Numeri e Crittografia

(Carlo Toffalori)

1. Crittografia

Qualche citazione dai libri gialli

S. Holmes (A. Conan Doyle, "La valle della paura"):

"Vi sono molti cifrari che saprei leggere con la stessa facilità con cui leggo gli annunci dei giornali: sono astuzie infantili che divertono l'intelletto senza affaticarlo."

Nel caso specifico

534 c2 12 127 36 31 4 17 21 41 ...

"Si tratta evidentemente di un riferimento alle parole contenute nella pagina di qualche libro. Ma finché non saprò di quale pagina e di quale libro si tratti sarò nell'impossibilità di agire."

The thinking machine (J.Futrelle, "Il cifrario fatale")

"Ci sono migliaia di tipi differenti di cifrari. Uno di questi è illustrato in modo eccellente da Poe nella storia "Lo scarabeo d'oro". In quel cifrario c'è un simbolo a rappresentare ogni lettera dell'alfabeto. Ci sono poi cifrari basati su libri, e questi sono forse i cifrari più sicuri perchè nessuno può decifrarli senza un indizio sul libro da cui le parole sono prese".

Una varietà ricchissima di altri esempi

- Ancora Sherlock Holmes
- Nero Wolfe
- Perry Mason
- S.S. Van Dine
- Agatha Christie

Lo schema generale

Due personaggi: **A** e **B**
Il "cattivo" (non necessariamente cattivo): **C**

L'obiettivo di **A** e **B**: trasmettersi informazioni senza che **C** le capisca;
le operazioni sono allora due
- cifrare
- decifrare

con una chiave opportuna.
In questo consiste la **crittografia.**

L'obiettivo di **C**: violare il sistema usato da **A** e **B**; questa è la **crittoanalisi**.

Procedimenti usuali di codifica e decodifica

Si può sostituire ogni lettera con un simbolo (un numero, una lettera dello stesso alfabeto o di un altro alfabeto) oppure permutare le lettere dell'alfabeto.

Le lettere diventano numeri

A	0	N	13
B	1	O	14
C	2	P	15
D	3	Q	16
E	4	R	17
F	5	S	18
G	6	T	19
H	7	U	20
I	8	V	21
J	9	W	22
K	10	X	23
L	11	Y	24
M	12	Z	25

Attenzione: si lavora modulo 26 (mod 26); 26 torna ad essere 0, 27 è 1, e via dicendo.

Esempio (Giulio Cesare)

Codifica $x \to x+3 \bmod 26$

$0 \to 3$, $1 \to 4$, ..., $25 \to 2$

$A \to D$, $B \to E$, ..., $Z \to C$

Decodifica $x \to x-3 \bmod 26$

Chiave ± 3

Essenziale: chiave di codifica / decodifica

- in genere la stessa;
- comunque tra loro direttamente collegate e computazionalmente equivalenti.

Procedimento usuale di crittoanalisi

Analisi di frequenza: si basa sul confronto tra le lettere più comuni nell'alfabeto e le lettere più frequenti nel messaggio.

In tutto questo la matematica ha un ruolo cruciale.

Ci sono cifrari sicuri ?

Esempio: Cifrario di Vernam, One-Time-Pad (taccuino monouso), 1917
L'informazione e la chiave sono sequenze di 0 ed 1

messaggio in chiaro	100010101001110
chiave	110111110110010
messaggio cifrato	010101011111100

Sia la codifica che la decodifica avvengono tramite la somma modulo 2 per la chiave.

Difetti: - generazione della chiave (sequenza casuale di 0 ed 1)
- trasmissione della chiave
- lunghezza della chiave (la stessa del messaggio)

Allora il "taccuino monouso" è buono una sola volta ed è molto costoso da produrre.

Teoria di Shannon

Shannon è uno dei padri della moderna informazione.
Crittosistema

- M insieme di messaggi
- K insieme delle chiavi
- $\forall k \in K,\ e_k, d_k : M \to M$ funzioni di codifica / decodifica, l'una inversa dell'altra

Alcune probabilità

- $\Pr_{m \in M}(m = m_0)$ per $m_0 \in M$
- $\Pr_{k \in K}(c_0 = e_k(m_0))$ per $m_0, c_0 \in M$ fissati

Definizione
Un crittosistema è **perfetto** se $\forall c_0 \in M \quad \Pr_{k \in K}(c_0 = e_k(m_0))$ resta costante al variare di $m_0 \in M$.
Equivalentemente: $\forall c_0, m_0 \in M \quad \Pr_{m \in M}(m = m_0) = \Pr_{m \in M, k \in K}(m = m_0 / c_0 = e_k(m_0))$

Teorema
In un crittosistema perfetto ci sono almeno tante chiavi quanti messaggi.
In altre parole: il Cifrario di Vernam è "l'unico" perfetto.

Crittografia a chiave pubblica

Crittografia classica

1. una chiave di codifica/decodifica preventivamente concordata e scambiata
2. la codifica è computazionalmente equivalente alla decodifica

Crittografia moderna

- Esigenze: trasmissione in rete, voto telematico, ...
- Caratteristiche: un gran numero di utenti A, B, C, ...

