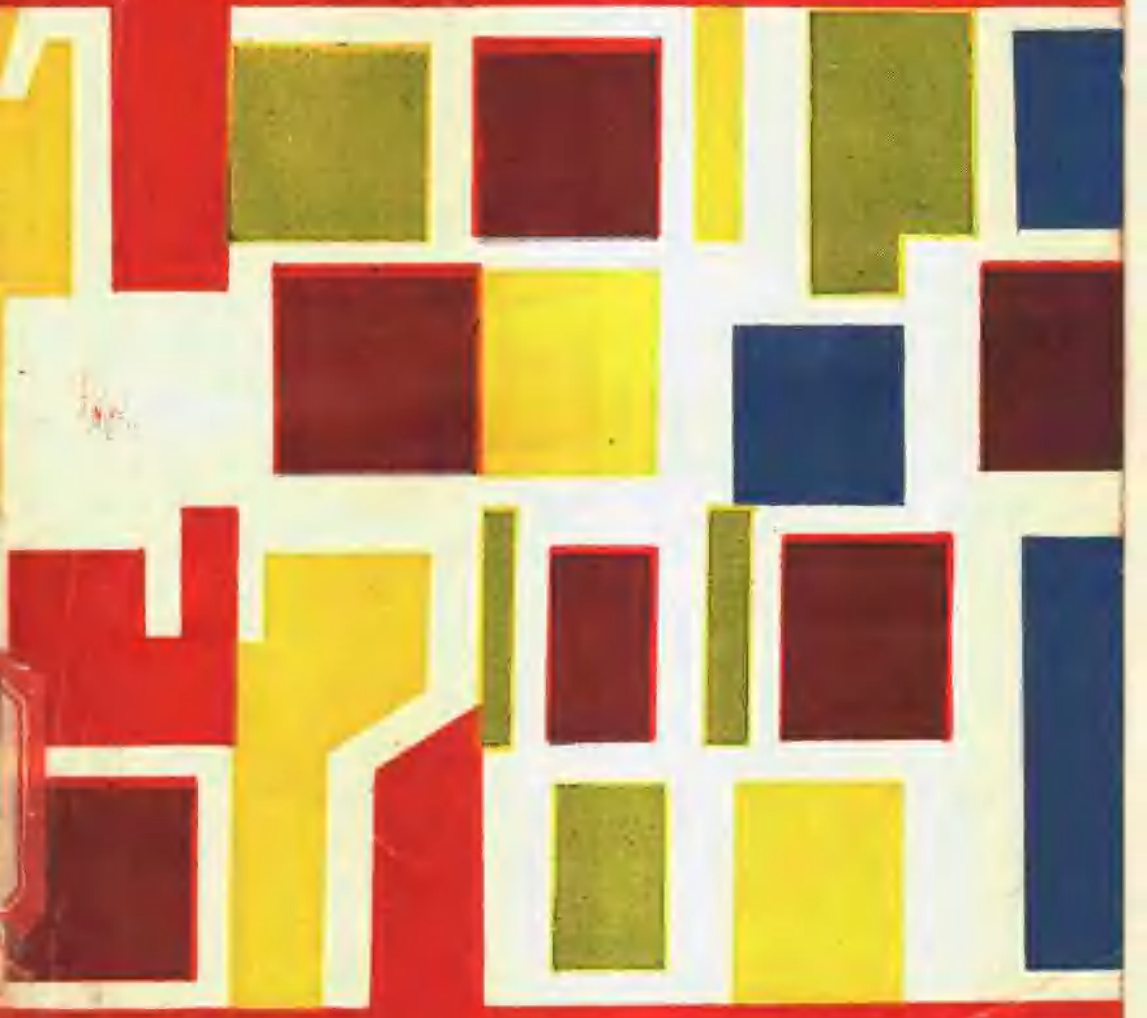
THE ART OF
COMPUTER PROGRAMMING

BASIC CONCEPTS

電腦與計算機



DONALD E. KNUTH 原著　　任清華譯

電腦與計算機

DONALD E. KNUTH 原著
任 清 華 譯

亞洲出版社印行

中華民國六十一年四月一日出版
出版登記證內版臺業字第〇五六二號

電腦與計算機

平裝特價：新台幣 [illegible] 元
精裝特價：新台幣 [illegible] 元

原著者 DONALD E. KNUTH
譯著者 任清華
發行人 任清華 住址：台北市潮州街八十五號
出版者 五洲出版社
總經銷 五洲出版社圖書有限公司 地址：臺北市重慶南路一段八十八號 郵政劃撥賬戶：二五三八號 電話：三三九六三〇號
經銷處 全省各大書局

