

国际大学生 程序设计竞赛例题解 (二)

广东省大学生程序设计竞赛试题
(2003-2005年)

郭嵩山 黎俊瑜 林祺颖 著



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

<http://www.phei.com.cn>

国际大学生程序设计竞赛例题解

(二)

广东省大学生程序设计竞赛试题
(2003—2005 年)

郭嵩山 黎俊瑜 林祺颖 著

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书收录了第 1 届至第 3 届广东省大学生程序设计竞赛（2003—2005 年 ACM/ICPC 广东省赛）和第 28 届 ACM 国际大学生程序设计竞赛（ACM/ICPC）亚洲预赛广州赛区（2003 年）的全部试题、完整的测试数据和答案。为了方便读者学习，本书对每个题目作了详尽的题目分析并详细地讲解其算法实现的原理，同时提供了完善的标准程序及其程序分析供读者参考。书中还提供了竞赛时评判用的基本测试数据，以方便读者测试自行完成上述题目的结果。随书还附有光盘，存放竞赛时评判用全部的测试数据，以便于有更高、更严格要求的同学能利用规模更大的测试数据进行训练和学习。

本书所提供的题目都是原创题，题目构思新颖，所涉及到的算法知识面广，其涉及的算法知识基本上覆盖大学计算机类本科专业所学到的基本算法。本书可以作为高等院校有关专业的研究生和本科学子参加国际大学生程序设计竞赛的辅导教材，也可作为高等院校有关专业相关课程的教学参考书和例题集，并且适于作为中学青少年信息学奥林匹克竞赛省级及省级以上优秀选手备战信息学奥林匹克竞赛的培训教材及训练题集。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

国际大学生程序设计竞赛例题解. 2, 广东省大学生程序设计竞赛试题: 2003—2005 年 / 郭嵩山, 黎俊瑜, 林祺颖著. —北京: 电子工业出版社, 2006.5

ISBN 7-121-02612-0

I. 国… II. ①郭… ②黎… ③林… III. 程序设计—竞赛—高等学校—解题 IV. TP311.1-44

中国版本图书馆 CIP 数据核字（2006）第 046041 号

责任编辑：龚立堇

印 刷：北京东光印刷厂

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

经 销：各地新华书店

开 本：787×1092 1/16 印张：20 字数：510 千字

印 次：2006 年 5 月第 1 次印刷

印 数：5000 册 定价：33.00 元（含光盘 1 张）

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件至 zlt@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

前 言

ACM 国际大学生程序设计竞赛 (ACM International Collegiate Programming Contest, 简称 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 (Association for Computer Machinery) 主办的, 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。该项竞赛从 1970 年举办至今已历 30 届, 因历届竞赛都荟萃了世界各大洲的精英, 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。ACM 所颁发的获奖证书也为世界各著名计算机公司、各知名大学所认可。该项竞赛分区域预赛和世界决赛两个阶段进行, 各预赛区第一名获得参加世界决赛的资格, 世界决赛安排在每年的 3~4 月举行, 而区域预赛安排在上一年 9~12 月在各大洲举行。ACM/ICPC 的区域预赛是规模很大、范围很广的赛事。以 2005 年为例, 全世界有 1702 所大学的 6007 支参赛队, 在六大洲 71 个国家 (或地区) 的 31 个赛站中争夺全球总决赛的 84 个名额, 其激烈程度可想而知。

与其他编程竞赛相比, ACM/ICPC 题目难度更大, 更强调算法的高效性, 不仅要解决一个指定的命题, 而且必须以最佳的方式解决指定的命题; 它涉及的知识面广, 与大学计算机系本科及研究生的课程直接关联, 如程序设计、离散数学、数据结构、人工智能、算法分析与设计等; 对数学要求更高; 由于采用英文命题, 对英语要求较高; ACM/ICPC 采用 3 人合作、共用一台电脑, 所以它更强调团队协作精神; 由于许多题目并无现成的算法, 需要具备创新的精神; ACM/ICPC 不仅强调学科的基础, 更强调全面素质和能力的培养。由于 ACM/ICPC 是采用 5 小时全封闭式竞赛, 参赛队员与外界完全隔离, 独立完成, 是参赛队员实际能力的真实表露, 其成绩可信度甚高。ACM/ICPC 又是一种“开卷考试”, 可以带任何书籍、资料, 甚至源程序代码清单 (但不能带电子媒体), 不需要死背算法, 强调的是算法的灵活运用。与其他计算机竞赛 (如软件设计、网站设计等) 相比, ACM/ICPC 有严谨而客观的评判规则 (严格的数据测试), 排除了因评委的主观因素而造成评审不公平的现象。所以, 对 ACM/ICPC 成绩的争议较少。