Si aprono allora nuove prospettive

1. non una sola chiave per codifica e decodifica ma, per ogni utente A,
    - una chiave pubblica di codifica (disponibile a chiunque voglia scrivere ad A),
    - una chiave privata di decodifica (conosciuta solo da A),
2. decodificare deve essere enormemente più difficile che codificare (salvo che ad A).

Un esempio pratico: il doppio lucchetto (Diffie-Hellman 1976)

- B invia un messaggio ad A in una scatola chiusa con la sua chiave (il primo lucchetto),
- A chiude ulteriormente con la sua chiave (il secondo lucchetto) e rispedisce a B,
- B apre il suo lucchetto e rispedisce ad A,
- A apre il suo lucchetto e la scatola.

In teoria: funzioni a senso unico

- facili da calcolare,
- invertibili,
- con inversa difficile da calcolare.

L'idea:

- f codifica,
- f^{-1} decodifica.

Osservazione

La nozione di funzione a senso unico è

- non rigorosa (che significa: facile/difficile da calcolare?)
- da controllare periodicamente (quello che è difficile oggi può diventare facile domani)

Problema: Dove cercare funzioni a senso unico?

2. Calcoli

Il ruolo del <u>calcolatore</u> nella vita di oggi

Illustri precursori: Leibnitz, Dissertatio de arte combinatoria, 1666

"Calculemus!" (Calcoliamo!)

"Io chiamo calcolo qualunque notazione che rappresenti il ragionamento, quand'anche non avesse alcun rapporto con i numeri"

$\Rightarrow$ modelli matematici anche per situazioni non matematiche per passare

"dai ragionamenti complicati ai calcoli semplici, dai vocaboli di significato vago ed incerto a caratteri determinati"

$\Rightarrow$ - "lingua characteristica": un linguaggio scientifico universale
- "calculus ratiocinator": un calcolo della ragione

(si anticipano Intelligenza artificiale e Deduzione automatica).

Dubbio: "Tutto" si può calcolare?

Motivazioni: difficoltà a risolvere i seguenti problemi

1. Il 10° problema di Hilbert: determinare un procedimento per stabilire, per ogni polinomio $P(\vec{x})$ a coefficienti interi (di qualunque grado, in qualunque numero di variabili), se $P(\vec{x})$ ha o no radici intere

   Esempi $2x_1 - 3,\ 2x_1 - 4,\ x_1^2 - 2,\ x_1^2 - 4,\ x_1^2 + 1,\ x_1^2 - 1,\ x_1^2 + x_2^2 - 2, \ldots$
   (alcuni con radici intere, altri -apparentemente simili- senza)

2. L'Entscheidungsproblem (il problema di decisione): determinare un procedimento per stabilire quali enunciati della logica del 1° ordine sono validi e quali no.

Una soluzione positiva: basta produrre un algoritmo che funziona
Una soluzione negativa: bisogna escludere ogni possibile procedimento

$\Rightarrow$ Da chiarire preliminarmente: che cosa è un algoritmo?
Meglio: quali sono i problemi che hanno un algoritmo di soluzione (calcolo)?

Turing, Church, Kleen, Gödel, ... (1936)

- Il concetto chiave: la Macchina di Turing
  - Informalmente: una vecchia macchina da scrivere
  - Formalmente: un programma con istruzioni del tipo

    simbolo in esame, stato di esame (corsivo, neretto, ...)

    $\Downarrow$

    simbolo da scrivere, stato in cui entrare, spostamento (destra, sinistra).

- La Tesi di Turing: "Un problema si può calcolare se e solo se c'è una macchina di Turing che lo calcola"

Argomenti a favore
- L'assenza di controesempi
- L'equivalenza con altri approcci alla computabilità (il λ - calcolo di Church, la ricorsività di Godel-Kleene, ...)
- Il riferimento al modo di procedere della mente umana (il modello dell'"Impiegato diligente")

In realtà il modello di Turing è
- discreto
- deterministico

Oggi nuovi orizzonti, talora alternativi
- Nanocomputazione (calcolo in ambienti microscopici)
- Computazione naturale (il sistema nervoso)
- Computazione quantistica

Comunque sulla base della tesi di Turing
- Soluzione negativa del 10° problema di Hilbert: nessun algoritmo possibile (Matijasevic, 1970)
- Soluzione negativa dell' Entscheidungsproblem (Turing, 1936)
- Impossibilità di decidere gli enunciati del 1° ordine veri o falsi in $(\mathbb{N}, +, \bullet)$ (Tarski: si collega al teorema di incompletezza di Godel)

In compenso
- Possibile decidere gli enunciati del 1° ordine veri o falsi in $(\mathbb{R}, +, \bullet, \leq)$ e $(\mathbb{C}, +, \bullet)$ (Tarski, anni '40)

Ma un risultato devastante:

Teorema (Fisher Rabin, 1970)
Qualunque algoritmo di decisione per $(\mathbb{R}, +)$ e $(\mathbb{C}, +)$ impiega talora tempo almeno esponenziale nella lunghezza dell'enunciato da esaminare.

Nota Un tempo esponenziale finisce con l'essere proibitivamente lungo: $n \to 2^n$ cresce "troppo" rapidamente.

