历届奥林匹克竞赛
典型试题剖析丛书

信息学
奥林匹克竞赛
典型试题剖析

吴耀斌　曹利国　朱全民　向期中　编著

湖南师范大学出版社

典 型 试 题 剖 析 丛 书

信息学奥林匹克竞赛

典型试题剖析

吴耀斌 曹利国 朱全民 向期中 编著

湖南师范大学出版社

图书在版编目（CIP）数据

信息学奥林匹克竞赛典型试题剖析 /吴耀斌编著 .—长沙：湖南师范大学出版社，2002.7

ISBN 7—81081—186—X/G·132

Ⅰ.信 … Ⅱ.吴 … Ⅲ.计算机课—中学—试题 Ⅳ.G634.675

中国版本图书馆 CIP 数据核字（2002）第 039758 号

信息学奥林匹克竞赛典型试题剖析

吴耀斌　曹利国　朱全民　向期中　编著

丛书策划：廖建军

责任编辑：廖小刚　施　游

责任校对：何　适

湖南师范大学出版社出版发行

（长沙市岳麓山）

湖南省新华书店经销　　衡阳印刷厂印刷

730×988　16 开　22.25 印张　457 千字

2002 年 7 月第 1 版　2002 年 7 月第 1 次印刷

印数：1—10000 册

ISBN7—81081—186—X/G·132

定价：24.00 元

内容提要

本书汇集了近几年的国际信息学奥林匹克竞赛(IOI)、全国信息学奥林匹克竞赛(NOI)、中国队组队选拔赛(CTSC)、省级信息学奥林匹克竞赛及集训、选拔赛和国际大学生程序设计竞赛(ACM)等试题。

基于读者已掌握了一门程序设计语言(如 Pascal、C、Basic 等)和基本数据结构知识,全书以算法为主线,以竞赛试题为载体,重点阐述了试题分析、解题思路和多种方法。本书是作者几年来培养参加国际、国内信息学奥林匹克竞赛获奖选手和部分优秀选手的实际经验总结,旨在提高参加信息学奥林匹克竞赛学生的分析和解决问题的能力。所有程序采用信息学竞赛流行的 Turbo Pascal7.0 语言编写,并注重结构化与可读性。

本书是一本大、中学生参加国际 ACM、IOI 和 NOI、全国信息学奥林匹克联赛(NOIp)竞赛的培训教材,也可作为大、中学生学习和研究算法设计的参考书。

前　言

江泽民主席在给第12届国际青少年信息学奥林匹克竞赛的贺信中指出："在人类即将进入新世纪之际，以信息科技和生命科技为核心的科技进步与创新，正在深刻地改变着人类的生产和生活方式，推动着世界文明的发展。青年是人类的未来，也是世界科技发展的未来。"国际信息学奥林匹克竞赛活动，对年轻一代了解和掌握现代科学技术，养成创新精神，具有重要作用。

国际信息学奥林匹克竞赛（International Olympiad in Informatics，IOI）始于1987年，是继数学、物理和化学之后的又一门国际（中学生）学科奥林匹克竞赛，是计算机知识在世界范围青少年中普及的产物。通过竞赛形式对有才华的青少年起到激励作用，促使其能力得以发展；让青少年彼此建立联系，推动经验交流，给学校这一类课程增加活力；建立起教育工作者与专家档次上的国际联系，推进学术思想交流。总之就是：启迪思路，激励英才，发展学科，促进交流。

早在19年前，邓小平同志在视察青少年校外计算机活动时指出："计算机的普及要从娃娃做起。"从此，全国性的青少年计算机竞赛活动每年都吸引着数以万计的青少年投身到这一活动当中，也成为我国校外计算机活动中最有代表性的形式。信息学奥林匹克是智力与能力的竞赛，注重考查全面素质与创新能力。正是因为坚持了全面素质教育的指导思想，把造就高素质、有创造精神的人才作为活动的目标，中国青少年在本学科国际竞赛中，届届名列前茅。

