21世纪高等学校规划教材 | 计算机科学与技术

ACM/ICPC 算法训练教程

余立功
南京理工大学ACM/ICPC集训队整理编辑

清华大学出版社

ACM/ICPC 算法训练教程

余立功 主编

清华大学出版社
北京

内 容 简 介

本书针对 ACM /ICPC 国际大学生程序设计竞赛的情况，较为系统和全面地介绍了竞赛中涉及的各种常见知识专题大类。通过专题讲解、赛题分析、源码介绍，重点阐述关于算法设计课程与数据结构课程要求的内容。全书共分为 8 章，分别介绍基础算法、数据结构、动态规划、数学问题、计算几何、搜索算法、图算法和字符串算法问题。内容翔实，每个专题都给出例题，并附有详细的题解代码，供读者边学边练。

本书适合高等院校开展 ACM/ICPC 竞赛训练，也适合 ACM/ICPC 竞赛爱好者、信息学竞赛爱好者、程序设计爱好者学习和实践竞赛中的算法，还适合本科生和研究生对算法和数据结构课程进行深入和拓展，尤其适合完成了 C/C++程序设计、具有一定数据结构和算法基础的学生用于 ACM/ICPC 竞赛入门。

图书在版编目(CIP)数据

ACM/ICPC 算法训练教程/余立功主编. —北京：清华大学出版社，2013.1
21 世纪高等学校规划教材·计算机科学与技术
ISBN 978-7-302-30513-2

Ⅰ. ①A… Ⅱ. ①余… Ⅲ. ①程序设计－算法－高等学校－教材 Ⅳ. ①TP311.1

中国版本图书馆 CIP 数据核字(2012)第 257270 号

责任编辑：闫红梅　李　晔
封面设计：傅瑞学
责任校对：梁　毅
责任印制：王静怡

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课 件 下 载：http://www.tup.com.cn,010-62795954
印 装 者：三河市李旗庄少明印装厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：20.25　　**字　　数**：504 千字
版　　次：2013 年 3 月第 1 版　　**印　　次**：2013 年 3 月第1 次印刷
印　　数：1～3000
定　　价：34.50 元

产品编号：041059-01

编审委员会成员

（按地区排序）

清华大学	周立柱　教授
	覃　征　教授
	王建民　教授
	冯建华　教授
	刘　强　副教授
北京大学	杨冬青　教授
	陈　钟　教授
	陈立军　副教授
北京航空航天大学	马殿富　教授
	吴超英　副教授
	姚淑珍　教授
中国人民大学	王　珊　教授
	孟小峰　教授
	陈　红　教授
北京师范大学	周明全　教授
北京交通大学	阮秋琦　教授
	赵　宏　副教授
北京信息工程学院	孟庆昌　教授
北京科技大学	杨炳儒　教授
石油大学	陈　明　教授
天津大学	艾德才　教授
复旦大学	吴立德　教授
	吴百锋　教授
	杨卫东　副教授
同济大学	苗夺谦　教授
	徐　安　教授
华东理工大学	邵志清　教授
华东师范大学	杨宗源　教授
	应吉康　教授
东华大学	乐嘉锦　教授
	孙　莉　副教授

浙江大学	吴朝晖 教授
	李善平 教授
扬州大学	李 云 教授
南京大学	骆 斌 教授
	黄 强 副教授
南京航空航天大学	黄志球 教授
	秦小麟 教授
南京理工大学	张功萱 教授
南京邮电学院	朱秀昌 教授
苏州大学	王宜怀 教授
	陈建明 副教授
江苏大学	鲍可进 教授
中国矿业大学	张 艳 教授
武汉大学	何炎祥 教授
华中科技大学	刘乐善 教授
中南财经政法大学	刘腾红 教授
华中师范大学	叶俊民 教授
	郑世珏 教授
	陈 利 教授
江汉大学	颜 彬 教授
国防科技大学	赵克佳 教授
	邹北骥 教授
中南大学	刘卫国 教授
湖南大学	林亚平 教授
西安交通大学	沈钧毅 教授
	齐 勇 教授
长安大学	巨永锋 教授
哈尔滨工业大学	郭茂祖 教授
吉林大学	徐一平 教授
	毕 强 教授
山东大学	孟祥旭 教授
	郝兴伟 教授
厦门大学	冯少荣 教授
厦门大学嘉庚学院	张思民 教授
云南大学	刘惟一 教授
电子科技大学	刘乃琦 教授
	罗 蕾 教授
成都理工大学	蔡 淮 教授
	于 春 副教授
西南交通大学	曾华燊 教授

