Broadview®
www.broadview.com.cn

本书前版已出版英文版
在全球发行

剑指Offer

第2版

名企面试官精讲典型编程题

何海涛◎著

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

剑指Offer

名企面试官精讲典型编程题

（第2版）

何海涛 著

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

内容简介

本书前身曾在全球范围内发行过英文版。这一版本在前版基础上进一步精选和增补试题，结合作者近年来在美国从事开发工作的实践经验及思考积累，使全书更加融会贯通、广泛适用。本书剖析了 80 道典型的编程题，系统整理基础知识、代码质量、解题思路、优化效率和综合能力这 5 个面试要点。全书共分 7 章，主要包括面试的流程，讨论面试每一环节需要注意的问题；面试需要的基础知识，从编程语言、数据结构及算法三方面总结程序员面试知识点；高质量的代码，讨论影响代码质量的 3 个要素（规范性、完整性和鲁棒性），强调高质量代码除完成基本功能外，还能考虑特殊情况并对非法输入进行合理处理；解决面试题的思路，总结编程面试中解决难题的有效思考模式，如在面试中遇到复杂难题，应聘者可利用画图、举例和分解这 3 种方法将其化繁为简，先形成清晰思路，再动手编程；优化时间和空间效率，读者将学会优化时间效率及用空间换时间的常用算法，从而在面试中找到最优解；面试中的各项能力，总结应聘者如何充分表现学习和沟通能力，并通过具体面试题讨论如何培养知识迁移、抽象建模和发散思维能力；两个面试案例，总结哪些面试举动是不良行为，而哪些表现又是面试官所期待的行为。

图书在版编目（CIP）数据

剑指 Offer：名企面试官精讲典型编程题 / 何海涛著. —2 版. —北京：电子工业出版社，2017.5

ISBN 978-7-121-31092-8

Ⅰ. ①剑… Ⅱ. ①何… Ⅲ. ①程序设计－资格考试－习题集 Ⅳ. ①TP311.1-44

中国版本图书馆 CIP 数据核字（2017）第 053903 号

策划编辑：张春雨
责任编辑：徐津平
特约编辑：赵树刚
印　　刷：三河市良远印务有限公司
装　　订：三河市良远印务有限公司
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：22　　字数：423 千字
版　　次：2011 年 11 月第 1 版
　　2017 年 5 月第 2 版
印　　次：2017 年 5 月第 1 次印刷
定　　价：65.00 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。

第 2 版序言

时间总是在不经意间流逝，我们也在人生的旅途上不断前行，转眼间我在微软的美国总部工作近两年了。生活总给我们带来新的挑战，同时也有新的惊喜。这两年在陌生的国度里用着不太流利的英语和各种肤色的人交流，体验着世界的多元化。这两年加过班、熬过夜，也为进展不顺的项目焦头烂额过。在微软 Office 新产品发布那天，也自豪过，忍不住在朋友圈里和大家分享自己的喜悦和兴奋。2015 年 4 月，我和素云又一次迎来了一个小生命。之后的日子虽然辛苦，但每当看着呼呼、阳阳两兄弟天真灿烂的笑容时，我的心里只有无限的幸福。

西雅图是一个 IT 氛围很浓的地方，这里是微软和亚马逊的总部所在地，Google、Facebook 等很多知名公司都在这里有研发中心。一群程序员聚在一起，总会谈到谁去这家公司面试了，谁拿到了那家公司的 Offer。这让我有机会从多个角度去理解编程面试，也更加深入地思考怎样刷题才会更加有效。我的这些理解、思考都融入《剑指 Offer——名企面试官精讲典型编程题》这本书的第 2 版里。

这次再版在第 1 版的基础上增加了新的面试题，涵盖了新的知识点。第 2 版新增了 2.4.3 节和 2.4.4 节，分别讨论回溯法、动态规划和贪婪算法。正则表达式是编程面试时经常出现的内容，本次新增了两个正则表达式匹配的问题（详见面试题 19 和面试题 20）。

这次新增的内容有些是原有内容的延伸。比如原书的面试题 35 要求找出字符串中第一个只出现一次的字符［在第 2 版中为面试题 50（题目一）］。这次新增的面试题 50（题目二）把要求改为从一个字符流中找出第一个只出现一次的字符。再比如，在原书的面试题 23［在第 2 版中为面试题 32（题

