信息科学与技术丛书 程序设计系列

程序算法与技巧精选

郭继展　郭勇　苏辉　编著

- 巧算π值 10000 位
- 哥德巴赫猜想的筛法验证
- 统计法排序——百万整数只需 1 秒
- 计算机辅助数学证明——3 个数学黑洞
- 辅助解智力测验题——12 只球中挑坏球
- 应用程序增加智能——显示解题过程和精确解

机械工业出版社
CHINA MACHINE PRESS

TP301.6/87
2008

信息科学与技术丛书·程序设计系列

程序算法与技巧精选

郭继展　郭勇　苏辉　编著

机械工业出版社

计算机科学是算法的科学。进行程序设计不仅需要掌握常用的算法、技术和方法，还要敢于创新、构思巧妙的算法和探索编程中的诸多技巧。

算法和技巧都需要学习、借鉴和交流。本书分17章，139个例题。书中介绍的算法和技巧涉及到随机数函数理论，基础数论，新意幻方，提高程序运行速度和精度，特定数据排序，穷举、递推、递归和迭代等诸多方面。这些算法和技巧大多是作者历年从事教学、软件开发、学术研究和学习的成果总结。

本书内容不涉及计算机专业课程的诸多概念、理论，读者只需要学过C语言，有算法、结构化程序设计和逻辑表达式的概念，并有独立上机编制30条左右语句小程序的经验，就能够掌握书中的程序设计思想、算法和技巧，并能举一反三，推广应用，使自己的编程水平上一个台阶。

本书可作为大专院校师生和计算机编程人员或自学人员参考。

图书在版编目（CIP）数据

程序算法与技巧精选/郭继展，郭勇，苏辉编著. —北京：机械工业出版社，2008.4
信息科学与技术丛书·程序设计系列
ISBN 978-7-111-23816-4

Ⅰ.程… Ⅱ.①郭…②郭…③苏… Ⅲ.算法程序－程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2008）第041793号

机械工业出版社（北京市百万庄大街22号 邮政编码 100037）
策　　划：丁　诚
责任编辑：丁　诚
责任印制：李　妍
保定市中画美凯印刷有限公司印刷
2008年5月第1版·第1次印刷
184mm×260mm·21印张·515千字
0001—5000册
标准书号：ISBN 978-7-111-23816-4
定价：36.00元

凡购本图书，如有缺页、倒页、脱页，由本社发行部调换
销售服务热线电话：（010）68326294
购书热线电话：（010）88379639　88379641　88379643
编辑热线电话：（010）88379753　88379739
封面无防伪标均为盗版

出 版 说 明

随着信息科学与技术的迅速发展，人类每时每刻都会面对层出不穷的新技术、新概念。毫无疑问，在节奏越来越快的工作和生活中，人们需要通过阅读和学习大量信息丰富、具备实践指导意义的图书，来获取新知识和新技能，从而不断提高自身素质，紧跟信息化时代发展的步伐。

众所周知，在计算机硬件方面，高性价比的解决方案和新型技术的应用一直备受青睐；在软件技术方面，随着计算机软件的规模和复杂性与日俱增，软件技术受到不断挑战，人们一直在为寻求更先进的软件技术而奋斗不止。目前，计算机在社会生活中日益普及，随着因特网延伸到人类世界的层层面面，掌握计算机网络技术和理论已成为大众的文化需求。由于信息科学与技术在电工、电子、通信、工业控制、智能建筑、工业产品设计与制造等专业领域中已经得到充分、广泛的应用，所以这些专业领域中的研究人员和工程技术人员越来越迫切需要汲取自身领域信息化所带来的新理念和新方法。

针对人们对了解和掌握新知识、新技能的热切期待，以及由此促成的人们对语言简洁、内容充实、融合实践经验的图书迫切需要的现状，机械工业出版社适时推出了“信息科学与技术丛书”。这套丛书涉及计算机软件、硬件、网络、工程应用等内容，注重理论与实践相结合，内容实用，层次分明，语言流畅，是信息科学与技术领域专业人员不可或缺的图书。

