普通高等教育“十一五”国家级规划教材

教育部“高等学校教学质量与教学改革工程”立项项目

俞经善 王宇华 于金峰 等编著

朴秀峰 审

ACM程序设计竞赛基础教程

计算机科学与技术专业实践系列教材

清华大学出版社

普通高等教育“十一五”国家级规划教

计算机科学与技术专业实践系列教材

ACM程序设计竞赛基础教程

俞经善 王宇华 于金峰 等编著

朴秀峰 审

清华大学出版社

北京

内 容 简 介

本书以循序渐进的方式对 ACM 程序设计竞赛中所涉及的基本题型和知识点进行了综合的介绍。全书共分 9 章，包括基础知识讲解、典型题目分析和算法设计，每道例题均给出完整的源程序作为参考。内容涵盖了基础算法、数据结构、字符串、搜索、图论、动态规划、组合数学和初等数论等。

本书内容全面，针对性强，言简意赅，讲解透彻，通俗易懂，图例丰富，所有源代码均可进行评测。本书作为 ACM 程序设计竞赛的培训教程，不仅为大学生们提供了竞赛入门的指导，而且对参赛学生拓展解题思路和提高训练水平也有很大的帮助。本书也可供喜爱程序设计的学生以及从事算法设计的教师学习参考。

图书在版编目（CIP）数据

ACM 程序设计竞赛基础教程/俞经善等编著. —北京：清华大学出版社，2010.10
（计算机科学与技术专业实践系列教材）
ISBN 978-7-302-23492-0

Ⅰ. ①A…　Ⅱ. ①俞…　Ⅲ. ①程序设计—教材　Ⅳ. ①TP311.1

中国版本图书馆 CIP 数据核字(2010)第 155611 号

责任编辑：张瑞庆　顾　冰
责任校对：李建庄
责任印制：李红英

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京密云胶印厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185×260　　**印　　张**：13.75　　**字　　数**：337 千字
版　　次：2010 年 10 月第 1 版　　**印　　次**：2010 年 10 月第 1 次印刷
印　　数：1～3000
定　　价：25.00 元

产品编号：038584-01

序

从1970年开始，ACM/ICPC赛事就影响着计算机与信息专业的许多大学生，引导着他们应用计算机技术展示自己分析问题解决问题的才能。哈尔滨工程大学于2005年开始积极投身于ACM/ICPC活动中。时至今日，令我欣慰的是，从仅有的几名程序设计爱好者到如今上百人参与集训的规模，我们的校代表队已形成了一套自己的训练方法。

在多年的磨炼中，我看到我们的队员走了一些弯路，也经历了一些波折。经过不断的尝试、努力，以及与其他高校参赛选手积极的交流，他们渐渐成长起来。今天，他们把自己的经验编辑成为一本系统的教材，既是对自己多年来训练学习的总结，也为了使更多的ACM/ICPC爱好者有“据”可依。

哈尔滨工程大学举办过多种形式的区域ACM/ICPC赛事，包括省级区域赛、东北四省区域赛、亚洲区域赛等。2010年，主办了第34届ACM/ICPC全球总决赛。这是我们举办的一次最高等级的ACM/ICPC赛事。赛事获得了圆满成功，得到了ACM/ICPC组委会以及世界各国参赛队员的一致好评。作为东北地区组委会，我们每年都会积极的展开省赛，东北地区赛以及其他各种赛事。所谓的不进沙场，一切皆为纸上谈兵。校代表队要在各种赛事中磨炼才能更坚毅。

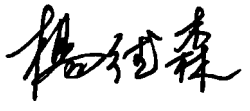
ACM/ICPC赛事活动是大学生自己的活动。学生们通过这些活动全面提高了自己的能力。从组织训练、参加比赛、赛队管理及举办比赛，都有学生自己参与。我们希望看到在一个十分自由的学术氛围里，学生可以放开手脚，自主地学习、创新。在ACM/ICPC竞赛中我们能够得到的不仅仅是算法上的知识，还有学生的动手能力，创造能力以及团队合作和组织能力等。这种能够全面培养学生各方面能力的赛事实属不多。我们希望借助这样的赛事，来提高学生的综合素质和专业技能。

