Broadview®
www.broadview.com.cn

程序员刷题宝典！编程能力提升秘籍！

程序员代码面试指南

IT名企算法与数据结构题目最优解

（第2版）

左程云 著

精选IT名企真实代码面试题，全面覆盖算法与数据结构题型

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

程序员代码面试指南

IT名企算法与数据结构题目最优解

（第2版）

左程云 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

这是一本程序员代码面试宝典！书中对IT名企代码面试各类题目的最优解进行了总结，并提供了相关代码实现。针对当前程序员面试缺乏权威题目汇总这一痛点，本书选取将近200道真实出现过的经典代码面试题，帮助广大程序员做充分的面试准备。“刷”完本书后，你就是“题王”！

本书采用“题目+解答”的方式组织内容，并把面试题类型相近或者解法相近的题目尽量放在一起，读者在学习本书时很容易看出面试题解法之间的联系，使知识的学习避免碎片化。本书将所有的面试题从难到易依次分为“将”“校”“尉”“士”四个档次，方便读者有针对性地选择“刷”题。本书收录的所有面试题都给出了最优解讲解和代码实现，并且提供了一些普通解法和最优解法的运行时间对比，让读者真切地感受到最优解的魅力！

本书中的题目全面且经典，更重要的是，书中收录了大量新题目和最优解分析，这些内容源自笔者多年来“死磕自己”的深入思考。

程序员们做好准备在IT名企的面试中脱颖而出、一举成名了吗？这本书就是你应该拥有的“神兵利器”。当然，对需要提升算法和数据结构等方面能力的程序员而言，本书的价值也是显而易见的。

图书在版编目（CIP）数据

程序员代码面试指南：IT 名企算法与数据结构题目最优解 / 左程云著.—2 版. —北京：电子工业出版社，2019.1
ISBN 978-7-121-35486-1

Ⅰ. ①程… Ⅱ. ①左… Ⅲ. ①程序设计－资格考试－自学参考资料 Ⅳ. ①TP311.1

中国版本图书馆 CIP 数据核字(2018)第 251338 号

策划编辑：牛　勇
责任编辑：李利健
印　　刷：三河市君旺印务有限公司
装　　订：三河市君旺印务有限公司
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
开　　本：787×980　1/16　印张：36　字数：871 千字
版　　次：2015 年 9 月第 1 版
　　2019 年 1 月第 2 版
印　　次：2019 年 1 月第 1 次印刷
定　　价：109.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

献给左军和谢桂兰

第 2 版说明

1．修改了第 1 版部分题目的解释，并增加了更多示例。

2．增加了很多近年来新出现的流行面试题，删掉了已经很少出现的低频面试题。

3．把经常出现的解题套路与算法原型做了结构化的调整和总结。

4．本书所有题目的代码都用 Java 语言实现，但这并不会妨碍其他语言使用者的阅读。这是因为笔者尽最大努力回避了与 Java 语言特性相关的代码实现，而且尽量遵循大多数编程语言共有的写法习惯。因此，将本书中的 Java 语言实现改写成其他语言的实现是非常容易的。

5．在 Java 语言中，如果想得到字符串 str 第 *i* 个位置的字符，则需用如下方式：

```
char p = str.charAt(i);
```

本书提供的函数中有大量参数为字符串类型的函数，但如上所示的方式并不符合大多数读者的阅读习惯。为了让代码更加易读，笔者在这些函数中把字符串类型的参数转换成 char 类型数组的变量来使用。例如：

```
char[] charArr = str.toCharArray();
```

此时得到字符串 str 第 *i* 个位置的字符，可以用如下方式：

```
char p = charArr[i];
```

在本书中，发生如上转换行为的函数在估算额外空间复杂度时，笔者并没有把 charArr 的空间计算在内，这是因为如果不转换成 char 数组，而是选择直接使用原参数 str，也是完全可以的，之所以选择转换，仅仅是为了让读者更容易读懂代码；是否进行转换对算法的逻辑没有任何影响，所以不把 charArr 的空间算作必须使用的额外空间。

推荐序

2015 年春节，因为公司业务的快速发展，我们开始寻觅优秀的笔试和面试算法讲师。几经周折，找到了当时在举办线下算法分享的程云，认认真真地听他讲了一堂课，当时就认定他就是我们要找的人。

我听过很多国内顶尖 ACM 选手的算法分享，但是每次听完后总觉得我和那些人永远隔着一个断裂带，算法对我来说遥不可及，而程云讲解算法的时候总能从最小的切口讲起，由浅入深，环环相扣，不知不觉引你走向算法的核心精髓，那种醍醐灌顶的感觉能激发大家学习算法的热情，并一直推着我们前进。