Ricordare L'aneddoto della scacchiera:

- 64 quadri
- 1 chicco di grano sul primo quadro
- ad ogni nuovo quadro si raddoppia il numero di chicchi

$\Rightarrow 1+2+2^2+...+2^{63}=2^{64}-1$ chicchi di grano in totale (una quantità irraggiungibile).

$\Rightarrow$ Una nuova prospettiva: <u>non più</u> problemi risolubili <u>ma</u> problemi risolubili a costo accessibile

a. Quale parametro per misurare il "costo"?
   - Tempo
   - Memoria
   - Energia $\Rightarrow$ teoria della complessità computazionale
   - Casualità
   - Interattività
   - ...

   Il più ragionevole (ma non l'unico): il <u>tempo</u>

b. Che significa "tempo rapido"?

Tesi di Edmonds-Cook-Karp (Von Neumann, Rabin, Cobham)
- A livello di slogan: rapido = polinomiale.
- In termini più rigorosi: un algoritmo opera rapidamente se e solo se la funzione che associa ad ogni intero positivo la massima lunghezza delle computazioni dell'algoritmo su input di lunghezza $\leq n$ è asintoticamente maggiorata da qualche funzione polinomiale.

Facile da accettare: esponenziale $\Rightarrow$ lento
Discutibile: (al più) polinomiale $\Rightarrow$ rapido
Consideriamo le funzioni
- $n \to n^{10^{50}}$ (polinomio di grado 10^{50})
- $n \to 10^{50} n$ (polinomio di grado 1)

Quanto accessibile è un algoritmo che richiede questi tempi di lavoro?
$\Rightarrow$ Tesi di Edmonds-Cook-Karp: da accettare per pigrizia più che per convinzione, non c'è in teoria una proposta più convincente.

Definizione P = classe dei problemi che si risolvono in tempo al più polinomiale.

<u>Una classe non così ampia come si vorrebbe</u>. Alcuni esempi in P

0. Calcolo del massimo comun divisore (l'algoritmo delle divisioni successive di Euclide lavora in tempo al più quadratico; serve anche a calcolare l'inverso modulo N di un intero primo con N - e dunque invertibile modulo N -).

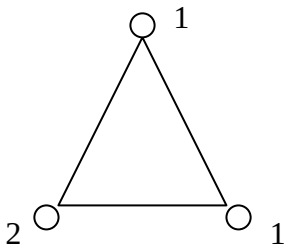
1. 2-COL
   Ricordare: un grafo è una coppia $G=(V,E)$ dove
   - $V \neq \varnothing$
   - E è una relazione binaria antiriflessiva simmetrica su V (<u>adiacenza</u>)

   Definizione: un grafo è 2-colorabile se esiste $c: V \to \{1,2\}$ tale che, per $v, v' \in V$ e $(v, v') \in E, c(v) \neq c(v')$
   (c si dice allora *2-colorazione* di G, 1 e 2 indicano i due possibili colori).

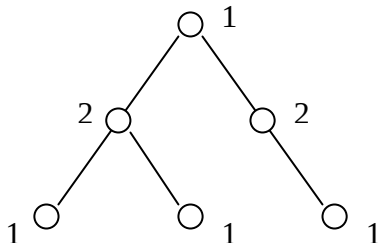
2-COL: Determinare un algoritmo per decidere, per ogni grafo finito $G=(V,E)$, se G è o no 2-colorabile.

Un algoritmo:

- Si sceglie un vertice $v \in V$ e gli si dà colore arbitrario, ad esempio 1 (una 2-colorazione di *G* si preserva per permutazioni dei colori 1 e 2).
- I vertici adiacenti a v hanno colore 2.
- Quelli adiacenti a questi ultimi tornano ad avere colore 1, se possibile.



Nessuna 2-colorazione possibile, l'algoritmo termina dando esito negativo.



Questa 2-colorazione va bene, l'algoritmo termina dando una possibile 2-colorazione del grafo.

$\Rightarrow$ effetto domino, tempo al più polinomiale.

2. PRIMI
   Determinare un algoritmo per decidere, per ogni intero $N \geq 2$, se N è primo o no

   Un algoritmo: AKS, Agrawal-Kayal-Saxena, 2002; tempo di grado 6 nelle più recenti e migliori implementazioni.

<u>Intorno a P: un limbo di problemi né manifestamente in P né manifestamente fuori</u>

1. 3-COL
   Variante di 2-COL con 3 colori 1, 2, 3 $\Rightarrow$ 3-colorabilità
   - Non più valido l'effetto domino.
   - Possibili 3-colorazioni di n vertici 3^n, permutazioni su 3 colori 3!=6; in totale un numero esponenziali di casi $\frac{3^n}{3!}$.
   - In compenso, se una soluzione c esiste,
     è rapido presentarla (basta dire $c(v)$ per ogni $v \in V$),
     è rapido controllare che c funziona.

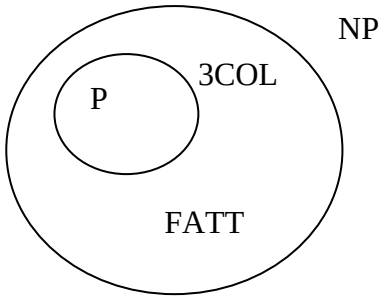
2. FATTORIZZAZIONE
   Determinare un algoritmo per calcolare, per ogni intero $N \geq 2$, la decomposizione in fattori primi di N.

