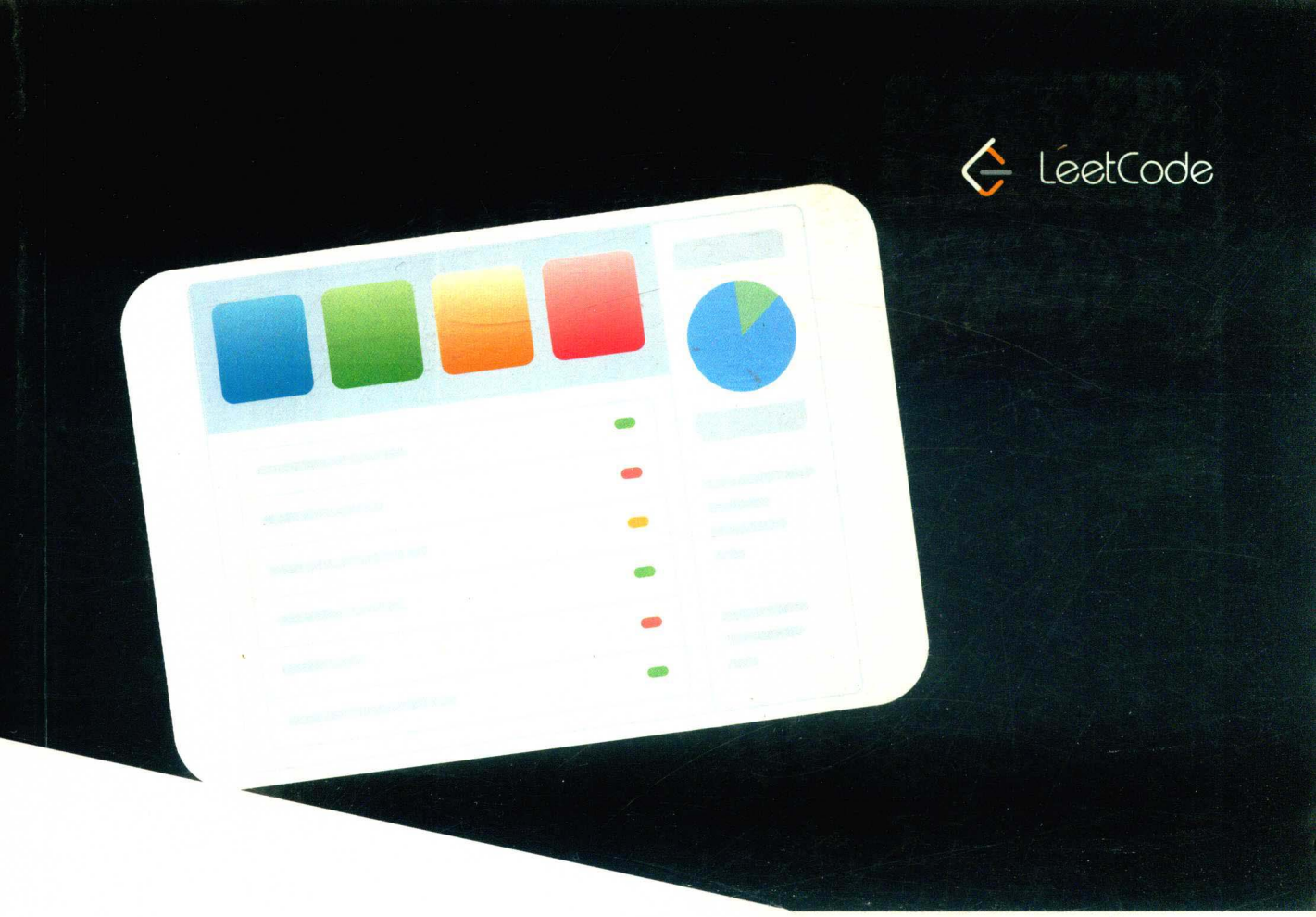


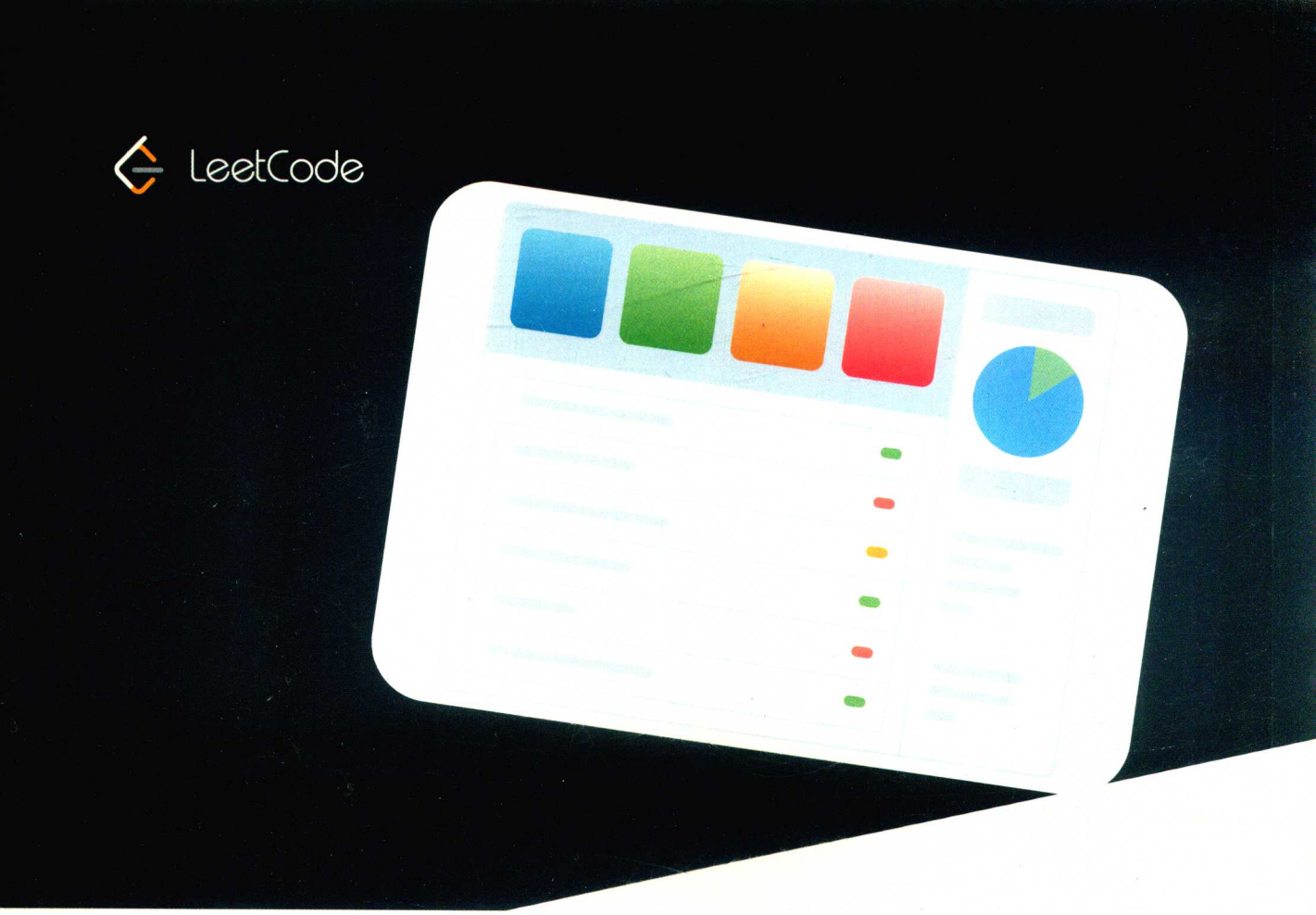
详解Python编程简单算法，深入理解数据结构和Python编程

图解LeetCode初级算法（Python版）

胡松涛 编著

清华大学出版社

需要全本请访问 www.ertongbook.com



图解LeetCode初级算法（Python版）

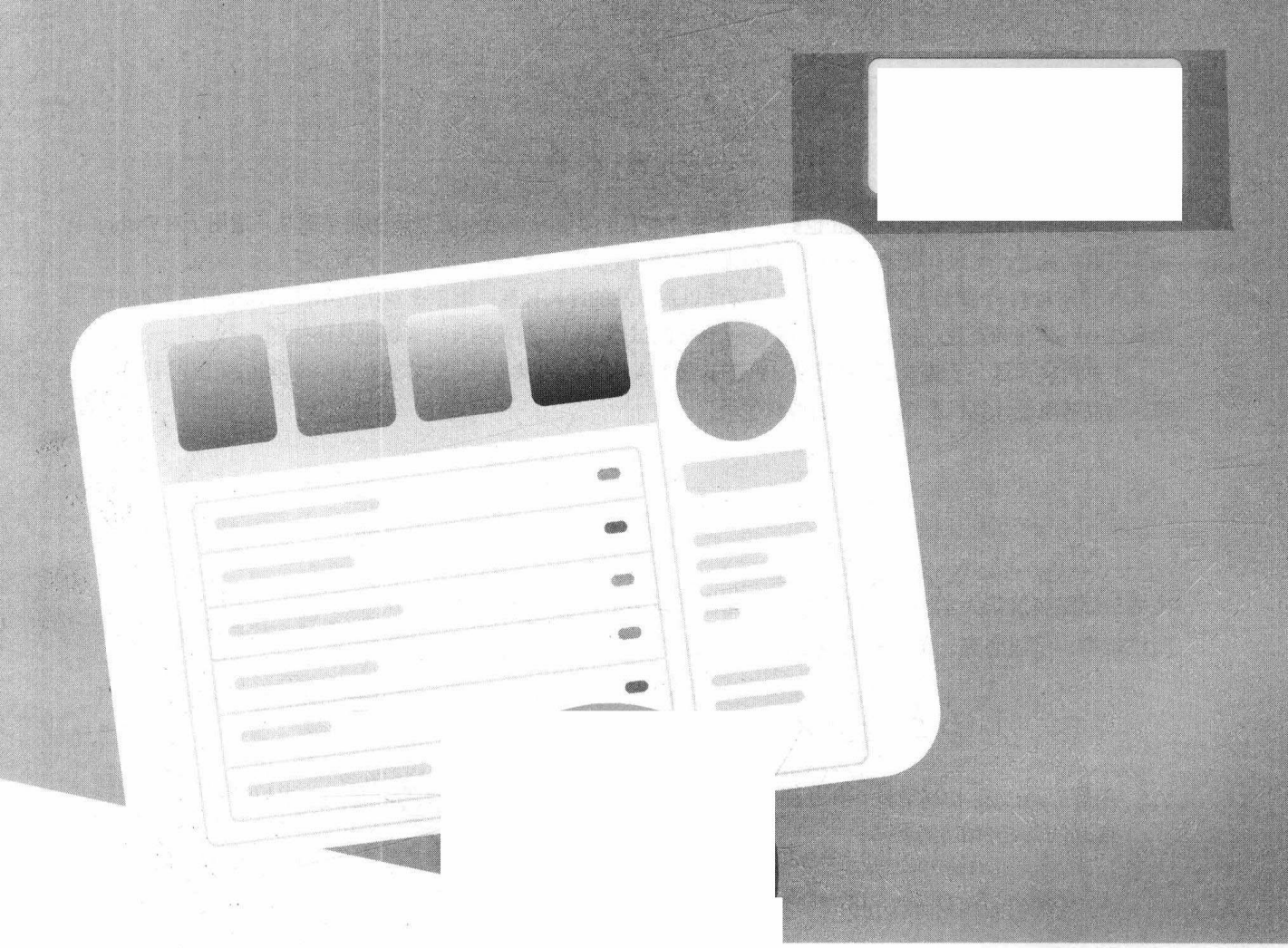
本书概括了几乎所有LeetCode的初级算法试题，以图例的形式展示了算法的原理，供读者学习研究，以提升算法水平和Python编程水平。本书适合Python编程和算法初学者。

清华社官方微信号

扫 我 有 惊 喜

ISBN 978-7-302-53989-6

定价：49.00元



图解LeetCode初级算法（Python版）

胡松涛 编著

清華大學出版社
北京

内 容 简 介

生活离不开算法，学习 Python 也离不开算法。本书主要讲解 Python 语言下的简单算法，适用于对 Python 有初步了解的读者，可以让读者对数据结构和语言本身有更好的理解。

本书分为 11 个章节：第 1 章简单介绍算法以及算法的评判标准；第 2~3 章用图示详细地说明计算机中常用的算法——排序和查找；第 4~11 章借用 LeetCode 的试题演示如何使用算法来解决具体问题。

本书内容详尽、示例丰富，是算法（Python 版）入门读者的参考书，非常适合大中专院校师生学习阅读，也可作为高等院校计算机及相关专业的教材使用。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售
版权所有，侵权必究。侵权举报电话：010-62782989　　13701121933