回顾13年国际赛事，正如中国队总教练吴文虎教授所说："参加高手云集的这种世界大赛是有相当难度的，第一，没有大纲，赛题范围没有界定，谁也无法去猜测每年主办国会出什么类型的难题；第二，计算机科学与技术发展很快，层出不穷的新思路和新成果会反映到试题中来；第三，所要解决的试题往往涉及图论、组合数学、人工智能等大学开设的课程知识；第四，比较短的给定解题时间与刁难的测试数据让选手必须拿出高超和精巧的解法，无论在时间上还是空间上都是优化的解法才能取得高分。有许多赛题没有固定的现成的解法，选手要在比赛现场凭借实力，理出思路，构建数学模型，写出算法，编出程序，运行并验证整个构思是否正确，出解的时间是否能达到题目的要求，等等。可以看出，在这一过程中最重要的是创造能力。我们为激发创新精神，培养创造能力，需要树立新的教育观念和教学方法，还要利用现代化的教学手段，引导学生学用电脑，在使用中帮助开发人脑，这可能是信息学奥林匹克活动的最重要的一个特点。在这项活动中应该培养学生的四种能力，即自学能力、实践动手能

力、创新能力和上网获取知识并能区分有用知识和无用知识的能力。”

在信息学奥林匹克的推动下，国际、国内一些优秀、新颖、有意义的试题层出不穷。为了进一步推广、普及信息技术，提高竞赛水平，我们将这些试题进行归纳、总结，并以算法为主线，分为数值算法、基本算法设计策略、搜索及搜索优化方法、图论算法处理和动态规划应用等内容。以竞赛试题为载体，重点阐述解题思路和方法，这些方法和思想是许多优秀选手、辅导教师的实际经验的总结，对于广大读者来说只是一个“抛砖引玉”的作用。我们相信随着信息技术的发展和信息学奥林匹克的不断深入，选手在解题中会有更多灵感和新的思想、新的方法。

本书是一本大、中学生参加国际 ACM、IOI 和 NOI、NOIp 竞赛的培训教材，也可作为大、中学生学习和研究算法设计的参考书。

本书由吴耀斌、曹利国、朱全民、向期中、王建新、肖建华、詹青松、黄烟波、王志坚等编写，全书由吴耀斌统稿，最后由湖南省政府信息化工作领导小组专家组组长、湖南省计算机学会副理事长、湖南省信息学奥林匹克竞赛委员会主任、中南大学计算机科学与技术专业首席教授、博士生导师陈松乔教授审定。

由于水平和时间有限，不妥之处在所难免，敬请读者批评指正。

编　者

2002 年 4 月

目　录

1　数学问题 …… (1)
1.1　瓷片项链(NOI2000) …… (1)
1.2　青蛙过河(NOI2000) …… (3)
1.3　01 统计(CTSC99) …… (7)
1.4　反正切函数的应用(NOI2001) …… (9)
1.5　均分纸牌(IOI99) …… (12)
2　统计问题 …… (16)
2.1　卫星覆盖(NOI97) …… (16)
2.2　机场跑道(IOI99) …… (24)
3　分治问题 …… (30)
3.1　导线开关(IOI95) …… (31)
3.2　中等硬度(IOI2000) …… (36)
3.3　地下城市(IOI99) …… (40)
4　模拟问题 …… (45)
4.1　SERNET 模拟(NOI98) …… (45)
4.2　公交车站(省级竞赛试题) …… (47)
4.3　灭鼠行动(CTSC2001) …… (57)
4.4　沙漠寻宝(HNOI2002) …… (67)
5　深度优先搜索 …… (71)
5.1　基本原理及方法 …… (71)
5.2　马的遍历 …… (76)
5.3　多处理机调度问题(原创试题) …… (78)
5.4　切分铁皮(湖南省集训试题) …… (85)
5.5　逻辑电路最优设计(CTSC2001) …… (88)
5.6　虎口脱险(2000 年国家集训队原创试题) …… (98)
5.7　文件系统(2000 年国家集训队原创试题) …… (108)
5.8　算符破译(NOI2000) …… (116)
5.9　新型防御系统(原创试题) …… (128)
5.10　文件匹配(NOI97) …… (135)

