongo X=R e ripeto, finché non ottengo un numero già visto. Nell'esempio in esame è facile verificare che ottengo ciclicamente i numeri 194, 64, 116, 194, … Ora, considerando due numeri qualsiasi della sequenza provo a calcolare, ad esempio, MCD(221, 194-64)=13, da cui trovo: 221=13*17. Nota: avendo trovato p=13, q=17, ho che p-1=12 ed anche q-1=16. I divisori primi (molto piccoli) di p-1 e q-1 implicano che la sequenza periodica sia molto corta.
Qualsiasi metodo Montecarlo può essere implementato in parallelo, utilizzando accorgimenti simili a quelli descritti a proposito del metodo di Pomerance. Si può dimostrare, però, che il tempo chiesto per ottenere il successo non varia in maniera significativa variando il numero iniziale o la funzione.
Altre applicazioni: Algoritmi di compressione e controllo, calcolo di particolari costanti matematiche, test deterministici di primalità, simulazione e verifica di modelli (fisici, biologici, economici, …)
Facciamo ora una veloce carrellata su altre possibili interazioni.
Un algoritmo di compressione/decompressione e controllo è un procedimento che permette di codificare un'informazione in un numero minore di byte per una successiva trasmissione più efficiente. Ai byte codificati vanno poi aggiunti dei byte di controllo. Il ricevente deve solo decomprimere i byte ricevuti, verificare la congruenza dei dati con il contenuto dei byte di controllo (ed eliminare questi ultimi in caso di congruenza). Le idee base di questi algoritmi sono state implementate da **Huffmann** (che ha anche fornito un limite massimo alla comprimibilità dei dati) e tuttora utilizzate in programmi tipo WinZip, Rar,… Il controllo è effettuato con il metodo CheckSum (versione elaborata del Parity Check, visto in precedenza).
Per il calcolo di particolari costanti matematiche, vengono utilizzate apposite librerie di calcolo in precisione multipla (scritte, di solito, in C++) che permettono di manipolare numeri di milioni di cifre (e più), utilizzando algoritmi elementari. Per esempio:
$e=2+1/2+1/3!+\ldots+1/n!+\ldots$; $\pi=16\text{atan}(1/5)-4\text{atan}(1/239)$, dove $\text{atan}(x)=x-x^3/3+\ldots+(-1)^n x^{2n+1}/(2n+1)$; $\ln(2)=2\{1/3+1/(3*3^3)+\ldots+1/[(2n+1)*3^{2n+1}]+\ldots\}$ utilizzando, ad esempio, gli sviluppi di Mc-Laurin $\ln[(1+x)/(1-x)]=2[x+x^3/3+\ldots+x^{2n+1}/(2n+1)+\ldots]$ con x=1/3).
Per le costanti in esame, esistono procedimenti che convergono molto più rapidamente, anche di tipo ricorsivo (formulati da Ramanujan) e con uso di costanti irrazionali algebriche (es: $\sqrt{2}$), che possono essere comunque calcolate rapidamente (v. algoritmo di Newton-Raphson per il calcolo approssimato di radici reali di un polinomio). Il risultato finale (e, π, ln(2), …) viene generalmente calcolato con almeno due procedimenti indipendenti e i risultati vengono poi confrontati.
Esiste anche la possibilità di utilizzare funzioni razionali fratte (es: tramite le frazioni continue), invece che razionali intere (come i polinomi di Mc-Laurin). Per esempio:
$(e^x+1)/(e^x-1)=[2/x,6/x,\ldots,(4n+2)/x,\ldots]$ dove, per convenzione tipografica, un'espressione come [a,b,c,d] va letta nel modo seguente: [a,b,c,d]=a+1/(b+1/(c+1/d)), se interpretata come frazione continua aritmetica semplice (ad ogni "piano" i numeratori valgono 1). Per x=1 trovo un valore (approssimato) di (e+1)/(e-1). Sottraendo 1 e considerando il reciproco, trovo: $(e-1)/2=[0,1,6,10,\ldots,(4n+2),\ldots]$.
Esistono espressioni analoghe anche per altre costanti, costruite usando frazioni continue generalizzate del tipo $[a,b,c,d,e,f,g,\ldots]=a+b/(c+d/(e+f/(g+\ldots)))$.