图书在版编目（CIP）数据

图解 LeetCode 初级算法：Python 版 / 胡松涛编著.— 北京：清华大学出版社，2020.1
ISBN 978-7-302-53989-6

Ⅰ. ①图… Ⅱ. ①胡… Ⅲ. ①计算机算法－图解 Ⅳ. ①TP301.6-64

中国版本图书馆 CIP 数据核字（2019）第 230743 号

责任编辑：夏毓彦
封面设计：王　翔
责任校对：闫秀华
责任印制：丛怀宇

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

印 装 者：三河市龙大印装有限公司
经　　销：全国新华书店
开　　本：190mm×260mm　　**印　　张**：15.75　　**字　　数**：403 千字
版　　次：2020 年 1 月第 1 版　　**印　　次**：2020 年 1 月第 1 次印刷
定　　价：49.00 元

产品编号：082910-01

前　言

本书写作初衷

1. 选择 Python 来实现算法的原因

Python 作为一门编程语言已经越来越流行。Python 的爬虫、Python 的大数据、Python 的机器学习、Python 的 Web 框架……一眼望去，皆是 Python。在最新的编程语言排行榜中，虽然 Python 上面还有两座大山（C & Java），但其进击势头极猛，榜单位置几乎不可动摇。

另外，作为一门动态语言，Python 使用简单方便，无须编译，即写即用。在解决一些简单问题时优势非常明显。

2. 算法的作用

“程序=算法+数据结构”这一说法流传已久。对编程有兴趣的不能不去学习算法。在解决实际问题时，有算法的支持和没算法的支持效率可能是天差地别。从最简单的排序，到目前流行的大数据，还有常见的广告精确推广……算法几乎无处不在。

3. 本书涉及的技术

- Python 经典算法之搜索
- Python 经典算法之排序
- Python 算法实用场景

4. 读者学习本书能达到的效果

Python 简单易学，是非常容易入门的一门语言，但从入门到提高有一定的门槛，尤其是算法方面。Leetcode 中由浅入深的算法试题可以帮助读者在学习算法开拓思维的同时加深对 Python 的理解。本书概括了几乎所有 Leetcode 的初级算法试题，以图例的形式展示算法的原理，供读者学习研究，以提升算法水平、Python 编程水平。

需要全本请访问 www.ertongbook.com

本书的特点

- 本书从理论知识出发，延伸到实例；精心选择算法中常见、经典的部分；讲解细致，图文并茂，分析透彻。
- 深入浅出、轻松易学。用简单的示例，以图例展示基本的原理，再扩展到代码说明，让读者能够真正了解 Python 的基本算法。
- 技术新颖、与时俱进，结合时下热门的专业网站 LeetCode，借用网站丰富的资源，为读者展示算法的魅力。
- 贴近读者。在展示算法的同时穿插 Python 编程语言的特点说明，以照顾不同基础的读者。