这几年 IT 技术蓬勃发展，日新月异，对技术人才的需求日益增长，程序员招聘市场也如火如荼。在有限的三五轮面试中，国外流行使用让面试者编程解决某些数据结构和算法的题目，通过观察面试者编码的熟练程度、思考的速度和深度来衡量面试者的能力和潜力。国内以百度、阿里、腾讯为代表的互联网企业也都开始采用算法面试来筛选人才。

程云出于对算法的热爱，长期“泡”在 CareerCup、LeetCode 等笔试和面试网站上，编码解决各种最新的笔试和面试编程题，对各种笔试和面试编程题的解题技巧了如指掌。

算法面试普及后，传统的数据结构和算法课本讲得太过基础，又远离求职需求，国内逐渐出现迎合求职需求的笔试和面试工具书，这些书籍有些过于应试，纯粹以通过面试为导向。程云的书和那些书相比，题目更前沿，讲解更注重思考思路和代码的实践技巧，对每个题目都深挖最优解，同时根据自己在线下讲课学员们的反馈，对每个编程考题的解题反复修改，让思路更清晰。

这本书不仅可以作为面试代码指南，还可以作为学生课后的辅助练习，“刷”题多年的经验悉数总结都沉淀在这本书里，相信读者跟着他的引导从头到尾逐一攻克难题，就一定会有所收获。

叶向宇

牛客网 CEO

自　序

我能出书挺意外的。

虽然我早就知道想进入那些大公司要靠“刷”代码面试题来练习编写代码的能力，可是在 6 年前的某一天，我突然有了心情去看代码面试题长什么样子，于是收集了代码面试的题目。了解得越深入，我就越有一种恐慌的感觉，因为感觉自己什么都不太在行，对一个归并排序（Merge sort）写出完整的代码都感觉挺费劲的，面对这个冯·诺依曼发明的排序算法，我真的有底气说自己是计算机专业的学生吗？这种打击并没有持续太久，因为爱耍小聪明的人总会特别自信。我决定开始认真面对“刷”题这件事，但那时我根本不知道我即将面对什么，更不会有写书的念头。

我把课余时间利用起来，心想：不就是“刷”题吗？别人能写出来，咱也能写出来。起初的心态是我不服，我就想告诉自己能行。过程虐心是肯定的，经常半夜因为看到一个复杂度特别低的算法自己真的不能理解而沮丧地睡不着觉。当时觉得找不到资料能彻底让我明白，书上讲得太粗浅，网上讲的太散乱，代码写得看不懂。起初我“刷”题的时候无数次地想放弃，因为觉得这些都是什么玩意儿！我为什么放着好好的日子不过，去找这种罪受？可是我又不甘心，虽然我不懂很多解法，但是我觉得它们真的很有意思。

我将能买到的所有相关书籍上的所有题目全都研究了一遍，无论是中文的还是英文的，我都硬着头皮“啃”。写完每道题后，我都和书上的方法进行反复对比。“啃”完了五六本书之后，距离我刚开始“刷”题已经过去 16 个月了。写书？别逗了，才刚看完。

“年轻人总会找借口说这个东西不是我感兴趣的，所以做不好是应该的。但他们没有注意的是，你面对的事情中感兴趣的事情总是少数，这就使得大多数时候你做事情的态度总是很懈怠、很消极，这使你变成了一个懈怠的人。当你真正面对自己感兴趣的东西时，你发现你已经攥不紧拳头了。”时常想起本科时的毕业设计指导老师——高鹏义老师说的这段话。说得对！对一个东西，如果你没有透彻研究过，就不要轻易说它不精彩。这不是博爱，而是对自己认真。

“刷”题代码达到 4 万行的时候，我基本上成了国内外所有热门“刷”题网站的日常用户，此时我确认了一件事情，今天的代码面试指导真的处在一个很初级的阶段，这种不健全是全方面的。

例如：

- 经常看到一篇文章前后的语境是割裂的，作者经常根据之前的一个优良解法提出更好的优化方式，但整篇文章都不提及之前的解法是什么。这就导致初学者根本无法看懂。
- 几乎所有的书籍都忽略例子带来的引导作用，甚至还有不少书籍在阐述一个解法的时候只写伪代码，这就使得读者在看懂意思和自己真正能写出代码之间其实还有很多的路要走。
- 代码面试题目的特点是“多”“杂”“难”，从着手开始学习到最终达到自己想要的效果之间，自己对自己的评估根本无从谈起。“慢慢练吧，学海无涯”成为主要的心态，这就难免会产生怀疑的情绪。
- 看见一道新的面试题时还是会无从下手，因为之前的学习无法做到举一反三，对自己做过的题目缺乏总结和归纳。