中山大学自 1997 年首次参加 ACM/ICPC 亚洲区预赛以来的 9 年中, 每年都派出多支队共参加过 25 次亚洲区预赛, 成绩全部在前 6 名之列, 其中有 16 次进入三甲, 夺得 3 次冠军 (1999 年台北, 2002、2003 年高雄)、4 次亚军 (2000 年香港、筑波, 2003 年北京、广州)、9 次季军 (1998—2000 年上海, 2001 年达卡, 2002 年北京, 2003 年高雄, 2004 年马尼拉, 2005 年台北、北京)。中山大学的参赛队曾 7 次进入全球总决赛 (1999—2001 年, 2003—2006 年): 2000 年在美国佛罗里达州奥兰多市举行的第 24 届全球总决赛中夺得了第 11 名的好成绩; 2001 年在加拿大温哥华市举行的第 25 届全球总

决赛中首获铜牌（世界第 14 名）；2003 年在美国洛杉矶市好莱坞举行的第 27 届全球总决赛中获得世界第 8 名并首获银牌的好成绩，跻身世界八强之列；2004 年在捷克布拉格市举行的第 28 届全球总决赛中获得世界第 11 名并再获铜牌，且在中国内地高校中排名第一；2005 年在上海举行的第 29 届全球总决赛中获得世界第 17 名；2006 年 4 月在美国得克萨斯州圣安东尼奥市举行的第 30 届 ACM/ICPC 全球总决赛中获得世界第 19 名。

为了帮助高等院校的大学生们备战国际大学生程序设计竞赛，帮助他们提高程序设计水平和培养更强的分析问题与解决问题的能力，我们编写了这套《国际大学生程序设计竞赛例题解》。本书是作为这套《国际大学生程序设计竞赛例题解》的第二册，编程所用的语言是 C++，全书共分 5 章。本书收录了第 1 届至第 3 届广东省大学生程序设计竞赛（2003—2005 年 ACM/ICPC 广东省赛）、2005 年中山大学程序设计竞赛预选赛（ZSUCPC 2005 年预选赛）和第 28 届 ACM 国际大学生程序设计竞赛（ACM/ICPC）亚洲预赛广州赛区（2003 年）的全部试题、完整的测试数据和答案。为了方便读者学习，本书对每个题目作了详尽的题目分析并详细地讲解其算法实现的原理，同时提供了完善的标准程序及其程序分析供读者参考。本书提供了基本测试数据以方便读者测试自行完成上述题目的结果。随书附带的光盘中存放所有例题中完整的测试数据，以便于有更高、更严格要求的同学，能利用规模更大的测试数据进行训练和学习用。参与上述竞赛命题的有：吴汉荣、崔昊、孔颖、李志业、金涛、李明、张磊、黎俊瑜、莫瑜、蔡文志、关沛勇、林祺颖等，他们大部分是硕士研究生，都是参加过世界决赛或亚洲多个赛区预赛并取得很好成绩的中山大学队的主力队员。

本书由郭嵩山教授和他指导的研究生编写。郭嵩山教授是国际大学生程序设计竞赛中山大学队的主教练，黎俊瑜、林祺颖是硕士研究生，也是参加过世界决赛或亚洲多个赛区预赛并取得很好成绩的中山大学队的主力队员。我们期望能将自己的知识、经验、心得和体会，奉献给广大的程序设计爱好者，以便与大家共同探讨和交流。

本书所提供的题目都是原创题，题目构思新颖，所涉及到的算法知识面广，其涉及的算法知识基本上覆盖大学计算机类本科专业所学到的基本算法。本书可以作为高等院校大学生和研究生们准备参加各级国际大学生程序设计竞赛活动的辅导教材和训练题集，也可以作为高等院校研究生和本科高年级学生学习相关课程的参考书，并适于作为中学省级以上信息学奥林匹克优秀选手准备高层次程序设计竞赛的参考用书。