出版说明

随着我国改革开放的进一步深化，高等教育也得到了快速发展，各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，加大了使用信息科学等现代科学技术提升、改造传统学科专业的投入力度，通过教育改革合理调整和配置了教育资源，优化了传统学科专业，积极为地方经济建设输送人才，为我国经济社会的快速、健康和可持续发展以及高等教育自身的改革发展做出了巨大贡献。但是，高等教育质量还需要进一步提高以适应经济社会发展的需要，不少高校的专业设置和结构不尽合理，教师队伍整体素质亟待提高，人才培养模式、教学内容和方法需要进一步转变，学生的实践能力和创新精神亟待加强。

教育部一直十分重视高等教育质量工作。2007 年 1 月，教育部下发了《关于实施高等学校本科教学质量与教学改革工程的意见》，计划实施“高等学校本科教学质量与教学改革工程”（简称“质量工程”），通过专业结构调整、课程教材建设、实践教学改革、教学团队建设等多项内容，进一步深化高等学校教学改革，提高人才培养的能力和水平，更好地满足经济社会发展对高素质人才的需要。在贯彻和落实教育部“质量工程”的过程中，各地高校发挥师资力量强、办学经验丰富、教学资源充裕等优势，对其特色专业及特色课程（群）加以规划、整理和总结，更新教学内容、改革课程体系，建设了一大批内容新、体系新、方法新、手段新的特色课程。在此基础上，经教育部相关教学指导委员会专家的指导和建议，清华大学出版社在多个领域精选各高校的特色课程，分别规划出版系列教材，以配合“质量工程”的实施，满足各高校教学质量和教学改革的需要。

为了深入贯彻落实教育部《关于加强高等学校本科教学工作，提高教学质量的若干意见》精神，紧密配合教育部已经启动的“高等学校教学质量与教学改革工程精品课程建设工作”，在有关专家、教授的倡议和有关部门的大力支持下，我们组织并成立了“清华大学出版社教材编审委员会”（以下简称“编委会”），旨在配合教育部制定精品课程教材的出版规划，讨论并实施精品课程教材的编写与出版工作。“编委会”成员皆来自全国各类高等学校教学与科研第一线的骨干教师，其中许多教师为各校相关院、系主管教学的院长或系主任。

按照教育部的要求，“编委会”一致认为，精品课程的建设工作从开始就要坚持高标准、严要求，处于一个比较高的起点上。精品课程教材应该能够反映各高校教学改革与课程建设的需要，要有特色风格、有创新性（新体系、新内容、新手段、新思路，教材的内容体系有较高的科学创新、技术创新和理念创新的含量）、先进性（对原有的学科体系有实质性的改革和发展，顺应并符合 21 世纪教学发展的规律，代表并引领课程发展的趋势和方向）、示范性（教材所体现的课程体系具有较广泛的辐射性和示范性）和一定的前瞻性。教材由个人申报或各校推荐（通过所在高校的“编委会”成员推荐），经“编委会”认真评审，最后由清华大学出版

社审定出版。

目前，针对计算机类和电子信息类相关专业成立了两个“编委会”，即“清华大学出版社计算机教材编审委员会”和“清华大学出版社电子信息教材编审委员会”。推出的特色精品教材包括：

(1) 21世纪高等学校规划教材·计算机应用——高等学校各类专业，特别是非计算机专业的计算机应用类教材。

(2) 21世纪高等学校规划教材·计算机科学与技术——高等学校计算机相关专业的教材。

(3) 21世纪高等学校规划教材·电子信息——高等学校电子信息相关专业的教材。

(4) 21世纪高等学校规划教材·软件工程——高等学校软件工程相关专业的教材。

(5) 21世纪高等学校规划教材·信息管理与信息系统。

(6) 21世纪高等学校规划教材·财经管理与应用。

(7) 21世纪高等学校规划教材·电子商务。

(8) 21世纪高等学校规划教材·物联网。

清华大学出版社经过三十多年的努力，在教材尤其是计算机和电子信息类专业教材出版方面树立了权威品牌，为我国的高等教育事业做出了重要贡献。清华版教材形成了技术准确、内容严谨的独特风格，这种风格将延续并反映在特色精品教材的建设中。

