信息学奥林匹克竞赛指导丛书

吴文虎 主编

信息学奥林匹克竞赛指导

——2002竞赛试题解析

吴文虎 王建德 编著

✓ 训练思维能力

✓ 提高解题技巧

清华大学出版社

信息学奥林匹克竞赛指导丛书
吴文虎　主编

信息学奥林匹克竞赛指导

——2002竞赛试题解析

吴文虎　王建德　编著

清华大学出版社
北京

内容简介

本书收集了2002年国际、国内有关信息学奥林匹克竞赛试题，重点在于分析解题思路和方法上，其中包括数学模型的构建、相应的算法分析以及程序的编写等，这些试题有相当的难度，是训练思维、提高解题技巧的很好参考资料。

书中对试题类型进行了归纳，增强了本书的可读性，既便于教师对参赛学生的辅导，又便于学生自学，所以本书既是参赛选手的必读书，也可作为理工科院校编程爱好者提高分析问题、解决问题能力的参考书。

版权所有，翻印必究。

本书封面贴有清华大学出版社激光防伪标签，无标签者不得销售。

图书在版编目(CIP)数据

信息学奥林匹克竞赛指导——2002竞赛试题解析/吴文虎，王建德编著. —北京：清华大学出版社，2003

(信息学奥林匹克竞赛指导丛书/吴文虎主编)

ISBN 7-302-07401-1

Ⅰ. 信… Ⅱ. ①吴… ② 王… 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(2003)第092476号

出 版 者：清华大学出版社

http://www.tup.com.cn

社 总 机：010-62770175

地　　址：北京清华大学学研大厦

邮　　编：100084

客户服务：010-62776969

责任编辑：宋　方

印 刷 者：清华大学印刷厂

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：185×260　**印张**：10.5　**字数**：239千字

版　　次：2004年1月第1版　2004年1月第1次印刷

书　　号：ISBN 7-302-07401-1/TP·5466

印　　数：1～5000

定　　价：15.00元

前　言

信息科学与技术正在对人类社会的发展产生难以估量的深远影响，已经成为新世纪的一个标志。作为人类总体智慧的结晶，电脑已经成为一种新的现代文化。“计算机的普及要从娃娃抓起”已经成为“科教兴国”的一项重要内容。

一个国家、一个民族要想不落伍，要想跻身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，说到底也是人才的竞争。电子计算机是现代科学与技术的基础和核心，它的飞速发展，把社会生产力水平提到前所未有的高度，人类进入了信息时代。电脑对人类社会的发展所起的巨大作用，特别是对人类智能的发展所起的促进作用，已为人们普遍认识。计算跟语言一样是人类社会每时每刻都不可缺少的。现在，人类已经拥有了帮助自己进行复杂计算与思维的工具，电子计算机起到了人脑延伸的作用。以往历史上的技术革命，只能起到创造和改进工具，用机器代替人的体力的作用；而计算机则是把人从重复性的或有固定程式的脑力劳动中解放出来，使自己的智能获得空前的发展。作为“人类通用智力工具”的计算机在开发人类智能方面所起的无与伦比的作用不容忽视。信息技术是以计算机为龙头的。这也就是信息技术与基础教育相结合，能够成为当今世界的大趋势的一个原因。从信息社会要求人才具备的科学素养看，数学、物理学、化学、生命科学和信息科学是五大支柱，这正是联合国教科文组织倡导举行五项国际学科奥林匹克竞赛的内容。

国际信息学奥林匹克（International Olympiad in Informatics，IOI）始于 1989 年，到 2003 年已成功地举办了 15 届。这是一种智力与应用计算机能力的大赛。从益智的角度看，是用电脑帮助开发人脑，重在提高思维能力，培养创新意识。在中国队的训练中强调德智体美全面发展；心态上自立、自尊、自信、自强，要怀着中华民族的自豪感和自信心去参赛；这种心态是学习、训练和取胜的重要条件。15 届比赛，中国队每届都取得了名列前茅的好成绩，59 人次参赛，夺得 59 块奖牌，其中金牌 30 块、银牌 17 块、铜牌 12 块。特别是 IOI’95（荷兰）突破了前 6 届比赛女孩与金奖无缘的纪录，两名中国女选手荣登金牌领奖台。在 IOI’96（匈牙利）上中国队又实现了全金的突破，四名选手，每人获得一块金牌。

