Cristiano Teodoro
(Ministero delle Comunicazioni - I.S.C.T.I.)

VERIFICA E GENERAZIONE DI NUMERI PRIMI RELATIVAMENTE GRANDI

(VERIFICATION AND PRODUCTION OF QUITE LARGE PRIME NUMBERS)

300000000000000000000000000000000000000000000000000000000000000000000000000000000000001 è primo ?

Sommario: l'articolo illustra la realizzazione di un programma scritto in QBASIC dedicato a verificare se un numero relativamente grande è primo o composto e, questione ancora più interessante, a generare numeri primi. Questo argomento è di grande interesse nella teleinformatica, con applicazioni in particolare nel campo crittografico, dove vengono presi in considerazione numeri costituiti da diverse decine o addirittura da qualche centinaio di cifre. Nell'articolo si pone l'accento sulle maggiori difficoltà e sulle principali problematiche che si incontrano nella realizzazione del programma. In effetti la manipolazione di tali numeri richiede che l'esecuzione delle quattro operazioni aritmetiche su di essi venga realizzata solo tramite opportune istruzioni in modo di poter lavorare in aritmetica a precisione multipla. In particolare, per l'operazione della divisione, che risulta la più complessa da programmare, viene utilizzato un algoritmo originale. Per quel che riguarda l'effettivo metodo scelto, si è preso in considerazione l'algoritmo di Rabin - Miller, che è di tipo probabilistico, ma con il quale si riesce a riconoscere con una probabilità alta a piacere, in tempi accettabili anche con l'uso di un PC non eccessivamente sofisticato, se un numero è primo o composto o a creare numeri primi. Viene anche illustrato in modo elementare l'uso della aritmetica modulare il cui impiego risulta qui indispensabile. Esistono pacchetti matematici, quali MATHEMATICA, MAPLE, MAT-CAD, ecc, che risolvono in modo brillante l'argomento trattato nel presente articolo, ma essi forniscono ovviamente scarse informazioni e delucidazioni sulle soluzioni che sono state impiegate e sulle problematiche che si sono dovute affrontare e superare per ottenere i risultati desiderati. Nell'articolo vengono illustrati alcuni risultati per numeri di varia grandezza in due tabelle, una relativa alla verifica della primalità o meno di un numero, l'altra relativa alla creazione di un numero primo, con l'indicazione dei tempi necessari di calcolo lavorando in QuickBASIC.. Si mettono in evidenza anche i tempi di calcolo estremamente grandi che sarebbero necessari se venissero utilizzati i metodi classici.

Abstract : the aim of this paper is to illustrate a Qbasic program dedicated to verify the primality of quite large numbers and, argument more interesting, to produce primes composed by several tens of digits or some hundreds of digits. To deal with large primes is an essential topic in several scientific and technical fields, specifically in the modern cryptography applied to communication.The paper stresses the bigger difficulties and the main problems in the program implementation, especially in carrying out multiple precision arithmetic for the four arithmetical operations, in particular the division. For this we use an original algorithm. In this paper we consider the probabilistic Rabin - Miller method. By this algorithms we are able to identify the number primality or to produce primes with the probability we like in acceptable time also using an unpretentious PC. We explain also the first principles of modular arithmetic: actually only using this characteristic arithmetic we can to carry out the indispensable computation, otherwise impossible. Mathematical Software packages are available (MATHEMATICA, MAPLE, MAT-CAD ,etc.) which can powerfully solve the matter of this paper,. but they provide poor explanations as regards the feasible solutions. In the paper we illustrate a few outcomes for numbers of different magnitude inside two Tables, one pertinent to primality verify, and the other relevant the primes generation with regard to Qbasic computation time.We also point out the extremely computation high time if we should use the classical methods and algorithms.

Introduzione

Il presente articolo è dedicato ai numeri primi.

Parleremo infatti di numeri primi costituiti anche da diverse decine di cifre (20, 50, 70, 100, 150, 300 e più cifre.....), manipolabili in modo esatto solo con l'aritmetica a precisione multipla.

L'intento è quello di mostrare come si possa costruire un programma, nel caso specifico usando un linguaggio evoluto quale il Qbasic, per **verificare** *in tempi brevi* se numeri anche di valore abbastanza elevato sono **primi** o **composti** e, argomento ancor più interessante, per **creare** numeri **primi** relativamente grandi.

I numeri che tratteremo saranno naturalmente degli interi dispari e potranno essere formati a partire da poche cifre sino a 500 cifre.

Il metodo impiegato è quello noto nella teoria dei numeri come ***test*** *di primalità di Rabin - Miller* [1] , [2] , [3] .

Perché utilizzare questo tipo di test e non usufruire invece di uno dei metodi classici: Crivello di Eratostene, Divisioni successive, ecc. ?

La risposta e' semplice: questione di tempi di calcolo per arrivare al risultato.

Poniamo il caso di prendere in esame numeri composti da due soli fattori abbastanza grandi, ad esempio dello stesso ordine di grandezza (numeri che chiameremo *composti difficili*) o numeri che effettivamente sono *primi*.

Con i metodi classici, come noto di tipo esaustivo, la verifica della primalità o meno di un numero è necessariamente legata all'algoritmo della sua scomposizione in fattori, per cui se il numero è effettivamente *primo od un composto difficile* i tempi di verifica o peggio i tempi per la creazione di primi, già per numeri costituiti da sole 14 cifre risultano più lunghi di quelli ottenibili con l'impiego del suddetto tipo di ***test***.

Figuriamoci per numeri di 50 e più cifre: questi tempi diventerebbero spaventosamente lunghi.

Volete qualche dato ?

Consideriamo in modo ottimistico anche per un PC abbastanza potente di impiegare **solo un decimo di secondo** per la *verifica* della primalità o meno di un numero di 14 cifre (che in effetti sia un primo o un composto difficile) utilizzando il metodo delle divisioni successive (*trial division*) con divisori, oltre a 2 ed a 3, della forma 6h ± 1 con:

$$h = 1,2,3,4,\ldots\ldots\ldots, \left\lfloor \frac{\sqrt{p}-1}{6} \right\rfloor + 1$$

metodo questo che risulta essere uno dei più efficienti fra quelli classici [4] .

Ebbene, estrapolando questo tempo per il calcolo di quelli relativi a numeri ancora più grandi, otterremmo per tali tipi di numeri, i seguenti tempi **t** di calcolo:

t = 100 secondi per un numero di 20 cifre

t > 115 giorni per un numero di 30 cifre

t > 31700 anni per un numero di 40 cifre

t > 31 miliardi di anni per un numero di 52 cifre

Sì, avete letto bene : 31 miliardi di anni !

Per rendervi conto di quanto sia incomparabilmente lungo questo tempo, immaginate di aver terminato alla data attuale la *verifica* o la *creazione* di un numero di 52 cifre; ebbene avreste dovuto iniziare con il vostro computer i calcoli *molto, ma molto prima del Big Bang , cio*è " *molto prima* " della nascita (15 miliardi di anni ?) dell'attuale Universo.

Utilizzando invece l'*algoritmo* ed il *test* impiegati nel programma illustrato i tempi di calcolo per la *verifica* o la *creazione* diventano irrisori a paragone dei precedenti (vedi tabelle).

Ad esempio per la *verifica* riguardante un generico numero di 77 cifre del tipo *composto difficile* il tempo risulta in pratica immediato, non superiore a 6 secondi se il numero invece è effettivamente *primo* (vedi tabella n° 1); per la *creazione* di un numero primo sempre di 77 cifre i tempi di elaborazione risultano e un po' più lunghi ma sempre contenuti in media entro 8 secondi.

Ci si può domandare ora quale importanza d'interesse applicativo possono avere, oltre quello puramente speculativo, *numeri primi* o *composti difficili* così grandi.

Una risposta è da ricercarsi nel campo della crittografia, campo che ha delle applicazioni molto interessanti nella teleinformatica e che si sta sviluppando in modo sempre più notevole, man mano che cresce la necessità di trasmettere a distanza in maniera *protetta* e *sicura* qualsiasi tipo d'informazione.

In effetti alcuni sistemi crittografici, fra i più robusti, si basano su algoritmi in cui risulta indispensabile la possibilità di verificare e soprattutto di creare in tempi *sufficientemente contenuti* nume-

ri **primi casuali** costituiti da diverse decine di cifre (80, 154, 161, 308, 312, 350 cifre e oltre) per formare con essi dei *composti difficili*.

Basti pensare alla importanza che assume la verifica e la creazione di numeri di questo tipo per la costruzione delle *chiavi*, pubblica e segreta, nell'algoritmo RSA, il più noto e diffuso sistema crittografico a chiave pubblica.

Algoritmo impiegato (Rabin -Miller)

Prima però di descrivere il programma realizzato occorre dire qualcosa in merito all'algoritmo impiegato e per illustrarlo occorre fornire qualche nozione tratta dalla teoria dei numeri. Cercheremo di usare una terminologia non troppo specialistica.

