中国计算机学会信息学奥林匹克系列丛书

全国信息学奥林匹克联赛

National Olympiad in Informatics

试题精解（2001—2003）（普及本）

主 编 中国计算机学会信息学奥林匹克科学委员会

本册编著 吴文虎 李立新 等

本册策划 王晓敏

清华大学出版社

中国计算机学会信息学奥林匹克系列丛书

全国信息学奥林匹克联赛

National Olympiad in Informatics

试题精解（2001—2003）（普及本）

主编　中国计算机学会信息学奥林匹克科学委员会

本册编著　吴文虎　李立新等　　本册策划　王晓敏

清华大学出版社

北　京

内 容 简 介

《中国计算机学会信息学奥林匹克系列丛书》由中国计算机学会信息学奥林匹克科学委员会主编，由全国著名专家学者精心编著而成。

本书收录了全国信息学奥林匹克联赛 2001 年至 2003 年的全部复赛试题，所有试题都给出了具体的算法分析和参考程序清单。对于其中一些试题，不仅给出了常用的基本算法，而且还提供了比较巧妙的优化算法，以开阔思路，启发思维。

本书深入浅出，可读性强，既适合教师辅导学生使用，也适合参加信息学奥林匹克联赛的学生自学，同时也是大专院校的计算机爱好者学习编程的优秀参考书。

图书在版编目(CIP)数据

全国信息学奥林匹克联赛试题精解（2001—2003）/ 吴文虎，李立新等编著. —北京：清华大学出版社，2004.7

（中国计算机学会信息学奥林匹克系列丛书）

ISBN 7-302-09024-6

Ⅰ. 全…　Ⅱ. ①吴…　②李…　Ⅲ. 计算机课-中小学-解题　Ⅳ. G634.675

中国版本图书馆 CIP 数据核字（2004）第 067888 号

出 版 者：清华大学出版社　　地　　址：北京清华大学学研大厦

http://www.tup.com.cn　　邮　　编：100084

社 总 机：010-62770175　　客户服务：010-62776969

组稿编辑：宋　方

文稿编辑：朱凌云

版式设计：肖　米

印 刷 者：北京鑫海金澳胶印有限公司

装 订 者：三河市新茂装订有限公司

发 行 者：新华书店总店北京发行所

开　　本：185×230　印张：12　字数：242 千字

版　　次：2004 年 7 月第 1 版　2004 年 7 月第1次印刷

书　　号：ISBN 7-302-09024-6/TP・6372

印　　数：1～6000

定　　价：18.60 元

FACE

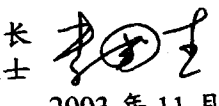
序 言

信息技术对人类社会的发展产生着深远的影响，已成为新世纪的一个标志。作为人类集体智慧的结晶，信息技术已成为一种时代文化。“计算机的普及要从娃娃抓起”成为“科教兴国”的一项重要内容。

一个国家、一个民族要立足于世界先进民族之林，关键在于拥有高素质的人才。综合国力的竞争，说到底是人才的竞争，培养和造就一大批优秀信息技术人才是当务之急。信息时代，信息技术已成为现代科学与技术的基础核心，成为人类的“通用智力工具”，在青少年中普及信息技术教育具有重要和深远的意义。

中国计算机学会从 1984 年起，就组织青少年参加信息学奥林匹克竞赛。二十余年，学会通过组织竞赛推动信息技术普及，促进青少年掌握信息技术知识，并提高他们的逻辑思维和解决问题的能力。为了培养和造就更多高素质的信息技术人才，中国计算机学会特别推出一套信息学奥林匹克系列指导丛书。这套丛书从基础知识开始，重点培养学生的创新思维方法和编程能力。本书的编著者大多是多年从事信息技术普及教学和信息学奥林匹克竞赛的指导者，有丰富的教学经验，他们编写的信息学奥林匹克系列丛书受到了全国青少年的喜爱，他们指导的学生曾多次在国际上获得金奖，为培养青少年的信息素养提供了很好的精神食粮。编著这套丛书的目的旨在培养学生逻辑思维、创新能力和全面提高青少年素质方面提供帮助。