从大局看，竞赛不是目的，是推动普及的手段，我们的目的只有一个：科教兴国。竞赛活动带有因材施教、因材施测的特点。普及是有层次的，与学科竞赛有关的普及活动，对青少年而言属于比较高的层次，当然就有相当的难度。我们编写的这套竞赛指导丛书，涉及程序设计语言、常用算法、组合数学、图论、人工智能搜索等的基本知识和基本方法。这些理论知识往往都是通过竞赛当中的一些实例来讲解的，重点放在解题思路上，书中有许多题目比较新颖，很难去套固定算法或固定模式，这中间有些招数

是选手们想出来的。从中可以看出信息学奥林匹克要求创新，鼓励创新。当然，书中给出的解法，对青少年读者而言，我们希望仅仅起到抛砖引玉的作用，并且热切盼望引出更多的“玉”来。作为老师，我和王建德都这样想，“精心育桃李，热望青胜蓝”就是我们编写这套丛书的初衷。

国际信息学奥林匹克中国队总教练
清华大学计算机系教授博士生导师
吴文虎

2003 年 11 月

试读结束：需要全本请在线购买： www.ertongbook.com

目　录

第1章　与课本知识相关的例题

在普及性的分区联赛中，为了考核学生课内知识的掌握程度，总是有意出一些相关的试题，让选手编程解答，这几乎成了历年的惯例。当然，选手要解这些题，需要有一定的编程技术，但最主要的还是解题思路和应用课内知识于实际问题的能力。

1.1　级数求和

问题描述

已知：$S_n=1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n}$。显然当 n 非常大的时候，S_n 可大于任何一个整数 k。现给出一个整数 $k(1\leqslant k\leqslant 15)$，要求计算出一个最小的 n，使得 $S_n>k$。

输入

键盘输入 k

输出

屏幕输出 n

输入输出样例

输入

1

输出

2

算法分析

此题编程并不难，关键是选手选择变量类型的能力。由于该数列是递减的，而 k 的上限为15，因此项数 n 很大，即使 longint 也容纳不下。但未必非高精度运算不可，只要启动浮点数运算({ $n+})，将项数设为 extended 类型，便可以得出正确解。

```
{$n+}                                    {启动浮点数运算}
var
    s,b,k:extended;                      {数列的和、项数、最接近 Sn(大于 Sn)的整数值}
begin
    s:=0;                                {数列的和初始化}
    b:=0;                                {项数初始化}
    readln(k);                           {读最接近 Sn(大于 Sn)的整数值 k}
    while s <= kdo                       {若目前数列的和小于 k,则项数 b+1,计算 Sb}
      beginb←b+1;s←s+1/b; end;{while}
    输出项数 round(b);
```

end. {main}

1.2 自由落体

问题描述

在高为 H 的天花板上有 n 个小球，体积不计，位置分别为 0，1，2，…，$n-1$。在地面上有一个小车（长为 L，高为 k，距原点距离为 s_1）。已知小球下落距离计算公式为 $s=\frac{1}{2}gt^2$，其中 $g=10$，t 为下落时间。地面上的小车以速度 v 前进。如图 1.1 所示。

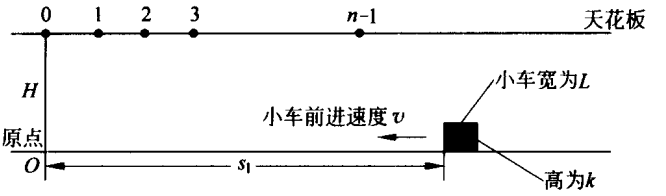

图 1.1

小车与所有小球同时开始运动，当小球距小车的距离小于等于 0.00001 时，即认为被小车接收（小球落到地面后不能被接收）。请你计算出小车能接收到多少个小球。

输入

$H,s_1,v,L,k,n(1\leqslant H,s_1,v,L,k,n\leqslant 100000)$

输出

小车能接收到的小球个数。

算法分析

这是分区联赛最容易失误的一道试题，关键是弄清小车行程的物理意义和计算的精度误差。由题意可知，车顶与天花板的距离为 $h-k$，小球由天花板落至车顶所花费的时间为 $t_1=\sqrt{\frac{2(h-k)}{10}}$，由天花板落至地面的时间为 $t_2=\sqrt{\frac{2h}{10}}$。小车与所有小球同时开始运动，当小球距小车的距离小于等于 0.00001 时，即认为小球被小车接收（小球落到地面后不能被接收）。