Un numero **p** *intero dispari* è detto *pseudoprimo forte* alla base ***b*** (dove ***b*** è un intero) se, posto **p** sotto la forma $\mathbf{p} = 1 + 2^k \cdot \mathbf{q}$ con **q** dispari, una delle seguenti condizioni è soddisfatta:

a) il resto della divisione di b^q per **p** ha valore **1** o valore **p-1**

oppure, non avendo il resto tali valori,

b) proseguendo con il calcolo si ottiene un resto di valore **p-1** nella divisione per **p** di uno dei seguenti termini :

$$b^{2q},\ b^{2^2 q},\ b^{2^3 q},\ b^{2^4 q},\ \ldots\ldots\ldots\ldots\ b^{2^{(k-1)} q}$$

Ebbene, considerato un numero **p** dispari, il summenzionato *test di Rabin -Miller* consiste nel controllare se **p** è uno *pseudoprimo forte* per diverse basi, ciascuna delle quali scelta in modo casuale nell'intervallo numerico da **2** a (**p**-1) ; se **p** non risulta essere uno *pseudoprimo forte*, in riferimento anche ad una sola delle basi scelte, il numero **p** è **certamente composto**; se invece il numero **p** risulta *pseudoprimo forte* per tutte le basi scelte, si dimostra che esso è *probabilmente primo*, con una probabilità che non lo sia

minore di $\frac{1}{4^{nb}}$

dove "nb" è il numero delle basi considerate [5].

Se ad esempio si effettua la verifica della primalità di un numero scegliendo un numero *nb* di basi pari a 25 e si riscontra che esso risulta essere *pseudoprimo forte* per tutte e 25 le basi scelte, lo si può considerare senz'altro primo .

Infatti la probabilità che non lo sia è minore di $\frac{1}{4^{25}}$

minore cioè di 0.00000000000000001, equivalente a dire che esso è primo con una probabilità

maggiore di $(1 - \frac{1}{4^{25}})$

cioè maggiore di 0.999999999999999)

Per rendersi conto di come si possa tranquillamente ritenere primo un numero che abbia superato il test con 25 basi, si può fare la seguente considerazione:

la probabilità che il computer tratti in modo errato qualche bit fra la mole enorme di bit manipolati nei calcoli elementari, a causa di possibili disturbi (rumore) sul suo hardware si dimostra essere molto più alta del valore 0.00000000000000001 suddetto, cioè della probabilità che il numero in esame dichiarato primo sia in realtà composto.

Un esempio banale ma più efficace può essere il seguente: giocando al *superenalotto* è risaputo quanto sia difficile vincere; ciò perché la probabilità di indovinare i 6 numeri è molto piccola

(pari a $\frac{1}{622614630} = 0.0000000016$) ;

ebbene questa così piccola probabilità è in effetti molto più elevata (più di un milione di volte) della probabilità che il numero dichiarato primo sia invece composto.

Ma ritorniamo al nostro test: ci limiteremo qui a illustrare un programma relativo a numeri sufficientemente piccoli da poter essere trattati con la sola aritmetica a doppia precisione disponibile su qualsiasi PC.

Questo allo scopo di mettere in evidenza le operazioni principali che realizzano il test, operazioni che poi si ritroveranno debitamente espresse in aritmetica a precisione multipla nell'effettivo programma dedicato ai numeri relativamente grandi.

Il programma in pratica traduce con istruzioni in Qbasic il test di Rabin-Miller esposto sopra.

Prima però di illustrarlo occorre dedicare un po' d'attenzione a come effettuare il calcolo del *resto* della divisione di b^q per p .

Aritmetica modulare

Se noi provassimo ad effettuare il calcolo in modo tradizionale, cioè come ci hanno insegnato a scuola, calcoleremmo prima l'effettivo valore del dividendo b^q , quindi effettueremmo la divisione di tale valore per il divisore p, trovando quoziente e resto. Questo modo di procedere però, quando b^q è un numero grande, non è più praticabile anche per valori modesti di b e di q.

Si consideri ad esempio il numero

p = 829459 = 1 + 2 · 414729

(quindi con k = 1 e q = 414729)

Posto ***b*** = 3 , vogliamo trovare il *resto* della divisione di b^q per p , vale a dire il resto di $\frac{3^{414729}}{829459}$; ma 3^{414729} è un numero costituito da ben 197877 cifre, cifre che riempirebbero da cima a fondo più di 25 pagine di questa rivista !

L'operazione di divisione con un dividendo così enorme è praticamente impossibile da effettuare anche utilizzando l'aritmetica a precisione multipla.

E allora quale altra via possiamo escogitare per trovare tale resto?

A questo punto ci viene in aiuto l'**aritmetica modulare** ideata dal grande matematico C.F. Gauss.

Senza entrare troppo in merito a questo tipo di aritmetica per la quale si rimanda alla bibliografia [6], [7] ci basterà illustrare la seguente *proprietà* facilmente dimostrabile, riguardante i numeri interi.

Considerati due numeri a e b , il resto r della divisione del prodotto (a·b) per un dato numero p risulta essere il resto della divisione del prodotto (r_a·r_b) per il numero p dove r_a ed r_b sono i resti trovati rispettivamente nelle divisioni di a per p e di b per p.

Naturalmente, se il valore di (r_a·r_b) fosse minore di p il resto r sarebbe il prodotto stesso (r_a·r_b).

Indicando tutto ciò sinteticamente scriveremo:

$$\text{resto di } \frac{a \cdot b}{p} = \text{resto di } \frac{r_a \cdot r_b}{p}$$

$$\text{dove } r_a = \text{resto di } \frac{a}{p} \; ; \quad r_b = \text{resto di } \frac{b}{p}$$

Ora, sfruttando tale proprietà, se b = a si ha:

$$\text{resto di } \frac{b^2}{p} = \text{resto di } \frac{r_b \cdot r_b}{p} = \text{resto di } \frac{r_b^{\,2}}{p};$$

Se poi si volesse trovare il resto di $\frac{b^4}{p}$ chiamato r_2 il resto di $\frac{r_b^{\,2}}{p}$ si avrebbe :

$$r_4 = \text{resto di } \frac{b^4}{p} = \text{resto di } \frac{b^2 \cdot b^2}{p} =$$

$$= \text{resto di } \frac{r_2 \cdot r_2}{p}.$$

Per mettere in evidenza l'importanza dell'applicazione della suddetta proprietà per semplificare i calcoli quando si voglia trovare il resto nella divisione per un dato numero della potenza di un altro numero, nel riquadro viene mostrato in dettaglio il seguente esempio elementare :

trovare il resto di $\frac{17^{19}}{11}$

Indicheremo per semplicità di esposizione con r_n il resto di $\frac{17^n}{11}$

dove n è un intero qualsiasi;

si ricordi inoltre dall'aritmetica che, posto $n = l + m$, la generica potenza 17^n può essere espressa come segue :

$$17^n = 17^{\,l+m} = 17^{\,l} \cdot 17^m$$

Come si può notare i calcoli mostrati nel riquadro sono molto semplici e si possono fare addirittura mentalmente. Se provassimo ad ottenere il resto eseguendo direttamente il calcolo di $\frac{17^{19}}{11}$ dovremmo trovare prima il valore effettivo di 17^{19} , che è un numero di ben 24 cifre, e poi effettuarne la divisione per 11, calcolando così il resto : è evidente la maggiore complessità di calcolo ; anzi si può constatare che il nostro computer, pur utilizzando la doppia precisione per eseguire i suoi calcoli, fornirebbe un risultato *errato* per il valore di r_{19} ,calcolato invece sopra così facilmente.

Dall'esempio illustrato si nota chiaramente la strategia impiegata per il calcolo del resto di $\frac{b^n}{p}$.

Si deve impostare un insieme di *operazioni di divisione* in cui, iniziando da piccole potenze e procedendo in modo opportuno con potenze di *b* crescenti, ad esempio come indicato in [8], in ogni divisione per *p* da effettuarsi, il *dividendo* formato dal prodotto di due potenze di *b* , in virtù della proprietà sopra illustrata, viene sostituito dal prodotto di due resti relativi alle due potenze considerate di *b*; resti a loro volta trovati in divisioni precedentemente eseguite con la stessa tecnica e relativi a potenze più piccole di *b*. E' evidente che qualsiasi *dividendo* implicato nei calcoli non potrà mai assumere un valore superiore al prodotto (p-1)·(p-1)dove p è il divisore comune per tutte le divisioni eseguite. Diciamo per inciso che volendo scrivere ed esprimersi nel linguaggio normalizzato dell'aritmetica modulare, tale resto si scriverebbe:

$$r_n \equiv \frac{b^n}{p} \pmod{p}$$

leggendo: r_n è il residuo modulo *p* di b^n.

Il Programma

E' ora venuto il momento di dare il listato delle istruzioni relative al programma dedicato a riconoscere se un numero **p** dispari, è primo o composto. Questo programma è relativo a numeri di dimensioni tali da poter utilizzare l'Aritmetica doppia precisione offerta dal software del Qbasic senza arrotondamenti.

Il test usato per la verifica è quello di RABIN-MILLER. Il numero " nb " delle basi è stato scelto pari a 25 per **p** > 100.

Il listato sottoripotato non è quello dell'effettivo programma relativo ai numeri primi grandi che si vorrebbe proporre alla vostra attenzione.

Esso tuttavia risulta utile per illustrare le principali parti che costituiscono il test di **Rabin-Miller**; parti che con le debite differenze formano anche il più elaborato programma dedicato a numeri primi molto grandi.