该丛书的编写是开放式的，凡有志于向青少年普及信息技术的科技工作者和教育工作者都可以加入到这个行列中。

中国计算机学会（代理）理事长
中国工程院院士 

2003 年 11 月

PREWORD

前 言

全国青少年信息学奥林匹克竞赛（National Olympiad in Informatics，NOI）是一项面向全国青少年的信息学竞赛和普及活动。NOI 以及全国青少年信息学奥林匹克联赛（NOIP）等系列活动由中国科学技术协会主管，中国计算机学会主办，旨在向青少年普及计算机科学知识，为学校的信息技术教育提供动力和新的思路，给那些有才华的学生提供相互交流和学习的机会，通过竞赛和相关的活动培养和选拔优秀的计算机人才。

竞赛的目的是为了在更高层次上推动普及。受中国计算机学会的委托与指导，从 1995 年起江苏省青少年科技中心已连续多年成功承办了全国信息学奥林匹克联赛活动，数以十万计的青少年从中受益。联赛是全国性的比赛，是 NOI 的基础，遵循开放性原则，任何有条件和有兴趣的学校和个人，都可以在业余时间自愿参加，不和现行的学校教学相冲突，也不列入教学计划，是课外性质的因材施教活动。

在多年的联赛中，参与此项工作的老师与专家们积累了许多宝贵经验，他们曾经对 1995 年至 2000 年前六届联赛的普及组与提高组的全部初、复赛试题加以分析解答并编写成书，对活动的普及起到了很好的推动作用。现根据广大读者的要求，由中国计算机学会组织安排，这些专家与老师们又将第七届（2001 年）至第九届（2003 年）联赛的复赛试题予以剖析，汇集而成本收。本书共分为 3 章，每章分别对应于这几届全国联赛的普及组与提高组的复赛试题。本书紧密围绕联赛复赛大纲所涉及的知识点，以算法分析为主线，针对每一道试题都提供了问题描述，输入输出样例，问题分析，数据结构，参考程序清单和测试数据等几部分。通过分析这些试题，讲思想，讲方法，侧重基础知识训练，引导学生在实践中掌握科学的思维方法，提高应用计算

机的能力。程序设计是一种创造性的劳动，因而在编写本书时，其指导思想侧重放在剖析思路与探讨思维方法方面，引导学生全面、灵活地把握解决问题的思路与方法，提高参赛选手的综合能力。

参加本书编写工作的有江苏省青少年信息学奥林匹克竞赛委员会（以下简称“奥赛委”）科学委员会的李立新教授、江苏省青少年信息学奥赛委普及委员会的高级教练章维铣、曹文老师以及中级教练员王静老师。全书由江苏省青少年信息学奥赛委科学委员会的王晓敏副主任策划并完成统稿，最后经李立新教授初审，清华大学计算机系教授、博士生导师、信息学奥林匹克竞赛中国队总教练吴文虎教授终审定稿。在成书的过程中，得到了中国计算机学会科学委员会诸多专家以及江苏省青少年信息学奥赛委科学委员会的宋方敏、沈军、朱玉珑等专家教授的指导，同时还得到江苏省青少年科技中心领导的大力支持，在此谨向他们表示感谢。

希望广大读者对本书提出宝贵的意见和建议，以便我们进一步修订，使之日臻完善。

编　者

2004 年 6 月

CONTENTS

目 录

第1章 2001年复赛试题解析

1.1 普及组

试题1 数的计数

问题描述

要求找出具有下列性质数的个数(包含输入的自然数 n):
先输入一个自然数 n（$n \leqslant 1000$），然后对此自然数按照如下方法进行处理。
① 不作任何处理;
② 在它的左边加上一个自然数，但该自然数不能超过原数的一半;
③ 加上数后，继续按此规则进行处理，直到不能再加自然数为止。

输入输出样例

输入:
6
输出:
6
此部分不必输出
满足条件的数为
6
16
26
126

36
136

问题分析

根据问题描述中的处理方法②，对自然数 n，可在它的左边加上一个自然数，但该自然数不能超过原数的一半；由描述③，n 加上数后，继续按此规则进行处理，直到不能再加自然数为止。很多同学不理解题意，不妨先看样例。我们把依次产生的数据放到集合 S 中。

① n=6　　S=[6]

② 在 n 的左边可添加 1，2，3，因此可产生新数 16，26，36，S=[6，16，26，36]