现今，信息科学与技术的发展可谓一日千里，机械工业出版社欢迎从事信息技术方面工作的科研人员、工程技术人员积极参与我们的工作，为推进我国的信息化建设作出贡献。

机械工业出版社

前　言

计算机科学是算法的科学。没有算法，人编不出程序，计算机也就无能为力。早年瑞士联帮技术学院沃思博士提出了一个著名的公式："程序＝算法＋数据结构"，后来我国计算机教育专家谭浩强教授将公式完善为：

程序＝算法＋数据结构＋程序设计方法＋语言工具和环境

所以，进行程序设计不仅要能根据具体问题选择数据结构、设计算法，还要熟练掌握方法（如：结构化程序设计方法，软件工程的原理和方法），熟悉语言工具和环境。初学者要想成功地进行程序设计，不仅要注意培养这四方面的重要技能，还要敢于创新、构思巧妙的算法，探索编程中的诸多技巧。程序设计是创造性的劳动，遇到的实际问题往往是错综复杂的。当使用我们熟悉的技术解决不了问题的时候，就要尝试某些技巧，有时小技巧能解决大问题。算法和技术需要学习，技巧也需要学习，还需要自己探索。发现一个技巧就等于完成一次创新。某位大师说过："一个技巧用过两次，就成为一门技术。"

一个成功的算法，应有较高的运行速度、令人满意的精度和良好的清晰度，其间蕴涵着诸多技巧和设计者的心血。有些问题算法是简单的，手算人人都会，但编程实现却很难。例如：求100位数的四则运算，求线性方程组的精确解等。程序设计人员，应当把自己对算法和技巧的研究成果整理出来，进行交流。本书所编程序设计算法和技巧不是常用算法和技巧集锦，主要是介绍作者"自己的东西"，或是提出的新问题和解法，或是传统问题算法有创新，或是编程有技巧。例如排序，提出特殊数据的无比较排序法——统计法、记录无交换排序法——标序法以及求万位的π值等，都是本书推出的新算法，探讨的新技巧；又如巧妙运用数组，将程序的运行速度提高百倍、千倍。

在多年教学中我们发现，许多在校大学生虽然学过计算机基础和一门高级语言程序设计（如C语言），但是还没有深刻领会到编写程序的真谛。在进行毕业设计时，他们把主要精力花费在计算机程序语言的学习和程序设计上，把毕业设计变成了名副其实的"程序设计"。尤其在编程遇到难关时，更不会改进算法，探索技巧，逾关而过。当我们有针对性地把书中的几个算法、技巧介绍给他们，时深受启发，能举一反三，成功完成毕业设计。因此我们也就不断体会到算法和技巧在程序设计中的重要性，并认为大学生们需要这方面的书籍。

本书分17章，有139个例题。针对的都是人们十分熟悉的问题或手工运算不可能完成的问题，如筛法选素数、称12只球、数学黑洞证明等，将它们变成计算机问题，求得答案。书中的题目均不涉及计算机专业课程（如图论、数据结构、计算方法等）的诸多概念和理论，而是属于程序设计基本算法与技巧的提高和深入。读者只需要学过一门高级程序设计语言，有算法、结构化程序设计和逻辑表达式的概念，并有独立上机编制30条左右语句小程序的经验，就能够掌握书中的程序设计思想、算法、程序和技巧，并能举一反三，推广应用，使编程水平明显提高。

本书是一本通过C语言（C语言实现不了的用了VB或汇编语言）介绍程序设计特殊问题算法和技巧的书籍。书中的算法和技巧适合任何一种程序设计语言，具有普遍性。书中近半数的程序，就是由其他语言，如Basic、Pascal、VB、VC等移植过来的，统一到一种语言即C语言之下。因为C语言是近年来国内外得到迅速推广应用的第一流语言，是各种大型