   In realtà basta determinare un algoritmo per calcolare, per ogni intero dispari $N \geq 2$, un intero d, 1<d<N, d divisore di N, e poi applicarlo finché necessario ad N e ai suoi divisori.
   - Problema più complicato di PRIMI;
   - N ha lunghezza $\lfloor \log_{10} N \rfloor + 1$ in base 10 $\Rightarrow$ esplorare gli N-2 interi d , 1<d<N, per verificare se uno di loro divide N è esponenzialmente lungo;
   - in compenso, se un divisore d esiste
     è rapido introdurlo (perchè d<N),
     è rapido controllare che d divide N (una divisione).

Definizione NP = classe dei problemi la cui soluzione è rapida da verificare: per risposte positive,
- c'è una informazione chiave (un testimone) t rapido da presentare
- con l'aiuto di t è rapido controllare la risposta.

Ovvio $P \subseteq NP$
(un problema rapido da risolvere è anche rapido da controllare)

Problema P=NP
(uno dei 7 problemi del Millennio della Matematica, 2000)
Sostanzialmente: un problema che ha un algoritmo rapido di verifica delle soluzioni ha anche un algoritmo rapido di soluzione?



Esempi: 3COL, FATTORIZZAZIONE $\in$ NP

Chiaro: se 3COL $\notin$ P, allora $P \neq NP$

(Apparente) errore di logica: se 3COL $\in$ P, allora P=NP

In realtà: vero per un teorema di Cook-Karp

Teorema 3COL è NP-completo, cioè
- 3COL $\in$ NP;
- $\forall S \in NP$ c'è una procedura deterministica che traduce in tempo al più polinomiale istanze di S in istanze di 3COL e istanze positive di S in istanze positive di 3COL
  (dunque un algoritmo rapido di soluzione di 3COL, combinato con questa procedura, genera un algoritmo rapido di soluzione di S).

Invece: non è chiaro se FATTORIZZAZIONE è NP-completo (per $P \neq NP$), si tende a credere

- FATTORIZZAZIONE $\in NP - P$
  $\Rightarrow$ NP-intermedio
- FATTORIZZAZIONE non NP-completo

La situazione attesa per FATTORIZZAZIONE (e anche per 3COL): per N composto,

- facile moltiplicare i fattori primi di N per ottenere N (tempo quadratico)
- difficile recuperare da N i fattori primi (non si conoscono algoritmi che impiegano tempo al più polinomiale)
- più facile se disponiamo di informazioni chiave come d (allora la verifica si fa in tempo polinomiale di grado 3)

$\Rightarrow$ ispirazione per funzioni a senso unico.

3. Primi e Composti

I numeri naturali 0, 1, 2, 3, 4, 5, ... ($\Rightarrow$ l'insieme $\mathbb{N}$)

- apparentemente banali
- nascondono i più grossi misteri della matematica: 2 citazioni (libere)
  L. Kronecker: "sono i soli creati da Dio",
  A.Weil: " dimostrano l'esistenza di Dio e anche quella del diavolo ".

Un esempio: la rappresentazione dei naturali.
Le cifre: in passato, assai diverse da quelle attuali

- Egiziana - geroglifica
  - ieratica
- Babilonese
- Greca - attica
  - ionica (alfabetica)
- Romana
- Cinese (a bastoncini)

∗ l'uso delle cifre attuali arriva dall'India a noi tramite gli Arabi sono nel tardo Medio Evo;

∗ una considerazione non banale: il ruolo dello zero come simbolo di posizione, per distinguere ad esempio 32 da 302, 3002, e via dicendo (si veda il racconto di R.Stout, The zero clue con Nero Wolfe protagonista).

$\Rightarrow$ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

La rappresentazione in base 10

$$1723 = 1\cdot 10^3 + 7\cdot 10^2 + 2\cdot 10^1 + 3\cdot 10^0$$

Notare:

- si fa riferimento al numero 10 delle dita delle 2 mani (per contare)
- un numero N ha lunghezza in base 10 $\lfloor \log_{10} N \rfloor + 1$
- non l'unica possibile

La rappresentazione in base 2

Due sole cifre 0,1 $\Rightarrow$ 0, 1, 10, 11, 100, 101, 110, 111, 1000, ...
Ad esempio $11001 = 1\cdot 2^4 + 1\cdot 2^3 + 0\cdot 2^2 + 0\cdot 2^1 + 1\cdot 2^0$ (= 25 in base 10)

Notare:

- la lunghezza di N in base 2 è $\lfloor \log_2 N \rfloor + 1$
  - maggiore che in base 10
  - sostanzialmente proporzionale tramite $\log_{10} 2$

La rappresentazione in base 8

Otto cifre 0, 1, 2, 3, 4, 5, 6, 7, dunque i numeri diventano 0, 1, ..., 7, 10, 11, 12, ...

Allora in base otto $31 = 3\cdot 8^1 + 1\cdot 8^0$ (= 25 in base 10).