前　言

這是您的書，您們數千人所要求出版的書。本書的編成耗去我們數年的時間，經過校對再校對，而帶給您唯一最好，最有趣，且最完整的書。可以說，無瑕疵可言，若您遵從本書所指示的方向，它將能爲您盡最大的服務，即使您以前從未學過。

二進位計算機的準備程序是很吸引人的，不僅因爲經濟和科學效能，並且類似討或音樂的組成，是一種耗時的經驗。本書是曾經用來訓練讀者熟練程序技巧的七册書中的前五册。

下面幾章並非對計算機程序的介紹，我們假定讀者已有這些經驗。所需要的實際上非常簡單，但在完全了解二進位計算機的觀念前，必須有更多的時間和實習。讀者應具有。

a）一些貯藏程序二進位計算機如何工作的觀念，並不需要完全電學的觀念，但應了解資料能保存在機器的記憶裏且能成功地操作，一些機械語的了解將是有幫助的。

b）具有能將問題解答以計算機所能了解的型式置入（機器沒有普通常識，亦不會思想，但能不多不少的做出所被告訴的，此事實是初學者最難抓住的觀念。）

c）大部分初級計算機程序的知識，像環（*looping*）（能重複執行一組命令），重複工作之使用，還有索引登記的使用。

d）一些關於普通計算機的隱語，例如"*memory*","*registers*","*bits*","*floating point*","*overflow*"等，大部分書中沒有定義

的字，皆在每册末之索引內有簡短的定義。

以上四點可歸納成一點，即是讀者至少已經學過或嘗試過四個程序。

我曾經試着以能適用幾種需要的方式來寫此書。第一點，此書是收集很多重要方面的知識的參考書。本書可用來作爲自讀的教科書，或是大學課程中計算機課程的教科書。爲要達成這些目的，我加入很多作業，並且對其中大部分皆附有答案，我盡最大的努力將此書以一般譯註充滿。

這組書是爲對計算機有興趣的人所寫的，但並非爲計算機專家。事實上，主要的目標是使這些計算機技術更易爲一般人所了解，且能利用計算機，還有那些不能夠有足夠時間瞭解技術上所必需的知識。

本書可被稱爲“非數目分析”雖然計算機傳統上與數值解，例如方程式的根，數值插入，積分等有關，像這些方式皆以通過形式處理。數值計算機程序是非常有趣且發展很快的部門，有很多書談論到它。最近幾年，有很多有趣的工作是用計算機作非數值題目，像分類，翻譯語言，解決高等數學題目和綜合分析。“*software*”（分開其它程序的程式）還有日常生活不同程序的模仿。數目通常發生在像符合等，和計算機決定能力而非作數學的能力上。在非數值題目中，吾人用加法和乘法，但不常需乘和除。注意，甚至一個初與數值計算機發生關係的人，亦可從學習非數值技術中得到益處，因這些在數值程序中亦呈現出來。

最近在非數值分析方面，分支成很多方面，同時其寫法也成爲混亂和無組織。學習這些基礎技術的用法是較迫切的，觀念上可以運用到很多型式的程序場合上，我曾嘗試將這些歸納成理論，並提供讀者這方面的理論，這些對於 *software* 程序技術上的運用亦列出。

當然，「非數值分析」，在學習這方面是可怕的陰性名詞，應該有較好，且較陽性的術語以描述此名詞。「消息程序」用來表示所指

的物質是太廣泛，而「程序技術」又太狹隘。因此，我以「電腦分析」（*analysis of algorithms*），作爲本書所涉及主題的名字，在本書中有詳細解釋，此名詞亦指「某種特殊計算機性質理論」。