CALCOLO del **resto di** $\frac{17^{19}}{11}$

r_1 = resto di $\frac{17}{11} = 6$

r_2 = resto di $\frac{17^2}{11}$ = resto di $\frac{17 \cdot 17}{11}$ = resto di $\frac{r_1 \cdot r_1}{11}$ = resto di $\frac{6 \cdot 6}{11}$ = resto di $\frac{36}{11} = 3$

r_3 = resto di $\frac{17^3}{11}$ = resto di $\frac{17^2 \cdot 17}{11}$ = resto di $\frac{r_2 \cdot r_1}{11}$ = resto di $\frac{3 \cdot 6}{11}$ = resto di $\frac{18}{11} = 7$

r_4 = resto di $\frac{17^4}{11}$ = resto di $\frac{17^2 \cdot 17^2}{11}$ = resto di $\frac{r_2 \cdot r_2}{11}$ = resto di $\frac{3 \cdot 3}{11}$ = resto di $\frac{9}{11} = 9$

r_8 = resto di $\frac{17^8}{11}$ = resto di $\frac{17^4 \cdot 17^4}{11}$ = resto di $\frac{r_4 \cdot r_4}{11}$ = resto di $\frac{9 \cdot 9}{11}$ = resto di $\frac{81}{11} = 4$

r_{16} = resto di $\frac{17^{16}}{11}$ = resto di $\frac{17^8 \cdot 17^8}{11}$ = resto di $\frac{r_8 \cdot r_8}{11}$ = resto di $\frac{4 \cdot 4}{11}$ = resto di $\frac{16}{11} = 5$

r_{19} = resto di $\frac{17^{19}}{11}$ = resto di $\frac{17^{16} \cdot 17^3}{11}$ = resto di $\frac{r_{16} \cdot r_3}{11}$ = resto di $\frac{5 \cdot 7}{11}$ = resto di $\frac{35}{11} = 2$

Tali parti in effetti risulterebbero più complicate da illustrare dato l'elevato numero di istruzioni necessarie alla loro realizzazione, in quanto viene utilizzata per esse l'aritmetica a precisione multipla. Come si può notare il suddetto programma tratta solo numeri piccoli vale a dire composti da 8 cifre.Che esso sia limitato a verificare solo numeri al massimo di 8 cifre e di questi solo quelli non superiori al valore *lm* = **94906267** lo si può comprendere se si osserva che vi sono istruzioni relative all'esecuzione di prodotti fra numeri che possono essere dello stesso ordine di grandezza del numero p in esame con risultati quindi per il loro prodotto anche di 16 cifre.

D'altra parte la condizione **indispensabile** per operare correttamente nell'esecuzione del test è quella di ottenere sempre risultati **esatti** in tutti i calcoli aritmetici effettuati, condizione questa non sempre ottenibile, anche utilizzando la doppia precisione, per prodotti relativi ad operandi costituiti ciascuno da 8 cifre e maggiori di *lm* (provate a verificare che risultato vi dà il vostro computer moltiplicando per se stesso il numero **94906267** od un qualsiasi numero dispari superiore a questo numero!).

Per facilitare l'esposizione del listato riportato, diverse righe sono state numerate .

Le parti principali che realizzano l'algoritmo proposto sono in successione le seguenti:

a) da riga 10 a riga 19 si hanno le istruzioni dedicate alla introduzione del numero **p** dispari con le opportune limitazioni ;

b) da riga 20 a riga 29 : istruzioni relative alla generazione di 25 "Basi", cioè di 25 numeri x (s) diversi tra loro, con s variabile da 1 a 25; ciascuna base è scelta in modo casuale di valore tra 2 e p - 1 utilizzando semplicemente la funzione RANDOMIZE sia per l'esponente es della potenza di 10 sia per la mantissa RND del numero x(s), (indicato d'ora in poi con x) messo sotto la forma :

x = INT (RND *10 ^{es})

e ciò per generare numeri costituiti da una diversa quantità di cifre ;

c) da riga 30 a riga 39 :
istruzioni dedicate al calcolo di **q** e di **k** una volta posto $\mathbf{p = 1 + 2^k \cdot q}$ con **q** dispari ;

d) da riga 40 a riga 49 : istruzioni per la conversione in forma binaria del numero **q**: ciò serve per poter applicare una *efficace procedura* per il progressivo calcolo in **aritmetica modulare** delle potenze crescenti di x sino alla potenza finale x^q [8].

e) da riga 50 a riga nes : istruzioni costituenti il loop principale del test da ripetersi con le modalità sottoillustrate per *ogni base* x; tale loop contiene i due loop interni seguenti :

1° loop : da riga 60 a riga nej : esso è relativo al calcolo in **aritmetica modulare** del resto

di $\dfrac{x^q}{p}$

e conseguente verifica della primalità del numero p ; il numero di ripetizioni da effettuarsi in questo loop è dato diminuito di uno, dal numero di cifre binarie (1, 0) che formano il valore di **q**, espresso appunto in numerazione binaria ;

2° loop : da riga 80 a riga 90 : relativo al calcolo sempre **in aritmetica modulare** del resto

di $\dfrac{x^{2^i}}{p}$

con i variabile da 1 a k-1, e conseguente verifica della primalità di p.

Nella realizzazione del programma si sono tenute presenti le seguenti osservazioni:

- le operazioni relative al 2° loop *non devono essere effettuate* se il resto trovato alla fine della esecuzione delle operazioni svolte nel 1° loop è di valore **1** oppure di valore **p-1** ; alla prima base x per cui si constata che il resto della divisione al termine del 2° loop risulta di valore diverso da **p-1** si arresta il test in quanto il numero p è certamente composto ; solo se per tutte le basi x considerate si verifica che il resto al termine del 1° loop ha valore **1** o **p-1**; se non si ottenesse ciò, ma il resto al termine del 2° loop fosse di valore **p-1**, si può dichiarare che il numero **p** è primo, con una minima probabilità che non lo sia pari a $\dfrac{1}{4^{nb}}$, dove nb = 25 è il numero delle basi considerate.

Se ci limitassimo però a questo programma non otterremmo praticamente nessun vantaggio di

tempo di calcolo rispetto a quello ottenibile con un test di primalità di tipo classico .

Anzi, per numeri così piccoli sarebbe forse più vantaggioso un test di verifica usando un metodo classico, se non altro perché per un numero composto si otterrebbe anche la sua completa scomposizione in fattori.

Per trattare numeri grandi si è già accennato che non è più sufficiente la doppia precisione disponibile sul computer; si dovranno effettuare le operazioni aritmetiche elementari con una *aritmetica a precisione multipla*, utilizzando per accelerare i calcoli, specie nell'esecuzione dell'operazione della divisione, non la base 10 quale base di numerazione, bensì la base 10^g , con g variabile tra 5 e 7 in dipendenza dalla quantità di cifre costituenti in tutte le divisioni il divisore comune, che è poi il numero p che si vuole verificare o generare.

PROGRAMMA PER LA VERIFICA DELLA PRIMALITA' DI UN NUMERO ≤ 94906267

```
REM   PROGRAMMA per la  VERIFICA della PRIMALITA' di  un NUMERO  ≤ 94906267
      CLS: DEFDBL  A-Z  :  lm  =  94906267
10    INPUT "p"; p :
      IF  p  =  INT  ( p  /  2 )  *  2  THEN  PRINT  " introdurre numero dispari ": GOTO 10
      IF  p  <= lm   GOTO  20'
      PRINT  " il numero non deve essere maggiore di ''';  lm
19    PRINT  " introdurre un numero piu basso" :  GOTO  10
20     nb = 25  :  IF p < 100  THEN  nb= 10 :  IF p < 30  THEN  nb =  5
      DIM  es (nb) ,  x ( nb ) :  cp  =  INT ( LOG ( p)  / LOG ( 10 ) )  +  1
      t1 = TIMER :  FOR  l  =  0  TO  nb  :  RANDOMIZE  t1 :  es(l )  =  INT (  RND * 10 )
      es = es ( l )  : IF es  =  0   OR  es  >  cp   THEN   l = l - 1 :  GOTO  nel
      x( l )  =  INT ( RND * (10 ^ es ) )
          IF x ( l )  <  2   OR  x ( l )   >  p - 1   THEN   l  =  l - 1 :  GOTO  nel
      FOR  h  =  1  TO  l - 1 :
        IF  x ( l )  =  x ( h )  THEN   l  =  l  -  1 :  GOTO  nel
      NEXT h
nel : NEXT  l
30    p0  =  p - 1 :  n  =  INT ( LOG ( p0 )  /  LOG( 2 ) )
       FOR   j  =  0  TO  n
        p0  =  p0  /  2  :  IF  p0  <>  INT ( p0  /  2 )  *  2   THEN   k  =  j  :  EXIT  FOR
       NEXT   j
     q  =  p0  :   m  =  INT ( LOG ( q )    /   LOG
( 2 )): DIM  a ( m )
39   ' PRINT  " k = "; k , "q = " ; q  :  PRINT " p  =   1  +  2  ^ ";  k ; " * " ;  q
40    y  = q
       FOR  i = 0  TO   m  :
                y1  = INT (y  / 2 ) :  a ( i )  =  y  -  2  *  y1  : y  =  y1  :  ' PRINT  a ( i )
49     NEXT i
50     FOR  s = 1  TO  nb
          b  =  x ( s ) :  c  =  x ( s )
60           FOR  j  =  1  TO   m
              z  =  b  *  b  :  b  =  z  -  INT ( z  / p )  *  p
                IF a ( j )  =  0   GOTO  nej
                c  =   c  *  b :   c  =  c  -  INT  ( c  / p )  *  p
nej :          NEXT j
70        IF c  = 1  OR  c  = p - 1  GOTO  nes
80           FOR  h  =  1  TO   k  - 1
               c  =   c  *  c  :  c  =  c  -   INT  (  c  /   p )  *  p
89             IF c  =  p - 1  goto nes
90           NEXT  h
100       GOTO composto
nes :  NEXT  s
PRINT   p  ;  " è  primo":   GOTO tempo
composto :   PRINT  p ; " è composto "
tempo :    PRINT  TIMER - t1  ;  " secondi "
```

Programma per numeri relativamente grandi

Il programma relativo a numeri grandi, il cui listato data la sua lunghezza (più di 400 righe in linguaggio QBasic) viene riportato in **ALLEGATO** al presente articolo e non è altro che la traduzione in aritmetica a precisione multipla del programma sopra illustrato.