Così Halloween (31 ottobre) diventa Natale (25 dicembre) : I. Asimov, "Il buon padre di famiglia", Casebook of the Black Widowers

Al mondo ci sono 10 categorie di persone: quelli che capiscono la numerazione in base 2 e gli altri.

I misteri più grandi sui naturali riguardano la divisione.

Definizione $N \in \mathbb{N}$, $N \geq 2$ è
- primo se gli unici divisori di N sono 1 e N
- composto altrimenti

Numeri primi: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, ...
2 è l'unico pari " the oddest prime " (Zassenhaus)
Notare: 3,5 o 5,7 primi gemelli (la loro differenza è 2), 7, 11 oppure 13, 17 no.

Teorema fondamentale dell'aritmetica: Ogni naturale $N \geq 2$ si decompone in uno ed un solo modo (a meno dell'ordine dei fattori) come prodotto di numeri primi.

Ad esempio $15 = 3\cdot 5$, $18 = 2\cdot 3^2$, $24 = 2^2\cdot 3$, $32 = 2^5$, ...

$\Rightarrow$ *Due problemi*: determinare algoritmi che, per ogni $N \in \mathbb{N}$, $N \geq 2$,
(PRIMI) decidano se N è primo o composto
(FATTORIZZAZIONE) decompongano N nei suoi fattori primi ($\Rightarrow$ per N composto, trovino d divisore di N, 1<d<N)

Un algoritmo elementare (noto già agli antichi Greci):
si considerano tutti i d con 1<d<N e si divide N per d,
- se la divisione è esatta per qualche d, si ha che N è composto (e si trova un suo divisore $d \neq 1, N$),
- se la divisione non è esatta per nessun d, N è primo.

Possibili scorciatoie:
- basta esplorare $1 < d \leq \sqrt{N}$ (se $N = d\cdot d'$ con $d, d' > 1$, allora $d \leq \sqrt{N}$ o $d' \leq \sqrt{N}$),
- se d non funziona, neppure 2d, 3d, ... funzionano.

Gauss, Disquisitiones Aritmeticae, 1801 (articolo 329)
" Il problema di separare i primi dai composti e di decomporre i secondi nei loro fattori primi è conosciuto essere uno dei più importanti ed utili in Matematica. La dignità stessa della scienza sembra richiedere di esplorare ogni possibile mezzo per la soluzione di un problema così elegante e famoso "

Perchè questa esigenza due millenni dopo i Greci?

Gauss: " Le tecniche conosciute in precedenza richiederebbero una fatica intollerabile anche per i più instancabili calcolatori "

Algoritmo elementare: richiede fino a $\sqrt{N} - 1$ divisioni per controllare N, cioè un numero esponenziale rispetto alla lunghezza di N.

Possibile separare i due problemi PRIMI e FATTORIZZAZIONE

Teorema di Wilson per $N \in \mathbb{N}$, $N \geq 2$,

N è primo $\Leftrightarrow (N-1)! \equiv -1 \bmod N$.

- controlla PRIMI, non FATTORIZZAZIONE
- richiede il calcolo di $(N-1)!$, dunque N-2 moltiplicazioni (e un tempo esponenziale rispetto alla lunghezza di N)

4 misteri sui primi (per maggiori dettagli, P.Ribenboim, The new book of prime number records, Springer 1996)

1. Numeri di Fermat

   Consideriamo le potenze di 2 1, 2, 4, 8, 16, ..., 2^m, ...
   e i loro successori 2, 3, 5, 9, 17, ..., 2^m+1, ...

   Trascuriamo 2, dunque supponiamo m>0. *Osserviamo*: se m ha un fattore dispari d >1 $d < m$, $m = d \cdot d'$ con $d, d' > 1$, cioè se m non è potenza di 2, allora 2^m+1 è composto

   $$2^m + 1 = \left(2^{d'}\right)^d + 1^d = \left(2^{d'}+1\right) \cdot \ldots$$

   Esempi: $2^3+1=9$ è composto, $2^5+1=33$ è composto, e così via.

   Invece $2^{2^0}+1=3$
   $2^{2^1}+1=5$
   $2^{2^2}+1=17$ tutti primi
   $2^{2^3}+1=257$
   $2^{2^4}+1=65537$

   Notazione: $F(n)=2^{2^n}+1$ n-simo numero di Fermat

   Congettura (Fermat): $F(n)$ è primo per ogni n.

   Eulero: $F(5)=2^{2^5}+1=641 \cdot 6700417$ è composto.

   - caratterizzazione dei possibili divisori primi di $F(n)$ $k \cdot 2^{n+1}+1$ (migliorata da Lucas $k' \cdot 2^{n+2}+1$)
   - applicazione del criterio a $F(5)$: $641 = 10 \cdot 2^6 + 1$

   Oggi
   - i primi di Fermat conosciuti sono ancora $F(n)$ per $n \leq 4$,
   - $5 \leq n \leq 30 \Rightarrow F(n)$ è composto,
     $5 \leq n \leq 11$: è nota anche la decomposizione in fattori primi di $F(n)$,
     n = 14, 20, 22, 24 : non si conosce alcun fattore primo di $F(n)$,
     per gli altri valori di $n \leq 30$: noto qualche fattore primo, non la decomposizione.