对于这个结果大家都很容易得到了，接下来对数据 16，26，36 怎么扩展，是本题的重点，也是本题的难点。

在处理方法描述②中，并没有对“原数”做出明确定义，怎么办？先猜想，再通过样例验证。

（1）猜想

① 原数≡n

由 16 可产生新数 116，216，316；26 可产生新数 126，226，326；36 可产生新数 136，236，336。但在样例中 116，216，316，226，326，236，336 均没有出现，猜想失败。

② 原数=新产生的数（如 16，26，36）

则由 16 可产生新数 116，216，316，416，516，616，716，816，同样，和样例比较，并不吻合，猜想也失败。

③ 原数=n 左边已添加的数据（如 16 中的 1，26 中的 2，36 中的 3）

因为要求左边可添加的自然数不能超过原数的一半，数据 16 无法产生新数；数据 26 可产生新数 126（左边添加 1）；数据 36 可产生新数 136（左边添加 1）。S=[6,16,26,36,126,136]。

126，136 能否继续扩展？根据猜想，数据 126 是在 6 的左边添加了 12，则可继续扩展出 1126，2126，3126，4126，5126，6126，与样例不符，猜想同样失败。

④ 原数=n 左边最近一次添加的数据

类似前面的分析，我们从 S=[6,16,26,36,126,136]开始。

因为 126 中最近一次添加的是 1，因此无法产生新数；同理 136 也无法产生新数。集合 S 扩展结束，与样例完全吻合。

（2）理论验证

根据问题描述中的处理方法③，要使程序能够终止，原数应该在 n 的基础上不断变

小，否则自然数的添加会无法终止，而只有猜想④符合这一要求。如 n=100，可扩展出50100，2550100，122550100，6122550100，36122550100，136122550100。在这里，原数依次为 100，50，25，12，6，3，1，是逐渐变小的。当原数为 1 的时候终止扩展。

很多同学喜欢一拿到题目，就凭主观臆想急着编程。俗话说“磨刀不误砍柴工”，前面的分析虽然花费了一定的时间，但它能使程序目标明确、意图清晰，何乐而不为？

分析到这步，很多同学感觉程序没有问题了。观察数据规模，n≤1000，取 n=1000，则可产生数据 1371531621252505001000（1000→500→250→125→62→31→15→7→3→1），怎样表示这些数据？可以采用高精度数。

再看试题要求：只需输出满足条件的数据个数即可。因此，能否把问题再简化？

根据变化规则，我们每次关心的只是最近一次添加的自然数，因此，可以只记录生成数据的个数，而忽略数据本身。这样，程序就简单多了。

算法 1

（1）数据结构

```
s: longint;                                    {记录满足条件的数据个数}
n: integer;                                    {表示原数}
```

（2）算法描述：（用 s 记录数据个数）

① 输入数据 n，s=0

② s=s+1，n 左边可添加的自然数为 $1\sim\left[\frac{n}{2}\right]$

③ 依次设置原数为 $1\sim\left[\frac{n}{2}\right]$，执行语句②

程序模拟：

假定 n=2，符合要求的数为②，12。

n=2，s=1→n 左边可添加的自然数为 1，s=s+1=2→程序结束。

假定 n=10，符合要求的数为 10，110，210，1210，310，1310，410，1410，2410，12410，510，1510，2510，12510，共 14 个。

图 1.1 表示了程序的模拟过程。

参考程序清单 1

```
program c01_1_2;
var
  s:longint;
  n:integer;
```

```
procedure change(n:longint);                    {统计 n 可扩展出的数据个数}
  var
    i:integer;
  begin
    inc(s);                                     {每出现一个原数，累加器加 1}
    for i:=1 to n div 2 do change(i);           {左边添加不超过原数一半的自然数，作为新原数}
  end;
begin
  read(n);
  s=0;
  change(n);
  writeln(s);
end.
```