Poiché si devono trattare numeri grandi e quindi costituiti da una quantità di cifre superiore a quella ammessa per la doppia precisione, altrimenti s'incorrerebbe in arrotondamenti assolutamente da evitare, il numero **p** deve essere introdotto nel programma come stringa di caratteri numerici (cifre), con la limitazione a 255 cifre, se ci si limita ad introdurre il numero come unica stringa.

Introducendo invece il numero come somma di due stringhe si possono trattare numeri fino a 510 cifre. Lo stesso trattamento subiranno le diverse Basi scelte in modo casuale, con la sola limitazione di contenere ciascuna non più di 15 cifre .

Si eseguiranno quindi le operazioni aritmetiche elementari di addizione, sottrazione e moltiplicazione in aritmetica a precisione multipla con gli stessi metodi che si adoperano abitualmente a mano con carta e penna e con le modalità esposte in [9].

Per l'operazione della divisione, si vuole qui solo accennare, senza approfondire il discorso, che per essa viene utilizzato invece un metodo originale, diverso da quello tradizionale che abbiamo imparato sui banchi di scuola e adoperato normalmente nei calcoli fatti a mano.

In effetti per la divisione il metodo tradizionale non risulta né il più semplice da programmare, né tantomeno il più efficiente.

Il programma presenta due opzioni : una riguardante la *verifica* se un numero è primo o composto, l'altra relativa alla *creazione* di un numero primo .

Per scegliere un'opzione o l'altra si devono seguire le indicazioni che compaiono sullo schermo.

Il linguaggio utilizzato può essere indifferentemente il BASIC o il QuikBASIC, (QBASIC) quest'ultimo consigliabile per sveltire l'esecuzione dei calcoli ed ottenere tempi di elaborazione 6 ÷ 7 volte più brevi di quelli ottenuti con il BASIC.

Una volta immagazzinato il programma con il nome "VERCREAN.BAS", seguendo le istruzioni lo si apra, e per eseguirlo si batta il tasto F5, seguendo sempre le istruzioni che compaiono sullo schermo.

Scelta ad esempio l'opzione di verifica (**ver**) e digitate le cifre del numero che si vuole controllare, si otterrà la risposta se esso è primo o composto dopo un tempo variabile dipendente dalla grandezza del numero e ovviamente dal PC utilizzato; se il numero digitato è primo la sua verifica richiederà un tempo di elaborazione all'incirca " nb" volte quello richiesto per un numero composto dello stesso ordine di grandezza; sullo schermo comparirà anche il tempo impiegato e, se il numero è primo, anche i valori numerici casuali delle " nb" *Basi* usate.

Vi è da sottolineare che invece di utilizzare subito l'algoritmo di Rabin - Miller, per accelerare nella maggioranza dei casi il tempo di calcolo, risulta conveniente controllare con un metodo classico efficace (ad esempio quello già menzionato) se il numero presenta un fattore primo di valore inferiore ad un numero prefissato, per esempio 10^4.

Così facendo, quanto più grande è il numero da verificare, maggiore sarà il risparmio di tempo necessario alla verifica o per la creazione .

Il listato dell'effettivo programma relativo ai numeri primi grandi che si vorrebbe proporre alla vostra attenzione viene riportato a causa della sua eccessiva stesura (più di 450 righe d'istruzione in linguaggio QBASIC viene riportato come allegato solo nella versione web di questo articolo.

Una precisazione: perché si possa utilizzare il programma occorre innanzitutto che sul vostro PC sia installato il pacchetto software relativo al QBASIC.

Una volta disponibile detto pacchetto software, si potrà ricopiare riga per riga tutto il listato, nel quale vengono contemplate diverse righe di REM, che sono state introdotte per renderlo un po' più comprensibile, anche se ciò ha dato luogo ad un suo allungamento.

Tali righe sono state poste, con l'indicazione sintetica del loro contenuto, all'inizio di ogni parte principale del programma e di ogni operazione aritmetica importante richiesta dall'algoritmo usato. Inoltre per chi lo richiedesse il programma è disponibile su floppy-disk.

Con detto programma si possono trattare numeri primi costituiti sino ad un massimo di 510 cifre pari a 1694 cifre binarie. Si sottolinea che con tale valore limite si è ampiamente entro il campo richiesto di generazione e verifica di primi utilizzabili negli algoritmi crittografici RSA e DSA per la formazione delle chiavi pubbliche e private relative alla *firma digitale*, come indicato nel Decreto del Presidente del Consiglio dei Ministri del 8 febbraio 1999, pubblicato sulla Gazzetta Ufficiale del 15/4/99, in cui viene prescritto che (vedi art. 4, punto 6): " *La lunghezza minima delle chiavi è stabilita in 1024 bit ".*

Un'ultima osservazione: il numero indicato sotto il titolo costituito da 82 cifre è primo e la sua verifica implica 6 secondi di calcolo.

TABELLE

Nella TABELLA n° 1 vengono riportati per il tipo di numero indicato i risultati dei tempi di calcolo ottenuti con un PC dotato di microprocessore, utilizzando il linguaggio Microsoft QuickBASIC versione 4.5. In essa compaiono anche i tempi di calcolo ottenibili, sempre utilizzando lo stesso PC e lo stesso linguaggio, con il *metodo classico delle Divisioni Successive* (Trial Division) già in precedenza citato.

Se si sceglie l'altra opzione (**crea**), una volta digitate a caso le cifre per formare un numero qualsiasi di partenza, si otterrà dopo un tempo variabile e assolutamente impossibile da prevedere il numero primo immediatamente successivo al numero digitato.

Per questa opzione si sfruttano abbondantemente le istruzioni relative alla opzione **ver**.

Pertanto per la verifica della primalità o meno di un numero, dopo averlo introdotto, viene controllato se esso presenta o no almeno un fatto- $6h \pm 1$ re, con il metodo classico delle divisioni successive con divisori oltre il 2 ed il 3, della forma

fino ad un divisore massimo di 10^4 ;

nel caso che non si trovasse nessun fattore inferiore a si prosegue con il test di Rabin-Miller, fino a che non si arriva a stabilire con le condizioni imposte da tale algoritmo se il numero in esame è composto o primo.

Per la generazione di un numero primo si esegue sul numero introdotto una stessa procedura, proseguendo poi se esso risultasse composto con il numero dispari immediatamente successivo e così via, fino a trovare un valore numerico soddisfacente le condizioni di primalità imposte dallo algoritmo di Rabin-Miller.

Nella TABELLA n° 2 vengono riportati per numeri di diversa grandezza il tempo minimo **Ta** di calcolo per l'ottenimento del numero primo, inteso come media dei tempi di calcolo trovati in diverse prove effettuate: tale tempo lo si ottiene nel caso *fortunato* di aver digitato proprio le cifre costituenti un numero primo.

Più di frequente invece si otterrà un tempo di calcolo maggiore impossibile da prevedere: il tempo **Tb** indicato in tabella è il *tempo medio* ricavato da diverse prove effettuate.

In tabella viene riportato anche il massimo valore del tempo **Tc** di calcolo che si potrebbe riscontrare nelle condizioni più sfavorevoli che si verificano allorché il numero digitato risulta alla massima distanza ($gap_{\max}$) dal primo immediatamente successivo al numero introdotto, dove con il termine **gap** si intende indicare la differenza fra il numero primo **p** ottenuto e il numero random digitato.

In effetti detti valori per **Tc** sono pessimistici all'estremo in quanto si suppone che il numero introdotto sia immediatamente successivo ad un numero primo.

I valori numerici di **Tc** che compaiono in tabella sono stati calcolati come segue.

Per ciascuna delle prove effettuate, presi in considerazione il tempo **tb** ed il corrispondente **gap** trovati nella creazione di ogni primo di data grandezza, tenendo conto del tempo **ta** impiegato per la verifica della sua primalità e rifacendosi alla *congettura* [10], [11], che la massima distanza o *massimo gap* ($gap_{\max}$) fra due primi consecutivi possa ricavarsi dalla seguente relazione [12]:

$$gap_{\max} \sim 2e^{-\gamma} \cdot (\ln \mathbf{p})^2$$

con:

e = 2.71828182…(base dei logaritmi naturali)

γ = 0.577215664…..(costante di Eulero) ,

e dove il segno ~ sta ad indicare che la relazione è asintoticamente valida, è facile risalire al tempo **tm** necessario per percorrere il *massimo gap*;

$$\mathbf{tm} = \Delta t \cdot \frac{gap_{\max}}{gap} + \mathbf{ta}$$

si avrà:

con $\Delta t = \mathbf{tb} - \mathbf{ta}$

Ebbene, il valore **Tc** in tabella è la media dei diversi tempi massimi **tm** trovati nelle diverse effettive prove eseguite relative alla creazione di numeri primi, ciascuno della grandezza riportata

TABELLA n° **I**
relativa ai **tempi** di **calcolo necessari** per la **verifica** della **primalità** di un Numero

Quantità di **Cifre Componenti** il Numero	**Tipo** di numero	**Tempo di calcolo massimo trovato** con algoritmo di **RABIN – MILLER** in QBasic	**Tempo di calcolo minimo necessario** con algoritmo **CLASSICO** in QBasic
14	Composto difficile	immediato	0.3 secondi
14	Primo	immediato	
16	Composto difficile	immediato	6 secondi
16	Primo	immediato	
20	Composto difficile	immediato	1 ora e 12 minuti
20	Primo	immediato	
28	Composto difficile	immediato	497 giorni *
28	Primo	immediato	
30	Composto difficile	immediato	> 13 anni *
30	Primo	immediato	
42	Composto difficile	immediato	> 13 milioni di anni*
42	Primo	1 secondo	
50	Composto difficile	immediato	Maggiore * dell'età dell'Universo
50	Primo	3 secondi **	
56	Composto difficile	immediato	
56	Primo	2 secondi **	
77	Composto difficile	immediato	
77	Primo	6 secondi	
154	Composto difficile	2 secondi	
154	Primo	42 secondi	
308	Composto difficile	5 secondi	
308	Primo	395 secondi	

* tali **tempi di calcolo** sono stati ovviamente ricavati per estrapolazione da quelli ottenuti per numeri costituiti da 14, 16 e 20 cifre.