由于我们水平所限，书中难免有不足之处，欢迎读者批评指正，谢谢！

作 者
2006 年 4 月

目 录

本书试题涉及知识点的说明	1
第 1 章 第 1 届广东省大学生程序设计竞赛暨第 6 届中山大学程序设计竞赛（2003 年）	
试题分析	3
1.1 学会奖章（简单统计）	3
1.1.1 试题	3
1.1.2 题目分析与算法实现	4
1.1.3 标准程序与程序分析	5
1.1.4 测试数据与输出结果	6
1.2 老师的数字（数论）	7
1.2.1 试题	7
1.2.2 题目分析与算法实现	8
1.2.3 标准程序与程序分析	10
1.2.4 测试数据与输出结果	12
1.3 国际象棋（复杂模拟）	13
1.3.1 试题	13
1.3.2 题目分析与算法实现	20
1.3.3 标准程序与程序分析	22
1.3.4 测试数据与输出结果	36
1.4 幻方（数学分析）	37
1.4.1 试题	37
1.4.2 题目分析与算法实现	39
1.4.3 标准程序与程序分析	40
1.4.4 测试数据与输出结果	41
1.5 工作安排（图论）	42
1.5.1 试题	42
1.5.2 题目分析与算法实现	43
1.5.3 标准程序与程序分析	43
1.5.4 测试数据与输出结果	47
1.6 邮递食物（搜索）	48
1.6.1 试题	48
1.6.2 题目分析与算法实现	50
1.6.3 标准程序与程序分析	50

1.6.4	测试数据与输出结果	53
1.7	谁是赢家（博弈演化为递推）	54
1.7.1	试题	54
1.7.2	题目分析与算法实现	55
1.7.3	标准程序与程序分析	56
1.7.4	测试数据与输出结果	58
1.8	排序算法（排序）	59
1.8.1	试题	59
1.8.2	题目分析与算法实现	60
1.8.3	标准程序与程序分析	60
1.8.4	测试数据与输出结果	62
1.9	TYLY 语言（图论）	62
1.9.1	试题	62
1.9.2	题目分析与算法实现	65
1.9.3	标准程序与程序分析	65
1.9.4	测试数据与输出结果	69
1.10	工作依赖（图搜索）	70
1.10.1	试题	70
1.10.2	题目分析与算法实现	71
1.10.3	标准程序与程序分析	71
1.10.4	测试数据与输出结果	73
1.11	总体题目分析与比赛情况	74
1.11.1	题目总结分析	74
1.11.2	比赛情况	74
第 2 章	第 2 届广东省大学生程序设计竞赛暨第 7 届中山大学程序设计竞赛（2004 年）	
	试题分析	75
2.1	二进制最大公约数（简单数论）	75
2.1.1	试题	75
2.1.2	题目分析与算法实现	76
2.1.3	标准程序与程序分析	76
2.1.4	测试数据与输出结果	79
2.2	船舶停靠（贪心和数据结构设计）	79
2.2.1	试题	79
2.2.2	题目分析与算法实现	82
2.2.3	标准程序与程序分析	86
2.2.4	测试数据与输出结果	96
2.3	凸边形外壳（凸包）	97
2.3.1	试题	97