5.11 软件安装(NOI98) …… (143)
5.12 积木搭建(IOI2000) …… (155)
5.13 最佳路线(ACM97) …… (169)
6 广度优先搜索 …… (181)
6.1 基本原理及方法 …… (181)
6.2 翻币问题 …… (190)
6.3 倒水问题(原创试题) …… (193)
6.4 房间问题 …… (196)
6.5 八数码难题 …… (201)
6.6 走棋子 …… (206)
6.7 数字转换 …… (210)
6.8 补丁与错误(CTSC99) …… (215)
6.9 八数码问题 …… (224)
7 图论问题 …… (225)
7.1 遥控赛车(HNOI2001 组队赛试题) …… (225)
7.2 家园(CTSC99) …… (234)
7.3 丘比特的烦恼(CTSC2000) …… (243)
8 动态程序设计 …… (251)
8.1 动态规划基本原理 …… (251)
8.2 机器分配(HNOI95) …… (253)
8.3 最长不下降序列(HNOI97) …… (255)
8.4 凸多边形三角划分(HNOI97) …… (258)
8.5 系统可靠性(HNOI98) …… (260)
8.6 快餐问题(HNOI99) …… (262)
8.7 求函数最大值(CTSC95) …… (268)
8.8 石子合并(NOI95) …… (270)
8.9 游览街区(NOI97) …… (272)
8.10 积木游戏(NOI97) …… (275)
8.11 免费馅饼(NOI98) …… (280)
8.12 棋盘分割(NOI99) …… (283)
8.13 钉子和小球(NOI99) …… (286)
8.14 Subset(NOI99) …… (290)
8.15 陨石的秘密(NOI2001) …… (295)
8.16 商店购物(IOI95) …… (300)
8.17 添括号问题(NOI96) …… (305)
8.18 最长前缀(IOI96) …… (308)

8.19 多边形(IOI98)…… (313)
8.20 花店橱窗布置(IOI99)…… (316)
8.21 选课(CTSC98)…… (319)
8.22 拯救大兵瑞恩(CTSC99)…… (324)
8.23 迷宫改造(winter99)…… (330)
8.24 奶牛浴场(winter2002)…… (341)

1 数学问题

数学是各学科基础的基础，利用数学方法处理问题往往是一种非常行之有效的办法。在信息学奥林匹克竞赛中，很多问题可以通过题设给定的条件和一些隐含的数学关系，寻找出解决问题的递推关系或递归关系，借助于数学方法来实现。在具体的用数学方法解决的问题中，数学方法主要体现以下几个方面：数值处理的精度问题、寻找问题的递归关系并建立问题的递推关系式和利用组合数学知识解决相应问题。在一些大规模搜索问题中，利用数学方法（判定）进行剪枝也是一种非常有效的手段。

1.1 瓷片项链（NOI2000）

问题描述

原始部落用一种稀有的泥土烧制直径相同的圆瓷片并串成项链，串的时候沿瓷片的直径方向顺次连接，瓷片之间没有空隙也不重叠，一条项链至少由一个瓷片构成。

图 1－1 表示四片同样大小的瓷片串接所成的项链，其总长为单个瓷片直径的四倍。

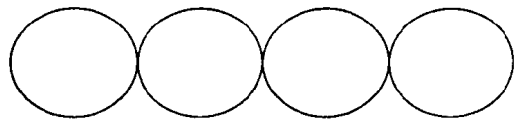

图 1－1

每个烧制的瓷片厚度是一定的，直径 D 和所用泥土的体积 V 有以下关系：

$$D=\begin{cases}0.3\sqrt{V-V_0} & (V>V_0)\\ 0 & (V\leqslant V_0)\end{cases}$$

其中 V_0 为烧制每一片的损耗，单位与 V 相同。当用料小于等于 V_0 时，不能烧制成瓷片。

例：$V_{总}=10, V_0=1$，若烧制成一片瓷片，$V=V_{总}=10, D=0.9$。如果把泥土均分成两份，每份泥土的体积为 $V=V_{总}/2=5$，单个瓷片的直径为 $D'=0.3\sqrt{5-1}=0.6$，串起来的总长为 1.2。

给定了泥土的总体积和烧制单个瓷片的损耗，烧制的瓷片数不同，能够得到的项链总长度也不相同，请计算烧制多少个瓷片能使所得到的项链最长。

输入文件:文件仅有两行,每一行仅包含一个整数和一个换行/回车符。第一行的数字为泥土总体积 $V_{总}$($0 < V_{总} < 60000$),第二行为烧制单个瓷片的损耗 V_0($0 < V_0 < 600$)。