难道“刷”题真的只适合“聪明人”？我不这么看，既然大多数内容处在有待商榷的阶段，那我就去学习原论文吧。

记得当时我一个人在国外，在初冬的一个下午，“刷”题已经两年之久，快吃晚饭的时候，我突然想起自己忘了吃午饭，就冲出家门去觅食。站在 7-11 门前的广场上，我拿着 1.5 美元的热狗和 75 美分的咖啡，微温的阳光撒在身上，远远地望着即将消失的太阳。我停下来，把咖啡放在斑驳的石头台子上，手里的热狗挺好看，香肠和洋葱都挺新鲜，清冷的空气吹过来，却让我的心绪更乱。旧金山的天空五彩斑斓，让漂泊者头晕目眩。哭得跟个鬼似的我除了想家，哪里敢设想自己会出书呢？

当我意识到在网上很难搜索到新鲜的题目时，我已经换了两家公司，反复实现了 600 多道题目，编写了差不多 10 万行代码。原来只是为了找份工作“刷”题这一初心早就忘了，而变成了兴趣并坚持了这么久，我自己也感到意外。更奇怪的是，我已经完全乐在其中，同时交流欲望越来越强，时常和同事们展开这方面的讨论。我发现很多书上的解法不是最优，很多题目其实和同事们讨论的做法更好，可以发现高手特别多，但好像都懒得动笔。

有一天，我看到自己写的题目，想到自己那些“抓心挠肝”的日子，突然觉得要不出书吧？我已经离不开这种感觉了，如果这不是真爱，那什么才是呢？

这不是一个励志的故事，是一个爱“刷”题的人决定把很多最优解讲出来的过程，就这么简单。

左程云

2015 年 7 月 20 日

目　录

第 1 章

栈和队列

设计一个有 getMin 功能的栈

【题目】

实现一个特殊的栈，在实现栈的基本功能的基础上，再实现返回栈中最小元素的操作。

【要求】

1．pop、push、getMin 操作的时间复杂度都是 $O(1)$。

2．设计的栈类型可以使用现成的栈结构。

【难度】

士 ★☆☆☆

【解答】

在设计时，我们使用两个栈，一个栈用来保存当前栈中的元素，其功能和一个正常的栈没有区别，这个栈记为 stackData；另一个栈用于保存每一步的最小值，这个栈记为 stackMin。具体的实现方式有两种。

第一种设计方案

（1）压入数据规则

假设当前数据为 newNum，先将其压入 stackData。然后判断 stackMin 是否为空：

- 如果为空，则 newNum 也压入 stackMin。
- 如果不为空，则比较 newNum 和 stackMin 的栈顶元素中哪一个更小：
  - 如果 newNum 更小或两者相等，则 newNum 也压入 stackMin；

➢ 如果 stackMin 中栈顶元素小，则 stackMin 不压入任何内容。

举例：依次压入 3、4、5、1、2、1 的过程中，stackData 和 stackMin 的变化如图 1-1 所示。

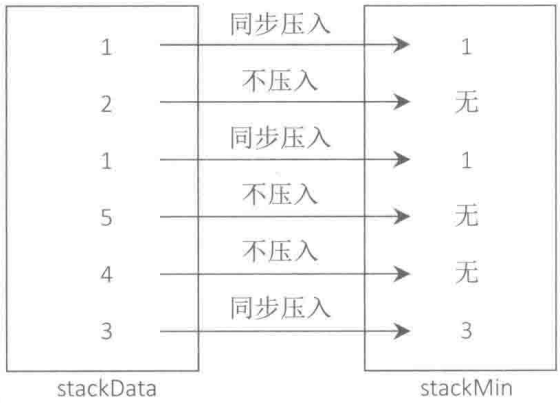

图 1-1

（2）弹出数据规则

先在 stackData 中弹出栈顶元素，记为 value。然后比较当前 stackMin 的栈顶元素和 value 哪一个更小。

通过上文提到的压入规则可知，stackMin 中存在的元素是从栈底到栈顶逐渐变小的，stackMin 栈顶的元素既是 stackMin 栈的最小值，也是当前 stackData 栈的最小值。所以不会出现 value 比 stackMin 的栈顶元素更小的情况，value 只可能大于或等于 stackMin 的栈顶元素。

当 value 等于 stackMin 的栈顶元素时，stackMin 弹出栈顶元素；当 value 大于 stackMin 的栈顶元素时，stackMin 不弹出栈顶元素，返回 value。

很明显可以看出，压入与弹出规则是对应的。

（3）查询当前栈中的最小值操作

由上文的压入数据规则和弹出数据规则可知，stackMin 始终记录着 stackData 中的最小值。所以，stackMin 的栈顶元素始终是当前 stackData 中的最小值。

方案一的代码实现如 MyStack1 类所示：

```
public class MyStack1 {
        private Stack<Integer> stackData;
        private Stack<Integer> stackMin;

        public MyStack1() {
                this.stackData = new Stack<Integer>();
                this.stackMin = new Stack<Integer>();
        }

        public void push(int newNum) {
                if (this.stackMin.isEmpty()) {
```