2.3.2	题目分析与算法实现	98
2.3.3	标准程序与程序分析	101
2.3.4	测试数据与输出结果	103
2.4	加密（简单模拟）	104
2.4.1	试题	104
2.4.2	题目分析与算法实现	105
2.4.3	标准程序与程序分析	105
2.4.4	测试数据与输出结果	106
2.5	发电站网络（树型动态规划）	107
2.5.1	试题	107
2.5.2	题目分析与算法实现	109
2.5.3	标准程序与程序分析	109
2.5.4	测试数据与输出结果	111
2.6	有趣的游戏（博弈）	112
2.6.1	试题	112
2.6.2	题目分析与算法实现	114
2.6.3	标准程序与程序分析	115
2.6.4	测试数据与输出结果	119
2.7	准备好了吗（简单排序）	119
2.7.1	试题	119
2.7.2	题目分析与算法实现	120
2.7.3	标准程序与程序分析	120
2.7.4	测试数据与输出结果	121
2.8	有用的论文（贪心）	122
2.8.1	试题	122
2.8.2	题目分析与算法实现	123
2.8.3	标准程序与程序分析	124
2.8.4	测试数据与输出结果	127
2.9	税收（数据结构设计）	128
2.9.1	试题	128
2.9.2	题目分析与算法实现	129
2.9.3	标准程序与程序分析	131
2.9.4	测试数据与输出结果	134
2.10	游览珠海校区（简单计算几何）	135
2.10.1	试题	135
2.10.2	题目分析与算法实现	136
2.10.3	标准程序与程序分析	137
2.10.4	测试数据与输出结果	138

2.11	总体题目分析与比赛情况	138
2.11.1	题目总结分析	138
2.11.2	比赛情况	139
第 3 章	第 8 届中山大学程序设计竞赛预选赛 (2005 年) 试题分析	140
3.1	兔子 (简单递推)	140
3.1.1	试题	140
3.1.2	题目分析与算法实现	141
3.1.3	标准程序与程序分析	141
3.1.4	测试数据与输出结果	142
3.2	任务调度 (拓扑排序)	143
3.2.1	试题	143
3.2.2	题目分析与算法实现	145
3.2.3	标准程序与程序分析	146
3.2.4	测试数据与输出结果	148
3.3	大学校区 (图论)	149
3.3.1	试题	149
3.3.2	题目分析与算法实现	151
3.3.3	标准程序与程序分析	152
3.3.4	测试数据与输出结果	153
3.4	加油站 (解方程组)	154
3.4.1	试题	154
3.4.2	题目分析与算法实现	156
3.4.3	标准程序与程序分析	160
3.4.4	测试数据与输出结果	161
3.5	城市道路 (动态规划)	162
3.5.1	试题	162
3.5.2	题目分析与算法实现	163
3.5.3	标准程序与程序分析	164
3.5.4	测试数据与输出结果	166
3.6	森林 (简单图论)	166
3.6.1	试题	166
3.6.2	题目分析与算法实现	167
3.6.3	标准程序与程序分析	168
3.6.4	测试数据与输出结果	169
3.7	DNA 匹配 (字符串匹配)	170
3.7.1	试题	170
3.7.2	题目分析与算法实现	171
3.7.3	标准程序与程序分析	172

3.7.4	测试数据与输出结果	173
3.8	题目总结分析与比赛情况	174
3.8.1	题目总结分析	174
3.8.2	比赛情况	174
第4章	第3届广东省大学生程序设计竞赛暨第8届中山大学程序设计竞赛(2005年)	
	试题分析	175
4.1	放球(递推)	175
4.1.1	试题	175
4.1.2	题目分析与算法实现	176
4.1.3	标准程序及程序分析	177
4.1.4	测试数据与输出结果	178
4.2	房子(简单图论)	179
4.2.1	试题	179
4.2.2	题目分析与算法实现	180
4.2.3	标准程序与程序分析	180
4.2.4	测试数据与输出结果	181
4.3	计算机(贪心+最小堆)	183
4.3.1	试题	183
4.3.2	题目分析与算法实现	184
4.3.3	标准程序与程序分析	186
4.3.4	测试数据与输出结果	189
4.4	大学排名(动态规划)	190
4.4.1	试题	190
4.4.2	题目分析与算法实现	191
4.4.3	标准程序与程序分析	192
4.4.4	测试数据与输出结果	194
4.5	椭圆相交(微积分)	195
4.5.1	试题	195
4.5.2	题目分析与算法实现	196
4.5.3	标准程序与程序分析	197
4.5.4	测试数据与输出结果	198
4.6	图灵机编程(构造)	199
4.6.1	试题	199
4.6.2	题目分析与算法实现	201
4.6.3	标准程序与程序分析	202
4.6.4	测试数据与输出结果	205
4.7	格雷码(分治)	208
4.7.1	试题	208