I valori di **Tempo di calcolo** riportati in terza colonna fanno riferimento al **massimo valore** trovato in diverse prove effettuate su numeri delle dimensioni e del tipo rispettivamente riportati. In prima e seconda colonna. ** il valore di 3 secondi per numeri con 50 cifre risulta da calcoli effettuati con una aritmetica in *base* 10^5 ; il valore di 2 secondi per numeri con 56 cifre risulta da una aritmetica in *base* 10^7 .

nella prima colonna della tabella.

Come si può notare i valori di **Tc**, anche se grandi rispetto ai corrispondenti valori **Tb**, sono un'inezia rispetto ai valori che si otterrebbero coi metodi classici; da considerare inoltre che questi tempi si vengono ad avere solo nella rara condizione di aver digitato numeri dislocati alla massima distanza dal primo immediatamente successivo.

TABELLA n° **2**
relativa ai tempi richiesti per la **generazione** di Numeri PRIMI

n° di **cifre componenti** il Numero **primo**	**Ta = Tempo minimo impiegabile** con algoritmo di **RABIN-MILLER** in QUICKBASIC	**Tb = Tempo medio impiegato** con algoritmo di **RABIN - MILLER** in QUICKBASIC	**Tc = Tempo massimo impiegabile** con algoritmo di **RABIN – MILLER** in QUICKBASIC	**Td = Tempo minimo necessario** con algoritmo **CLASSICO** in QUICKBASIC
14	immediato	immediato	< 2 secondi	0.3 secondi
16	immediato	immediato	< 3 secondi	6 secondi
20	immediato	immediato	< 5 secondi	1 ora e 12 minuti
28	immediato	< 1 secondo	30 secondi	497 giorni *
30	1 secondo	1.5 secondi	45 secondi	> 13 anni *
42	1 secondo	1.6 secondi	50 secondi	>13 milioni di anni *
50	3 secondi	7 secondi **	203 secondi	Maggiore * dell'età dell'universo
56	2 secondi	4 secondi **	212 secondi	
77	5 secondi	8 secondi	< 0.25 ore	
119	18 secondi	31 secondi	< 2 ore	
154	40 secondi	350 secondi	< 7.5 ore	
308	329 secondi	895 secondi	7 giorni e 11 ore	

Note :

- il tempo **Ta** è ricavato quale *valore medio* dei tempi necessari per la *verifica* della primalità di un numero *primo* della grandezza riportata nella prima colonna, trovati in varie prove effettuate;
- il tempo **Tb** è il valore medio ricavato dagli effettivi tempi impiegati nelle diverse prove eseguite per creare numeri primi della grandezza riportata in prima colonna ;
- il valore limite **Tc** è anch'esso il valore medio ricavato dagli effettivi tempi impiegati nelle diverse prove che si riferiscono al caso più sfavorevole in cui il numero digitato risulta alla massima distanza dal primo che si vuole creare (secondo la congettura accennata nel testo) ;
- i valori limite **Td** sono i *tempi minimi necessari* per generare un numero primo, dell'ordine di grandezza riportata in prima colonna, con l'impiego dell'algoritmo classico delle divisionisuccessive di prova utilizzando quali divisori i numeri 2, 3 ed i numeri della forma $6h \pm 1$ con h=1, 2, 3, 4, 5, ……;
- * i tempi **Td** contrassegnati da un asterisco in tabella, di valore elevato, sono ovviamente desunti per estrapolazione dai valori trovati nella generazione di numeri primi più piccoli costituiti cioè da 16, 18 e 20 cifre.
- ** il valore di 7 secondi per numeri con 50 cifre risulta da calcoli effettuati con una aritmetica in *base* 10^5 ; il valore di 4 secondi per numeri con 56 cifre risulta da una aritmetica in base 10^7 .

RIFERIMENTI

1 - D. E. Knuth, *The art of computer programming*, Vol. 2: Seminumerical algorithms, 2nd ed., Addison - Wesley, Reading, M.A., 1981
2 - B. Schneier, *Applied Cryptography*, John Wiley & Sons, Inc.
3 - F. Arnault, *Rabin - Miller Primality test : composite numbers wich pass it,* Mathematics of Computation, Vol. 64, n. 209 (January 1995) , 355 - 365
4 - H. Riesel, *Prime numbers and computer methods for factorization*, 2nd ed., Birkhäuser, Boston 1994
5 - M.O. Rabin, *Probabilistic algorithms for testing primality*, Journal of Number Theory 12 (1980), 128 -138
6 - D. E. Knuth, *The art of computer programming*, Vol. 2: Seminumerical algorithms, 2nd ed., Addison - Wesley, Reading, M.A., 1981
7 - C. Pomerance, *Alla ricerca dei numeri primi*, Le Scienze n. 174, febbraio 1983
8 - D. E. Knuth , *The art of computer programming*, Vol. 2: Seminumerical algorithms, 2nd ed., Addison - Wesley, Reading, M.A., 1981
9 - C. Teodoro, *Aritmetica a precisione multipla*, Mcmicrocompter n.56, ottobre 1986
10 - D. Shanks, *On Maximal Gap between Successive Primes* , Mathematics of Computation, Vol. 18, n. 88 (October 1964) , 646 - 651
11 - H. Riesel , *Prime numbers and computer methods for factorization*, 2nd ed., Birkhäuser, Boston 1994
12 - G.Tenenbaum, M. Mendès France, *Les nombres premiers*, 1re édition : 1997, mars, collection *Que sais-je?*, Presse Universitaire de France (PUF)

Cristiano Teodoro, si è laureato in Ingegneria Elettronica presso l'Università "La Sapienza" di Roma. Ha percorso la sua carriera professionale presso l'Istituto Superiore delle Poste e delle Telecomunicazioni (ISPT) e quindi presso l'attuale ISCTI. Ha svolto incarichi in diversi campi ed attività delle telecomunicazioni fra cui in particolare quelli di normazione, di omologazione e di collaudo di apparati terminali telegrafici (telescriventi) telefonici e dati (modem), di apparati di multiplazione e trasmissione a lunga distanza su portante fisico di segnali Telefonici e Dati relativi alla multiplazione PCM, alla Gerarchia Plesiocrona (PDH) e alla gerarchia sincrona (SDH).
Ha svolto inoltre una intensa attività Normativa e di Standardizzazione sia in ambito Nazionale che Internazionale quale Relatore Nazionale di due Commissioni dell'UIT - T e come membro di diverse Commissioni dello stesso Organismo Internazionale.
E' docente della materia "Esercitazioni di Telegrafia e Telefonia" presso e la Scuola Superiore di Specializzazione in Telecomunicazioni (SSST) dell'ISCTI.