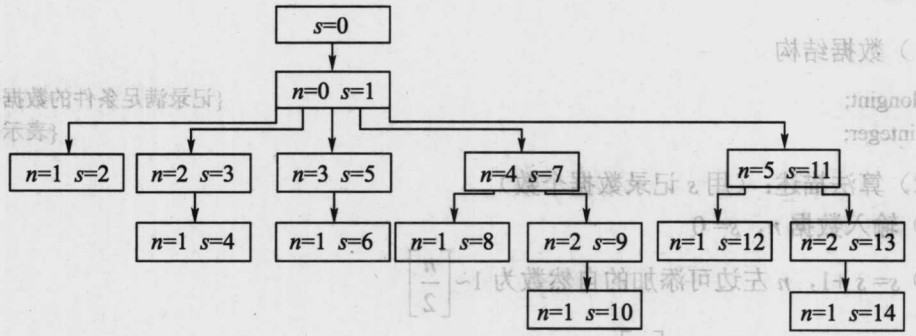


图 1.1

运行结果如表 1.1 所示。

表 1.1

序号	输入（n）	输出（s）
1	2	2
2	10	14

算法 2

算法 1 主要从问题描述中的处理方法①、②出发，每出现一个新数，s 便加 1，直到不能扩展为止。算法 2 主要从问题描述中处理方法②、③出发，对于原数 n，考虑 n 经过一次扩展可以得到的新数个数，直接累加至 s，更新 n 进行下一次扩展，当原数为 1 时，扩展结束。

（1）数据结构

s: longint；{记录满足条件的数据个数}
n: integer；{表示原数}

（2）算法描述：（用 s 记录数据个数）

① 输入数据 n，s=1

② n 左边可添加的自然数为 1~$\left[\frac{n}{2}\right]$，s=s+n div 2

③ 用②继续统计 2~$\left[\frac{n}{2}\right]$左边可添加的自然数个数

程序模拟：假定 n=10，模拟过程如图 1.2 所示。

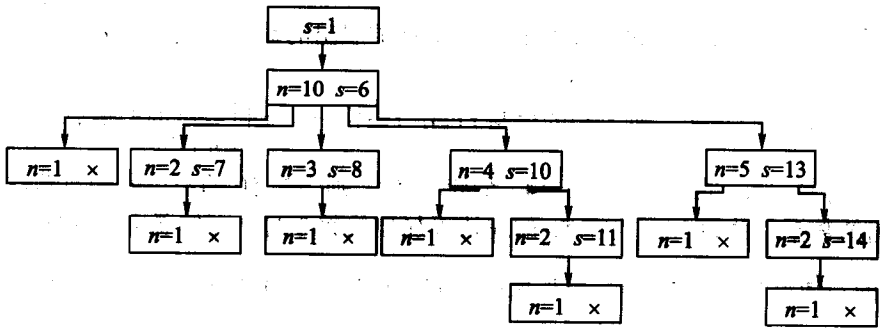


图 1.2

参考程序清单 2

```
program c01_1_2;
var s:longint;
    n:integer;

Procedure change(n:integer);                                    {递归扩展过程}
var i:integer;
  begin
        s:=s+n div 2;
        for i:=2 to (n div 2) do change(i);                     {当 n=1 时，循环不再执行，递归结束}
  end;

begin
  read(n);
  s:=1;
  change(n);
  write(s);
end.
```

运行结果如表 1.2 所示。

表 1.2

序号	输入	输出
1	2	2
2	10	14

算法 1 和算法 2 是两种类似的深度优先搜索算法，都属于递归算法。

算法 3

（1）算法分析

换一个角度看问题，通过前面的分析，我们可以发现这样一个事实，对于任意自然数 n，它能扩展出的数据个数受限于 $1\sim\left[\frac{n}{2}\right]$ 所能扩展出的数据个数。用 $f(n)$表示自然数 n 所能扩展的数据总个数，则 $f(n)$的值与 $f(1)$，$f(2)$，…，$f\left(\left[\frac{n}{2}\right]\right)$ 的取值有关。由前面的分析已知 $f(1)=1$，$f(2)=2$，$f(10)=14$，假定 $n=50$，则当在 50 的左边添加自然数 10 时，可扩展出的数据总数为 14，当在 50 的左边添加自然数 2 时，可扩展出的数据总数为 2，当在 50 的左边添加自然数 1 时，可扩展出的数据总数为 1。能否用 $f(1),f(2),\cdots,f\left(\left[\frac{n}{2}\right]\right)$ 来表示 $f(n)$?