4.7.2	题目分析与算法实现	209
4.7.3	标准程序与程序分析	210
4.7.4	测试数据与输出结果	210
4.8	英雄（宽度优先搜索）	212
4.8.1	试题	212
4.8.2	题目分析与算法实现	214
4.8.3	标准程序与程序分析	215
4.8.4	测试数据与输出结果	218
4.9	程序重构（简单模拟）	219
4.9.1	试题	219
4.9.2	题目分析与算法实现	220
4.9.3	标准程序与程序分析	220
4.9.4	测试数据与输出结果	221
4.10	多边形旋转（计算几何）	225
4.10.1	试题	225
4.10.2	题目分析与算法实现	227
4.10.3	标准程序与程序分析	228
4.10.4	测试数据与输出结果	233
4.11	总体题目分析与比赛情况	235
4.11.1	题目总结分析	235
4.11.2	比赛情况	235
第 5 章	第 28 届 ACM/ICPC 国际大学生程序设计竞赛亚洲预赛广州赛区竞赛（2003 年）	
	试题分析	236
5.1	原子实验（动态规划）	236
5.1.1	试题	236
5.1.2	题目分析与算法实现	238
5.1.3	标准程序与程序分析	239
5.1.4	测试数据与输出结果	243
5.2	电梯调度计划（动态规划）	245
5.2.1	试题	245
5.2.2	题目分析与算法实现	246
5.2.3	标准程序与程序分析	247
5.2.4	测试数据与输出结果	249
5.3	新围棋游戏（回溯）	251
5.3.1	试题	251
5.3.2	题目分析与算法实现	253
5.3.3	标准程序与程序分析	255
5.3.4	测试数据与输出结果	263

5.4 互联网（模拟）	265
5.4.1 试题	265
5.4.2 题目分析与算法实现	270
5.4.3 标准程序与程序分析	271
5.4.4 测试数据与输出结果	276
5.5 折纸条（计算几何）	278
5.5.1 试题	278
5.5.2 题目分析与算法实现	279
5.5.3 标准程序与程序分析	280
5.5.4 测试数据与输出结果	286
5.6 阶乘之和（简单数学）	287
5.6.1 试题	287
5.6.2 题目分析与算法实现	288
5.6.3 标准程序与程序分析	289
5.6.4 测试数据与输出结果	290
5.7 任务序列（图论）	291
5.7.1 试题	291
5.7.2 题目分析与算法实现	293
5.7.3 标准程序与程序分析	294
5.7.4 测试数据与输出结果	295
5.8 Vivian 的问题（数论）	298
5.8.1 试题	298
5.8.2 题目分析与算法实现	299
5.8.3 标准程序与程序分析	300
5.8.4 测试数据与输出结果	302
5.9 总体题目分析与比赛情况	305
5.9.1 题目总结分析	305
5.9.2 比赛情况	306
参考文献	307
作者简介	308

本书试题涉及知识点的说明

本书中的题型所覆盖的知识面相当广泛,包括了常用的多种算法设计方法,数据结构,图论,组合数学,计算几何,线性代数,高等数学,数论和博弈等方面的知识。

1. 算法设计方法

(1) 贪心。具体策略是并不从整体最优上来加以考虑,而是选取某种意义下的局部最优解,当然很多时候贪心所得到的结果也是整体最优的。例如本书中的“2.8 有用的论文”和“4.3 计算机”。

(2) 动态规划。动态规划的特点是将问题的解决过程划分为若干阶段,按照阶段来进行状态转移,同时状态必须具有无后效性,即后面阶段的状态不会反过来影响前面阶段的状态。例如本书中的“2.5 发电站网络”、“4.4 大学排名”、“5.1 原子实验”和“5.2 电梯调度计划”。

(3) 分治。分治的思想是将问题划分为若干子问题,通过将子问题的求解和对子问题的解进行合并来得到原问题的解,例如本书中的“4.7 格雷码”。

(4) 搜索。搜索技术包括宽度优先搜索和深度优先搜索,在其他算法设计方法难以奏效的情况下,搜索通常是解决问题的有效手段。搜索是对问题的解空间进行遍历的过程,它从某一初始解出发,通过一些邻域扩展手段来不断产生新的解,从而达到遍历解空间的目的。当然有时对于问题解空间相当庞大的情况,完全遍历解空间是不现实的,此时就必须充分发掘问题所包含的约束条件,在搜索过程中应用这些条件来进行剪枝,从而减少搜索量。例如本书中的“1.6 邮递食物”、“1.10 工作依赖”、“4.8 英雄”和“5.3 新围棋游戏”。