小车在行驶了 $t_1v-0.00001$ 路程后开始接收第 1 个小球，由于第 1 个小球的编号上限为 $n-1$，且在行驶 l 距离的过程中，该小球都会落至车上，因此第 1 个小球的编号 $n_1=\min\{n-1,\lfloor s_1-t_1v+l+0.00001\rfloor\}$，如图 1.2 所示。公式中取下整的目的是为了舍去无用的小数部分。

小车在行驶了 $t_2v+0.00001$ 路程后小球落至地面，由于小车接收的最后一个小球的编号下限为 0，因此其编号为 $n_2=\max\{0,\lfloor s_1-t_2v-0.00001+0.5\rfloor\}$，如图 1.3 所示。公式中加 0.5 且取上整的目的是为了得出小球落地前小车接收的最后一个小球编号。

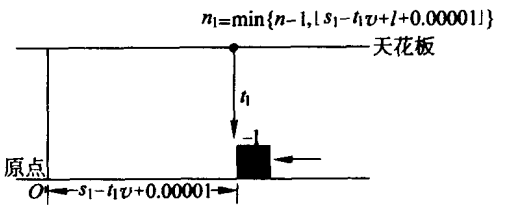

图 1.2

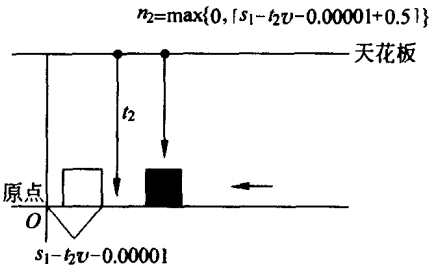

图 1.3

显然，若 $n_1 \geqslant n_2$，则小车接收的小球数为 n_1-n_2+1；否则小车未接收任何一个小球。由此得出算法：

读天花板的高度 h，小车的初始位置 s_1、速度 v、长 l，高 k 和小球数 n；

```
    t1←sqrt(2 * (h-k)/10);                {计算小球由天花板落至车顶所花费的时间}
    t2←sqrt(2 * h/10);                    {计算小球由天花板落至地面的时间}
    n1←trunc(s1-t1 * v+l+0.00001);        {计算小车接收的第 1 个小球编号}
    n2←round(s1-t2 * v-0.00001+0.5);      {计算小车接收的最后 1 个小球编号}
    if n1>n-1 then n1←n-1;
    if n2<0 then n2←0;
if n1>=n2 then writeln(n1-n2+1)           {输出小车接收的小球数}
    else writeln(0);
```

第2章　数据结构类的例题

数据结构专门研究各种数据的表示、数据的类型以及它们之间关系的集合，其研究范围主要包括各种数据结构的性质，即它们的逻辑结构、物理结构以及施于其上的操作。虽然2002年全国和国际信息学奥赛的每一道题都有如何为解题选择合适的数据结构的问题，都与数据结构相关，但是其中也有一些重点考核数据结构知识的试题，这些知识包括字符串处理和并查集与路径压缩。

2.1　有关字符串处理的例题

在实际生活中我们经常要对一串元素进行操作。如信息检查系统、文字编辑系统、自然语言翻译系统以及音乐分析程序等，都是以字符串数据作为处理对象的。随着非数值的广泛应用，字符串的处理将越来越重要，因此竞赛中也经常出现一些字符串处理的试题。为此，我们不仅应该熟悉字符串处理的经典算法，例如KMP等，而且还应该熟悉Pascal系统提供的库函数，并能驾轻就熟地应用于解题过程。