清华大学出版社教材编审委员会

联系人：魏江江

E-mail：weijj@tup.tsinghua.edu.cn

序

ACM 国际大学生程序设计竞赛(ACM/ICPC) 是由国际计算机界历史悠久、颇具权威性的组织 ACM 学会主办,是世界上公认的规模最大、水平最高的国际大学生程序设计竞赛,其目的旨在使大学生运用计算机来充分展示自己分析问题和解决问题的能力。因历届竞赛都荟萃了世界各地的精英,云集了计算机界的"希望之星",而受到国际各知名大学的重视,并受到全世界各著名计算机公司的高度关注,成为世界各国大学生最具影响力的国际级计算机类的赛事。

南京理工大学参与该项赛事 10 年,获得亚洲区银奖 10 个、铜奖 17 个。同时在训练中也积累了一些训练的资料。南京理工大学 ACM/ICPC 集训队根据多年训练积累,整理的《ACM/ICPC 算法训练教程》适合 ACM/ICPC 初学者及具有一定基础的计算机算法和编程爱好者。适合作为 ACM/ICPC 训练教材,本科及研究生算法与数据结构类课程的参考教材。

本书资料来自南京理工大学 ACM/ICPC 集训队训练讲义,所选例题分别来自于南京理工大学 OPEN JUDGE(NJUSTOJ)地址为 http://icpc.njust.edu.cn,北京大学 ONLINE JUDGE(POJ) 地址为 http://acm.poj.org,浙江大学 ONLINE JUDGE(ZOJ) 地址为 http://acm.zju.edu.cn。由于主要针对 ACM/ICPC 算法训练和高级数据结构和算法训练,故加强了方法的应用性,而简化了理论论述。每个知识点都从网上题库选择相应例题进行阐述,并附有正确的代码。

本书参考了部分网上公开的资料,本书例题代码来自南京理工大学 ACM/ICPC 集训队代码库。集训队队员刘铁俊、朱艺楠、华嘉炜、黄典典、孙健波、周宇哲、薛斌、崔嵬、符瑞盛等参与了专题整理和审校工作。因成书仓促,错误在所难免,望批评指正。

编　者

2012 年 8 月

目 录

第1章 基础算法

在开始 ACM/ICPC 算法训练之前，首先明确几个概念：什么是算法？什么是程序？

算法(Algorithm)是指解决问题的一种方法或一个过程。严格地讲，算法是由若干条指令组成的有穷序列。对于计算机科学来说，算法的概念至关重要。计算机软硬件的一切问题几乎都能够落实到具体的算法。对于 ACM/ICPC(国际大学生程序设计竞赛)而言，解题的核心是算法设计。只有设计了巧妙的算法，写出程序，才能解决千变万化的问题。

1. 算法的特征

算法应该具有以下 4 个重要特征。

1) 输入

有零个或多个外部提供的量作为算法的输入，以刻画运算对象的初始情况。所谓 0 个输入是指算法本身定出了初始条件。

2) 输出

算法产生一个或多个输出，以反映对输入数据加工后的结果。没有输出的算法是毫无意义的。

3) 确切性

组成算法的每条指令是清晰的、无歧义的。

4) 有穷性

算法中的每条指令的执行次数和执行时间都是有限的。

程序(Program)是算法用某种程序语言的具体实现。对于一个大的软件系统而言，有时候，一个好的算法将决定整个系统的成败。因此说，算法是一个程序的灵魂。

2. “好”算法的衡量标准

一般来说，“好”算法应该满足以下标准：

1) 正确性

算法应该满足具体问题的需求，通过某种输入能够得到特定的输出。解决相应的问题。

2) 可读性

算法是联系数学模型和程序的桥梁，可读性好有助于人对算法的理解。现代计算机软件设计，往往需要团队协同工作，代码可读性强的算法能够促进团队交流，提高开发效率。

3) 健壮性

算法对于异常情况要有充分的考虑和处理。

4）效率高和存储量少

算法需要在一定的时间结束，即具有可以接受的时间复杂度。同时算法必须在一定的内存空间运行。ACM/ICPC 竞赛题正是从以上 4 个方面对选手进行多方面考查。为了设计出高效可行的程序，就必须获得一个既有效又优美的算法。设计出满足以上 4 个特性的算法，完成程序解决问题，本身也是竞赛训练需要达到的目的。

本章将对一些基本和常用的算法设计思想进行探讨。选用相关的竞赛赛题进行举例。

1.1 枚举法

专题讲解

枚举法，又称穷举法。指的是从可能的解的集合中一一枚举各元素，用给定的检验条件判定哪些是无用的，哪些是有用的。能使命题成立，即为其解。这是最直观的算法，也是最容易想到的方法。