   Notare:
   - $F(10)=2^{2^{10}}+1$ si compone di più di 300 cifre in base 10

- $F(n) = F(0) \cdot F(1) \cdot ... \cdot F(n-1) + 2$

Possibile: infiniti primi di Fermat?

Teorema di Gauss: Sia $N \geq 3$. Allora è possibile costruire un poligono regolare di N lati con riga e compasso se e solo se

$N = 2^h \cdot p_0 \cdot p_1 \cdot ... \cdot p_r$ con i p_i primi di Fermat due a due distinti.

$\Rightarrow$ impossibile costruire con riga e compasso poligoni regolari di 7, 9, 11, 13, 14, ... lati

2. <u>Numeri di Mersenne</u>

Stavolta consideriamo i predecessori delle potenze di 2

0, 1, 3, 7, 15, ..., $2^m - 1$

Trascuriamo 0 e 1, dunque assumiamo m>1.

Osserviamo che, se $m = d \cdot d'$ è composto $(1 < d, d' < m)$, anche $2^m - 1$ lo è:

$$2^m - 1 = \left(2^{d'}\right)^d - 1^d = \left(2^{d'} - 1\right) \cdot ...$$

Esempio $2^4 - 1 = 15 = 3 \cdot 5$ è composto.

Se m è primo, non è detto che $2^m - 1$ sia primo

m = 11 è primo, $2^{11} - 1 = 2047 = 23 \cdot 89$ no.

Definiamo $M(m) = 2^m - 1$ m-simo numero di Mersenne.

Si trovano primi di Mersenne

$M(2) = 2^2 - 1 = 3$, $M(3) = 2^3 - 1 = 7$, $M(5) = 2^5 - 1 = 31$

$\Rightarrow$ esempi sempre maggiori (infiniti esempi?)

* $M(13)$ (Eulero, 1722)
* $M(22)$ (Lucas, 1876)
* $M(216091)$ (Slowinski, 1987)
* $M(20996011)$ (Shafer, 2003)
* $M(24036583)$ (Findley, 2004)
* $M(25964951)$ (Nowack, 2005) (il numero ha 8 milioni di cifre decimali)
* $M(30402457)$ (Curtis Cooper e Steven Boone, 03-01-2006) (il numero ha 9 milioni di cifre decimali).

Gli ultimi quattro numeri sono stati scoperti grazie al software GIMPS (Great Internet Mersenne Primes) creato nel 1996 da George Woltman.

3. <u>Primi gemelli</u>: coppie della forma p, p + 2 (un'infinità ?)

Si alternano a lacune comunque grandi

Teorema: per ogni k > 1, ci sono $k!$ numeri consecutivi nessuno dei quali è primo.

Tra 11 e 20: 4 primi 11, 13, 17, 19 il massimo possibile

Tra 10^7 e $10^7 + 100$: 2 soli primi $10^7 + 19$ e $10^7 + 79$

4. La congettura di Goldback

   (A.Doxiadis, Zio Petros e la congettura di Goldbach, Bompiani)

   C. Goldbach 1742, lettera a Eulero: "Ogni $N \geq 6$ è la somma di al più 3 primi" (non necessariamente distinti)
   L. Eulero: "Basta provare che ogni numero pari $N \geq 4$ è la somma di 2 primi" (perchè ?)

   4 = 2 +2, 6 = 3 +3, 8 = 3 +5, 10 = 5 +5 = 3 + 7, ...

Primi

PRIMI input: $N \in \mathbb{N}$, $N \geq 2$ (senza perdita di generalità, N dispari e $N > 2$)
output: N è primo o composto ?

Qualche premessa

1. Il piccolo teorema di Fermat
   Risultato attribuito a Fermat (~ 1600)
   Teorema: se N è primo, allora ogni intero a primo con N soddisfa $a^{N-1} \equiv 1 \pmod{N}$.
   Cenni sulla dimostrazione:
   - Basta $a \geq 0$.
   - Si procede per induzione su a.
   - Si usano un po' di calcolo combinatorio e argomenti elementari.

   Notare: $(a, N) = 1$ e $a^{N-1} \equiv 1 \pmod{N}$ sono rapidi da controllare rispetto alla lunghezza di N, perchè richiedono
   - divisioni (costo quadratico rispetto alla lunghezza di N),
   - elevamento a potenza modulo N $a \to a^{N-1}$
     Esempio: $a^{100}, N = 101$
     1. possibile evitare 99 moltiplicazioni
     2. si procede così

        $$a^{100} = \left(a^{50}\right)^2 = \left(\left(a^{25}\right)^2\right)^2 = \left(\left(\left(a^{12}\right)^2 \cdot a\right)^2\right)^2 = \ldots$$

        con elevamenti al quadrato e moltiplicazioni per a.

   Dubbio
   1. Fissiamo a primo con N (ad esempio a = 2, visto che N è dispari). Se vale $a^{N-1} \equiv 1 \pmod{N}$ (rapido da verificare), N è primo ? NO
   2. Se $a^{N-1} \equiv 1 \pmod{N}$ per **ogni** a primo con N (lungo da controllare, perchè richiede fino a N-1 controlli), N è primo ? NO

   Definizione: N è pseudoprimo di Carmichael se N è composto ma ogni a primo con N soddisfa $a^{N-1} \equiv 1 \pmod{N}$

   Minimo esempio: 561
   Infiniti esempi: (Alford, Granville, Pomerance, 1994)

   Una generalizzazione di Eulero del piccolo teorema di Fermat: usa la funzione φ di Eulero.