输出文件:文件中仅包含一个数字和一个换行/回车符。该数字为能获得最长项链而烧制的瓷片数。如果不能烧制成瓷片或者最优解不惟一(存在两个或者两个以上方案均能获得最长项链),输出数字 0。

输入输出文件样例 1:

Input.txt

```
10
1
```

Output.txt

```
5
```

输入输出文件样例 2:

Input.txt

```
10
2
```

Output.txt

```
0
```

分析

本题是一道十分简单的试题,最简单的方法就是用枚举来实现,在最坏情况下枚举次数不会超过 60000,但由于实数的运算存在一个精度的问题,实数运算中在没有必要的情况下,我们应尽量减少除法(/)和乘法(小数相乘)运算,因为过多的这种运算将会导致运算精度的不准确。如果我们完全按照题目的描述进行计算,即

```
repeat
  int(i); {i 记录瓷片数}
  each: = v/i; {each 表示单个瓷片可用泥土体积}
  if v < = v0 * i then break;
  d: = 0.3 * sqrt(each - v0) * i; {利用题设公式计算瓷片直径}
until false;
```

从理论上讲,上述方法应该不存在任何问题,但在实际的运行过程中,我们可以发现部分测试数据无法通过,其实这正是由于处理过程中的精度问题造成的。如果我们在判断时去掉 sqrt 运算,同时平方,再省去除法运算,即将 $d: = 0.3 * sqrt(each - v_0) * i$ 改成下面的式子:

$d^2 = 0.3 * 0.3 * (v/i - v_0) * i * i = 0.3 * 0.3 * i * (v - v_0 * i)$,0.3 为常数,判断时可以舍去。

这样,算法本身并没有发生任何改变,但我们可以完全通过测试数据。

事实上,本题完全可以借助数学知识来简化问题的求解,用 n 表示制作瓷片的个数,则有:

$d: = n * 0.3 * sqrt(v/n - v_0)$

$d^2 = n * n * 0.3 * 0.3 * (v/n - v_0) = -0.09v_0n^2 + 0.09vn$

可见，d^2 是一个关于 n 的二次函数，利用数学知识可知，当 $n = v/(2 * v_0)$ 时，d^2 有最大值。这样我们可以得到原问题如下的简单求解方法：

```
x: = v div (2 * v0);
if v mod (2 * v0) > v0 then writeln(x + 1)
                       else if v mod (2 * v0) < v0 then writeln(x)
                                                   else writeln(x + 1);
```

在此基础上，由于 n 必须满足

$(n-1) * 0.3 * \mathrm{sqrt}(v/(n-1) - v_0) \leqslant n * 0.3 * \mathrm{sqrt}(v/n - v_0)$，

$n * 0.3 * \mathrm{sqrt}(v/n - v_0) \geqslant (n+1) * 0.3 * \mathrm{sqrt}(v/(n+1) - v_0)$.

因此，有 $(v - v_0)/(2 * v_0) \leqslant n \leqslant (v + v_0)/(2 * v_0)$，上述程序段可写成如下形式：

```
x: = (v + v0) div (2 * v0);
y: = (v - v0) div (2 * v0) + ord((v - v0) mod (2 * v0)) < >0);
if x = y then writeln(x) else writeln(0);
```

至此问题得圆满解决。

本题尽管是一道非常简单的问题，但我们在求解的过程中也可以得到一些启示，即善于充分挖掘问题给出的数学关系，建立起问题的数学模型，利用数学方法解决的问题程序往往简捷明了，准确率高。同时注意在实数运算中应尽量避免小数运算、除数运算及开方运算等可能带来精度问题的运算。

1.2 青蛙过河(NOI2000)

问题描述

大小各不相同的一队青蛙站在河左岸的石墩(记为 A)上，要过到对岸的石墩(记为 D)上去。河心有几片荷叶(分别记为 $Y_1, \cdots, Y_m$)和几个石墩(分别记为 $S_1, \cdots, S_n$)。图示如图 1-2。