2. 数据结构

算法和数据结构是计算机程序中最基本的两个部分,如果说算法是对数据的处理,那么数据结构则是对数据的存储和组织。除了常用的简单数据结构如数组、链表、堆栈、队列等,还包括较为复杂的数据结构,如最小堆、AVL 树(平衡二叉树)、胜者树、线段树、并查集和散列表等,它们在某些应用场合可以极大地提高程序效率。例如,“2.2 船舶停靠”涉及了 AVL 树,“2.9 税收”涉及了胜者树,“4.3 计算机”涉及了最小堆。

3. 图论

图论中的知识点相当广泛,在书中涉及的有:

- (1) 最短路问题,例如“3.3 大学校区”;
- (2) 拓扑排序,例如“3.2 任务调度”;
- (3) 树的遍历,例如“3.6 森林”;
- (4) 计算图中顶点度数,例如“4.2 房子”;
- (5) 最优匹配问题,例如“1.5 工作安排”;

(6) 竞赛图中的哈密尔顿路问题, 例如“5.7 任务序列”。

4. 组合数学

主要是组合计数问题, 例如“3.1 兔子”和“4.1 放球”。

5. 计算几何

计算几何中的知识点也相当多:

- (1) 求平面点集的凸包, 例如“2.3 凸边形外壳”;
- (2) 计算多边形面积, 例如“2.10 游览珠海校区”;
- (3) 求两直线交点, 例如“4.10 多边形旋转”和“5.5 折纸条”;
- (4) 判断直线与多边形相交, 例如“4.10 多边形旋转”和“5.5 折纸条”;
- (5) 求点关于直线的对称点, 例如“4.10 多边形旋转”和“5.5 折纸条”;
- (6) 求多边形质心, 例如“4.10 多边形旋转”。

6. 线性代数

主要是求解线性方程组, 例如“3.4 加油站”。

7. 高等数学

主要是微积分, 例如“4.5 椭圆相交”。

8. 数论

数论涉及的知识点包括:

求最大公约数, 例如“2.1 二进制最大公约数”;

中国剩余定理, 例如“1.2 老师的数字”;

梅森素数的性质, 例如“5.8 Vivian 的问题”。

9. 博弈

主要是游戏中的致胜策略问题, 包括“1.7 谁是赢家”和“2.6 有趣的游戏”。

此外本书还包括以下知识点:

- (1) 模拟, 例如“1.3 国际象棋”、“2.4 加密”和“5.4 互联网”;
- (2) 排序, 例如“1.8 排序算法”和“2.7 准备好了吗”;
- (3) 统计, 例如“1.1 学会奖章”;
- (4) 字符串处理, 例如“3.7 DNA 匹配”;
- (5) 构造, 例如“4.6 图灵机编程”;
- (6) 其他数学问题, 例如“1.4 幻方”和“5.6 阶乘之和”。

第 1 章

第 1 届广东省大学生程序设计竞赛暨第 6 届中山大学 程序设计竞赛（2003 年）试题分析

1.1 学会奖章（简单统计）

1.1.1 试题

Problem A

Academy Awards

(Input File: oscar.in / Output File: oscar.out)

Selected from 3,850 teams from 1,329 universities in 68 countries competing at 106 sites and preliminary contests worldwide, sixty-eight teams competed for bragging rights and prizes at The 27th Annual ACM International Collegiate Programming Contest World Finals sponsored by IBM on March 25, 2003, in Hollywood, California. The 2003 World Champion is Warsaw University. And Zhongshan University won the 8th place. During those days, another world famous event was held in the same place. It was the 75th Annual Academy Awards. It's also known as Oscar.

We always say that the Best Picture is the most important award of all the awards. Before the Oscar Night, we can't tell which film will win Best Picture. Fortunately, we can do it out from the Nominee List of all the awards other than the Best Picture. I suggest that you should follow my 3 rules here.