2.1.1　字符近似查找

问题描述

设有 n 个单词的字典表($1 \leqslant n \leqslant 100$)。计算某单词在字典表中的4种匹配情况(字典表中的单词和待匹配单词的长度上限为255)：

i：该单词在字典表中的序号；

E_i：在字典表中仅有一个字符不匹配的单词序号；

F_i：在字典表中多或少一个字符(其余字符匹配)的单词序号；

N：其他情况。

当查找时有多个单词符合条件，仅要求第1个单词的序号即可。

输入文件

输入文件名为a.in，文件的格式如下：

n(字典表的单词数)

n 行，每行一个单词

待匹配单词

输出文件

输出文件名a.out，格式如下：

i

E_i

F_i

试读结束：需要全本请在线购买：　www.ertongbook.com

其中 i 为字典表中符合条件的单词序号($1 \leqslant i \leqslant n$),若字典表中不存在符合条件的单词,则对应的 $i=0$。若上述 3 种情况不存在,则输出 N。

输入输出样例

输入 1:

5
abcde
abc
asdfasfd
abcd
aacd
abcd

输出 1:

4
E5
F1

输入 2:

1
a
b

输出 2:

0
E0
F0
N

算法分析

我们将字典表中的单词分成 3 类:

第 1 类:单词与待匹配单词多或少一个字符,其余字符匹配;

第 2 类:单词仅有一个字符与待匹配单词不匹配;

第 3 类:单词与待匹配单词完全匹配。

设

```
const
  note :array[1..3] of string=('F','E','');        {匹配情况的标志}

var
  want:string;                                     {待匹配单词}
  list:array[1..100] of string;                    {字典表。其中 list[i]为字典 i}
```

```
ans:array[1..100] of integer;          {单词的类别序列。其中
```

$$ans[i]=\begin{cases}1 & \text{在 want 中添加或删除一个字符得到 list [i]}\\ 2 & \text{want 与 list[i]有一个字符不同}\\ 3 & \text{want 与 list[i]相同}\\ 0 & \text{其他情况}\end{cases}$$

1. 匹配情况的计算

(1) 计算两个等长字串中不同字符的个数

```
function find(a,b:string):integer;      {输入两个等长字串 a,b,计算和返回不同字符的个数}
var i,tot:integer;
begin
    tot←0;
    for i←1 to length(a) do if a[i]<>b[i] then inc(tot);
    find←tot;
end;{find}
```

(2) 判别一个字串是否比另一个字串多一个字符(其余字符匹配)

我们不知道长度大 1 的字串究竟在哪个位置上多出一个字符,所以只能将该字串的每一个字符试插在另一个字串的对应位置上。如果插入后使得两串相同,则说明猜想成立,否则猜想不成立。

```
function check(a,b:string):integer;      {输入字串 a,b。若 b 能够在 a 的基础上
                                           添加一个字符得到的话,则返回 1;否则
                                           返回 0}
var i:integer;
begin
    check←0;
    for i←0 to length(a) do               {搜索 a 的每一个插入位置}
    begin
      a←copy(a,1,i)+b[i+1]+copy(a,i+1,255);   {在 a[i]后插入 b[i+1]}
      if a=b                              {若插入后两串相同,则成功退出}
        then begin check←1;exit;end;{then}
      delete(a,i+1,1);                    {删去 a 中的插入字符}
    end;{for}
end;{check}
```

2. 计算字典表中的每一类单词

首先,依次计算每一个单词的类别序号

在单词 i 与待匹配单词等长的情况下,若两串相同,则单词 i 的类别记为 3;若两串仅有一个字符不同,则单词 i 的类别记为 2;

若单词 i 比待匹配单词多或少一个字符(其余字符匹配),则单词 i 的类别记为 1;

否则单词 i 的类别记为 0；

然后根据 ans 序列在字典表中依次搜索类别 3..类别 1 的单词，输出对应的单词序号。如果在字典表中不存在上述 3 种类别的单词，则输出“N”。

```
fillchar(ans,sizeof(ans),0);                {单词的类别序列初始化}
for i←1 to n do begin                       {对字典中的每个单词进行分类}
  if length(list[i])=length(want)           {若单词 i 与待匹配单等长}
    then begin
        k←find(list[i],want);               {计算单词 i 与待匹配单词的不同字符个数}
        if k=0 then ans[i]←3;               {记下类别序号}
        if k=1 then ans[i]←2;
    end;{then}                              {若单词 i 比待匹配单词多或少一个字符(其余字符匹
                                             配),则单词 i 的类别记为 1;否则单词 i 的类别记为 0}
      if length(list[i])+1=length(want) then ans[i]←check(list[i],want);
      if length(list[i])=length(want)+1 then ans[i]←check(want,list[i]);
    end;{for}
    have←false;                             {匹配情况存在的标志初始化}
    for i←3 downto 1 do begin               {依次输出每一类别的单词在字典表最先出现的序号}
      k←0;
      for j←1 to n do if ans[j]=ithen begin k←j;break;end;{then}
      have←have or (k>0);
      writeln(note[i],k);
    end;{for}
```