要趕上在經濟方面有利，通常是困難的，因此期望這裏所提到的很多技術較其它者好是很自然的事。當然對我們而言，趕在前頭兩年是不可能的事，而前面所提到的亦會改變。在這方面，我混合我的情緒，因爲我確實希望這組書會有更深的研究，而不變爲陳腐。

實際上，此地所提到的電腦在五年以前即被一些不同的人所有，因此這些理論皆以成熟且是最佳的形式。故將其置入本書中，希望同學們能努力研習它。

DONALD E. KNUTH *California Institute of Technology*

目　次

電腦與計算機

許多不熟悉數字的人認爲既然 [*Babbage's Analytical Engine*] 能以數字表示結果，那麼它的過程必定是具算術性及數字性而非代數性和分析性。其實不然，此機器所排列組合出來的數量與字母或其他代替符號所表示的一樣，事實上只要條件定義它也可能產生代數記號。

—*ADA AUGUSTA, Countess of Lovelace*(1844)

本文中的"電子計算機"及"數字計算機"（*digitall computer*）都稱爲"資料組合器"（*Data Processor*）。

—1977 *disgital computer* 的錯誤更正表

1.1 算 法
(*ALGORITHMS*)

Al 是所有電子計算機程式的基礎現在我們仔細地分析此觀念。

這字本身非常有趣祇把"對數"的前四個字母搞亂，1957年的韋氏新字辭典中，找不到這字我們只能找出 *algorism* 的意義，即使用阿拉伯數字做算術，中世紀時使用算盤和算術計算。以後這個字的本意不清起初語言學家認爲是 *algiros*（*painful*）與算術（數字）的結合。有人認爲這是出自"凱斯提爾王愛爾格"，終於數學家肯定它出自阿拉伯一位極出名的教本作者，住在今天俄國的山城 *Khiva* 他寫了一本有關還原與減法的書，*algebra* 這字是來自他所出版的書名，但

是這本書並不是全部談代數。

漸漸地 *algorion* 字的意義有了偏差，牛津英文字典中把它與算術混淆在一起，從 *algoriom* 改爲 *algorithm* 是因爲人們已忘記了它最原始的意義，德國早期數學字典 *Vollstandiges Mathematisches Lexicon* (*Leipzig*, 1747) 給 *Algorithmus* 下個定義：把數字計算之四種型加減乘除連合起來做一種設計。

1950年以前 *algorithm* 常與 *Euclid's algorithm* 連在一起找出最大公約數，*Euclid's hlements*（書名）現在參看 *Euclid's algorithm*。

Algorithm E. 二正整數 m,n 找出最大公約數，即最大的正整數可以同時除盡 m,n。

E 1. [找出除數] m 被 n 除，r 爲除數（$0 \leq r < n$）。

E 2. [是否爲0] 如 $r = 0$ 則 n 爲其解。

E 3. [互換] 把 m n 互換 r 互換回重做 **E** 1。

當然，尤克利並不是如此解說 *algorithm* 以後就同一型式列出。

每個 *algorith* 都以一個字母表示，步驟以一個數字表示各章以數字分成若干部門，每部門以字母命名，例如在 1.1；尤克利 *algo* 稱爲 *Algorithm E* 以後各部門稱爲 *Algo* 1.1 *E*。

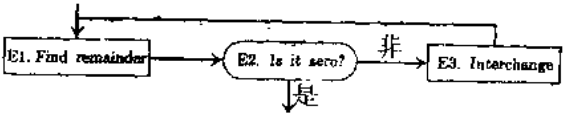


圖 1

algo 之過程以括弧內簡單註語開頭通常附有流程圖以使讀者更能了解。

接着以字句圖 1 或一些行動說明有時也有註解（**E** 1. 的第二句即是）包括這步驟的解釋，變數的性質或這個步驟的目的，註解只供讀

者了解並不顯示下個步驟。

記號“←”意謂代替“$m \leftarrow n$”即m之值換成n值*algo E*開始時m n都是原來的數值，符號是用來區別等號，我們不說“$m = n$”但却可能問m是否等於n“=”是可以測量的，←是個可行的改變n增加1以“$n \leftarrow n + 1$”表示（讀做n以$n + 1$代替之）“變數←公式”意謂公式用現在任一變數計算之，其結果代替變數的原數值放在箭號左方，不熟悉電腦工作者常會把“$n \rightarrow n + 1$”說成“n變成$n + 1$”這是應當避免的因與標準設想不符。