猜想：

$$f(n)=f(1)+f(2)+\cdots+f\left(\left[\frac{n}{2}\right]\right)$$

验证：

若 $n=2$，则 $f(2)=f(1)$，错误。问题在于忽略了处理规则①，$f(n)$存在初始值 1。因此，修正递推表达式：

$$f(n)=1+f(1)+f(2)+\cdots+f\left(\left[\frac{n}{2}\right]\right)$$

$f(3)=1+f(1)=2$　　$f(4)=1+f(1)+f(2)=4$　　$f(5)=1+f(1)+f(2)=4$

$f(10)=1+f(1)+f(2)+f(3)+f(4)+f(5)=1+1+2+2+4+4=14$

n 的最大值为 1000，因此，时间和空间都能承受，虽然在运算过程中 $f\left(\left[\frac{n}{2}\right]+1\right)\sim f(n-1)$等数据对 $f(n)$不起作用，不妨将 $f(1)\sim f(1000)$或 $f(1)\sim f(n)$全部计算出来。请比较以下两个程序，做出选择。

（2）数据结构

```
a: array[1..1000] of longint;                    {a 表示原数 i（1≤i≤1000）所能转换的数据总个数}
```

参考程序清单 3

```
program c01_1_3_1;
var
    a: array[1..1000] of longint;
    i,j,n:integer;
begin
     for i:=1 To 1000 Do a[i]:=1;                                    {设初始值为 1}
     for i:=2 To 1000 Do
          for j:=1 To (i div 2) Do                                   {a[i] = Σ(j=1 to n/2) a[j]}
               a[i]:=a[i]+a[j];
     read(n);
     write(a[n]);
end.
```

```
program c01_1_3_2;
var s:longint;
    a:array[1..1000] of longint;
    i,j,n:integer;

begin
     read(n);
     for i:=1 to n do a[i]:=1;
     for i:=2 to n div 2 do                                          {先计算 a[1]~a[n div 2]}
          for j:=1 to (i div 2) do
               a[i]:=a[i]+a[j];
for j:=1 to n div 2 do                                               {计算 a[n]}
a[n]:=a[n]+a[j];
     write(a[n]);
end.
```

将程序变形，增加如下数组：

```
h:array[1..1000] of longint;
```

用 $h[i]$表示从 i 开始能够扩展出的所有数据；用 sum(i)代表从 $h(1)$到 $h(i)$的加和。则

由前面的分析可知：

$$h(i)=1+\text{sum}(i \text{ div } 2) \qquad \text{sum}(i)=\text{sum}(i-1)+h(i)$$

参考程序清单 4

```
program c01_1_4;
const
  maxn = 1000;
var
  n,i:longint;
  h:array[0..maxn] of longint;
  sum:array[0..maxn] of longint;                    { sum[k]=h[1]+h[2]+…+h[k] }

  begin
    read(n);
    for i:=1 to n do
    begin
      h[i]:=1 + sum[i div 2];
      sum[i]:=sum[i-1]+h[i];
    end;

writeln(h[n]);
  end.
```

程序评估

为了验证程序是否正确，我们通常需要通过一些数据进行尝试，同学们可以自己设计一些测试数据，就本题而言，不妨选取如下数据进行检验：

① 小数据：n=1 和 n=2

这两个数据可以看作是本题的边界情形，很多同学会在这些细节上失分。

② 中等数据：n=10

这类数据我们可以通过直接推导，写出所有满足题意的数，以此作为程序正确性的判断依据。

③ 大数据：n=300 和 n=1000

对于大数据的测试，我们的目的并不是检验能否得出正确解，而是检查程序在数据类型表示、空间、时间等方面是否存在欠缺，以提高程序的稳定性。

对以上 3 类数据的测试如表 1.3 所示。

表 1.3

序号	输入	输出
1	1	1
2	10	14
3	50	786
4	100	9828
5	198	195830
6	300	1564308
7	1000	1981471878

试题 2　最大公约数与最小公倍数问题

问题描述

输入两个正整数 x_0，y_0（$2 \leqslant x_0 < 100000$，$2 \leqslant y_0 \leqslant 1000000$），求出满足下列条件的 p，q 的个数：

条件：

① p，q 是正整数；

② 要求 p，q 以 x_0 为最大公约数，以 y_0 为最小公倍数。

试求：满足条件的所有可能的两个正整数的个数。

输入输出样例

输入：

3　60　（x_0=3，y_0=60）

输出：

4

此部分不必输出

此时的 p　q 分别为

3　60

15　12

12　15

60　3

所以，满足条件的所有可能的两个正整数的组合共 4 种。

问题分析

有些同学拿到这道题，首先想到的是盲目穷举，当 x_0，y_0 偏小的时候尚且可以，但