源代码下载

本书配套示例源代码下载地址可扫描右侧二维码获得。



如果下载有问题，请联系 booksaga@163.com，邮件主题为“图解 LeetCode 初级算法”。

本书的读者

- Python 开发初学者
- 对计算机算法有兴趣的人员
- 数学爱好者
- 喜欢 Python 的大中专院校学生
- 培训学校的学生

编　者

2020 年 1 月

目 录

第 1 章 浅谈算法 1

1.1 算法概述 1

1.2 度量算法 2

1.2.1 时间复杂度 2

1.2.2 空间复杂度 2

1.3 Python&Pythonic 3

第 2 章 基础算法之排序 5

2.1 冒泡排序 5

2.1.1 原理 5

2.1.2 代码 8

2.2 选择排序 10

2.2.1 原理 11

2.2.2 代码 12

2.3 插入排序 13

2.3.1 原理 14

2.3.2 代码 15

2.4 归并排序 17

2.4.1 原理 17

2.4.2 代码 20

2.5 快速排序 21

2.5.1 原理 21

2.5.2 代码 24

2.6 计数排序 26

2.6.1 原理 26

2.6.2 代码 28

2.7 算法小结 30

第 3 章　基础算法之查找 31

3.1　顺序查找 31

3.1.1　原理 31

3.1.2　代码 33

3.2　二分法查找 36

3.2.1　原理 36

3.2.2　代码 37

3.3　斐波那契查找 39

3.3.1　原理 39

3.3.2　代码 41

3.4　插值查找 43

3.4.1　原理 43

3.4.2　代码 45

3.5　分块查找 47

3.5.1　原理 47

3.5.2　代码 48

第 4 章　数组 51

4.1　从排序数组中删除重复项 51

4.1.1　算法要求 51

4.1.2　解题思路 52

4.1.3　解题代码 53

4.2　买卖股票的最佳时期 II 55

4.2.1　算法要求 55

4.2.2　解题思路 55

4.2.3　解题代码 57

4.3　旋转数组 58

4.3.1　算法要求 58

4.3.2　解题思路 59

4.3.3　解题代码 61

4.4　存在重复 65

4.4.1　算法要求 65

4.4.2　解题思路 65

4.4.3　解题代码 66

4.5　只出现一次的数字 71

4.5.1　算法要求 71

4.5.2　解题思路 71

4.5.3 解题代码 72
4.6 两个数组的交集 II 75
4.6.1 算法要求 75
4.6.2 解题思路 76
4.6.3 解题代码 78
4.7 加一 82
4.7.1 算法要求 82
4.7.2 解题思路 82
4.7.3 解题代码 84
4.8 移动零 87
4.8.1 算法要求 87
4.8.2 解题思路 87
4.8.3 解题代码 89
4.9 两数之和 92
4.9.1 算法要求 92
4.9.2 解题思路 92
4.9.3 解题代码 93
4.10 有效的数独 96
4.10.1 算法要求 96
4.10.2 解题思路 97
4.10.3 解题代码 99
4.11 旋转图像 100
4.11.1 算法要求 100
4.11.2 解题思路 101
4.11.3 解题代码 102

第 5 章 字符串 104

5.1 反转字符串 104
5.1.1 算法要求 104
5.1.2 解题思路 104
5.1.3 解题代码 105
5.2 整数反转 108
5.2.1 算法要求 108
5.2.2 解题思路 109
5.2.3 解题代码 110
5.3 字符串中的第一个唯一字符 113
5.3.1 算法要求 113

5.3.2 解题思路 114
5.3.3 解题代码 115
5.4 有效的字母异位词 118
5.4.1 算法要求 118
5.4.2 解题思路 119
5.4.3 解题代码 120
5.5 验证回文字符串 123
5.5.1 算法要求 123
5.5.2 解题思路 124
5.5.3 解题代码 125
5.6 字符串转整数 128
5.6.1 算法要求 128
5.6.2 解题思路 129
5.6.3 解题代码 130
5.7 实现 strStr() 132
5.7.1 算法要求 132
5.7.2 解题思路 132
5.7.3 解题代码 133
5.8 报数 135
5.8.1 算法要求 135
5.8.2 解题思路 135
5.8.3 解题代码 137
5.9 最长公共前缀 139
5.9.1 算法要求 139
5.9.2 解题思路 139
5.9.3 解题代码 140

第 6 章 链表 142

6.1 删除链表中的节点 142
6.1.1 算法要求 142
6.1.2 解题思路 143
6.1.3 解题代码 143
6.2 删除链表的倒数第 n 个节点 145
6.2.1 算法要求 145
6.2.2 解题思路 145
6.2.3 解题代码 146
6.3 反转链表 150

6.3.1 算法要求 150
6.3.2 解题思路 150
6.3.3 解题代码 151
6.4 合并两个有序链表 154
6.4.1 算法要求 154
6.4.2 解题思路 154
6.4.3 解题代码 156
6.5 回文链表 157
6.5.1 算法要求 157
6.5.2 解题思路 158
6.5.3 解题代码 158
6.6 环形链表 160
6.6.1 算法要求 160
6.6.2 解题思路 161
6.6.3 解题代码 161

第 7 章 树 164

7.1 二叉树的最大深度 164
7.1.1 算法要求 164
7.1.2 解题思路 164
7.1.3 解题代码 165
7.2 验证二叉搜索树 167
7.2.1 算法要求 167
7.2.2 解题思路 167
7.2.3 解题代码 168
7.3 对称二叉树 170
7.3.1 算法要求 170
7.3.2 解题思路 170
7.3.3 解题代码 171
7.4 二叉树的层次遍历 173
7.4.1 算法要求 173
7.4.2 解题思路 173
7.4.3 解题代码 174
7.5 将有序数组转换成二叉搜索树 176
7.5.1 算法要求 176
7.5.2 解题思路 176
7.5.3 解题代码 177