由*E* 3我們已注意到次序很重要設“$m \leftarrow n$，$n \leftarrow r$”與“設$n \leftarrow r$”D有限，*algo*經有必在限序步驟以後解出*Al E*滿足這個條件，因*E* 1步驟後r少於n如果$r \neq 0$，下次*E* 1時n值將減少一遞減的正整數數列終有極限，所以*E* 1步驟事實上是有限次，但步驟次數也可能很大如m，n極大*E* 1會有上萬次。

次了無限外所有的*algo*性質稱爲“計算機方法”尤克利除發明求最大公約數的方法外，他還有一套幾何理論，對於*algo E*有同等重要性，這套理論用於求得兩線後最大相同度量，如果已知長度爲確定的則此法適用。

2）確定性，一個*algo*的每個步驟必須是確定的，每個變換必需是精確而不含糊的，本書所述可以做到這點，但由於它的特殊因而使得讀者可能不易了解，爲了破除這個困難，就產生了程序語言又稱爲電腦語言嚴格加以定義。本書中許多*algo*均以英文和電腦語言述及電腦語言中計算的方法稱之爲“程式”。

Algo E 中*E* 1步驟以n除m即含有確定性，事實上，如果m與n不爲正整數我們沒有一致判斷，以$-\pi$除-8餘數如何？以0來研究它否則你就會東西南北都弄不清楚它到底是什麼，假如我們用筆計算，讀者會很容易了解，因而其他方法大半不會成功。

現在做一題*algo E*的例題，設$m = 119$，$n = 544$，第一步*E* 1

（讀者應以 n 除 m 商爲 0 而餘數爲 119，$r \leftarrow 119$，第二步 E 2 因 $r \neq 0$ 沒有任何改變，第三步 E 3 設 $m \leftarrow 544$，$n \leftarrow 119$，如果原來 $m < n$ 第一步定得 0 其後必交換 m，n，我們可以再增加一個步驟。

E 0 [須 $m \geq n$] 如 $m < n$，交換 $m \leftrightarrow n$

回到 E 1 $\frac{544}{119} = 4\ \frac{68}{119}$，$r \leftarrow 68$，$E$ 2 不適用，E 3 $m \leftarrow 119$ $n \leftarrow 68$ 第二次設 $r \leftarrow 51$，$m \leftarrow 68$，$n \leftarrow 51$，再一次則 $r \leftarrow 17$，$m \leftarrow 51$，$n \leftarrow 17$，最後 51 被 17 除，$r \leftarrow 0$ 故在 E 2 得到答案 119 和 544 的最大公約數爲 17。

Algo 的意義與過程方法，技術，手續，規律相似，只不過有一點點不同而已，它除了有一些規則外還有五個重要性質，“$m \leftarrow n$”不同，因後者謂 n 之原值在設 m 之前已喪失。因此後者等於“設 $n \leftarrow r$，$m \leftarrow r$”假若我們設好幾個變數具有相同的數量時，可以使用複符號“$n \leftarrow r$，$m \leftarrow r$”可寫成“$n \leftarrow m \leftarrow r$”兩變數之值互換寫成“互換 $m \leftrightarrow n$”；也可設一新變數 t 寫作“設 $t \leftarrow m$，$m \leftarrow n$，$n \leftarrow t$”。

Al 以最低數起頭，通常爲 1，除非另有定義都是連續數，自 E 3 “回到 E 1.”是顯示了計算的順序，E 2 須以 $r = 0$ 爲條件；“如 $r \neq 0$，回復 E 3”。

“■”出現於 E 3 之最後表示一個 *Algo* 之結束與本文之。

以上我們討論的是本書中的符號，此外還有一個符號，當我們述說 n 個數量，$v_1, v_2 \cdots\cdots v_n$ 第 j 個量不寫 v_j 而用 $v[j]$ 代替同樣的 $a[i,j]$ 代替 a_{ij} 某些字也用來表示一個變數的值例如：$TEMP$ 而 $PRIME[K]$ 表示第 n 個 *prime* 數。