2.2 并查集与路径压缩

将所有相关的事物归并到一个集合里，这个集合简称为并查集。并查集的数据类型采用树型，查询的效率最高。

寻找并查集元素的时效与树的深度成线性关系，平均的时间复杂度为 $O(\log_x n)$（x 指树的度），但在树退化成链表的最坏情况下，时间复杂度为 $O(n)$，因此需要进行路径压缩的优化：首先找到树根，然后将路径上所有结点的父结点改为根，使得树的深度为 1，如图 2.1 所示。

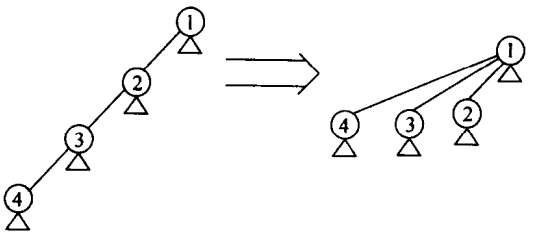

图 2.1

路径压缩后，寻找并查集元素的时间复杂度降低为 $O(1)$。

2001 年的全国赛曾经出过一道并查集与路径压缩的试题(食物链)，引起了了对并查集的讨论。无独有偶，2002 年的全国赛又出了同类型的一道试题，但增加了计算并查集中元素间相对位置关系的要求，把讨论又引深了一步。

2.2.1 银河英雄传说

问题描述

公元 5801 年，地球居民迁移至金牛座 α 第二行星，在那里发表银河联邦创立宣言，同年改元为宇宙历元年，并开始向银河系深处拓展。

宇宙历 799 年，银河系的两大军事集团在巴米利恩星域爆发战争。泰山压顶集团派宇宙舰队司令莱因哈特率领 10 万余艘战舰出征，气吞山河集团点名将杨威利组织麾下 3 万艘战舰迎敌。

杨威利擅长排兵布阵，巧妙运用各种战术屡次以少胜多，难免滋生骄气。在这次决战中，他将巴米利恩星域战场划分成 30 000 列，每列依次编号为 1，2，…，30 000。之后，他把自己的战舰也依次编号为 1，2，…，30 000，让第 i 号战舰处于第 i 列($i = 1, 2, \cdots, 30\,000$)，形成“一字长蛇阵”，诱敌深入，这是初始阵形。当进犯之敌到达时，杨威利会多次发布合并指令，将大部分战舰集中在某几列上，实施密集攻击。合并指令为 M_{ij}，含义为让第 i 号战舰所在的整个战舰队列，作为一个整体(头在前尾在后)接至第 j 号战舰所在的战舰队列的尾部。显然战舰队列是由处于同一列的一个或多个战舰组成的。合并指令的执行结果会使队列增大。

然而，老谋深算的莱因哈特早已在战略上取得了主动。在交战中，他可以通过庞大的情报网络随时监听杨威利的舰队调动指令。

在杨威利发布指令调动舰队的同时，莱因哈特为了及时了解当前杨威利的战舰分布情况，也会发出一些询问指令：C_{ij}。该指令的意思是，询问电脑，杨威利的第 i 号战舰与第 j 号战舰当前是否在同一列中，如果在同一列中，那么它们之间布置有多少战舰。

作为一个资深的高级程序设计员，你被要求编写程序分析杨威利的指令，以及回答莱因哈特的询问。

最终的决战已经展开，银河系的历史又翻过了一页……

输入文件

输入文件 galaxy.in 的第 1 行有一个整数 $T(1 \leqslant T \leqslant 500\,000)$，表示总共有 T 条指令。

以下有 T 行，每行有一条指令。指令有如下两种格式。

M_{ij}：i 和 j 是两个整数($1 \leqslant i, j \leqslant 30\,000$)，表示指令涉及的战舰编号。该指令是莱因哈特窃听到的杨威利发布的舰队调动指令，并且保证第 i 号战舰与第 j 号战舰不在同一列。