此为试读，需要完整PDF请访问：www.ertongbook.com

机、小型机和微机上普遍使用的通用程序设计语言，并且多数用户是熟悉C语言的。

本书是作者在多年校内外教学、软件开发、科学研究和指导毕业设计积累的经验和成果的基础上，参考了多种资料后编写的。书中的题目很多都做了数学上的分析，不少题目给出了两种以上的算法。选题大多来源于作者的学术论文、著作、教案、科研成果，有些是近期的新作。选题范围很广，涉及到随机数函数理论，基础数论，新意幻方，提高程序的运行速度、运算精度，特定数据排序，获得计算机的几种编码，穷举、递推、递归和迭代，使应用程序增加智能，用计算机进行逻辑推理、解智力测验题、辅助数学证明，以及警惕计算机犯错误等。其中不乏名题、趣题、妙题，有些算法和技巧也许会令人喜出望外。

本书具有较高的实用价值，既可作为大专院校编程算法和C语言等多种课程的辅助教材，又可供计算机编程人员、自学人员学习和参考。

在本书编写过程中，炮兵指挥学院计算机教研室的同志们对本书提出了许多宝贵意见，给予了大力帮助和支持，在此向他们表示衷心感谢。由于编者水平有限、时间仓促，缺点错误在所难免，敬请专家、读者批评指正。

编　者

目　录

第1章 算法——程序的灵魂

算法，是程序设计的灵魂。没有算法，就编不出程序。设计算法是软件工作者的重要本领，一名合格的软件工作者既要深刻理解、熟练掌握常用的基本算法，又要敢于创新、构思巧妙的算法。

计算机的发展是靠“两条腿”走路的，一条是硬件，一条是软件，但二者很不协调。计算机诞生后的60年里，硬件的速度已提高了约10亿倍，成本也降低了约10亿倍，但软件的发展却相对缓慢，其重要的原因就是对算法的研究不足、重视不够或算法难度大。

算法的研究不仅仅是为了能用计算机解决具体问题，它往往能够开辟科学的新天地。能解决别人提出的问题是水平，能提出别人解决不了或某段时间内解决不了的问题也是水平。正是在寻求算法的过程中，诞生了许多副产品（有的更加珍贵），或延伸了学科前沿，或开辟了新的学科，或使学科实现交叉。有的科学家因此而改行，在新的领域里做出了巨大贡献。如哥德巴赫猜想、费马定理等问题的研究，推动了现代数论的建立和发展；大素数已用于计算机加密领域，如公开密钥、加密密钥、数字签名等。

1.1 计算机科学是算法的科学

1. 什么是算法

算法是对特定问题求解步骤的一种描述，包含操作的有限规则和操作的有限序列。

通俗一点讲，算法就是一个解决问题的公式（数学手册上的公式都是经典算法）、规则、思路、方法和步骤。算法可以用自然语言描述，也可以用流程图描述，但最终要用计算机语言编程，上机实现。

有些问题算法是简单的，手算人人都会，但编程上机实现却很难（本书将对这一问题做较多的讨论）。例如求100位数的四则运算，求线性方程组的精确解等。一个成功的算法，应该有良好的速度、精度和清晰度。

人没有算法，计算机也无能为力。算法错了，计算机将误入歧途。20世纪初的数学家希尔伯特提出了23个问题。一个世纪过去了，有的问题还没有解决，就是因为数学大师们仍找不到一个合适的算法。从这一点上讲，计算机的理论基础是数学。

2. 好的算法应具备的属性

（1）确定性。算法应当满足具体问题的需求，正确反映问题对输入、输出和加工处理等方面的要求。每一条指令必须有确定的含义和作用，对于相同的输入必须得出相同的执行结果。

（2）有限性。算法中每条指令的执行次数是有限的，执行时间也是有限的。算法执行有限步后必然结束，不能要求无休止地执行下去。如近似计算，当达到要求的精度或要求的计算项数时，即可停止。

此为试读，需要完整PDF请访问：www.ertongbook.com

(3) 可行性。算法的每一个步骤或动作都是可以实行的，最后应能达到预期的目的。不能做违背科学原理的事，如用 0 作除数。还有一点要注意，算法具有可行性，但程序运行不一定有可行性。如运行所需时间要以日、月、年计的问题。