讓我們做一個 *Algo* 的實例切記不可以用讀小說的心情除 59/13？因此利用 E 1 步驟我們須確定 m，n 爲正整數，如此 E 1 的餘數是

個非負整數，且非0才能進行第三步 $E3$ 。

3）內含數：一個 *algo* 有許多個或沒有內含數，所謂內含數即從某集合而來在 *algo* 前就有的例如：*Algo E* 中 m ，n 都定義於正整數。

4）外含數：一個 *algo* 有一個或許多個外含數與內含數有關。

Algo E 有一個外含數第二步 $E2$ 的 n 是兩個內含數的最大公約數。

吾人極易證明此數之最大公約數如下：第一步 $E1$ 後，

$$m = qu + r$$

q 爲整數，$q \in I$ ，若 $r = 0$ ，m 是 n 三倍數，則 n 是 m 和 n 的最大公約數，若 $r \neq 0$ ，m 及 n 之任一公約數必須除盡 $m - qn = r$ ；任何 n 與 r 之公約數必須除盡 $qn + r = m$ ；m 與 n 的公約數所成的集合必與 n 與 r 公約數所成的集合相同，因此第三步 $E3$ 對於原問題答案不變。

5）有效性：所有過程須爲基本的，在一定時間內可做完 *AlgoE* 只是用一正整數除另一個正整數試“看是否爲0”，同時設一變數等於另一變數值，此過程，整數可以用特定形式表現於紙上，且至少有一方法（除法 *algo*）互相除，但若數值爲實數且爲無限的或是物理線段就不能適用，“若2是最大整數 n ，方程式 $x^n + y^n = z^n$ ”有解 x y z 皆正整數再依第4步 $E4$ ”除非有人證明有一 *algo* 可以決定是否2爲最大整數，否則這句話不能算眞。

讓我們把 *algo* 的概念與食譜比較一下：每個製法具有限性（雖說看着的茶壺不會沸）內含數（蛋、麵粉等）外含數（*TV* 晚餐等）但缺乏確定性通常它會沒有碼定性“加一匙鹽”“一匙定義成”少於 $\frac{1}{8}$ “茶匙”鹽可能可以確定多少但應加在那裏（頂上，旁邊）？又如

"輕輕放入直等溶和""這類指示"，對的有訓練的廚師或許夠了，但*algo*須精確到電腦可以依著指示操作，但學習電腦程式仍可像研習食譜一般。

有限性不能實際、應用，一個有用的*algo*應不僅需要有限次的步驟而且需要一個非常有限的合理的數目，例如：一個*algo*被用來決定是否這盤棋是對白棋而言被迫勝利，我們始終不會知道它的答案，因必須花費很多時間，參看8.3節分閱有限數字大到難以理解的程度。

我們需要的是"好"*Algo*也就是費時不太長，電腦運算起來簡單又精確等等。如果一個問題有許多計算方法我們可以挑選最好的一種*algo*分析就是要決定問題的性質，例如："假設n值已知，但m可以是所有正整數*algo* E的E 1平均次數，Tn是多少？"首先，先要知道這問題眞有答案（因要求無限m的平均數）和這有關的是以n除m設的餘數從$m=1$，$m=2\cdots\cdots m=n$計算E 1步驟的總次數有多少。

決定Tn的性質是很重要的，等於$\frac{1}{3}n$，或$\sqrt{n}$等？（事實上很困難4.5節會詳細談到n極大則Tn近似（12 $ln\,2/\pi^2$）$ln\,n$其他請參看4.5節。

"*algo*的分析"（*Analyois of algo*）是作家喜用的名詞，討論其一般性與適應性："*algo*理論"（*theory of algo*）又是另外一回事，討論*algo*是否存在，本書11章中會大略研討一番。

