硅谷资深工程师凝结心力之作

精选128道经典的算法和编程题目

覆盖编程面试常见题型

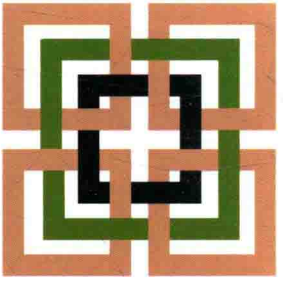
提高分析和解决问题能力，提升编程素养

Coding Puzzles

编程谜题

codingtmd 著

PUZZLES



中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

Coding Puzzles

编程谜题

codingtmd 著

人民邮电出版社
北京

图书在版编目（CIP）数据

编程谜题 / codingtmd著. -- 北京 : 人民邮电出版社, 2016.5
ISBN 978-7-115-41901-9

Ⅰ. ①编… Ⅱ. ①c… Ⅲ. ①程序设计 Ⅳ. ①TP311.1

中国版本图书馆CIP数据核字(2016)第062878号

内 容 提 要

本书精选128道经典的算法和编程题目，有针对性地做出分析和解答，并给出代码解决方案。本书的主要思路是，利用计算机算法知识，以分析和解决谜题的形式，总结如何把计算机常用算法及数据结构等知识应用到相关的问题上，提高读者分析问题、解决问题的能力。进而，希望培养读者的编程素养，帮助读者更好地从事程序设计的相关工作。

本书中的题目涉及递归、分而治之、二及树搜索、树遍历、图遍历、动态规划、字符串搜索等经典的算法问题，也是编程面试以及程序设计实践中经常遇到的问题。本书最后提供一个附录，包含一些精选的论文、图书和参考资料，可以帮助读者了解IT行业内最新的进展，并藉此在IT职业生涯中提高解决问题的能力。

本书适合对程序设计和算法问题感兴趣的读者阅读，尤其适合准备编程面试或者想要提高自身程序设计能力和素养的程序员学习参考。

◆ 著　　codingtmd
责任编辑　陈冀康
责任印制　张佳莹　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路11号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
固安县铭成印刷有限公司印刷

◆ 开本：720×960　1/16
印张：15.25
字数：207千字　　2016年5月第1版
印数：1－3 000册　　2016年5月河北第1次印刷

定价：49.00元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315

作者简介

codingtmd，2001～2005 年于中国科学技术大学获得计算机学士学位。2008 年获中国科学院软件所软件工程中心工学硕士学位。2008～2014 年供职于微软，从事于数据库、分布式系统、云计算基础架构及服务等方向的工作，参与了 Bing 和 Windows Azure 等系统的研发工作。目前就职于 Facebook，负责新产品研发及第三方云服务平台建设。

爱好算法及系统架构设计，酷爱读书，喜欢冒险。

前　言

《Coding Puzzles》英文版最早起源于我在博客（http://fisherlei.blogspot.com）上的连载文章。因为我个人喜欢对 leetcode 的算法和编程题目做出分析和解答并写成博客在网络上分享，长期以来，积累了不少解题分析和思路方面的素材。编纂成书后受到了很多读者的欢迎，当然，也收到了很多的建议和批评。

读者的热情反馈，超出了我最早编写本书的预期，也促使我进一步编写和出版了英文版《Coding Puzzles》（第 2 版）（电子版）。该版本发布不久，我就收到了国内出版社的邀约，于是，有了大家手中这本《编程谜题》。

和最早的英文版相比，本书主要做了以下的修改和改进。

1．根据读者反馈，修改了第 1 版的错误及疏漏，补充了新的图示及分析，使得内容上可读性更好。

2．增加了更多的趣题。相比第 1 版，本版增加了 23 道新的题目。

3．本书最后增加了一个附录，包括了业界一些常用分布式理论及产品设计的一些资料。希望可以帮助读者更好地了解业界的发展。

这本书的主要思路，是利用计算机算法知识，以分析和解决谜题的形式，总结如何把计算机常用算法及数据结构等知识应用到相关的问题上，提高读者分析问题、解决问题的能力；进而，希望培养读者的编程素养，帮助读者更好地从事程序设计的相关工作。