目一）]中讨论了如何把二叉树按层打印到一行里，这次新增了两个按层打印二叉树的面试题：面试题 32（题目二）要求把二叉树的每一层单独打印到一行；面试题 32（题目三）要求按之字形顺序打印二叉树。

计算机领域的知识更新很快，编程面试题也需要推陈出新。本书的参考代码以 C++为主，这次再版根据 C++新的标准在内容上进行了一些调整。例如，原书的面试题 48 要求用 C++实现不能继承的类。由于在 C++ 11 中引入了关键字 final，那么用 C++实现不能继承的类已经变得非常容易。因此，这次再版时用新的面试题替代了它。

自本书出版以来，收到了很多读者的反馈，让我受益匪浅。例如，面试题 20“表示数值的字符串”根据 GitHub 用户 cooljacket 的意见做了修改。在此对所有提出反馈、建议的读者表示衷心的感谢。

本书所有源代码（包含单元测试用例）都分享在 GitHub 上，欢迎读者对本书及 GitHub 上的代码提出意见。如果发现代码中存在问题，或者发现还有更好的解法，则欢迎读者递交代码。本书所有源代码均以 BSD 许可证开源，欢迎大家共同参与，一起提高代码的质量。

通过读者的 E-mail，我很高兴地得知《剑指 Offer——名企面试官精讲典型编程题》一书陪伴很多读者找到了心仪的工作，拿到了满意的 Offer。实际上，这本书不仅仅是一本关于求职面试的工具书，同时还是一本关于编程的技术书。书中用大量的篇幅讨论数据结构和算法，讨论如何才能写出高质量的代码。这些技能在面试的时候有用，在平时的开发工作中同样有用。希望本书能陪伴更多的读者在职场中成长。

何海涛

2016 年 12 月 7 日深夜于美国雷德蒙德

推荐序一

海涛2008年在我的团队做过软件开发工程师。他是一名很细心的员工，对面试这个话题很感兴趣，经常和我及其他员工讨论，积累了很多面试方面的技巧和经验。他曾跟我提过想要写本有关面试的书，如今他把书写出来了！他是一个有目标、有耐心和持久力的人。

我在微软做了很多年的面试官，后面 7 年多作为把关面试官，也面试了很多应聘者。应聘者要想做好面试，确实应把面试当作一门技巧来学习，更重要的是要提高自身的能力。我遇到很多应聘者可能自身能力也不差，但因为不懂得怎样回答提问，不能很好地发挥。也有很多刚走出校园的应聘者也学过数据结构和算法分析，可是在处理具体问题时不能用学过的知识来有效地解决。这些朋友读读海涛的这本书，会受益匪浅，在面试中的发挥也会有很大提高。这本书也可以作为很好的教学补充资料，让学生不仅学到书本知识，也学到解决问题的方法。

在向我汇报的员工中有面试发挥很好但工作平平的，也有面试一般但工作优秀的。对于追求职业发展的人来说，通过面试只是迈过一道门槛而不是目的，真正的较量是在入职后的成长。就像学钓鱼，你可能在有经验的垂钓者的指导下能钓到几条鱼，但如果没有学到垂钓的真谛，离开了指导者，你可能就很难钓到很多鱼。我希望读这本书的朋友不要只学一些技巧来应付面试，而是通过学习如何解决面试中的难题来提高自己的编程和解决问题的能力，进而提高自信心，在职场中迅速成长。

徐鹏阳（Pung Xu）
Principal Development Manager, Search Technology Center Asia
Microsoft

推荐序二

I had the privilege of working with Harry at Microsoft. His background and industry experience are a great asset in learning about the process and techniques of technical interviews. Harry shares practical information about what to expect in a technical interview that goes beyond the core engineering skills. An interview is more than a skills assessment. It is the chance for you and a prospective employer to gauge whether there is a mutual fit. Harry includes reminders about the key factors that can determine a successful interview as well as success in your new job.

Harry takes you through a set of interview questions to share his insight into the key aspects of the question. By understanding these questions, you can learn how to approach any question more effectively. The basics of languages, algorithms and data structures are discussed as well as questions that explore how to write robust solutions after breaking down problems into manageable pieces. Harry also includes examples to focus on modeling and creative problem solving.