青蛙的站队和移动方法规则：

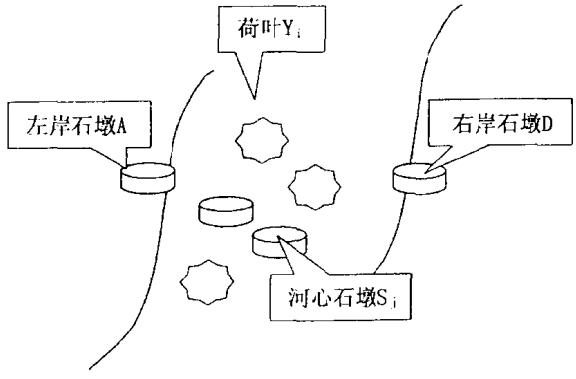

图 1-2

(1)每只青蛙只能站在荷叶、石墩,或者仅比它大一号的青蛙背上(统称为合法的落脚点);

(2)一只青蛙只有背上没有其他青蛙的时候才能够从一个落脚点跳到另一个落脚点;

(3)青蛙允许从左岸 A 直接跳到河心的石墩、荷叶和右岸的石墩 D 上,允许从河心的石墩和荷叶跳到右岸的石墩 D 上;

(4)青蛙在河心的石墩之间、荷叶之间以及石墩和荷叶之间可以来回跳动;

(5)青蛙在离开左岸石墩后,不能再返回左岸;到达右岸后,不能再跳回;

(6)假定石墩承重能力很大,允许无论多少只青蛙都可呆在上面。但是,由于石墩的面积不大,至多只能有一只青蛙直接站在上面,而其他的青蛙只能依规则(1)落在比它大一号的青蛙的背上。

(7)荷叶不仅面积不大,而且负重能力也有限,至多只能有一只青蛙站在上面。

(8)每一步只能移动一只青蛙,并且移动后需要满足站队规则;

(9)在一开始的时候,青蛙均站在 A 上,最大的一只青蛙直接站在石墩上,而其他的青蛙依规则(6)站在比其大一号的青蛙的背上。

青蛙希望最终能够全部移动到 D 上,并完成站队。

设河心有 m 片荷叶和 n 个石墩,请求出这队青蛙至多有多少只,在满足站队和移动规则的前提下,能从 A 过到 D。

例如,在 m = 1 且 n = 1 时,河心有一片荷叶(Y_1)和一个石墩(S_1),此时至多有 4 只青蛙能够过河(由小到大称为 1、2、3、4),过河的一种方法为:

开始		1 2 3 4 A	S_1	Y_1	D
Step 1	1 from A to Y_1	2 3 4 A	S_1	1 Y_1	D
Step 2	2 from A to S_1	3 4 A	2 S_1	1 Y_1	D

Step 3	1 from Y_1 to S_1	3	1		
		4	2		
		A	S_1	Y_1	D
Step 4	3 from A to Y_1		1		
		4	2	3	
		A	S_1	Y_1	D
Step 5	4 from A to D		1		
			2	3	4
		A	S_1	Y_1	D
Step 6	3 from Y_1 to D				3
			2	1	4
		A	S_1	Y_1	D
Step 8	2 from S_1 to D				2
					3
				1	4
		A	S_1	Y_1	D
Step 9	1 from Y_1 to D				1
					2
					3
					4
		A	S_1	Y_1	D

此例中，当河心有一片荷叶和一个石墩时，4 只青蛙能够跳动 9 步过河。

输入文件：文件仅有两行，每一行仅包含一个整数和一个换行/回车符。第一行的数字为河心的石墩数 n($0 \leqslant n \leqslant 25$)，第二行为荷叶数 m($0 \leqslant m \leqslant 25$)。

输出文件：文件中仅包含一个数字和一个换行/回车符。该数字为在河心有 n 个石墩和 m 片荷叶时，最多能够过河的青蛙的只数。

输入输出文件样例：

Input.txt

```
1
1
```

Output.txt

```
4
```

分析

本题青蛙的移动规则非常繁杂，按照一般的思考方式，运用产生式系统进行解题

有一定难度，同时，模拟青蛙的移动过程也是一项非常艰巨的任务。但我们注意到，题中只要求我们求出最多的青蛙只数，并不需要清楚知道青蛙的具体移动方案，这就给我们提供了一个是否可以用数学方法来解决问题或从中得到某些规律的启示。

设 F[n,m] 表示河心有 n 个石墩和 m 片荷叶时按照移动规则可以过河的最多的青蛙只数。考虑其与在河心减少一个石墩或减少一片荷叶的条件下可以过河的青蛙只数之间的关系，即考虑 F[n,m]与 F[n-1,m]之间和 F[n,m]与 F[n,m-1]之间的关系。