“教书育人”四个掷地有声的字是我们永远的教学宗旨。我们时常会思索，应该教什么样的书，育什么样的人。从多年的国际教育研讨会上，不难发现，众多世界顶级的高等院校纷纷将创新性人才培养作为教育的重中之重，这和我们的教学理念是不谋而合的。几年来，我校ACM代表队不仅在各种ACM赛事中收获了奖牌，而且在各种科技创新活动中也非常活跃。他们开发的在线考试系统和Online Judge系统，在实际使用中表现了先进的设计思想、良好的人机环境和可靠的软件质量。

希望本书的出版可以引起更多程序设计爱好者的兴趣，可以成为ACM/ICPC参赛队员道路上的一块垫脚石。

第34届ACM/ICPC全球总决赛执行主席

哈尔滨工程大学副校长



前　言

ACM 国际大学生程序设计竞赛(ACM International Collegiate Programming Contest, ACM/ICPC),其目的是使大学生充分展示分析问题和运用计算机解决问题的能力。与其他程序设计竞赛相比,ACM/ICPC 题目难度更大,更强调算法的高效性,它在一定程度上要求参赛学生要以最佳的方式解决指定的命题。它几乎涉及所有与计算相关的知识:计算机科学、程序设计、数据结构、操作系统、算法分析与设计、离散数学、数论和图论等。其题目并无现成的固定解法,这就要求参赛学生具有较强的创造力。ACM/ICPC 具有严格、严谨而客观的评判规则,通过评判系统运行严格的测试数据来验证,因此 ACM/ICPC 竞赛成绩的客观、公正性得到公认,极少引起争议。

自 2005 年开始,ACM/ICPC 哈尔滨工程大学代表队便开始有针对性的训练。并于 2007 年秋季将 ACM/ICPC 的实践形式引入教学,开设了 ACM/ICPC 竞赛入门课程。每年分层次的暑期集训,吸引了大批对算法、程序设计等感兴趣的学生。同时,代表队的实力也在逐年提高,多次在 Regional 地区赛上摘得奖牌。作为第 34 届 ACM/ICPC 全球总决赛的举办方,哈尔滨工程大学之前成功举办过黑龙江省赛、东北(黑龙江、吉林、辽宁和内蒙古自治区)地区赛和 Regional 等不同层次的多场比赛。在培训队员,参加比赛,组织比赛过程中,我们收获了许多经验。多年来的教学实践和培训经历,使我们逐渐形成了一套适合自己学校学生的训练方法。

为了使更多的学生更加直观地了解 ACM/ICPC 竞赛并且投身其中,提高学生用计算机解决问题的能力,现将我校代表队历年培训内容的精华整理编写成书。本书综合、全面地介绍计算机程序设计竞赛中的基本题型和相关知识,内容涵盖基础算法、数据结构、字符串、搜索、图论、组合数学和初等数论等知识点。将竞赛知识点与竞赛题型结合,通过对基础知识讲解、典型题目分析和算法实现,使读者能够对问题有深入直观的了解。同时,每道例题均给出完整的源程序作为参考,以帮助读者加深对算法的理解。

本书提供配套网站(http://acm.hrbeu.edu.cn),读者可以从中下载到所有书中题目的源代码用来参考。并在 HEU Online Judge(http://acm.hrbeu.edu.cn)上开设包括本书中出现的所有题目的评测专区,供读者提交、评测自己的程序以检验解题结果。同时,还开辟专门的论坛以方便沟通、交流和讨论。

本书的出版是我们数届新老队员辛勤劳动的结晶,代表了大家共同的心愿。为本书的形成做出重要贡献的有张文涛、李超、周伟、周松、刘俊宏、尤波、张兴彪、常智、吴青松、白一岚、韩佳彤、徐嘉明、王熙魁、严鑫、陆路、刘俊峰、胡光、袁喆。在本书形成的后期,张文涛、胡光、李超和白一岚对其内容和文字上进行了处理。另外,本书的形成也得益于方效林、玄世昌、莫锡昌和盛中华的早期工作。正是由于他们的辛勤劳动才使得本书得以出版,在此我们表示衷心的感谢!