枚举法本质上属于搜索算法。但是它与搜索有所不同，因为适用于枚举法求解的问题必须满足两个条件：

（1）可预先确定解的个数 n；

（2）解变量 $A1,A2,\cdots,An$ 的值的可能变化范围预先确定。

例如某问题的解变量有 3 个：$A1,A2,A3$。其中：

$A1$ 解变量值的可能范围 $A1 \in \{X11,X12,X13\}$

$A2$ 解变量值的可能范围 $A2 \in \{X21,X22,X23\}$

$A3$ 解变量值的可能范围 $A3 \in \{X31,X32;X33\}$

则问题的可能解有 27 个：$(A1,A2,A3) \in \{(X11,X21,X31),(X11,X21,X32),(X11,X21,X33)$

$(X11,X22,X31),(X11,X22,X32),(X11,X22,X33)$

$\vdots$

$(X13,X23,X31),(X13,X23,X32),(X13,X23,X33)\}$

在上述可能解的集合中，满足题目给定的检验条件的解元素，即为问题的解。

一般来说，如果预先对问题确定了解的个数以及各个解的值域，则可以利用 for 语句和条件判断语句逐步求解或证明：

```
1.  for A1 : = X11 to X11 do
2.          ⋮
3.   for Ai : = Xi1 to Xiq do
4.          ⋮
5.       for An : = Xn1 to Xnk do
6.           if(A1,…, Ai ,…, An 满足检验条件)
7.               输出问题的解(A1,…, Ai ,…, An)
```

枚举法的特点是算法简单，对于可确定解的值域又一时找不到其他好的算法时就可用它。但枚举法又是一种比较笨拙、原始的方法，运算量大是它的弱点。算法的复杂度是指数

级的。对于枚举法也不要过于悲观，从全局观点使用枚举法，计算量容易过大。但是，如果能够排除那些明显不属于解集的元素，在局部地方使用枚举法，其效果会十分显著。

例题 1.1 Safecracker (ZOJ 1403)。

题意简述

密码序列由一系列大写字母组成，在解密序列不唯一的情况下，按字典序输出最后一个。解密公式：

$$v - w^2 + x^3 - y^4 + z^5 = \text{target}$$

其中，target 是数字，由题目给出。v、w、x、y、z 属于同一个集合且各不相同，该集合由题目给出，由 26 个大写字母中的任意 5～12 个组成。每个字母对应的数值，以此类推($A=1, B=2, \cdots, Z=26$)。

思路

这是一道典型的枚举题。题目中，解的值域已经确定，解元素中的 v、w、x、y、z 都是题目给定集合中的一个元素。尽管枚举法的算法复杂度是指数级的，但鉴于给定集合最多只有 12 个元素，完全可以直接用枚举。另外，题目中的约束条件，v、w、x、y、z 各不相同，明显不属于解集的元素可以很容易地通过判断来排除。需要注意的是，题目求的密码序列是按字典序输出最后一个，所以需要事先将集合元素排好序。

代码

```
1.  /*  Language:C++Run Time(ms): 70 Run Memory(KB): 188 */
2.  #include<iostream>
3.  #include<cmath>
4.  #include<cstdio>
5.  #include<algorithm>
6.  #include<cstring>
7.  using namespace std;
8.  char letters[15];
9.  int value[15], target;
10. void process(int len)                    //主要计算过程,枚举
11. {
12.     int a, b, c, d, e;
13.     for (a = 0; a < len; a++)
14.         for (b = 0; b < len; b++)
15.             if (a != b)                  //排除明显不属于解集的元素,下同
16.                 for (c = 0; c < len; c++)
17.                     if (a != c && b != c)
18.                         for (d = 0; d < len; d++)
19.                             if (a != d && b != d && c != d)
20.                                 for (e = 0; e < len; e++)
21.                                     if (a != e && b != e && c != e && d != e)
22.                                      //找到了解
23.                                         if (value[a] - pow(value[b], 2.0)
24.                                               + pow(value[c], 3.0)
25.                                               - pow(value[d], 4.0)
26.                                               + pow(value[e], 5.0)
27.                                               == target) {
```

```
28.                                                    printf("%c%c%c%c%c\n",value[a]
        + 'A' - 1,value[b] + 'A' - 1,value[c] + 'A' - 1,value[d] + 'A' - 1,value[e]
        + 'A' - 1);
29.                                                    return;
30.                                                }
31.     printf("no solution\n");
32. }
33. bool compare(int a, int b)
34. {
35.     return a>b;                                    //降序排列,如果改为 return a<b,则为升序
36. }
37. int main()
38. {
39.     int i;
40.     while (~scanf("%d %s", &target, letters)) {
41.         if (target == 0 && strcmp(letters, "END") == 0) {
42.            return 0;
43.         }
44.         i = 0;
45.         while (letters[i]) {
46.             value[i] = letters[i] - 'A' + 1;
47.             i++;
48.         }
49.         sort(value, value + i, compare);
50.         process(i);
51.     }
52.     return 0;
53. }
```