在分析解题思路的过程中，涉及一些算法和数据结构的知识，由于本书是面向解题方法和思路的，并没有针对这些算法和数据结构知识给出详细的讲解和介绍。如果读者需要温习计算机常用算法及数据结构的相关知识，建议阅读Thomas H. Cormen 的《Introduction to Algorithms》或相关教科书。

需要说明的是，本书中的题目只是在程序员面试中一些比较有代表性的题目，而不是一个包罗万象的习题集。如何在纷繁复杂的题目中，找到其后不变的东西，才是算法分析的真谛。

如果读者希望亲自实践书中的题目，可以访问北美最流行的网站https://leetcode.com/。如果希望参与后续的讨论，可以访问我的博客：http://www.cnblogs.com/codingtmd/或 http://fisherlei-blogspot.com。书中的示例代码，可以到 https://github.com/codingtmd/leetcode 下载。

因为水平有限，书中难免有错误纰漏和不足之处，还请广大读者将相关反馈发至 codingtmd @outlook.com。期望能够与读者共同学习和探讨。

■ 读者对象

本书的读者对象如下：

- 计算机行业从业者；
- 计算机相关专业学生；
- 对计算机算法应用感兴趣的人。

本书的示例代码都是用 C++写成，所以需要读者能够对 C/C++语言有所了解。

■ 致谢

感谢我的家人对我的全力支持。

感谢包子 IT 培训（http://baozitraining.org/）对于本书的支持和帮助。

感谢责任编辑陈冀康先生及人民邮电出版社将本书付梓出版。最后，祝大家一切顺利！

Codingtmd 于 Mountain View

2016 年 1 月

目　录

1. 两数之和 …… 1
2. 3 个数之和 …… 4
3. 3 个数之和最接近 …… 6
4. 4 个数之和 …… 8
5. 二进制数相加 …… 9
6. 两个数相加 …… 11
7. anagrams …… 12
8. 购买和销售股票的最佳时机 Ⅰ …… 14
9. 购买和销售股票的最佳时机 Ⅱ …… 15
10. 购买和销售股票的最佳时机 Ⅲ …… 15
11. 平衡二叉树 …… 17
12. 前序遍历二叉树 …… 18
13. 中序遍历二叉树 …… 21
14. 层次遍历二叉树 …… 23
15. 二叉树最大路径和 …… 25
16. 爬楼梯 …… 27
17. 复制图 …… 28
18. 组合求和 Ⅰ …… 31
19. 组合求和 Ⅱ …… 33
20. 组合 …… 35
21. 从前序遍历和中序遍历构造二叉树 …… 36

22. 能装最多的水的容器……39
23. 把排序的数组转换为二叉树……40
24. 将排序的链表转换为二叉树……41
25. 复制带有随机指针的链表……43
26. 数数并读出……45
27. 解码方法……47
28. 不同子序列……48
29. 两个整数相除……50
30. 编辑距离……51
31. 计算逆波兰式……54
32. 第 1 个非正整数……55
33. 将二叉树扁平化为链表……57
34. 加油站……60
35. 生成括号……62
36. 格雷码……64
37. 实现 strStr()……65
38. 插入间隔……68
39. 整数转换为罗马数字……70
40. 插值字符串……72
41. 跳跃游戏……76
42. 跳跃游戏 II……77
43. 柱状图中最大的矩形……79
44. 最后一个单词的长度……82
45. 一个手机号码的字母组合……84
46. 链表中的环……85