- All the films in the list have chances to win the Best Picture
- The film which will win the Best Picture is the film which has been nominated the most times in the list
- If there are more than one film which have been nominated the most times in the list, we will choose the first one which appears in the list

Let's see such a List below.

VISUAL EFFECTS

THE LORD OF THE RINGS: THE TWO TOWERS

SPIDER-MAN

STAR WARS EPISODE II ATTACK OF THE CLONES

SOUND EDITING

THE LORD OF THE RINGS: THE TWO TOWERS

MINORITY REPORT

ROAD TO PERDITION

From the list, we can find that *THE LORD OF THE RINGS: THE TWO TOWERS* has been nominated twice. And each of the other films has been nominated only once. So we can say *THE LORD OF THE RINGS: THE TWO TOWERS* will win the Best Picture.

Your task is to write a program to figure out the anticipatory winner from the list.

Input:

The input file will consist of several lists. The first line of each list contains only one integer n ($1 \leq n \leq 100$), representing the number of awards in the list. Then you get n blocks. Each block indicated the nominees of a distinct award. The first line of each block is the name of the award which is not longer than 80. The second line is m_i ($1 \leq m_i \leq 10$, $1 \leq i \leq n$) - the number of nominated films. In the following lines are m_i film names, one per line. For make the question simple, you can assume that there isn't any space in the film names.

The input is terminated by a line with one zero.

Output:

For each list, you are supposed to figure out the winner of Best Picture in a single line.

Sample Input:

```
2
VISUAL_EFFECTS
3
THE_LORD_OF_THE_RINGS:_THE_TWO_TOWERS
SPIDER-MAN
STAR_WARS_EPISODE_II_ATTACK_OF_THE_CLONES
SOUND_EDITING
3
THE_LORD_OF_THE_RINGS:_THE_TWO_TOWERS
MINORITY_REPORT
ROAD_TO_PERDITION
0
```

Output for the Sample Input:

```
THE_LORD_OF_THE_RINGS:_THE_TWO_TOWERS
```

1.1.2 题目分析与算法实现

本题是这届比赛中最简单的一道题，主要考察编程者对基本的字符串处理和对数据统

计能力。也就是说，考察参赛选手最基本的编程能力。

题目的大意是，给出 n 个不同的奖项，每个奖项一共有 m_i ($m_i \leq 10$, $1 \leq i \leq n$) 个提名，所有的提名都是电影名字，要求找出提名最多的电影名字，并且输出该电影名字。

题目说明了一共有 n 个不同的奖项 ($n \leq 100$)，每个奖项的提名数为 m_i ($m_i \leq 10$)，显然，一共最多有 $100 \times 10 = 1000$ 个不同的电影名字，而电影名字最多只有 80 个字符，所以把全部电影名字存下来是绝对足够的。

分析本题时要注意两个地方：一是当出现频率一样时怎样处理；二是电影名字单词间是否出现空格。在正式比赛时，这两点一定要弄清楚，不然会出现错误。

既然可以把全部的电影名字保存下来，那么就只需要考虑每个电影的出现次数，使用一个数组变量内统计对应的电影名字出现的频率即可。最后再把提名最多的电影名字输出就行了。

算法是建立两个数组，一个数组是存放电影名字，另一个数组是存放对应的电影出现的频率。当读入一个电影名字时，搜索该电影名字是否在之前已经出现，如果出现，就把频率加 1，如果没有出现，就新增加一个电影名字。最后在所有已统计的电影名字中找出出现频率最大的一个。本题的算法复杂度为 $O(n \cdot m)$ 。

1.1.3 标准程序与程序分析

```
#include "stdio.h"
#include "string.h"

char a[1000][100]; //a 存放电影名字
int b[1000],t; //b 存放对应电影出现的频率，t 存放一共有多少个不同的电影

int index(char *s) //处理电影名字为 s
{
    int i;

    for (i=0; i<t; i++)
        if (strcmp(s,a[i])==0)
            return i; //如果电影名字已经出现，放回位置
    memcpy(a[t],s,strlen(s)); //新增加一个电影
    t++;
    return t-1;
}

int main()
{
    int n,m,i,j,k;
    char s[100];
```