吴文虎教授说过:“竞赛不是目的,是推动普及的手段”。作为 ACM/ICPC 竞赛活动的参与者,我们也经常思索这个问题:竞赛的目的是什么呢?我们的体会是:竞赛是手段,育

人是目的。最根本的出发点还得回到教育本身。我们希望借助比赛来从另一个方面施教，带给学生另外一些收获，如合作与竞争、团队与个人、理论与实践、责任与信任，以此来弥补一些现阶段教育的不足。

由于作者能力和认识有限，出现不足和错误在所难免，诚盼广大同行批评指正！在众多的程序设计竞赛相关书籍中，希望我们抛出的这块“砖”，能够引出更多的“玉”！

ACM/ICPC 哈尔滨工程大学代表队
俞经善
2010 年 9 月

目　录

第1章 基础算法

1.1 分治

分治算法的基本思想是将一个规模为 N 的问题分解为 K 个规模较小的子问题，这些子问题相互独立且与原问题性质相同。求出子问题的解，就可得到原问题的解。

例 1-1 计数问题

1. 题目描述

给定两个数 a 和 b，计算出 1 在 a 和 b 之间出现的次数。例如，如果 $a=1024$，$b=1032$，那么 a 和 b 之间的数就是：

1024 1025 1026 1027 1028 1029 1030 1031 1032

则有 10 个 1 出现在这些数中。

输入：

输入不会超过 500 行。每一行有两个数 a 和 b，a 和 b 的范围是 $0<a,b<100\ 000\ 000$。输入两个 0 时程序结束，两个 0 不作为输入样例。

输出：

对于每一对输入的 a 和 b，输出一个数，代表 1 出现的个数。

样例输入：

```
1 10
44 497
346 542
1199 1748
1496 1403
1004 503
1714 190
1317 854
1976 494
1001 1960
0 0
```

样例输出：

```
2
185
40
666
113
105
```

1133
512
1375
1256

2. 解题思路

本题要求出 1 在两个数 a 和 b 之间出现的次数。可以由分治算法的思想，先求出 1 在 0～a 之间出现的次数，再求出 1 在 0～b 之间出现的次数，然后两者相减即可。现在的问题转换为如何求出 1 在 0～a 之间出现的次数。

将 0～197 的数列出来后可以看出规律：

① 可以求出 1 在 190～197 之间出现的次数，然后对于 0～189。190～197 中 1 在个位数上出现了 1 次。

② 个位考虑完后，直接考虑 197/10－1(即 18)中 1 出现的次数，同时考虑到，数字减小了，每一位的权值会增加，也就是说每一个数字出现的次数会增加 10 倍。例如，现在的 1，是原来 10～19 之间的所有的 1，即权值变为了原来的 10 倍。

3. 参考程序

```
/*本算法算出了 0~9 在 a 和 b 之间分别出现的次数,取答案时直接取 d[1]即可*/
#include<iostream>
using namespace std;
/*******************************************/
const int N=11;
int d[N];                                   //d[N]中存储数字 0~9 分别出现的次数
int value;                                  //记录相应的权值变化
void deal(int n);
/*******************************************/
void deal(int n)
{
    if(n<=0)return;
    int one, ten;                           //one,ten 分别表示个位和十位
    one=n%10;
    n/=10;
    ten=n;

    for(int i=0; i<=one; i++)               //将个位上出现的数统计下来
        d[i]+=value;
    while(ten)
    {
        d[ten%10]+=(one+1)*value;
        ten/=10;
    }
    for(int i=0; i<10; i++)
        d[i]+=value*n;
    d[0]-=value;                            //将第一位是 0 的情况排除
```

```
    value*=10;                                    //权值变化,变为原来的10倍
    deal(n-1);
}
/*******************************************/
int main()
{
    int a, b;
    while(cin>>a>>b)
    {
        if(a==0&&b==0)break;
        if(a<b){int tmp=b; b=a; a=tmp;}           //将较大值存入a中,较小值存入b中
        for(int i=0; i<10; i++)                   //初始化操作
            d[i]=0;
        /*处理过程*/
        value=1;
        deal(a);
        value=-1;                 //此处value=-1是为了求出最后的答案deal(a)-deal(b)
        deal(b-1);
        /*输出结果*/
        cout<<d[1]<<endl;
    }

//  system("pause");
    return 0;
}
```

1.2 递归

程序调用自身的编程技巧称为递归(recursion)。

一个过程或函数在其定义或说明中又直接或间接地调用自身的一种方法,通常可以把一个大型复杂的问题层层转化为一个与原问题相似的、规模较小的问题来求解。递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算,大大地减少了程序的代码量。递归的能力在于用有限的语句来定义对象的无限集合。一般来说,递归需要有边界条件、递归前进段和递归返回段。当边界条件不满足时,递归前进;当边界条件满足时,递归返回。