由题设条件可知，F[n,m]与 F[n-1,m]关系是：如果当前增添了一个石墩，则可以先将前 F[n-1,m]个青蛙移到新增添的石墩上（即把新增添的石墩看成原来的右岸），把后 F[n-1,m]个青蛙移到右岸上，然后再把先前移到石墩上的 F[n-1,m]个青蛙移到右岸上（把新增添的石墩看成是原来的左岸）。这样，我们可以得到以下的一个递推关系式：

F[n,m] = 2 * F[n-1,m]

而 F[n,m]与 F[n,m-1]之间的关系是：如果当前增添了一片荷叶，按照移动规则，我们可以先将当前最小的青蛙安放在其上面，然后把余下的 F[n,m-1]个移到右岸，再把最小的青蛙移到右岸。同样，我们又可以得到以下的一个递推关系式：

F[n,m] = F[n,m-1] + 1

综合上述两个递推式，我们得到 F[n,m]的递推关系式如下：

F[n,m] = max(F[n-1,m] * 2,F[n,m-1] + 1)

易知该递推关系式的终止条件是 F[0,m] = m + 1。

至此，我们已经求得了上述递推关系式，程序的实现已经非常简单。事实上，借助于数学的推导，我们可以得到一个更简单的解决问题的通项公式。

我们注意到，假设当前有 n 个石墩，m 片荷叶，由于荷叶起到的只是一个过渡作用，即可以将(m+1)只青蛙按顺序从起始石墩移到目标石墩。因此，如果换一种思考方式，设用 G[n]表示有 n 个石墩且所有的青蛙均跳到河中心的石墩时最多可跳过的青蛙数，则有 G[n] = G[n-1] * 2 + m + 1，G[0] = m + 1。此时利用数学中等比数列的性质，我们可以直接得到解决问题的公式。即

$$G[n] := \sum (m+1) * 2^i \quad (i = 1,2,3,\cdots,n-1)$$

$$= (m+1) * \sum 2^i$$

$$= (m+1) * (2^n - 1)$$

然后将最后的 m + 1 只青蛙从左岸跳到右岸，则问题的答案变为：

$$Ans := (m+1) * (2^n - 1) + (m+1) = (m+1) * 2^n$$

从本题的求解过程可以看出，剔除题目中过多的条件描述，挖掘出问题的本质和内在关系，利用数学知识和数学推导方法，建立起解决问题的递推关系式，可以大大地简化问题的求解过程，达到事半功倍的效果。

1.3 01 统计(CTSC99)

问题描述

近来 IOI 的专家们在进行一项有关二进制数的研究,研究涉及的一个统计问题令他们大伤脑筋。问题是这样的:

对于一个自然数 n,可以把它转换成对应的二进制数$\overline{a_k a_{k-1}\cdots a_1 a_0}$,其中:$n = a_k \times 2^k + a_{k-1} \times 2^{k-1} + \cdots + a_1 \times 2^1 + a_0$,且 $a_i = 0$ 或 $1(0 \leqslant i < k)$,$a_k = 1$。如:$10 = \overline{1010}$,$5 = \overline{101}$。我们统计一下 $a_0 \sim a_k$ 这 k+1 个数中 0 的个数和 1 的个数,如果在这 k+1 个数中,0 的个数比 1 的个数多,就称 n 为 A 类数。

现在的任务是,对于一个给定的 m,求 1 ~ m 中 A 类数的个数。

输入:输入文件中只有一个自然数 $m(1 \leqslant m \leqslant 10^{30})$。

输出:输出文件也只有一个自然数:1 ~ m 中 A 类数的个数。

输入输出示例:

输入文件	输出文件
3	0

分析

01 统计问题的题义可以归纳为:

A 类数的定义:对于自然数 n,如果 n 的二进制数中 0 的个数比 1 的个数多,那么 n 为 A 类数,输入自然数 m,求 1 ~ m 中 A 类数的个数。

本题的重点、难点在于数据量大,m 的取值范围在$[1 \cdots 10^{30}]$,大大超过机器定义的整数范围。虽然本题是统计问题,但使用1 ~ m的循环来逐个判断显然耗时过多,对于 m 较大时无法在规定的时间内解出。所以我们希望通过分类统计的方法,进一步抽象问题,得到可行的算法。