47. 链表中的环 II ····· 86
48. 最大公共前缀 ····· 88
49. 最长连续序列 ····· 89
50. 最长回文子字符串 ····· 91
51. 没有重复字符的最长的子字符串 ····· 93
52. 最长的有效的括号 ····· 94
53. LRU 缓存 ····· 95
54. 二叉树的最大深度 ····· 98
55. 一条直线上的点的最大数目 ····· 99
56. 最大子数组 ····· 101
57. 两个排序的数组的中位数 ····· 103
58. 合并间隔 ····· 106
59. 合并 *k* 个排序的链表 ····· 107
60. 合并排序的数组 ····· 109
61. 合并两个排序的链表 ····· 110
62. 二叉树的最小深度 ····· 111
63. 最小路径和 ····· 112
64. 最小的窗口子字符串 ····· 114
65. 字符串相乘 ····· 117
66. 下一个排列 ····· 118
67. 回文数字 ····· 120
68. 回文划分 ····· 121
69. 回文划分 II ····· 123
70. 划分链表 ····· 125
71. Pascal 三角 ····· 127

72. 路径加和 …… 128
73. 路径加和 II …… 130
74. 变换 …… 131
75. 变换 II …… 133
76. 变换序列 …… 135
77. 加 1 …… 137
78. 填充每一个节点的 Next 右指针 …… 138
79. 填充每一个节点的 Next 右指针 II …… 140
80. Pow(x, n) …… 143
81. 从排序的数组中删除重复的元素 …… 144
82. 从排序的数组中删除重复的元素 II …… 145
83. 从排序的链表中删除重复元素 …… 146
84. 从排序的链表中删除重复元素 II …… 147
85. 删除元素 …… 148
86. 删除从链表末尾开始的第 N 个节点 …… 149
87. 重新排序链表 …… 151
88. 还原 IP 地址 …… 153
89. 整数翻转 …… 154
90. 翻转链表 II …… 156
91. 以 k 为一组翻转节点 …… 157
92. 从罗马数字转换为整数 …… 159
93. 旋转图像 …… 161
94. 旋转链表 …… 162
95. 搜索一个 2D 矩阵 …… 163
96. 搜索一个范围 …… 165

97. 在旋转后的排序数组中搜索 …… 167
98. 在旋转后的排序数组中搜索 II …… 169
99. 搜索插入位置 …… 170
100. 序列化和反序列化一个树 …… 171
101. 设置矩阵为 0 …… 173
102. 简化路径 …… 175
103. 单个数字 …… 177
104. 单个数字 II …… 178
105. 排序颜色 …… 179
106. 排序链表 …… 181
107. Sqrt(x) …… 184
108. 字符串转换为整数 …… 186
109. 子集 …… 188
110. 子集 II …… 190
111. 对根到叶子的数字求和 …… 192
112. 包围的区域 …… 193
113. 成对地交换节点 …… 196
114. 对成树 …… 197
115. 装雨水 …… 200
116. 三角形 …… 202
117. 唯一的二叉搜索树 …… 204
118. 唯一的二叉搜索树 II …… 206
119. 唯一路径 …… 208
120. 唯一路径 II …… 209
121. 大写和小写排列 …… 211

122. 验证回文 …… 213
123. 验证括号 …… 214
124. 验证数独 …… 215
125. 验证二叉搜索树 …… 217
126. 通配符匹配 …… 218
127. 单词拆分 …… 220
128. 单词拆分 II …… 222
附录 阅读列表 …… 224

1. 两数之和

【题目】

给定整数的一个数组，找出这样的两个数，它们的加和等于一个特定的目标数字（target）。

twoSum 函数应该返回两个数的索引，这两个数相加等于目标数字，其中 index1 必须小于 index2。请注意，返回的结果（index1 和 index2）不是基于 0 的。