cristianoteodoro@virgilio.it

ALLEGATO

```
REM ---------------------->  PROGRAMMA "VERCREAN.BAS  <----------------------

REM Il presente programma è dedicato alla VERIFICA della primalità o meno di
REM un numero  ed alla generazione di numeri primi relativamente grandi
REM costituiti  da  più di 255 cifre decimali
REM si possono trattare numeri composti da non meno di 5 cifre sino ad un
REM massimo di 510 cifre.

REM E'comunque sufficiente introdurre numeri di 308 cifre decimali pari a
REM 1024 cifre binarie per creare numeri primi quali chiavi pubbliche per la
REM FIRMA DIGITALE ELETTRONICA, come indicato dal Decreto del Presidente del
REM Consiglio dei Ministri 8 febbraio 1999, comparso sulla Gazzette Ufficiale
REM del 15/4/99 in cui vengono  prescritti (vedi art.4 ,punto 6) come chiavi
REM numeri binari costituiti da 1024 bit.
REM Il programma contempla due opzioni:
REM   con la 1^ opzione si verifica se il numero introdotto è primo o
REM    composto;
REM   con la 2^ opzione si crea (si genera) un numero primo casuale (random)
REM    derivato dal numero introdotto digitando a caso delle cifre (nextprime)
REM ------------------------------------------------------------------------
SCREEN 0
CLS : DEFDBL A-Z: 'WIDTH 40, 25
DIM b(30), gs(30): 'pr(30),
DIM a(200), a$(200), b$(200), ab$(200), ar$(100), rn$(200)
DIM x(200), z(200), c(200), d(200), pp(200)
DIM p(200), f(200), p0(200), f0(200), w0(2000), q(200):  ' w(1000)
h = 0: pri = 0: pb = .000000000000001#
nb = 25: REM nb Š il numero delle basi (ovvero dei test da effettuare)

REM se il numero e' a 3 o 4 cifre si pone nb = 20: 'SCREEN 9
PRINT : PRINT

PRINT "ATTENZIONE ------MODALITA DI INTRODUZIONE DEL NUMERO -------"
PRINT
PRINT "  Se il numero è composto da meno di 256, introducilo, battendo"
PRINT "  poi il tasto Invio due volte di seguito"
PRINT
PRINT "  Se il numero è composto da C cifre con C> 255  (max 510  cifre),"
PRINT "     a) -  introduci le prime C1 cifre del numero con C1< 256,";
PRINT "           e batti quindi il tasto Invio"
PRINT "     b) -  introduci poi le rimanenti C2 cifre del numero (C2 = C-C1)";
PRINT "           e batti di nuovo il tasto Invio"
PRINT
REM ------------------------ 2 OPZIONI -------------------------------
```

```
COLOR 15: PRINT " vuoi verificare se un numero Š primo o composto ?"
PRINT " digita"; : COLOR 14: PRINT " ver "; : COLOR 15:
PRINT " e premi poi tasto "; : COLOR 14: PRINT "Invio": COLOR 15: PRINT
PRINT " vuoi invece creare un numero primo random ?": PRINT " digita";
COLOR 14: PRINT " crea "; : COLOR 15: PRINT "e premi poi tasto "; : COLOR 14
PRINT "Invio"
10 INPUT " ", op$: v$ = "ver": cr$ = "crea"
IF op$ = v$ GOTO verifica
IF op$ = cr$ GOTO crea
IF op$ <> v$ OR op$ <> cr$ THEN PRINT "ridigita": GOTO 10
verifica: CLS : PRINT : COLOR 15, 1
CLS : COLOR 15
REM ------------------ 1^ OPZIONE -----------------------------------
PRINT "                VERIFICA  della PRIMALITA' di un numero"
PRINT " digita il numero e poi premi tasto Invio : ": COLOR 3
PRINT "nøcifre: 10        20        30        40        50        60        ";
PRINT " 70        8"
PRINT "---------|---------|---------|---------|---------|---------|--------";
PRINT "-|---------|"
COLOR 15
INPUT "", d1$
INPUT "", d2$
t1 = TIMER
cf1 = LEN(d1$): p1 = VAL(d1$):
cf2 = LEN(d2$): p1 = VAL(d2$):
dn$ = d1$ + d2$: dni$ = dn$
cf = LEN(dn$): p = VAL(dn$):
IF cf < 5 THEN nb = 15: pb = .000000001#
FOR k = 1 TO cf: us$ = MID$(dn$, k, 1)
IF ASC(us$) < 48 OR ASC(us$) > 57 THEN PRINT : GOTO noncor
NEXT k
GOTO tre
pausa: PRINT " hai digitato in modo non corretto :"
PRINT " premi un tasto qualsiasi e quindi ridigita il numero;"
DO: y$ = INKEY$: LOOP WHILE y$ = "": GOTO verifica
tre: COLOR 3: PRINT "( cifre del numero :"; cf; ")"
COLOR 7: IF cf < 3 THEN PRINT " hai introdotto un numero troppo piccolo!"
IF cf < 3 THEN PRINT " il numero deve avere almeno 3 cifre": GOTO 253
u$ = RIGHT$(dn$, 1): u = VAL(u$): ui = INT(u / 2) * 2
IF u = ui THEN PRINT " hai introdotto un "; : COLOR 14: PRINT "numero pari:";
IF u = ui THEN PRINT " : manifestamente COMPOSTO": GOTO 253
IF u$ = "5" THEN PRINT " il numero Š manifestamente";
IF u$ = "5" THEN COLOR 14: PRINT " COMPOSTO e divisibile per 5"; : GOTO 253
FOR k = 1 TO cf: um = VAL(MID$(dn$, k, 1)): u3 = u3 + um: NEXT k: u = u3 / 3
IF u = INT(u) THEN PRINT " il numero e'"; : COLOR 14
IF u = INT(u) THEN PRINT " COMPOSTO e divisibile per 3": GOTO 253
GOTO alloca
```

```
253 : COLOR 7: PRINT " vuoi tornare al programma ? digita lettera ";
COLOR 14: PRINT "p"; : COLOR 7: PRINT " e premi"; : COLOR 14: PRINT " 2 ";
PRINT "volte"; : COLOR 7: PRINT " tasto"; : COLOR 14: PRINT " Invio": COLOR 7
PRINT " vuoi verificare un altro numero ?"; : PRINT " digita lettera ";
COLOR 14: PRINT "v"; : COLOR 7: PRINT " e premi"; : PRINT "; tasto; "; ""
COLOR 14: PRINT " Invio ": INPUT " ", us$
24 IF us$ = "v" GOTO verifica
IF us$ = "p" GOTO fineu
PRINT "hai premuto tasto sbagliato"
INPUT ""; us$: GOTO 24
alloca: LOCATE 15, 1: COLOR 3: PRINT STRING$(80, "-")
LOCATE 28, 1: COLOR 3: PRINT STRING$(80, "-")
COLOR 10
VIEW PRINT 27 TO 28: PRINT STRING$(26, " "); "computer in elaborazione v"
GOTO iniziot
REM istruzioni relative ad incrementare di 1 il numero introdotto se pari
REM oppure di 2 se esso finisce rispettivamente per 5
REM ---------- 2^ OPZIONE ----------
crea: CLS :  COLOR 15, 1
CLS : COLOR 12
'crea: CLS : COLOR 14, 1: COLOR 12
PRINT "                CREAZIONE di un numero PRIMO RANDOM": ' PRINT
PRINT "componi un numero digitando a caso tante cifre quante ne vuoi:";
PRINT " premi poi Invio"
COLOR 3
PRINT "nøcifre: 10        20        30        40        50        60        ";
PRINT " 70        8"
PRINT "---------|---------|---------|---------|---------|---------|--------";
PRINT "-|---------|"
COLOR 12: 'dn$ = " 111222333444555666777888"
INPUT "", d1$: cf1 = LEN(d1$): p1 = VAL(d1$):
INPUT "", d2$
t1 = TIMER
cf2 = LEN(d2$): p1 = VAL(d2$):
dn$ = d1$ + d2$: dni$ = dn$
cf = LEN(dn$): p = VAL(dn$):
IF cf < 5 THEN nb = 15: pb = .000000001#
FOR k = 1 TO cf: us$ = MID$(dn$, k, 1)
IF ASC(us$) < 48 OR ASC(us$) > 57 THEN PRINT : GOTO noncor
NEXT k
GOTO minortre
noncor: COLOR 15: PRINT " hai digitato in modo non corretto :"
PRINT " premi un tasto qualsiasi e quindi ridigita il numero;"
DO: y$ = INKEY$: LOOP WHILE y$ = "": GOTO crea
minortre: IF cf > 2 GOTO trebis
COLOR 15: PRINT " hai introdotto un numero troppo piccolo !":
PRINT " il numero deve avere almeno tre cifre"
```

```
PRINT " premi un tasto qualsiasi e quindi ridigita il numero"
DO: y$ = INKEY$: LOOP WHILE y$ = "": GOTO crea
trebis: COLOR 3: PRINT "( cifre del numero+ :"; cf; ")"

uc$ = RIGHT$(dn$, 1): rd$ = LEFT$(dn$, cf - 1): pg = 0:  ' PRINT uc$, rd$
uc = VAL(uc$): IF uc = 4 THEN uc = uc + 3: pg = 3: GOTO ulci
IF INT(uc / 2) * 2 = uc THEN uc = uc + 1: pg = 1
IF uc = 5 THEN uc = uc + 2: pg = 2
ulci: uc$ = RIGHT$(STR$(uc), 1)
dn$ = rd$ + uc$
'DO: y$ = INKEY$: LOOP WHILE y$ = ""
limita: LOCATE 15, 1: COLOR 3: PRINT STRING$(80, "-")
LOCATE 28, 1: COLOR 3: PRINT STRING$(80, "-")
COLOR 10:
VIEW PRINT 27 TO 28: PRINT STRING$(26, " "); " computer in elaborazionep"
dai$ = DATE$: ori$ = TIME$
iniziot: t1 = TIMER: IF op$ = v$ GOTO emmem
REM -------------- ------------------------------------------------------
RI = 0: cf = LEN(dn$): cf1 = cf
ricomin:  mm = 0: nn = nn + 1: ' PRINT "nn="; nn, "ri="; ri