Infine, restando nell'ambito dell'aritmetica, parliamo dei test di primalità di tipo deterministico. Il primo (e più generale) di questi risale a Eduard Lucas (fine 1800), ma è possibile implementarne di particolari, per numeri di una determinata forma. Vediamo un esempio.
Supponiamo di voler verificare se un numero della forma $M=2^n-1$ è primo. Per tali numeri M (detti di Mersenne) esiste un test molto veloce per decidere se M è primo (lo è per n=2, 3,…, 25964951). L'ultimo primo (certificato) ha più di 7.800.000 cifre! Il certificato si può ottenere costruendo un sottoinsieme di N il cui primo numero è $X_0=4$, mentre ciascuno dei successivi è ottenuto per ricorrenza ($X_{n+1}=X_n^2-2$). L'insieme contiene, pertanto, 4, 14, 194, 37634,… Per verificare, ad esempio, M con n=3, ho: M=7, n-1=2. Prendo il secondo elemento dell'insieme e verifico che 14/7 non dà resto (e concludo che 7 è primo). Con n=5 avrei M=31, n-1=4 e 37634/31 non dà resto (31 è primo). Con n=4, M=15, n-1=3, 194/15 dà resto (15 è composto. Bella forza! Lo stesso n=4 è composto!!). Nota: nei calcoli pratici, la divisione per M viene fatta ad ogni ricorrenza e viene considerato solo il resto (**$X_{n+1}=X_n^2-2$ modulo M**).
Uscendo un poco dalle applicazioni evidenti di matematica pura è possibile ricondurre a problemi di questo tipo modelli costruiti con riferimento ad altri ambiti. Per esempio, in ambito economico, tra i metodi di ottimizzazione (lineare) di funzioni (lineari sia nella loro espressione che nei vincoli) figura il metodo del simplesso (oggi molto migliorato, ma non molti anni fa utilizzato per il collaudo di nuove macchine elaboratrici da mettere in commercio).
Non vanno poi dimenticate applicazioni di più ampio respiro. Per esempio il sequenziamento del DNA è stato ottenuto con modelli matematici molto sofisticati, dopo anni di calcoli, con utilizzo di tante risorse in parallelo. Restano poi diversi problemi aperti, tra i quali, non dimentichiamo, il problema tutt'altro che banale delle previsioni a lungo termine del tempo (atmosferico) e di simulazioni (con metodi Montecarlo) di tanti altri modelli (es. frattali), ma questa è tutta un'altra storia…

Bibliografia
Languasco-Zaccagnini: Introduzione alla crittografia (ed. Hoepli)
Conway-Guy: Il libro dei numeri (ed. Hoepli)
Weil: Teoria dei numeri (Einaudi paperbacks)
Davenport: Aritmetica superiore (ed. Zanichelli)
Devlin: Dove va la Matematica (ed. Boringhieri)
Courant-Robbins: Che cos'è la matematica (ed. Boringhieri)
Singh: Codici e segreti (ed. Rizzoli), L'ultimo teorema di Fermat (ed. Rizzoli)
Delahaye: Stupefacenti numeri primi (ed. Ghisetti & C.), L'affascinante numero π (ed. Ghisetti & C.)

Sul Web
http://gallica.bnf.fr (contiene molti testi in formato elettronico, gratuitamente scaricabili).
http://www.wolfram.com (contiene una intera enciclopedia di matematica, in lingua inglese e aggiornata in continuazione. È anche scaricabile e facilmente navigabile).
http://www.utm.edu oppure http://www.mersenne.org (contengono diversi link e molti spunti interessanti da utilizzare su qualche motore di ricerca).
http://www.matematicamente.it (uno dei siti più ricchi in lingua italiana).

per contattarmi scrivere a:
marco.frigerio@istruzione.it oppure a frigerio.marco@gmail.com