The skills that Harry teaches for problem solving can help you with your next interview and in your next job. Understanding better the key problem solving techniques that are analyzed in an interview can help you get the first job after university or make your next career move.

Matt Gibbs
Direct of Development, Asia Research & Development
Microsoft Corporation

前　言

自 2011 年 9 月以来，我的面试题博客（http://zhedahht.blog.163.com/）点击率上升很快，累计点击量超过 70 万次，并且平均每天还会增加约 3000 次点击。每年随着秋季新学期的开始，新一轮招聘高峰也即将来到。这不禁让我想起几年前自己找工作的情形。那个时候的我，也是在网络的各个角落搜索面试经验，尽可能多地搜集各家公司的面试题。

当时网上的面试经验还很零散，应聘者如果想系统地搜集面试题，则需要付出很大的努力。于是我萌生了一个念头，在博客上系统地搜集、整理有代表性的面试题，这样可以极大地方便后来人。经过一段时间的准备，我于 2007 年 2 月在网易博客上发表了第一篇关于编程面试题的博文。

在之后的日子里，我陆续发表了 60 余篇关于面试题的博文。随着博文数目的增加，我也逐渐意识到一篇篇博文仍然是零散的。一篇博文只是单纯地分析一道面试题，但对解题思路缺乏系统性的梳理。于是，2010 年 10 月，我有了把博文整理成一本书的想法。经过努力，这本书终于和读者见面了。

本书内容

全书分为 7 章，各章的主要内容如下：

第 1 章介绍面试的流程。通常整个面试过程可以分为电话面试、共享桌面远程面试和现场面试 3 个阶段，每轮面试又可以分为行为面试、技术面试和应聘者提问 3 个环节。本章详细讨论了面试中每个环节需要注意的问题。其中，1.3.2 节深入讨论了技术面试中的 5 个要素，是全书的大纲，

接下来的第 2～6 章将逐一讨论每个要点。

第 2 章梳理应聘者在接受技术面试时需要用到的基础知识。本章从编程语言、数据结构及算法 3 个方面总结了程序员面试的知识点。

第 3 章讨论应聘者在面试时写出高质量代码的 3 个要点。通常面试官除了期待应聘者写出的代码能够完成基本的功能，还能应对特殊情况并对非法输入进行合理的处理。读完这一章，读者将学会如何从规范性、完整性和鲁棒性 3 个方面提高代码的质量。

第 4 章总结在编程面试中解决难题的常用思路。如果在面试过程中遇到复杂的难题，那么应聘者最好在写代码之前形成清晰的思路。读者在读完这一章之后，将学会如何用画图、举例和分解这 3 种思路来解决问题。

第 5 章介绍如何优化代码的时间效率和空间效率。如果一个问题有多种解法，那么面试官总是期待应聘者能找到最优的解法。读完这一章，读者将学会优化时间效率及用空间换时间的常用算法。

第 6 章总结面试中的各项能力。在面试过程中，面试官会一直关注应聘者的学习能力和沟通能力。除此之外，有些面试官还喜欢考查应聘者的知识迁移能力、抽象建模能力和发散思维能力。读完这一章，读者将学会如何培养和运用这些能力。

第 7 章是两个面试案例。在这两个案例中，读者将看到应聘者在面试过程中的哪些举动是不好的行为，而哪些表现又是面试官所期待的行为。衷心地希望应聘者能在面试时少犯甚至不犯错误，完美地表现出自己的综合素质，最终拿到心仪的 Offer。

本书特色

正如前面提到的那样，本书的原型是我多年来陆陆续续发表的几十篇博文，但这本书也不仅仅是这些博文的总和，它在博文的基础上添加了如下内容：

本书试图以面试官的视角来剖析面试题。本书前 6 章的第一节都是“面试官谈面试”，收录了分布在不同 IT 企业（或者 IT 部门）的面试官对代码质量、应聘者如何形成及表达解题思路等方面的理解。在本书中穿插着几十条“面试小提示”，是我作为面试官给应聘者在面试方法、技巧方面的建议。在第 7 章的案例中，“面试官心理”揭示了面试官在听到应聘者不同回