(4) 有零个或多个输入。输入多指程序运行时从键盘输入数据，这样程序才有通用性和灵活性，实现人－机交互。零个输入指程序本身已经具有了运行所需要的数据。

(5) 有一个或多个输出。解只有输出才能得到。算法的目的是求问题的解，解的形式可以是多种多样的，如文字、数字、图形、声音、动画等。

(6) 健壮性。这是系统（算法转化为程序）在异常和危险情况下生存的关键，当输入非法数据时，算法应能适当地作出反应或进行处理，输出表示错误性质的信息，不死机、不崩溃。

(7) 可读性。算法除了用于编程，在计算机上执行之外，另一个重要作用是阅读与交流。高可读性的算法有助于人们对算法的理解，便于交流和推广，以及软件的维护。

(8) 经济性（时间效率和存储占用量）。程序执行时间短的算法效率高，占用存储空间少的算法好。

1.2 算法具有多样性

解决同一问题的算法不是唯一的，可能有多种。它们的思路截然不同，有的很“笨”，有的很“灵”。在选择或设计上不能保守，要采用和研究最先进、最高效的算法。

【例 1-1】 求圆周率 π 有多种算法，如：

(1) 几何方法。有割圆法。刘徽、祖冲之相继割圆，树立了求 π 的里程碑。还有方格法。如图 1-1 所示，圆内的小方格的总数（不足一格者“估凑”一格）与正方形内小方格总数之比乘以 4，为 π 的近似值，当然可仅取$\frac{1}{4}$圆计算。如果圆半径 r＝1000 毫米，格数计数误差不超过 10 个的话，π 的值可达 5 位有效数字。

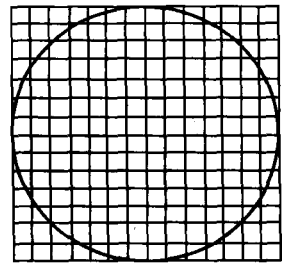

图 1-1

(2) 物理方法。有称量法。制作一块质地均匀的、等厚的正方形木板（或用其他材料），称出其重量 P_1，计算出其面积 S_1。画正方形的内切圆，将圆外部分垂直地锯、锉掉，称出其重量 P_2，可按正比例关系计算出 $\pi \approx S_1 \times \frac{P_2}{P_1}$。

还有排液法。选一种不渗水的等厚材料，制作出上述的“圆图板”，沉入长方体的水箱中。算出“圆图板”排开的水的体积，再除以板的厚度，即得圆面积，进而推算出 π 的值。

(3) 概率方法。有蒲丰提出的抛针实验方法（真随机数模拟，略）。

还有抽取随机数模拟的方法，详见例 2-10。

(4) 代数方法。利用有关 π 的无穷级数做近似计算（后面 8.5 节详细介绍）。

【例 1-2】 求多个正整数的最大公约数和最小公倍数的三种算法。

如求最大公约数（756，504，630，2226），最小公倍数［756，504，630，2226］。

(1) 算法1：分解质因数法

$$756=2\times2\times3\times3\times3\times7$$
$$504=2\times2\times2\times3\times3\times7$$
$$630=2\times3\times3\times5\times7$$
$$2226=2\times3\times7\times53$$

最大公约数为4个数的公因子相乘：(756，504，630，1470) $=2\times3\times7=42$

最小公倍数为4个数不重复因子相乘：[756，504，630，1470] $=756\times2\times5\times53=400680$

程序略。

(2) 算法2：辗转相除——递推法

先求最大公约数：

1) (756,504):$756\%504=252$,$504\%252=0$,所以(756,504)=252。

2) (630,252):$630\%252=126$,$252\%126=0$,所以(630,252)=126。

3) (2226,126):$2226\%126=84$,$126\%84=42$,$84\%42=0$,所以(1470,126)=42。

所以，4个数的最大公约数为：(756，504，630，2226) =42。

再求最小公倍数：

1) $[756,504]=\dfrac{756\times504}{252}=1512$

2) [1512,630]:$1512\%630=252$,$630\%252=126$,$252\%126=0$,所以(1512,630)=126

$[1512,630]=\dfrac{1512\times630}{126}=7560$

3) [7560,2226]:$7560\%2226=882$,$2226\%882=462$,$882\%462=420$,$462\%420=42$,

$420\%42=0$,所以(7560,2226)=42,最小公倍数$[7560,2226]=\dfrac{7560\times2226}{42}=400680$