例 1-2 汉诺塔问题

1. 题目描述

汉诺塔(又称河内塔)问题其实是印度的一个古老的传说。

开天辟地的神勃拉玛(和中国的盘古差不多的神)在一个庙里留下了3根金刚石的棒,第一根上面套着64个圆的金片,最大的一个金片在底下,其余金片一个比一个小,依次叠上去。庙里的众僧不倦地把第一根棒上的金片一个个地搬到第三根棒上,规定:可利用中间的第二根棒作为帮助,但每次只能搬一个金片,而且大的不能放在小的上面。移动圆片的次

数的计算结果非常恐怖：18 446 744 073 709 551 615 次，众僧们即便是耗尽毕生精力也不可能完成金片的移动。

输入：

输入一个正整数 n，表示有 n 个盘片在第一根柱子上。

输出：

输出操作序列，格式为 move t from x to y。每个操作一行，表示把 x 柱子上的编号为 t 的盘片挪到柱子 y 上。柱子编号为 a、b、c，你要用最少的操作把所有的盘子从 a 柱子上转移到 c 柱子上。

样例输入：

```
3
```

样例输出：

```
move 1 from a to c
move 2 from a to b
move 1 from c to b
move 3 from a to c
move 1 from b to a
move 2 from b to c
move 1 from a to c
```

2. 解题思路

其实算法非常简单，当盘子的个数为 n 时，移动的次数应等于 2^n-1（有兴趣的读者可以自己证明）。后来一位美国学者发现一种出人意料的简单方法，只要轮流进行两步操作就可以实现。首先把 3 根柱子按顺序排成“品”字形，把所有的圆盘按从大到小的顺序放在柱子 A 上，根据圆盘的数量确定柱子的排放顺序：若 n 为偶数，按顺时针方向依次摆放 A、B、C；若 n 为奇数，按顺时针方向依次摆放 A、C、B。

① 按顺时针方向把圆盘 1 从现在的柱子移动到下一根柱子，即当 n 为偶数时，若圆盘 1 在柱子 A，则把它移动到 B；若圆盘 1 在柱子 B，则把它移动到 C；若圆盘 1 在柱子 C，则把它移动到 A。

② 接着把另外两根柱子上可以移动的圆盘移动到新的柱子上。即把非空柱子上的圆盘移动到空柱子上，当两根柱子都非空时，移动较小的圆盘。这一步没有明确规定移动哪个圆盘，你可能以为会有多种可能性，其实不然，可实施的行动是唯一的。

③ 反复进行①、②操作，最后就能按规定完成汉诺塔的移动。

所以，方法非常简单，就是按照移动规则向一个方向移动圆的金片。例如 3 阶汉诺塔的移动为：A→C，A→B，C→B，A→C，B→A，B→C，A→C。

汉诺塔问题也是程序设计中的经典递归问题，下面给出递归和非递归的不同实现源代码。

3. 参考代码

```
#include<fstream>
#include<iostream>
```

```
using namespace std;
void Move(int n,char x,char y)
{
cout<<"move "<<n<<" from "<<x<<" to "<<y<<endl;
}
void Hannoi(int n,char a,char b,char c)
{
          if(n==1)
              Move(1,a,c);
        else
    {
          Hannoi(n-1,a,c,b);
            Move(n,a,c);
            Hannoi(n-1,b,a,c);
    }
}
int main()
{
        int n;
        scanf("%d",&n);
        Hannoi(n,'a','b','c');

        return 0;
}
```

1.3 枚举

枚举算法就是列举出问题中所有可能的解，然后根据问题的实际意义排除错误答案，最终找到正确答案的过程。下面来分析一道具体可以用到枚举算法的题目。

例 1-3 木棒三角形

1. 题目描述

小 A 家里有很多长度不一样的木棍，有一天他很无聊，便摆弄这些木棒来解闷。小 A 的数学学得很好，所以他想在这些木棒中挑出 3 根来组成一个直角三角形，当然他有可能有很多种选法，所以他还想挑出一个面积最大的。

输入：

输入有多组，每组输入包括两行，第一行输入一个 $n(0\leqslant n\leqslant 100)$，表示小 A 有 n 根木棍，接着一行有 n 个整数($\leqslant 1000$)，表示木棍的长度(长度从小到大给出)。