$\varphi(N)$ = numero dei naturali tra 1 e n primi con N, per ogni N > 0.

Esempi:

- $\varphi(N) = N-1$ se N è primo,
- $\varphi(1) = 1$,
- se N è prodotto di due primi $p \neq q$, $\varphi(N) = (p-1)(q-1)$
  ($\varphi(6) = \varphi(2\cdot 3) = 2 = 1\cdot 2, \varphi(15) = \varphi(3\cdot 5) = 8 = 2\cdot 4$, ...)

Teorema di Eulero: Sia $N \in \mathbb{N}$, $N \geq 2$ (N dispari). Allora per ogni a primo con N

$$a^{\varphi(N)} \equiv 1 (\mathrm{mod}\, N)$$

C*onseguenza*: $a^{\varphi(N)+1} \equiv a (\mathrm{mod}\, N)$ (valido anche se N divide a)

2. $x^2 = 1$ ha 2 radici ± 1 in $\mathbb{Z}$ ($\Rightarrow$ in $\mathbb{Q}, \mathbb{R}, ...$)
   Ma se lavoriamo modulo N le cose cambiano.

   Esempio: $N = 8 \Rightarrow \pm 1, \pm 3$ soddisfano $x^2 = 1 \mathrm{mod}\, 8$ (4 radici distinte modulo 8)
   Tuttavia si ha
   Osservazione: Se N è primo, $x^2 = 1 \mathrm{mod}\, N$ ha solo 2 radici $x \equiv \pm 1 \mathrm{mod}\, N$
   (distinte per N dispari)
   Infatti, se N divide $x^2 - 1 = (x-1)(x+1)$ e N è primo, allora N divide uno dei fattori
   $x-1$ o $x+1$

<u>L'algoritmo più efficiente per i primi:</u> *l'algoritmo di Miller-Rabin* (fine anni settanta)

L'idea: sacrificare la precisione per aumentare la velocità
$\Rightarrow$ si ammettono risposte sbagliate, purché la probabilità di errore sia bassa.

E.Borel: "un evento che ha probabilità $< 10^{-50}$ non accadrà mai, e se anche accade non sarà mai rilevato"
$\Rightarrow$ algoritmi probabilistici veloci

- Montecarlo: risposte probabilmente vere in tempi certamente rapidi,
- Las Vegas: risposte certamente vere in tempi probabilmente rapidi.

L'algoritmo: è dato $N > 2$, N dispari. Dunque N – 1 è pari. Invochiamo un testimone a, $1 < a < N$.

* $(a, N) \neq 1$: si dichiara N COMPOSTO (anzi si ha un'informazione su un divisore $\neq 1$, N di N)
* $(a, N) = 1$: si calcola a^{N-1} modulo N
  - $a^{N-1} \not\equiv 1 (\mathrm{mod}\, N)$: si dichiara N COMPOSTO
  - $a^{N-1} \equiv 1 (\mathrm{mod}\, N)$: si calcola $a^{\frac{N-1}{2}}$ modulo N (ricordando che N - 1 è pari)
    - $a^{\frac{N-1}{2}} \not\equiv \pm 1 (\mathrm{mod}\, N)$: si dichiara N COMPOSTO

- $a^{\frac{N-1}{2}} \equiv -1 \pmod N$: si dice N PROBABILMENTE PRIMO
- $a^{\frac{N-1}{2}} \equiv 1 \pmod N$, $\frac{N-1}{2}$ dispari: si dice N PROBABILMENTE PRIMO
- $a^{\frac{N-1}{2}} \equiv 1 \pmod N$, $\frac{N-1}{2}$ pari: si calcola $\frac{N-1}{4}$ modulo N ,

...

La risposta N COMPOSTO è sicura, quella N PROBABILMENTE PRIMO può essere sbagliata.

Ma: la <u>probabilità di errore</u> al variare di a, dopo il primo tentativo, è $\le \frac{1}{4}$ (almeno 3 testimoni su 4 sono onesti).

Tempi di lavoro: a meno di una costante moltiplicativa, n^5 per n lunghezza di N.

Dopo 100 applicazioni con esito concorde N PROBABILMENTE PRIMO

- *probabilità di errore* $\frac{1}{4^{100}} < \frac{1}{10^{50}}$
- *tempo di lavoro* a meno di una costante moltiplicativa (moltiplicata per 100), ancora n^5.

<u>L'algoritmo AKS (Agrawal-Kayal-Saxena, 2002)</u>

Il fondamento.

Teorema: siano $N \in \mathbb{N}$, $N > 2$, un intero a primo con N. Allora

N è primo $\Leftrightarrow$ i polinomi $x^N - a$ e $(x-a)^N$ sono congrui modulo N

Si intende che i coefficienti di ugual grado sono a 2 a 2 congrui modulo N, dunque ci sono N – 1 congruenze da controllare: TROPPE!
Il teorema è

- elegante ma poco pratico,
- la dimostrazione si fonda sul Piccolo Teorema di Fermat.