COLOR 14: VIEW PRINT 16 TO 21:  PRINT dn$,
'DO: y$ = INKEY$: LOOP WHILE y$ = ""
REM -------------- inizio ciclo verifica primalit… numeri in successione ---
ERASE w0, p, f, a$, p0, x, d, z, c, f0
p0 = 0: cv = 0: dv = 0: sv = 0: xv = 0: zv = 0: s = 0: k2 = 0: c = 0
'pv = 0
REM ----- determinazione della migliore Base di Numerazione :pr = 10^g ----
IF cf = cf1 AND RI = 1 GOTO numero
emmem: mm = 0: a = cf - INT(cf / 5) * 5: IF a = 0 THEN a = 5
   b = cf - INT(cf / 6) * 6: IF b = 0 THEN b = 6
   c = cf - INT(cf / 7) * 7: IF c = 0 THEN c = 7
   mm = b: g = 6: IF a > b THEN mm = a: g = 5
   IF mm <= c THEN mm = c: g = 7
   IF cf < 8 THEN g = INT((cf + 1) / 2)
pr = 10 ^ g: ' PRINT "g="; g; "pr="; pr,
REM --- il numero p introdotto come stringa dn$  viene memorizzato in modo
REM     opportuno nel vettore numerico p(h) ---------------------------------
numero: ' PRINT "NUMERO", dn$
pv = INT(cf / g): IF pv = cf / g THEN pv = pv - 1
ERASE p:    ' PRINT "^^^^^^^^^^^^^^^^"; pv; cf; g
'DO: y$ = INKEY$: LOOP WHILE y$ = ""
   FOR h = 0 TO pv - 1: c = cf + 1 - (h + 1) * g
     a$ = MID$(dn$, c, g): p(h) = VAL(a$): ' PRINT "?"; p(h), c; cf
     NEXT h
   IF c > 1 THEN a$ = LEFT$(dn$, c - 1): p(pv) = VAL(a$):
    'PRINT : PRINT "(("; p(pv);
    'FOR h = pv - 1 TO 0 STEP -1: PRINT p(h); : NEXT h
```

```
IF op$ = v$ GOTO hk6
REM ----------CONDIZIONI DI DIVISIBILITA' PER 3 ecc.----------
IF RI = 1 THEN p(0) = p(0) + 2
rx = 0: ' PRINT "ri ="; ri, "++++++++++p0="; p(0)
'DO: y$ = INKEY$: LOOP WHILE y$ = ""
IF p(0) / 5 = INT(p(0) / 5) THEN p(0) = p(0) + 2
ripr: rx = 0
FOR k = 0 TO pv: f = p(k) + rx: rx = INT(f / pr): p(k) = f - rx * pr: NEXT k
sf3 = 0: FOR k = 1 TO pv: sf3 = sf3 + p(k): NEXT k: ' PRINT "***"
si3 = (sf3 + p(0)) / 3: IF si3 = INT(si3) THEN p(0) = p(0) + 2:
FOR k = 0 TO pv: f = p(k) + rx: rx = INT(f / pr): p(k) = f - rx * pr: NEXT k
IF p(0) / 5 = INT(p(0) / 5) THEN p(0) = p(0) + 2
IF rx = 1 THEN pv = pv + 1: p(pv) = rx
ERASE pp
hk6: r = 0: h = -1
'DO: y$ = INKEY$: LOOP WHILE y$ = ""
acca: 'IF h > 10000 GOTO 250
IF op$ = v$ AND h > 1000 GOTO genbas
IF h > 10000 GOTO 250
h = h + 6: r = 0: ppv = p(pv): cpv = LEN(STR$(ppv)) - 1:
pih: FOR j = pv TO 0 STEP -1: b = r * pr + p(j): pp(j) = INT(b / h)
        r = b - pp(j) * h
        NEXT j
      IF r = 0 AND op$ = v$ THEN hh = h: GOTO composto
      IF r = 0 THEN p(0) = p(0) + 2: GOTO cinque
      'IF r = 0 THEN PRINT "h="; h: p(0) = p(0) + 2: GOTO cinque
      IF r <> 0 GOTO pik
pik: IF x = 1 THEN h = h - 2: x = 0: GOTO acca
r = 0: ERASE pp: h = h + 2: x = 1: GOTO pih
cinque: IF p(0) / 5 = INT(p(0) / 5) THEN p(0) = p(0) + 2: ' GOTO errex
GOTO ripr
rx = 0
FOR k = 0 TO pv: f = p(k) + rx: rx = INT(f / pr): p(k) = f - rx * pr: NEXT k
IF rx = 1 THEN pv = pv + 1: p(pv) = rx
250 : dn$ = "": z$ = "0": rx = 0
cp$ = STR$(p(pv)): c1 = LEN(cp$) - 1: cf = c1 + g * pv
FOR k = pv TO 0 STEP -1
cc$ = STR$(p(k)): lr = LEN(cc$) - 1
cr$ = RIGHT$(cc$, lr)
IF cf = g + 1 OR k = pv THEN aa$ = cr$: GOTO dienne
zz$ = STRING$(g - lr, z$): aa$ = zz$ + cr$: ' PRINT "aa$="; aa$
dienne: dn$ = dn$ + aa$: ' PRINT "aa$="; aa$
NEXT k
COLOR 14: VIEW PRINT 16 TO 21:  PRINT dn$,
'DO: y$ = INKEY$: LOOP WHILE y$ = ""
REM ---------- GENERAZIONE delle BASI come numeri random < 10^15  --------
genbas: cl = INT(LOG(p) / LOG(10)) + 1: t3 = TIMER
```

```
FOR l = 1 TO nb: RANDOMIZE t3: gs(l) = INT(RND * 16)
IF gs(l) = 0 OR gs(l) > cl THEN l = l - 1: GOTO nel
b(l) = INT(RND * (10 ^ gs(l)))
IF p <= 10 ^ 15 AND b(l) > p - 1 THEN l = l - 1: GOTO nel
IF b(l) < 2 OR b(l) >= p - 1 THEN l = l - 1: GOTO nel
FOR h = 1 TO l - 1: IF b(l) = b(h) THEN l = l - 1: h = l - 1: GOTO nel
NEXT h
nel: NEXT l
REM ------------ calcolo di q ------> q(j)  ( posto p = 1+2^k*q ) --------
REM    (il vettore numerico q(j) viene dedicato per contenere il valore di q)
FOR h = pv TO 1 STEP -1: q(h) = p(h): NEXT h
q(0) = p(0) - 1: p = VAL(dn$): n = INT(LOG(p) / LOG(2))
 po = pv
FOR j = 1 TO n: r = 0
    FOR h = po TO 0 STEP -1
      b = r * pr + q(h): q(h) = INT(b / 2): r = b - q(h) * 2
    NEXT h
      IF q(po) = 0 THEN po = po - 1
    pu = r * pr + q(0): IF pu <> INT(pu / 2) * 2 THEN k1 = j: j = n
NEXT j
k = k1
'PRINT "p = 1 + 2 ^"; k; "* (";
' FOR j = po TO 0 STEP -1: PRINT q(j); : NEXT j: PRINT ")"

REM --------- conversione in binario con sua dislocazione nel vettore w0(l)
REM            del valore numerico di q memorizzato nel vettore  q(h) -------
FOR h = po TO 0 STEP -1: f0(h) = q(h): NEXT h
  IF po = 0 THEN v1 = q(po): GOTO enneo
  v = q(po) * pr + q(po - 1) + 1: g1 = g * (po - 1): ' v1 = v * 10 ^ g1

enneo: lv = LOG(v) + g1 * LOG(10): n0 = INT(lv / LOG(2))
FOR l = 0 TO n0: r = 0
   FOR h = po TO 0 STEP -1:
      b = r * pr + f0(h): f0(h) = INT(b / 2): r = b - f0(h) * 2
   NEXT h
 w0(l) = r
NEXT l
IF w0(n0) = 0 THEN n0 = n0 - 1

REM ----------------- messa in forma delle BASI ---------------------------
m = 0
basem: m = m + 1: x = b(m): c = b(m):  ' PRINT m; : IF m < 15 THEN PRINT",";
cc = 0: cv = 0: cx = 0: xv = 0: zv = 0
cc = LEN(STR$(c)) - 1: cv = INT(cc / g): IF cv = cc / g THEN cv = cv - 1
FOR s = 0 TO cv: a = INT(c / pr): c(s) = c - a * pr: c = a: NEXT s
cx = LEN(STR$(x)) - 1: xv = INT(cx / g): IF xv = cx / g THEN xv = xv - 1
FOR s = 0 TO xv: a = INT(x / pr): x(s) = x - a * pr: x = a: NEXT s
```

```
REM -+-+-+-+-+-+-+-+-+  PRIMO CICLO  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

REM ============ inizio loop  riguardante l'OPERAZIONE : b^q mod p =======
REM ---  calcolo del RESTO della divisione ( b^q / p ) ovvero  del
REM ---  RESIDUO  di  x(s)^q(j)  mod p(h)
nk = n0 + k1
FOR j = 1 TO nk:  aa = 1000: ' PRINT "j="; j;
  IF j <= n0 GOTO ciclo1
   FOR k = 0 TO pv: x(k) = c(k): c(k) = 0: NEXT k
  xv = cv
REM ------------------------------------------------------------------------
REM  1) ELEVAZIONE al QUADRATO di x: x*x [(x=x(xv),x(xv-1).....,x(1),x(0)]
REM     z=x(j-1)*x(j-1)
ciclo1: FOR k = 0 TO xv: r = 0
       FOR h = 0 TO xv
       a = z(h + k) + x(h) * x(k) + r: r = INT(a / pr): z(h + k) = a - r * pr
       NEXT h
        z(h + k) = r
        NEXT k
   zv = xv + xv + 1: IF z(zv) = 0 THEN zv = zv - 1
REM ------------------------------------------------------------------------
REM  2a)           RESTO della operazione di DIVISIONE DI x*x=z  per  p
REM  2b) od anche  RESTO della operazione di DIVISIONE DI c*x=z  per  p
dividi:  d = p(pv) + 1: r = 0
       FOR h = zv TO pv + 1 STEP -1: a = z(h): sv = pv
         IF a < d THEN a = z(h) * pr + z(h - 1): sv = pv + 1
         q = INT(a / d)
          FOR k = 0 TO pv
           x = q * p(k) + r: r = INT(x / pr): z = x - r * pr: s = k + h - sv
           IF z > z(s) THEN z(s) = pr + z(s): z(s + 1) = z(s + 1) - 1
           z(s) = z(s) - z
          NEXT k
         z(h) = z(h) - r: r = 0: IF z(h) <> 0 THEN h = h + 1
        NEXT h
REM ---------- controllo sul resto  z  della divisione ------------------
REM --- se il vettore z(k) ancora > di  p(k) si saltera' a "prose0"
REM --- se il vettore z(k) < di p(k) allora c(k) e' gia'l'effettivo resto
REM --- si andra' a "resto0"
trovare0:  FOR I = pv TO 0 STEP -1: IF z(I) < p(I) THEN I = 0: GOTO resto0
             IF z(I) > p(I) THEN I = 0: GOTO prose0
           NEXT I
REM --- il resto z e' uguale a p (ovvero e' uguale a 0) -----------------

 GOTO composto
prose0: r = 0: d = p(pv) + 1
         IF z(pv) = p(pv) THEN d = p(pv)
         q = INT(z(pv) / d):
```