所以，4个数的最小公倍数为：[756，504，630，2226] = 400680。程序略。

(3) 算法3：逐次相减、相除法

先看逐次相减法，用来求多个数（设为m个）的最大公约数，是辗转相除法思想的推广。

算法是：先将m个数由大到小排序，再逐次相减修改各元素的值。设排序后相邻的两个数为A、B，关系为A≥B。再设n=1，2，3，…，各个A同时按下述方法重新赋值：

如$A>n\times B$，则$A=A-n\times B>0$，n是满足条件的最大的整数，相当于A=A%B;

如A=B，A不变。

如$A=n\times B$，$n\neq1$，则$A=A-(n-1)\times B$，即取A=B，A不能是0或负数；

例如，求最大公约数（756，504，630，2226)：

1) 4个数由大到小排序，再从左到右逐次相减：

$$\begin{aligned}&(756,504,630,2226)\\=&(2226,756,630,504)\\=&(2226-2\times756,756-630,630-504,504)\\=&(714,126,126,504)\end{aligned}$$

2）再由大到小排序，从左到右逐次相减：

$$(714,126,126,504)$$
$$=(714,504,126,126)$$
$$=(210,504-3\times126,126,126)\quad\text{（第 2 项减 3 倍的 126，不可为 4 倍）}$$
$$=(210,126,126,126)$$

3）再排序，逐次相减，循环几次：

$$(210,126,126,126)=(84,126,126,126)=(126,126,126,84)$$
$$=(126,126,42,84)=(126,126,84,42)=(126,42,42,42)$$
$$=(126-2\times42,42,42,42)=(42,42,42,42)$$