输出：

输出面积最大的直角三角形的面积，且保留 3 位小数，如果不能组成，输出“My God!”。

样例输入：

```
4
```

```
1 2 3 4
5
2 3 4 5 6
6
3 4 5 6 8 10
2
1 1
```

样例输出：

```
My God!
6.000
24.000
My God!
```

2. 解题思路

看到题目很容易想到，如果能知道把从 n 根木棍中选出 3 根的所有情况的解，则答案也就出来了。

现在主要的问题是怎么用程序来枚举所有情况，已知直角三角形的 3 条边中斜边是最长的，题目给出了一个“长度从小到大给出”的条件，这样可以依次枚举三角形中长度最短、第二长和最长的边，具体实现见以下代码。

3. 参考程序

```
#include<stdio.h>
#include<stdlib.h>
int main(){
    int i,j,k;
    double ans;
    int n;
    int len[110];
    while(scanf("%d",&n)!=EOF){
      for(i=1;i<=n;i++)
          scanf("%d",&len[i]);                          //存储木棍长度
      ans=-1;
      for(i=1;i<=n;i++){                                //枚举最短木棍
            for(j=i+1;j<=n;j++){                        //枚举第二长的木棍
                  for(k=j+1;k<=n;k++){                  //枚举第三长的木棍
                        if(len[i]*len[i]+len[j]*len[j]==len[k]*len[k]){
//如果是直角三角形
                          if(0.5*len[i]*len[j]>ans)     //取最优解
                            ans=0.5*len[i]*len[j];
                        }
                  }
            }
      }
      if(ans==-1)
```

```
            printf("My God!\n");
        else
            printf("%.3lf\n",ans);
    }
}
```

1.4 贪心

所谓贪心算法是指在对问题求解时，总是做出在当前看来是最好的选择。也就是说，不从整体最优上加以考虑，它所做出的仅是在某种意义上的局部最优解。

贪心算法不是对所有问题都能得到整体最优解，但是对范围相当广泛的许多问题能产生整体最优解或者是整体最优解的近似解。

例 1-4 A 公司的烦恼

1. 题目描述

A 公司的计算机管理系统受到了千年虫病毒的攻击，因此 A 公司丢失了向 MS 公司做年终汇报的数据。

A 公司目前掌握的数据是 MS 公司每次公布的公司盈亏报表，而 MS 公司公布盈亏的方式与众不同，它每次都是将连续 5 个月的盈或亏的总和做一次性的公布，因此 A 公司不知道每个月具体的盈亏状况。已知的情况是所有盈利月的盈利固定为 s，而亏损月的亏损固定为 d。

写一个程序，确定 MS 公司是否盈利，若盈利的话，计算可能的盈利最大值。

输入：

输入为两个正整数 s 和 d。

输出：

对于每一组的输入数据，若盈利的话，那么输出可能的盈利最大值；若亏损的话，输出 Deficit。

样例输入：

```
59 237
375 743
200000 849694
2500000 8000000
```

样例输出：

```
116
28
300612
Deficit
```

2. 解题思路

首先每连续 5 个月中至少有哪几个月是亏损的是可以计算出来的。

用贪心的思想可以解决如何安排亏损月份的问题;假设5个月至少亏损3个月,那么对前5个月来说,亏损的月份必定是3、4、5月,这样才能保证这3个亏损月能最大可能地被后面的连续的5个月利用,以减少出现更多的亏损月。

用贪心思想合理的安排12个月的亏损盈利情况,即可解题。

解题步骤如下:

① 计算出连续的5个月至少有哪几个月是亏损的。

② 根据贪心的思想计算全年的盈亏。注意,当至少亏损月份为4个月和5个月时予以特殊的考虑,其余的情况用通用的公式计算盈亏。

3. 参考程序

```
#include<iostream>
using namespace std;
int main()
{
    int s,d;
    while(scanf("%d%d",&s,&d)!=EOF)
    {
        int i,ans;
        for(i=1;i<=5;i++)
            if(s*(5-i)-d*i<0)
                break;                          //枚举出5个月中至少有i个亏损月
        if(i==4)
            ans=3*s-9*d;
        else
            ans=s*(12-2*i)-d*2*i;               //i==5时此公式不适用,但可直接判断全年亏损
        if(i==5||ans<0)                         //"i==5"这个判断条件必须放在"ans<0"前
            printf("Deficit\n");
        else
            printf("%d\n",ans);
    }
}
```