C_{ij}：i 和 j 是两个整数($1 \leqslant i, j \leqslant 30\,000$)，表示指令涉及的战舰编号。该指令是莱因哈特发布的询问指令。

输出文件

输出文件为 galaxy.out。你的程序应当依次对输入的每一条指令进行分析和处理：

如果是杨威利发布的舰队调动指令，则表示舰队排列发生了变化，你的程序要注意到这一点，但是不要输出任何信息；

如果是莱因哈特发布的询问指令，你的程序要输出一行，仅包含一个整数，表示在同一列上，第 i 号战舰与第 j 号战舰之间布置的战舰数目。如果第 i 号战舰与第 j 号战舰当前不在同一列上，则输出 -1。

输入输出样例

输入

4

M 2 3

C 1 2

M 2 4

C 4 2

输出

-1

1

样例说明

战舰位置图：如表 2.1 所示。表格中阿拉伯数字表示战舰编号。

表 2.1

	第 1 列	第 2 列	第 3 列	第 4 列	……
初始时	1	2	3	4	……
M_{23}		1	3 2	4	……
C_{12}	1 号战舰与 2 号战舰不在同一列，因此输出 -1				
M_{24}	1			4 3 2	……
C_{42}	4 号战舰与 2 号战舰之间仅布置了一艘战舰，编号为 3，输出 1				

算法分析

同一列的战舰组成一个并查集，集合中的一个结点对应一艘战舰。在集合中，我们以当前列的第一艘战舰作为集合的代表元。并查集的数据类型采用树型，树的根结点即为集合的代表元。为了使查询的效率达到最优，我们进行了路径压缩的优化：首先找到树根，然后将路径上所有结点的父结点改为根，使得树的深度为 1。

问题是，试题不仅要求判别两个结点是否在同一个集合（即两艘战舰是否在同一列），而且还要求计算结点在有序集合的位置（即每一艘战舰相隔同列的第一艘战舰几个位置，简称相对位置）。设

```
var
  Father,Count,Behind:array[1..maxn] of integer;
```

在同一队列中的战舰组成一个树型结构的并查集。

Father [x]——战舰 x 的父指针。Father [x]=x,表明战舰 x 为根结点。路径压缩后,树中所有子结点的父指针指向根结点;

Count [x]——以结点 x 为根的树的结点数;

Behind [x]—战舰 x 在列中的相对位置;

初始时,我们为每一艘战舰建立一个集合,即

Father [x]=x Count [x]=1　Behind [x]=0　(1≤x≤30000)

1. 查找根结点并进行路径压缩

设被查找的结点为 x,计算 x 所在树的根结点,并对该树进行路径压缩。显然,从结点 x 出发,沿 Father 指针一直找到 Father[f]=f,f 即为 x 所在树的根结点。其间经过的各个子结点即为最近被归并的各个集合的代表元,将它们的父指针设为 f 即可完成路径压缩。问题是这些子结点的相对位置是多少。我们分别从每一个子结点 i 出发,沿 Father 指针一直找到根,累计其间经过的每一个子结点的 Behind 值,其和即为 Behind[i],如图 2.2 所示。

```
function Find_Father(x:integer):integer;        {查找结点 x 所在树的根结点,并对该树进行路径压缩}
var i,j,f,next:integer;
begin
  i←x;                                          {找出结点 x 所在树的根结点 f}
  while Father[i]<>i do i←Father[i];
    f←i;
  i←x;
  while i<>f do                                 {按照自下而上的顺序处理 x 的祖先结点}
    begin
    next←Father[i];
    Father[i]←f;                                {把结点 i 的父结点设为 f,完成路径压缩}
    j←next;
    repeat
      Behind[i]←Behind[i]+Behind[j];            {迭代求出路径上每一个子结点相对于 f 的相对位置}
      j←Father[j];
    until Father[j]=j;
    i←next;
  end;{while}
  find_Father←f;                                {返回 x 所在树的根结点}
end;{Find_Father}
```


试读结束：需要全本请在线购买：　www.ertongbook.com