4 个数相同，则该数为最大公约数：(756，504，630，2226) = 42。

流程如图 1-2 所示。由于本书问题都比较复杂，后面的流程图均省略某些不言而喻的细节（如提示信息），或只画出算法的关键部分，或只画出局部框作功能性描述。

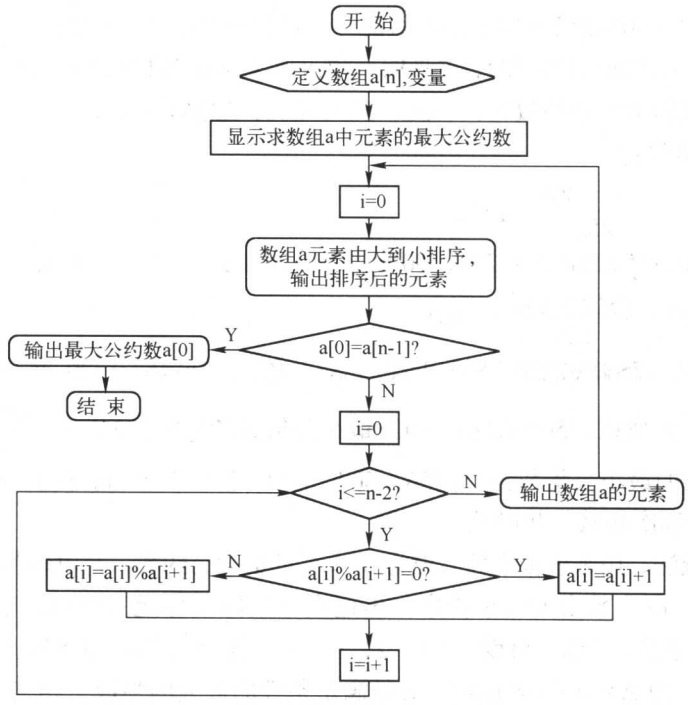

图 1-2

type p1－2－1.c

```
main()
{long a[4]={756,504,630,2226},i,j,t;
 printf("最大公约数:(%ld,%ld,%ld,%ld)\n",a[0],a[1],a[2],a[3]);
 h:for (i=0;i<3;i++)                    /* 数大到小排序,可能多轮 */
   {for (j=i+1;j<=3;j++)
      if (a[i]<a[j]) {t=a[i];a[i]=a[j];a[j]=t;}
```

```
    }
  printf(" =(%ld,%ld,%ld,%ld)\n",a[0],a[1],a[2],a[3]);
  if (a[0]==a[3])                          /* 首尾元素已相等 */
  { printf(" =%ld\n\n",a[0]); exit(0);}/* 输出最大公约数,结束程序 */
 else
  {for (i=0;i<=2;i++)                      /* 逐次相减 */
     { if (a[i]%a[i+1]! =0) a[i]=a[i]%a[i+1];
       else a[i]=a[i+1];
     }
    printf(" =(%ld,%ld,%ld,%ld)\n",a[0],a[1],a[2],a[3]);
    goto h;
  }
}
```

程序运行，给出几轮排序、逐次相减的详细过程。

再看逐次相除法，用来求多个数的最小公倍数。

设有若干个数，其中最大的数为 a，用 a 的 1 倍、2 倍、3 倍、…，除以其余的各数，若第 n 次恰好都能除尽，则此时的 n×a 即为它们的最小公倍数。

例如，756、504、630、2226 中最大的数是 2226，2226 的 180 倍，恰好能被其余的数除尽，所以，4 个数的最小公倍数为：[756，504，630，2226] =2226×180=400680。

type p1-2-2.c

```
main()
{long a[4]={756,504,630,2226},i,t,k=0;
 printf("最小公倍数:[%ld,%ld,%ld,%ld]= ",a[0],a[1],a[2],a[3]);
 for (i=1;i<=3;i++)
     if (a[k]<a[i]) k=i;                  /* k为最大数下标 */
 t=a[0];a[0]=a[k];a[k]=t;t=a[0]; /* 将最大数换到a[0]的位置,令t=a[0] */
 while (a[0]%a[1]! =0 || a[0]%a[2]! =0 || a[0]%a[3]! =0 ) /* 逐次相除 */
       a[0]+=t;
 printf(" %ld\n\n",a[0]);
}
```

1.3　奇妙算法是智慧的结晶

软件（众多算法的集成）的发展，虽然已从效率第一过渡到清晰度第一，但追求效率仍然是永恒的。常用算法是最基本的算法，是一切编程的基础。在此基础上，根据具体问题，可以构思出一些奇妙的算法来解决问题。

奇妙算法，与传统算法截然不同，它是奇特而巧妙的算法，是创新的算法，是高效的算法，它是智慧的结晶。

奇妙算法只垂青善于思考、有准备的头脑，多是苦思冥想的升华。计算机可以帮助人，

使勤快人更聪明；也可以贻误人，使懒惰人更笨拙。这就看和计算机交互时，人有没有把自己放到主导的地位。人要计算机聪明，人就必须比计算机更聪明。人的聪明之处在于能够设计奇妙的算法，而计算机不行，至少现在还不行。

下面的例子，多是考小学生的。但是，如果不善于思考，恐怕中学生、大学生也未必想得出这样奇妙的算法。

1. 奇妙的一定是非凡的

【例 1-3】 1－2＋3＝?