Idea per l'algoritmo AKS: N primo $\Leftrightarrow x^N - a \equiv (x-a)^N \pmod{N, x^r - 1}$ con r opportuno, logaritmico in N.

Tempi di lavoro: (nell' "implementazione" di Lenstra-Pomerance, 2005) approssimativamente n^6.

Fattorizzazione

<u>FATTORIZZAZIONE</u> input: $N \in \mathbb{N}$, $N > 2$, N <u>composto</u> dispari,
output: $d \neq 1, N$, d divisore di N.

- vari metodi conosciuti (curve ellittiche, ...)
- nessuno polinomiale nella lunghezza di N

Ma attenzione ai procedimenti quantistici (P.Shor, 1994) !

4. Ancora crittografia

RSA

Il criptosistema RSA (Rivest-Shamir-Adleman, 1978)

La funzione a senso unico: la moltiplicazione.
Per $p \neq q$ primi grandi

- facile calcolare $N = p \cdot q$
- lento recuperare p, q dal prodotto N.

Sia A un utente, allora A

- genera $p \neq q$ primi (A dispone di algoritmi rapidi per PRIMI),
- calcola $N = p \cdot q$,
- ricorda $\varphi(N) = (p-1)(q-1)$ e $a^{\varphi(N)} \equiv 1 \bmod N$ per ogni a primo con N, cioè con p, q,
- sceglie d_A, e_A l'uno inverso dell'altro modulo $\varphi(N)$ (con l'algoritmo rapido per il massimo comun divisore): dunque
  $e_A \cdot d_A \equiv 1(\bmod \varphi(N))$, $e_A \cdot d_A = 1 + k\varphi(N)$ per un opportuno k,
- divulga e_A, N come <u>chiave pubblica</u> di codifica (e per encript)
- custodisce d_A, p, q come <u>chiave privata</u> di decodifica (d per decript)
-

Unità di messaggio: $a \in \mathbb{N}, a < p, q \ (\Rightarrow a < N)$

Si noti a = 0 oppure a primo con p, q e dunque con N.

Codifica di un messaggio per A: $a \to a^{e_A} \bmod N$

Decodifica del messaggio: $a' \to a'^{d_A} \bmod N$

Infatti $\left(a^{e_A}\right)^{d_A} = a^{e_A \cdot d_A} = a^{1+k\varphi(N)} \equiv \begin{cases} a & \text{se } a = 0 \\ a^{k\varphi(N)} \cdot a \equiv 1 \cdot a = a & \text{altrimenti} \end{cases} \equiv a \,(\bmod N)$

Sicurezza del criptosistema: il pirata C conosce N, e_A; se intercetta a^{e_A}, deve estrarre una radice e_A-ma per ottenere a.

Via maestra per trovare la radice:

- conoscere $d_A = \text{inverso di } e_A \bmod \varphi(N)$,
- $\Rightarrow$ conoscere $\varphi(N) = (p-1)(q-1)$.
- $\Rightarrow$ conoscere p, q, cioè i fattori primi di N,
- $\Rightarrow$ difficoltà di FATTORIZZAZIONE.

Qualche precauzione

- cambiare N
- evitare d_A, e_A piccoli

(parziali esposizioni della chiave privata permettono di infrangere il criptosistema)

Il criptosistema di Diffie-Hellman (1976)

La funzione a senso unico: il logaritmo discreto

Premessa algebrica: per p primo, $\mathbb{Z}_p$ è un campo e $\mathbb{Z}_p^* = \mathbb{Z}_p - \{0\}$ è un gruppo ciclico moltiplicativo (c'è un intero positivo g tale che $1, g, g^2, ..., g^{p-2}$ formano gli elementi di $\mathbb{Z}_p^*$ modulo p, e $g^{p-1} \equiv 1 (\mathrm{mod}\, p)$)

Si verifica che, per $n \in \mathbb{N}, n < p-1$,

- facile calcolare g^n da n,
- difficile recuperare n da g^n (il logaritmo discreto di g^n rispetto a g).

Un utente A:

- sceglie n_A e calcola g^{n_A},
- divulga g^{n_A} come chiave pubblica,
- custodisce n_A come chiave privata.

Due utenti A e B concordano $g^{n_A \cdot n_B}$ come chiave comune classica di corrispondenza:

- A lo calcola come $\left(g^{n_B}\right)^{n_A}$ con g^{n_B} chiave pubblica di B e n_A chiave privata di A,
- B lo calcola come $\left(g^{n_A}\right)^{n_B}$.

Sicurezza del criptosistema: il pirata C conosce g, g^{n_A}, g^{n_B}, ma deve ottenere $g^{n_A \cdot n_B}$.

Via maestra:

- calcolare n_A, n_B da g^{n_A}, g^{n_B} rispettivamente (cioè due logaritmi discreti)
- ricavare $n_A \cdot n_B$ e poi $g^{n_A \cdot n_B}$.

Ipotesi di Diffie-Hellman: calcolare $g^{n_A \cdot n_B}$ da g^{n_A}, g^{n_B} è computazionalmente equivalente a calcolare un logaritmo discreto (dunque n_A, n_B da g^{n_A}, g^{n_B} rispettivamente).