1.2 递归法

专题讲解

设一个未知函数 f,用其自身构成的已知函数 g 来定义

$$f(n)=\begin{cases} g(n,f(n-1)) & n>0 \\ a & n=0 \end{cases}$$

为了定义 $f(n)$,必须先定义 $f(n-1)$;为了定义 $f(n-1)$,又必须先定义 $f(n-2)$ ……上述这种用自身的简单情况来定义自己的方式称为递归定义。

一个递归定义必须是有确切含义的,也就是说,必须一步比一步简单,最后是有终结的,决不能无限循环下去。也就是说递归必须满足 3 个要素:

- 递归边界。

递归算法必须有一个边界出口,能够结束程序。

- 参数收敛。

每次调用的时候,涉及递归调用的参数都是收敛于递归边界的。

- 自身调用。

递归在于调用同名方法，这也是其为何简单的原因。

在 $f(n)$ 的定义中，当 n 为 0 时定义一个已知数 a，是最简单的情况，称为递归边界，它本身不再使用递归的定义。

说起递归，首先联想到的应该就是那个经典的 Hanoi 塔问题。虽然这个问题也有非递归算法，但多年来，它已经是递归算法的典例了。

例题 1.2 Hanoi 塔问题。

题意简述

设 a、b、c 是 3 个塔座。开始时，在塔座 a 上有一叠共 n 个圆盘，这些圆盘自下而上，由大到小地叠在一起。各圆盘从小到大编号为 1，2，…，n，现要求将塔座 a 上的这一叠圆盘移到塔座 b 上，并仍按同样顺序叠置。在移动圆盘时应遵守以下移动规则：

(1) 每次只能移动 1 个圆盘。

(2) 任何时刻都不允许将较大的圆盘压在较小的圆盘之上。

(3) 在满足移动规则 1 和 2 的前提下，可将圆盘移至 a、b、c 中任一塔座上。

思路

Hanoi(n，a，b，c) 表示将塔座 a 上自上而下，由大到小叠在一起的 n 个圆盘依移动规则移至塔座 b 上并仍按同样顺序叠排。在移动过程中，以塔座 c 作为辅助塔座。move(n，a，b) 表示将塔座 a 上编号为 n 的圆盘移至塔座 b 上。

代码

```
1.  void hanoi(int n, int a, int b, int c)
2.  {
3.     if(n>0) {
4.       hanoi(n-1,a,c,b);
5.       move(n,a,b);
6.       hanoi(n-1,c,b,a);
7.     }
8.  }
```

如果 Hanoi 问题不用递归进行设计，很难想象算法将会有多么复杂，但使用递归设计之后，方法变得非常简单。

递归算法和递推十分相似。与递推一样，每一个递归定义都有其边界条件。但不同的是，递推是内边界条件出发。通过递归式求 $f(n)$ 的值，从边界到求解的全过程十分清楚；而递归则是从函数自身出发来达到边界条件。

在通往边界条件的递归调用过程中，系统用堆栈把每次调用的中间结果(局部变量和返回地址值)保存起来，直至求出递归边界值 $f(0)=a$。然后返回调用函数。返回过程中，中间结果相继出栈恢复，$f(1)=g(1,a)->f(2)=g(2,f(1))->\cdots$直至求出 $f(n)=g(n,f(n-1))$。

由上面递归的定义可见，由于递归函数常常使用系统堆栈，递归算法的效率往往很低，费时和费内存空间。但是递归也有其长处，它能使一个蕴涵递归关系且结构复杂的程序简洁精练，增加可读性。特别是在难于找到从边界到解的全过程的情况下，如果把问题推进一步，其结果仍维持原问题的关系，则采用递归算法编程比较合适。