答时的心理活动。应聘者如果能了解面试官的心理活动，则无疑能在面试时更好地表现自己。

本书总结了解决面试难题的常用方法，而不仅仅是解决一道道零散的题目。在仔细分析、解决了几十道典型的面试题之后，我发现，其实是有一些通用的方法可以在面试的时候帮助我们解题的。举个例子，如果面试的时候遇到的题目很难，那么我们可以试着把一个大的、复杂的问题分解成若干小的、简单的子问题，然后递归地去解决这些子问题。再比如，我们可以用数组实现一个简单的哈希表解决一系列与字符串相关的面试题。在详细分析了一道面试题之后，很多章节都会在“相关题目”中列举同类型的面试题，并在“举一反三”中总结解决这一类型题目的方法和要点。

本书收集的面试题都是各大公司的编程面试题，极具实战意义。包括Google、微软在内的知名IT企业在招聘的时候都非常重视应聘者的编程能力，编程技术面试也是整个面试流程中最为重要的环节。本书选取的题目都是被各大公司面试官反复采用的编程题。如果读者一开始觉得书中的有些题目比较难，那也正常，没有必要感到气馁，因为像Google、微软、阿里巴巴、腾讯这样的大企业的面试本身就不简单。读者逐步掌握了书中总结的解题方法之后，编程能力和分析复杂问题的能力将会得到很大的提升，再去大公司面试将会轻松很多。

本书附带提供了80道编程题的完整的源代码，其中包含每道题的测试用例。很多面试官在应聘者写完程序之后，都会要求应聘者自己想一些测试用例来测试自己的代码，而一些没有实际项目开发经验的应聘者不知道如何进行单元测试。相信读者在读完本书后就会知道如何从基本功能测试、边界值测试、性能测试等方面去设计测试用例，从而提高编写高质量代码的能力。

本书体例

在本书的正文中间或者章节的末尾穿插了不少特殊体例。这些体例或用来给应聘者提出建议，或用来总结解题方法，希望能够引起读者的注意。

面试小提示：

本条目是从面试官的角度给应聘者提出的建议，或者希望应聘者能够注意到的细节。

源代码：

读者将在本条目看到一个指向 GitHub 的链接，可以到对应的网页上浏览代码。同时，读者也可以把代码下载到本地，用 Visual Studio 打开 CodingInterviewChinese2.sln 文件阅读或者调试代码。

测试用例：

本条目列举应聘者在面试时可以用来测试代码是否完整、鲁棒的单元测试用例。通常本书从基本功能、边界值、无效的输入等方面测试代码的完整性和鲁棒性，针对在时间效率或者空间效率方面有要求的面试题还包含性能测试的测试用例。

本题考点：

本条目总结面试官采用一道面试题的考查要点。

相关题目：

本条目列举一些和详细分析的面试题相关或者类似的面试题。

举一反三：

本条目从解决面试例题中提炼出常用的解题方法。这些解题方法能够应用到解决其他同类型的问题中去，达到举一反三的目的。

面试官心理：

在第 7 章的面试案例中，本条目用来模拟面试官听到应聘者的回答之后的心理活动。

关于遗漏的问题

由于时间仓促，再加上笔者的能力有限，书中难免会有一些遗漏。今后一旦发现遗漏的问题，我将第一时间在博客（http://zhedahht.blog.163.com/）上公布勘误信息。读者如果发现任何问题或者有任何建议，那么也请在博客上留言、评论，或者通过电子邮件（zhedahht@hotmail.com）和我联系。

致谢

在写博客及把博文整理成书的过程中，我得到了很多人的帮助。没有他们，也就没有这本书。因此，我想在这里对他们诚挚地说一声：谢谢！

首先我要谢谢个人博客上的读者。网友的鼓励让我在博客上的写作从2007年2月开始坚持到了现在。也正是由于网友的鼓励，我最终下定决心把博文整理成一本书。

在本书的写作过程中，我得到了很多同学、同事的帮忙，包括Autodesk的马凌洲、刘景勇、王海波、蓝诚，支付宝的殷焰，百度的张珺、张晓禹，英特尔的尹彦，交通银行的朱麟，淘宝的尧敏，微软的陈黎明、田超，英伟达的吴斌，SAP的何幸杰和华为的韩伟东（在书稿写作阶段他还在盛大工作）。感谢他们和大家分享了对编程面试的理解和思考。同时还要感谢GlaxoSmithKline Investment的Recruitment & HRIS Manager蔡咏来（也是2008年把我招进微软的HR）和大家分享了微软所推崇的STAR简历模型。还要感谢在微软期间我的两个老板徐鹏阳和Matt Gibbs，他们都是在微软有十几年面试经验的资深面试官，对面试有着深刻的理解。感谢二位在百忙之中抽时间为本书写序，为本书增色不少。