我們定義一個計算方法有關四個元素的向量（Q，I，Ω，f），Q爲一集合包含I和Ω，f是自Q映至它本身的一個函數，Ω映射固定，即對所有Ω中原系q而言$f(q)$應等於q，Q，I，Ω，f，代表運算的狀態，*input output*和計算規則I集合中每個*input* x定義一個計算數列$x_0, x_1, x_2, \cdots\cdots$如下：

$$x_0=x \quad 且 \quad x_{k+1}=f(x_k) \qquad k \geq 0 \tag{1}$$

如 k 是使 x_k 在 Ω 內的最小整數的話此數列稱爲在 k 步驟終止，且稱 x 產生出 x_k。（注意，如 x_k 在 Ω 內，x_{k+1} 亦然，因 $x_{k+1}=x_k$）有些數列不會終止，*algo* 是一種計算方法在有限步驟後必終止。

Algo E 可能依下列條件形成：讓 Q 爲 (n)，序對 (m,n)，及有序四數 $(m,n,r,1)$，$(m,n,r,2)$，$(m,n,p,3)$ 所成集合，m，n，p 是正整數，r 爲非負整數。令 I 爲所有序對 (m,n) 的部分集合，Ω 是所有 (n) 的部分集合，令 f 定義如下：

$$\begin{aligned}
&f(m,n)=(m,n,0,1);\quad f(n)=(n);\\
&f(m,n,r,1)=(m,n,m\div n\text{的餘數},2);\\
&\text{如 } r=0\text{，則 } f(m,n,r,2)=(n)\text{ 否則爲 } (m,n,r,3)\\
&f(m,n,p,3)=(n,p,p,1)
\end{aligned}\tag{2}$$

由上可看出這個定義與 *algo E* 的關係。

上例並不合乎“有效性”的原則，例如：Q 可能爲此限數列，或 f 可能需較複雜的手續才能解出，如我們要簡化它，則可限制 Q，I，Ω，f，例如令 A 爲有限字母所成的集合，A^* 爲 A 的數列所成集合（有序數列 $a_1a_2\cdots\cdots a_n$，$n\geq 0$，a_j 爲 A 中一元素，$1\leq j\leq n$），其目的在於把運算以 A^* 代替。令 N 爲非負整數，令 I 爲 Q 的部分集合，$j=0$，再令 Ω 爲 $j=N$ 的部分集合。如果 θ 和 σ 爲 A^* 的數列，對序列 α 和 ω 言，σ 具 $\alpha\theta\omega$ 的形式則稱 θ 於 σ 內產生，接着令 f 爲下列形式的函數定義，序列 θ_j，ϕ_j 和整數 a_j，b_j，$0\leq j<N$：

$$\begin{aligned}
&f(\sigma,j)=(\sigma,a_j)\quad \text{如 } \theta_j \text{ 不產生於 } \sigma;\\
&f(\sigma,j)=(\alpha\phi_j\omega,b_j)\ \text{如 } \alpha \text{ 是 } \sigma=\alpha\theta_j\omega \text{ 的最短序列}\\
&f(\sigma,N)=(\sigma,N)
\end{aligned}\tag{3}$$

這樣的計算方法具有“有效性”可以解答許多問題，當然也有許多另外可以說明其他計算法（例如：*Turing* 機器）*A. A. Markov*

所著“ *Algo* 理論”（ *The Theory of Algorithms* ）中亦述及此書譯自 *J. J. Schorr-kon*，由美國商務部技術協助，書號 *OTS* 60-51085 發行。

習　題

1. [10] 利用數列交換把（a, b, c, d）變換成（b, c, d, a），亦即把原來的 b 變成 a 等等，並儘量採取最少步驟。

2. [15] 證明 E 1 中 m 代大於 n，除了起初可能有例外。

3. [20] 爲了使 E 3 不改變數值 n 立刻可被 r 除，試改變 *Algo* E，m 爲約數，增加步驟以免三數同時改變新的 *Algo* 以 *Algo* E 的形式表示，稱爲 *Algo* F。

4. [16] 2166 與 6099 的最大公約數？

▶ 5. [12] 說明前面所述讀此書的程序，不滿足我們說述五個條件中的三個並說出它與 *Algo* E 不同處。

6. [20] 這節末尾的 T_n 是什麼？

▶ 7. [*M* 21] 假設 m 已知，n 爲正整數，令 U_n 爲 *Algo* E 內 E 1 的平均次數，定義 U_n，U_m 與 T_m 有關連嗎？

8. [*M* 25] 爲計算正整數 m 與 n 的最大公約數列出一可行的 *algo Eqs*(3) 中引用 $\theta_j, \phi_j, a_j, b_j$ 四數，以 $a^m b^n$ 代表存入（*input*），亦即 nb's 接在 ma's 的後面，答案儘量簡短［提示：利用算法 E，E 1 中，令 $r \leftarrow |m-n|$，$n \leftarrow min(m, n)$］。