可以假设对每一个输入来说，都只有一个解决方案。

输入：numbers={2, 7, 11, 15}，target=9

输出：index1=1，index2=2

【解析】

最直观明了的解法是通过暴力搜索，使用两个嵌套的循环来遍历所有可能，时间复杂度为 O(n*n)。但是很明显，这不是出题者想看到的结果。

有两种解法。

解法一，哈希方法。

从左往右扫描一遍，然后将整数及其索引存到 map 中。然后再扫描一遍，对每一个整数 K，搜索 target-K 在 map 中是否存在即可。如果存在，则输出 K 及 target-K 的下标。时间复杂度为 $O(n)$。

解法二，双指针扫描。

首先将数组排序，然后双指针从首尾往中间扫描。时间复杂度为 $O(n*\lg n)$。因为要求返回原数组的下标，所以在排序的时候还要有额外的数组来存储下标信息。

注意	**双指针扫描** 双指针问题是对于普通问题的一种特殊解法，往往要求问题是线性的，可以用双指针来遍历，从许多解中找到一个最大或者最小的解。双指针一般可以分别放在首尾处，用来往中间靠拢，或者都放在头部，用来划分区间。

【代码】

解法一的实现如下：

```
1:   vector<int> twoSum(vector<int>&numbers, int target) {
2:     map<int, int> mapping;
3:     vector<int> result;
4:     for(int i =0; i< numbers.size(); i++)
5:     {
6:       mapping[numbers[i]]=i;
7:     }
8:     for(int i =0; i< numbers.size(); i++)
9:     {
10:       int searched = target - numbers[i];
11:       if(mapping.find(searched) != mapping.end()&&i!=mapping
[searched])
12:       {
13:         result.push_back(i+1);
14:         result.push_back(mapping[searched]+1);
15:         break;
16:       }
17:     }
18:     return result;
19:   }
```

解法二的实现如下：

```
1:struct Node
2: {
3:    int val;
4:    int index;
5:    Node(int pVal, int pIndex):val(pVal), index(pIndex){}
6: };
7: static bool compare(const Node &left, const Node &right)
8: {
9:    return left.val < right.val;
10:}
11: vector<int> twoSum(vector<int>&numbers, int target) {
12:    vector<Node> elements;
13:    for(int i =0; i< numbers.size(); i++)
14:    {
15:       elements.push_back(Node(numbers[i], i));
16:    }
17:    std::sort(elements.begin(), elements.end(), compare);
18:    int start = 0, end = numbers.size()-1;
19:        vector<int> result;
20:        while(start < end)
21:       {
22:         int sum = elements[start].val + elements[end].val;
23:        if(sum == target)
24:          {
25:          result.push_back(elements[start].index+1);
26:              if(elements[start].index < elements[end].index)
27:        result.push_back(elements[end].index+1);
28:              else
29:        result.insert(result.begin(),elements[end].index+1);
30:              break;
31:          }
32:          else if(sum > target)
33:             end--;
34:          else
35:             start++;
36:        }
37:        return result;
38: }
```

2. 3个数之和

给定 n 个整数的一个数组 S，S 中是否有元素 a、b 和 c 满足 $a+b+c=0$？找出数组中所有满足加和为 0 的不同的三个数组合。

注意，(a,b,c)中的元素必须是非降序的排列方式（即，$a \leqslant b \leqslant c$）。

解决方案中给出的集合不能包含重复的三元组。

例如，给定数组 $S=\{-1\ 0\ 1\ 2\ -1\ -4\}$，

一个解决方案集合是

```
(-1, 0, 1)
(-1, -1, 2)
```

【解析】

这是上一题的扩展，把 2 个变量变成 3 个变量。对于一般的扩展题，要通过一些转化将其还原为已有的问题，才能更好地处理。

如果我们把等式稍稍修改一下，

```
a + b + c =0   =>   a + b = -c
```

这时候，问题就转换为，寻找两个变量 a 和 b，使得其之和为$-c$，又回到了“两数之和”的问题。

因为题目提到了可能出现重复解的问题，所以要注意“去重”。

1. 代码第 19～24 行