我同样要感谢现在思科的老板Min Lu及TQSG上海团队的同事王劭、赵斌和朱波对我的理解。他们在我写作期间替我分担了大量的工作，让我能够集中更多的精力来写书。

感谢电子工业出版社的工作人员，他们大到全书的构架，小到文字的推敲，都给予了我极大的帮助，从而使本书的质量有了极大的提升。

本书还得到了很多朋友的支持和帮助，限于篇幅，虽然不能在此一一说出他们的名字，但我一样对他们心存感激。

最后，我要衷心地感谢我的爱人刘素云。感谢她在过去一年中对我的理解和支持，为我营造了一个温馨而又浪漫的家，让我能够心无旁骛地写书。我无以为谢，谨以此书献给她及我们的孩子。

何海涛

2017 年 2 月于上海三泾南宅

轻松注册成为博文视点社区用户（www.broadview.com.cn），您即可享受以下服务。

- **下载资源：**本书所提供的示例代码及资源文件均可在【下载资源】处下载。
- **提交勘误：**您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **与我们交流：**在页面下方【读者评论】处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/31092

二维码：

目　录

第 1 章

面试的流程

1.1 面试官谈面试

“对于初级程序员，我一般会偏向考查算法和数据结构，看应聘者的基本功；对于高级程序员，我会多关注专业技能和项目经验。”

——何幸杰（SAP，高级工程师）

“应聘者要事先做好准备，对公司近况、项目情况有所了解，对所应聘的工作很有热情。另外，应聘者还要准备好合适的问题问面试官。”

——韩伟东（盛大，高级研究员）

“应聘者在面试过程中首先需要放松，不要过于紧张，这有助于后面解决问题时开拓思路。其次不要急于编写代码，应该先了解清楚所要解决的问题。这时候最好先和面试官多做沟通，然后开始做一些整体的设计和规划，这有助于编写高质量和高可读性的代码。写完代码后不要马上提交，最好自己检查并借助一些测试用例来测试几遍代码，找出可能出现的错误。”

——尧敏（淘宝，资深经理）

“‘神马’都是浮云，应聘技术岗位就是要踏实写程序。”

——田超（微软，SDE II）

1.2 面试的 3 种形式

如果应聘者能够通过公司的简历筛选环节，那恭喜他取得了阶段性的成功。但要想拿到心仪的 Offer，应聘者还有更长的路要走。大部分公司的面试都是从电话面试开始的。通过电话面试之后，有些公司还会有一两轮远程面试。面试官让应聘者共享自己的桌面，远程观察应聘者编写及调试代码的过程。如果前面的面试都很顺利，应聘者就会收到现场面试的邀请信，请他去公司接受面对面的面试。整个面试的流程我们可以用图 1.1 表示。

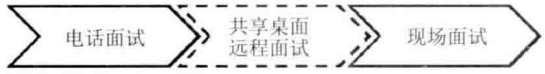

图 1.1 面试的形式和流程

注：只有少数公司有共享桌面远程面试环节。

1.2.1 电话面试

顾名思义，电话面试是面试官以打电话的形式考查应聘者。有些面试官会先和应聘者预约好电话面试的时间，而还有些面试官却喜欢搞突然袭击，一个电话打过去就开始面试。为了应付这种突然袭击，建议应聘者在投出简历之后的一两个星期之内，要保证手机电池能至少连续通话一小时。另外，应聘者不要长时间待在很嘈杂的地方。如果应聘者身在闹市的时候突然接到面试电话，那么双方就有可能因为听不清对方而倍感尴尬。