小学生还没有学过负数，此题怎么做？奇妙的算法是将“＋3”前移（这就是非凡）：1＋3－2＝2。能悟出这一点的小学生百里不挑一。但明白这个道理后，马上能将“3÷7×14＝?”变为“3×14÷7＝6”的儿童就超过 50％了。这就叫智力开发！

2. 奇妙的一定是简捷的

【例 1-4】 韩信大点兵。韩信校场点兵，2 人一伍多 1 人，3 人一伍多 2 人，4 人一伍多 3 人，5 人一伍多 4 人，6 人一伍多 5 人，7 人一伍多 6 人，8 人一伍多 7 人，9 人一伍多 8 人，10 人一伍多 9 人，11 人一伍多 10 人，12 人一伍多 11 人。请问：韩信至少有多少兵？

用方程组，难列难解。分析题意和特点，可换一种说法：“2 人一伍少 1 人，3 人一伍少 1 人，…，12 人一伍少 1 人”，都是“少一人”，这就变成了求最小公倍数（再减 1）的问题，何等简捷：

$$(2,3,4,5,6,7,8,9,10,11,12)-1=27719$$

【例 1-5】 阶乘 N＝2008!，末尾有多少个 0?

当然不可能真的去做 2007 次乘法，再数 0 的个数，这种方法用计算机解亦非易事。问题问的是“末尾有多少个 0”，而不是阶乘的精确值。所以，简捷的算法是：N 的 2 的因子多于 5 的因子，有一个 5 的因子就可得到一个 0，但有的数可以分解出几个 5 的因子就可得到几个 0（下面的式子为除法的商取整后相加）：

$$2008/5+2008/25+2008/125+2008/625=500$$

或

$$2008/5+401/5+80/5+16/5=500$$

3. 奇妙的一定是创新的

【例 1-6】 N 个人参加乒乓球单打淘汰赛，至决出冠军需要打多少场?

传统算法一定要知道 N 的具体值，还要考虑可能的轮空。如 N＝100，先有 28 人轮空，其余 72 人比 36 场出线 36 人，28 人加 36 人共 64 人再比 32 场，……

$$36+32+16+8+4+2+1=99$$

创新算法：N－1。每场比赛要淘汰掉一个人，也就是说每淘汰一个人就要进行一场比赛。N 个人参加比赛，到最后只有冠军一个人不被淘汰，要淘汰掉 N－1 个人，所以要打 N－1 场。N＝100 时，N－1＝99。

1.4 穷举法——编程的瑰宝

有些问题的算法可描述为确定的步骤，每一步都是有用的、必须的，没有无用功。而有的问题很难找到这样的算法，或根本就不存在这样的算法，但它们具有这样的特点：如果问

题有解，一组或多组，必定全在某个集合之内，如果集合内无解，集合外也肯定无解。这样，我们就可以将集合中的元素一一列举出来，验证是否是问题的解，这就是穷举法，亦称枚举法。

穷举法不是理想的方法，也不是万能的方法，而是没有办法的办法，但往往又是高效的方法（算法构造快，程序编写快、运行快），有时还是唯一有效的方法。用穷举法编程，第一步是如何把实际问题定义成穷举问题（这是关键，常被忽视），将可能的解限定在一个容易表达的集合之内，然后用循环或循环嵌套编程，验证是否满足问题要求的条件。

最简单的穷举如图 1-3 所示，设待穷举的元素为 100 个，且问题的条件可用变量 i 的表达式表示；变量 t 为有解无解的标志（如果能肯定问题有解，也可不设此标志）。

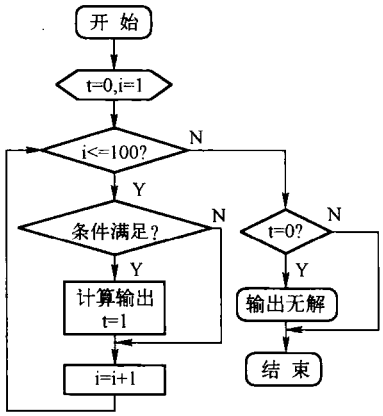

图 1-3

可以这样说，没有程序不会用到穷举法（除非它太简单了，只用公式就够了）。程序中只要有循环，循环中有 if 语句，就是穷举。所以，穷举法是基本的重要的编程方法之一。一个初学编程者，如果他还不会穷举法，应该说他编程还没有入门。如果他仅限于用穷举法编出了程序，还不会优化程序、提高运行速度，可以说他对程序执行过程的理解还欠深入。至于如何优化程序，后面的章节要多次讨论。

此为试读，需要完整PDF请访问：www.ertongbook.com