需要全本请访问 www.ertongbook.com

第 8 章　排序和搜索&设计问题 179

8.1　合并两个有序数组 179
8.1.1　算法要求 179
8.1.2　解题思路 179
8.1.3　解题代码 181
8.2　第一个错误的版本 184
8.2.1　算法要求 184
8.2.2　解题思路 184
8.2.3　解题代码 185
8.3　数组洗牌（Shuffle an Array） 187
8.3.1　算法要求 187
8.3.2　解题思路 187
8.3.3　解题代码 188
8.4　最小栈 190
8.4.1　算法要求 190
8.4.2　解题思路 190
8.4.3　解题代码 190

第 9 章　动态规划 193

9.1　爬楼梯 193
9.1.1　算法要求 193
9.1.2　解题思路 194
9.1.3　解题代码 195
9.2　买卖股票的最佳时机 198
9.2.1　算法要求 198
9.2.2　解题思路 199
9.2.3　解题代码 199
9.3　最大子序和 202
9.3.1　算法要求 202
9.3.2　解题思路 203
9.3.3　解题代码 203
9.4　寻找宝物 205
9.4.1　算法要求 205
9.4.2　解题思路 205
9.4.3　解题代码 206

第 10 章 数 学 208
10.1 FizzBuzz 208
10.1.1 算法要求 208
10.1.2 解题思路 209
10.1.3 解题代码 209
10.2 计算质数 211
10.2.1 算法要求 211
10.2.2 解题思路 211
10.2.3 解题代码 216
10.3 3 的幂 219
10.3.1 算法要求 219
10.3.2 解题思路 220
10.3.3 解题代码 220
10.4 罗马数字转整数 222
10.4.1 算法要求 222
10.4.2 解题思路 223
10.4.3 解题代码 223
第 11 章 其他 226
11.1 位 1 的个数 226
11.1.1 算法要求 226
11.1.2 解题思路 226
11.1.3 解题代码 227
11.2 汉明距离 228
11.2.1 算法要求 228
11.2.2 解题思路 229
11.2.3 解题代码 229
11.3 颠倒二进制位 231
11.3.1 算法要求 231
11.3.2 解题思路 231
11.3.3 解题代码 232
11.4 帕斯卡三角形 233
11.4.1 算法要求 233
11.4.2 解题思路 234
11.4.3 解题代码 234
11.5 有效的括号 236
11.5.1 算法要求 236

11.5.2 解题思路 236
11.5.3 解题代码 237
11.6 缺失数字 239
11.6.1 算法要求 239
11.6.2 解题思路 239
11.6.3 解题代码 239

第 1 章

◂ 浅谈算法 ▸

学习计算机编程，需要解决的不会总是简单的计算问题，也不会总是简单的逻辑判断。总会有一些具体、实际的问题需要解决。要解决这些问题，光靠计算和逻辑判断恐怕是不够的。即使勉强达到目的，效率也不会太高。这时候就该算法上场了。

1.1 算法概述

【算法的定义】

在学计算机时，曾流行一种观点，程序=算法+数据结构。虽然这种观点一直都颇具争议，但也间接地说明了算法的重要性。

算法，简单地说就是利用计算机解决问题的步骤。算法必须具备两个重要条件。

- 第一，有效性。算法必须要给出正确的结果。
- 第二，终止性。算法不能无限循环，必须要有个终止条件。

【算法的作用】

到达一个目的地，可以步行，可以乘交通工具，还可以自驾。选择任何一种出行方式，并不影响到达目的地这个结果，但总有一个最合适的方法。

例如，从武汉到北京，最快的方法是乘飞机，最经济的方法是高铁（相信我，步行的开销一定比高铁贵），最自由的方法是自驾。换个目的地，从武汉到长沙呢？最快的方法变成了高铁。最经济的方法是自驾。

出门旅行可以根据不同的目的地、不同的需求选择一个最合适的方法。同样，用计算机解决一个问题，也可以有不同的算法。选择任何一个算法都不会影响结果的正确性。有的算法快，有的算法省空间（不需要重新申请空间，in-place），有的算法简单明了，有的算法思路巧妙……可以根据不同的需求，选择最合适的那个算法。

1.2 度量算法

衡量一个算法是不是好算法，一般是看这个算法的时间复杂度和空间复杂度。时间复杂度决定了算法运行时间的长短。空间复杂度决定了计算时所需的资源（空间资源）多少。通常很难两者兼顾。根据不同的需求，优先满足一方面即可。大多数情况下都是优先缩短运行时间。