为了加深对问题的认识,我们手工计算一个例子。

设 $m = (112)_{10} = (1110000)_2$,求 1 ~ m 的 A 类数个数。

如果采用逐个循环判断的方法,数学意义是将 112 个数分成 112 类,每一类是一个自然数,分别统计后应用加法法则得到最后的结果。为了提高效率,我们可以尝试减少所分的类型,将 112 个数分为以下几类:

(1)数字形式为$(111xxxx)_2$

这一类数只有 1 个,即$(1110000)_2$,是 A 类数。

(2)数字形式为$(110xxxx)_2$

设 4 个填数位置填 0 的个数为 S_0,填 1 的个数为 S_1,

则 $S_0 + S_1 + 3 = 7$。

若为 A 类数,则 $S_0 + 1 > S_1 + 2$,

可以取 $S_0 = 4, S_1 = 0$;$S_0 = 3, S_1 = 1$。

这一类数中的 A 类数个数：$C_4^4 + C_4^3 = 5$。

(3)数字形式为$(10xxxxx)_2$

设 5 个填数位置填 0 的个数为 S_0，填 1 的个数为 S_1，

则 $S_0 + S_1 + 2 = 7$。

若为 A 类数，则 $S_0 + 1 > S_1 + 1$，

可以取 $S_0 = 5, S_1 = 0; S_0 = 4, S_1 = 1; S_0 = 3, S_1 = 2$。

这一类数中的 A 类数个数：$C_5^5 + C_5^4 + C_5^3 = 16$。

(4)数字形式为$(0xxxxxx)_2$

这一类数字的分析与前几类略有不同，因为前几类二进制数的位数都是 7，而这一类数还需对位数进行讨论：

①1 位数，即$(1)_2$，不是 A 类数。

②2 位数，即$(1x)_2$，$(10)_2$ 和$(11)_2$ 都不是 A 类数。

③3 位数，即$(1xx)_2$，A 类数个数为 $C_2^2 = 1$。

④4 位数，即$(1xxx)_2$，A 类数个数为 $C_3^3 = 1$。

⑤5 位数，即$(1xxxx)_2$，A 类数个数为 $C_4^4 + C_4^3 = 5$。

⑥6 位数，即$(1xxxxx)_2$，A 类数个数为 $C_5^5 + C_5^4 = 6$。

最后的答案是 $1 + 5 + 16 + (1 + 1 + 5 + 6) = 35$。

通过这个具体例子的计算，可看出根据二进制数的前缀来分类是合理的，既没有重复也没有遗漏。当 m 的二进制数长度为 n 时，最多有 2n 种类型，即 $2([\log_2 m] + 1)$ 种，比穷举法的 m 种类型有了很大进步。以下是这种组合数学算法的完整描述：

原理：组合数学的加法法则，即将原计数问题分为互不交叉的几个子计数问题，分别统计后将结果累加。

分类：设 m 的二进制数为 $a_1 a_2 a_3 \cdots a_n$，设 $a_1 a_2 a_3 \cdots a_k$ 中 0 的个数为 $S_0(k)$，1 的个数为 $S_1(k)$，设 $b_1 b_2 b_3 \cdots b_n$ 为 1 ~ m 的一个数，此时可分为：

第一大类，当 $b_1 = 1$ 时，根据前缀的不同，这类数中的 A 类数个数为

$$Ans1 = \sum_{i=2}^{n} \sum_{j=0}^{n-i} C_{n-i}^{j} (\text{当 } a_i = 0 \text{ 且 } S_0(i-1) + 1 + j > S_1(i-1) + n - i - j)$$

为了统一起见，当计算到 C_0^0 时，约定 $C_0^0 = 1$。

第二大类，当 $b_1 = 0$ 时，根据二进制数位数的不同，这类数中的 A 类数个数为

$$Ans2 = \sum_{i=2}^{n} \sum_{j=\frac{n-i}{2}+1}^{n-i} C_{n-i}^{j}$$

以上两大类统计的均是小于 m 的 A 类数，如果 m 本身是 A 类数，那么 Ans3 = 1，否则 Ans3 = 0。

问题的解即为 Ans1 + Ans2 + Ans3，问题得到解决。