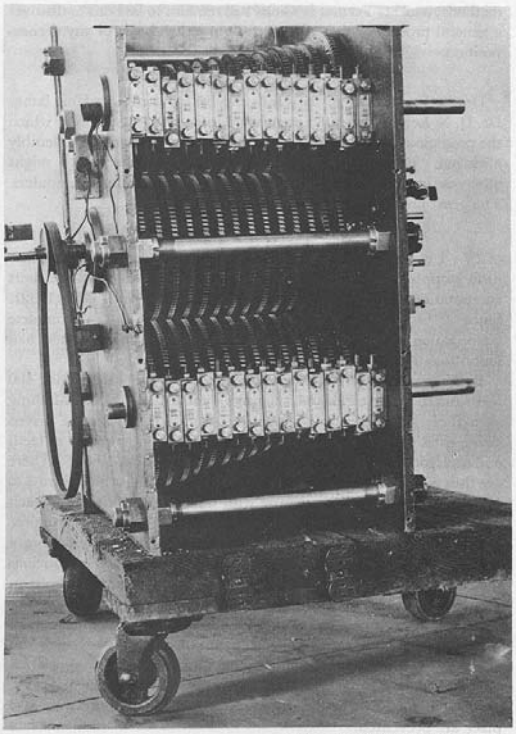
```
        FOR l = 0 TO pv
         x = q * p(l) + r: r = INT(x / pr): z = x - r * pr
         IF z > z(l) THEN z(l) = pr + z(l): z(l + 1) = z(l + 1) - 1
         z(l) = z(l) - z
        NEXT l
       IF z(pv) > 0 GOTO trovare0
resto0:  FOR l = pv TO 0 STEP -1: IF z(l) > 0 THEN zv = l: l = 0
          NEXT l

IF j <= n0 GOTO jminorn0

REM ======== SECONDE  CONDIZIONI DI PRIMALI-
TA'===3=========================
 FOR k = zv TO 0 STEP -1: c(k) = z(k): z(k) = 0: NEXT k
 cv = zv
  IF cv <> pv GOTO nej
      FOR h = pv TO 1 STEP -1: IF c(h) <> p(h) THEN h = 1: GOTO nej
      NEXT h
  IF c(0) = p(0) - 1 AND m = nb THEN j = nk: GOTO primo
  IF c(0) = p(0) - 1 THEN j = nk: GOTO 323
REM ------------------------------------------------------------------------
jminorn0: IF aa = 1111 GOTO 20
REM  3a)  RESTO espresso con il vettore  x(i),x(i-1).....x(1),x(0)
REM       ed  AZZERAMENTO di z(k)
   FOR k = zv TO 0 STEP -1: x(k) = z(k): z(k) = 0: NEXT k
   xv = zv: zv = 0
aa = 1111
IF w0(j) = 0 GOTO nej
REM ------------------------------------------------------------------------
REM  4)  PRODOTTO  c(k)*x(h)
   FOR h = 0 TO xv: r = 0
     FOR k = 0 TO cv
    xc = z(h + k) + c(k) * x(h) + r: r = INT(xc / pr): z(h + k) = xc - r * pr
     NEXT k
    z(h + k) = r
   NEXT h
  zv = xv + cv + 1: IF z(zv) = 0 THEN zv = zv - 1
REM ------------------------------------------------------------------------
GOTO dividi
REM  3b)  RESTO espresso con il vettore  c(i),c(i-1).....c(1),c(0)
REM       ed  AZZERAMENTO di z(k)
20   FOR k = zv TO 0 STEP -1: c(k) = z(k): z(k) = 0: NEXT k
    cv = zv: zv = 0
    aa = 1000
723    IF j < n0 GOTO nej
REM ------- VERIFICA DELLE PRIME CONDIZIONI di PRIMALITA' del numero
IF c(0) <> 1 AND c(0) <> p(0) - 1 GOTO nej
```

```
IF cv = 0 AND c(cv) = 1 AND m = nb GOTO primo
IF cv = 0 AND c(cv) = 1 GOTO 323
IF cv <> pv GOTO nej
FOR h = cv TO 1 STEP -1: IF c(h) <> p(h) THEN h = 1: GOTO nej
NEXT h
IF m = nb GOTO primo
GOTO 323
nej: NEXT j
REM ====== SECONDO CICLO: inserito nel loop di j per j>n0 sino a no+k1 ======
GOTO composto
323 : FOR k = 0 TO pv: c(k) = 0: NEXT k
IF m < nb GOTO basem
composto: IF op$ = cr$ GOTO pzero
IF op$ = v$ THEN CLS 2: COLOR 14, 1: VIEW PRINT 16 TO 21:
IF op$ = v$ THEN PRINT dn$; " Š COMPOSTO ":  cmp = 1: GOTO tempo
REM ------------------------------------------------------------
pzero: COLOR 10: 'nf = nf + 1
VIEW PRINT 27 TO 28: PRINT STRING$(26, " "); "pp computer in elaborazione"
RI = 1: GOTO ricomin
REM ------------------------------------------------------------------------
primo: COLOR 14: VIEW PRINT 16 TO 21
PRINT ""; dn$; "  Š PRIMO ": pri = 111
REM ------------------------------------------------------------------------
tempo: dt = TIMER - t1
ss = INT(dt): tv = dt - ss: td = ss: IF tv > .5 THEN td = ss + 1
mt = INT(td / 60):
COLOR 3: VIEW PRINT 27 TO 27: PRINT "( nø di cifre :"; cf; ")",
PRINT "tempo impiegato :"; td; " sec.";
IF mt = 0 GOTO basi
PRINT " ="; mt; "' e "; td - mt * 60; "sec.";
basi: 'LOCATE 31, 1
VIEW PRINT 31 TO 38:
IF cmp = 1 THEN PRINT " valore della base :"; : GOTO bs
COLOR 10:
PRINT "N.B.: la probabilit… che il numero non sia primo è inferiore a"; pb
COLOR 3: PRINT "valore delle"; nb; "Basi:"; :
bs: FOR l = 1 TO m - 1: PRINT b(l); ";"; : NEXT l: PRINT b(m),
daf$ = DATE$: orf$ = TIME$
PRINT : PRINT "inizio:"; dai$; "**"; ori$, "   fine:"; daf$; "**"; orf$
x = INT(cf / 78)
END
IF x = 0 AND op$ = "crea" THEN LPRINT : LPRINT dni$; "  numero iniziale": LPRINT
IF x = 0 THEN LPRINT dn$; "  Š PRIMO ": LPRINT "( nø di cifre :"; cf; ")"
IF x = 0 GOTO ltempo
DIM si$(x + 1), sp(x + 1), lsi(x + 1), lsp(x + 1)
si$(0) = LEFT$(dni$, 78): lsi(0) = LEN(si$(0))
sp$(0) = LEFT$(dn$, 78): lsp(0) = LEN(sp$(0))
```

```
FOR k = 1 TO x - 1
si$(k) = MID$(dni$, k * 78 + 1, 78): lsi(k) = LEN(si$(k))
sp$(k) = MID$(dn$, k * 78 + 1, 78): lsp(k) = LEN(sp$(k))
NEXT k
si$(x) = RIGHT$(dni$, cf - x * 78): lsi(x + 1) = LEN(si$(x + 1))
sp$(x) = RIGHT$(dn$, cf - x * 78): lsp(x + 1) = LEN(sp$(x + 1))
END
LPRINT
FOR k = 0 TO x - 1
LPRINT si$(k)
NEXT k
IF x <> cf / 78 THEN LPRINT si$(x);
LPRINT " numero iniziale"
LPRINT
460 IF op$ = v$ THEN PRINT "  Š COMPOSTO ": LPRINT dn$; " Š composto":
IF op$ = v$ THEN LPRINT " cifre del numero:"; cf: GOTO ltempo
'IF op$ = v$ GOTO ltempo
FOR k = 0 TO x - 1
LPRINT sp$(k)
NEXT k
IF x <> cf / 78 THEN LPRINT sp$(x);
LPRINT " Š primo"
LPRINT
LPRINT "( nø di cifre :"; cf; ")"
ltempo: LPRINT "tempo impiegato :"; td; " sec.";
LPRINT " ="; mt; "' e "; td - mt * 60; "sec."
IF op$ = v$ GOTO fineu
'LPRINT : LPRINT "inizio:"; dai$; "**"; ori$, "    fine:"; daf$; "**"; orf$
LPRINT "N.B.:la probabilit… che il numero non sia primo Š inferiore a"; pb
LPRINT "valore delle"; nb; "Basi:"; :
FOR l = 1 TO m - 1: LPRINT b(l); ";"; : NEXT l: LPRINT b(m)
LPRINT : LPRINT "inizio:"; dai$; "**"; ori$, "    fine:"; daf$; "**"; orf$
LPRINT
LPRINT "*****************************************************************"
fineu: END
```

VERIFICA E GENERAZIONE DI NUMERI PRIMI RELATIVAMENTE GRANDI

(VERIFICATION AND PRODUCTION OF QUITE LARGE PRIME NUMBERS)



Courtesy Carnegie Institution of Washington

PLATE A. Dr. Lehmer's Factoring Machine