▶ 9. [*M* 30] 假設 $C_1 = (Q_1, I_1, \Omega_1, f_1)$，$C_2 = (Q_2, I_2, \Omega_2, f_2)$ 是一種計算方法，C_1 可能表示 *Eqs* 中的算法 E，m，n 的大小受限制，C_2 可能表示算法依一種電腦程序，（Q_1 可能表示這部機器所有計算的集合，亦即所有可能的記憶與記錄 ϵ_2 可能定義單一計算；I_1 可能是包括決定最大公約數的程式）。

下一個集合理論定義“C_2代表C_1”，意謂所有C_1的計算數列均適合C_2，除非C_2需要更多的步驟才能算出（由此我們可說“X代表算法Y的）。

1.2 數 學
(PRELIMINARIES)

此節我們將討論本文中提到的數學名詞。如對若干程式不明，反則亦不能明瞭更深的理論。

數學定義有二目的：

①描述算法的基礎；②分析運算的特性；③用來描述算法的定義很簡單前節已述，如要分析運算則需更多定義。

本書中大多算法都附有數學運算以便迅速求得結果，大多數的算法都可用大學代數求之，讀者自能充分了解，有些地方須用 的變數理論、群論、數字論、或然率等等。

注意事項：許多讀者不了解到底資料與電腦程式有何關連，（除了1.2.1）有的讀者或許會簡略讀過後又回頭來讀前面之節次。

1.2.1 數學歸納
(Mathematical induction)

令$P(n)$爲n的函數，$P(n)$可能是“n乘$(n+3)$爲偶數”或“若$n \geq 10$，則$2^n > n^3$”假如須證明對所有正整數n，$P(n)$是眞。

a) 證明$P(1)$爲眞；

b) 證明：對所有正整數而言“若$P(1), P(2), \cdots\cdots P(n)$爲眞，則$P(n+1)$爲眞，例如：

$$1=1^2,\ 1+3=2^2,\ 1+3+5=3^2,\ 1+3+5+7=4^2,$$
$$1+3+5+7+9=5^2 \tag{1}$$

設成方程式即：

$$1+3+\cdots\cdots+(2n-1)=n^2 \tag{2}$$

令這方程式爲$P(n)$，須證明$P(n)$對所有正整數n皆眞。

a) "$P(1)$爲眞，因$1=1^2$"

b) "若$P(1)$，……，$P(n)$皆眞，則$P(n)$爲眞，故*Eq.*(2) 爲眞，兩邊各加$2n+1$則：

$$1+3+\cdots\cdots+(2n-1)+(2n+1)=n^2+2n+1$$
$$=(n+1)^2$$

即$P(n+1)$爲眞"。

這種方法稱爲計算證明過程（*algorithmic proof procedure*）事實上，以下$P(n)$所證明均假設(*a*)與(*b*)已知：

算法*I*（構成證明），正整數n證明$P(n)$爲眞。

*I*1. [證明$P(1)$]令$k\leftarrow 1$，根據(*a*)證明$P(1)$。

*I*2. [$k=n$？]如$k=n$，結果。

*I*3. [證明$P(k+1)$]根據(*b*)證明"如$P(1)$，……$P(k)$爲點，則$P(k+1)$爲眞""已知$P(1)$，……$P(k)$爲眞，故$P(k+1)$爲眞"。

*I*4. [增加k]k增加1，做*I*2。

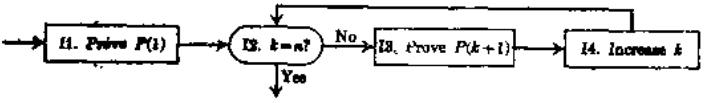


圖2　算法*I*：數學歸納

數學歸納法應與歸納推理（*inductive reasoning*）分開，科學