1.2.1 时间复杂度

一个算法执行所耗费的时间，从理论上是不能算出来的，必须上机运行测试才能知道。但我们不可能也没有必要对每个算法都上机测试，只需知道哪个算法花费的时间多、哪个算法花费的时间少就可以了。并且一个算法花费的时间与算法中语句的执行次数成正比例，哪个算法中语句执行次数多，它花费的时间就多。对于时间复杂度，通常都记作大写字母 O。

以一个长度为 n 的列表 sList 求和为例。遍历一次 sList 列表，代码如下：

```
sum = 0
for I in sList: #len(sList) = n
    sum += i
```

此时的时间复杂度为 $O(n)$。

从长度为 n 的列表 sList 中获取两个数，要求这两个数的和是一个定值，最直接的算法是嵌套循环，代码如下：

```
sum = 16
for I in sList:
    for j in sList:
        if I + j == x:
            return (I, j)
```

此时的时间复杂度是 $O(n^2)$（当然，这绝对不是最优的算法）。如果 3 个数的和是定值呢？用最笨的算法，3 层嵌套循环，时间复杂度是 $O(n^3)$。

常见的时间复杂度有 $O(1)$、$O(\text{Log}_2{}^n)$、$O(n)$、$O(n\ \text{Log}_2{}^n)$、$O(n^2)$……

1.2.2 空间复杂度

空间复杂度（Space Complexity）是对一个算法在运行过程中临时占用存储空间大小的量度。一个算法在计算机存储器上所占用的存储空间包括存储算法本身所占用的存储空间、算法的输入输出数据所占用的存储空间和算法在运行过程中临时占用的存储空间这 3 个方面。算法的输入输出数据所占用的存储空间是由要解决的问题决定的，是通过参数表由调用函数传递而来的，它不随本算法的不同而改变。

存储算法本身所占用的存储空间与算法书写的长短成正比，要压缩这方面的存储空间，就必须编写出较短的算法。算法在运行过程中临时占用的存储空间因算法而异，有的算法只需要占用少量的临时工作单元，而且不随问题规模的大小而改变，称这种算法是“就地”（in-place）进行的，是节省存储的算法。

当一个算法的空间复杂度为一个常量，即不随被处理数据量 n 的大小而改变时，可表示为 $O(1)$；当一个算法的空间复杂度与以 2 为底的 n 的对数成正比时，可表示为 $O(\text{Log}_2{}^n)$；当一个算法的空间复杂度与 n 成线性比例关系时，可表示为 $O(n)$。

1.3 Python&Pythonic

不管什么算法，最终都是以编程语言来运行实现的。本书所有章节都是以最新版本的 Python 3 来运行实现算法的。

虽然不管什么编程语言大致都是一个模式，即顺序、选择、循环……但是不同的编程语言各有特色风格。例如，C 语言的 i++、++i，C++的 for(;;)，无须去看代码的头部，一眼就能分辨出这些代码属于什么编程语言。有鲜明 Python 风格的代码被称为 Pythonic。在《Python 之禅》中，对 Pythonic 的定义为：

- 优美胜于丑陋（Python 以编写优美的代码为目标）。
- 明了胜于晦涩（优美的代码应当是明了的，命名规范，风格相似）。
- 简洁胜于复杂（优美的代码应当是简洁的，不要有复杂的内部实现）。
- 复杂胜于凌乱（如果复杂不可避免，那代码间也不能有难懂的关系，要保持接口简洁）。
- 扁平胜于嵌套（优美的代码应当是扁平的，不能有太多的嵌套）。
- 间隔胜于紧凑（优美的代码有适当的间隔，不要奢望一行代码解决问题）。
- 可读性很重要（优美的代码是可读的）。

或者可以简单地理解为尽可能使用 Python 特有的函数，尽可能简洁、扁平，尽可能通俗易懂……

以求 100 以内所有整数和为例，代码如下：

```
s = 0
i = 1
while i <= 100:
s += i
    i += 1
```

这样当然是对的，但是这种写法偏 C 语言风格。如果只看这一部分代码，那么无法分辨这段代码是 C、C++还是 Python。这不是 Python 风格。比较 Pythonic 的写法如下：

```
s = 0
for i in range(101):
    s += i
```


需要全本请访问 www.ertongbook.com