电话面试和现场面试最大的区别就是应聘者和面试官是见不到对方的，因此双方的沟通只能依靠声音。没有了肢体语言、面部表情，应聘者清楚地表达自己想法的难度就比现场面试时要大很多，特别是在解释复杂算法的时候。应聘者在电话面试的时候应尽可能用形象化的语言把细节说清楚。例如，在现场面试的时候，应聘者如果想说一棵二叉树的结构，则可以用笔在白纸上画出来，就一目了然了。但在电话面试的时候，应聘者就需要把二叉树中有哪些节点，每个节点的左子节点是什么、右子节点是什么都说得很清楚，只有这样面试官才能准确地理解应聘者的思路。

很多外企在电话面试时都会加上英语面试的环节，甚至有些公司全部

面试都会用英语进行。电话面试时应聘者只能听到面试官的声音而看不到他的口型，这对应聘者的听力提出了更高的要求。如果应聘者在面试的时候没有听清楚或者听懂面试官的问题，则千万不要不懂装懂、答非所问，这是面试的大忌。当不确定面试官的问题的时候，应聘者一定要大胆地向面试官多提问，直到弄清楚面试官的意图为止。

 面试小提示：

应聘者在电话面试的时候应尽可能用形象的语言把细节说清楚。

如果在英语面试时没有听清或没有听懂面试官的问题，则应聘者要敢于说 Pardon。

1.2.2　共享桌面远程面试

共享桌面远程面试（Phone-Screen Interview）是指利用一些共享桌面的软件（如微软的 Skype、思科的 WebEx 等），应聘者把自己电脑的桌面共享给远程的面试官。这样两个人虽然没有坐在一起，但面试官却能通过共享桌面观看应聘者编程和调试的过程。目前只有为数不多的几家大公司会在邀请应聘者到公司参加现场面试之前，先进行一两轮共享桌面远程面试。

这种形式的面试，面试官最关心的是应聘者的编程习惯及调试能力。通常面试官会认可应聘者下列几种编程习惯：

- **思考清楚再开始编码。**应聘者不要一听到题目就匆忙打开编程软件如 Visual Studio 开始敲代码，因为在没有形成清晰的思路之前写出的代码通常会漏洞百出。这些漏洞被面试官发现之后，应聘者容易慌张，这个时候再修改代码也会越改越乱，最终导致面试的结果不理想。更好的策略是应聘者应先想清楚解决问题的思路，如算法的时间、空间复杂度各是什么，有哪些特殊情况需要处理等，然后再动手编写代码。
- **良好的代码命名和缩进对齐习惯。**一目了然的变量和函数名，加以合理的缩进和括号对齐，会让面试官觉得应聘者有参与大型项目的开发经验。

- **能够进行单元测试。**通常面试官出的题目都是要求写函数解决某一问题，如果应聘者能够在定义函数之后，立即对该函数进行全面的单元测试，那就相当于向面试官证明了自己有着专业的软件开发经验。如果应聘者先写单元测试用例，再写解决问题的函数，那么我相信面试官定会对你刮目相看，因为能做到测试在前、开发在后的程序员实在是太稀缺了，他会毫不犹豫地抛出橄榄枝。

通常我们在写代码的时候都会遇到问题。当应聘者运行代码发现结果不对之后的表现，也是面试官关注的重点，因为应聘者此时的反应、采取的措施都能体现出他的调试功底。如果应聘者能够熟练地设置断点、单步跟踪、查看内存、分析调用栈，就能很快发现问题的根源并最终解决问题，那么面试官将会觉得他的开发经验很丰富。调试能力是在书本上学不到的，只有通过大量的软件开发实践才能积累出调试技巧。当面试官发现一个应聘者的调试功底很扎实的时候，他在写面试报告的时候是不会吝啬赞美之词的。

面试小提示：

在共享桌面远程面试过程中，面试官最关心的是应聘者的编程习惯及调试能力。

1.2.3 现场面试

在通过电话面试和共享桌面远程面试之后，应聘者不久就会收到 E-mail，邀请他去公司参加现场面试（Onsite Interview）。

在去公司参加现场面试之前，应聘者应做好以下几点准备：

- **规划好路线并估算出行时间。**应聘者要事先估算在路上需要花费多长时间，并预留半小时左右的缓冲时间以应对堵车等意外情况。如果面试迟到，那至少印象分会大打折扣。
- **准备好得体的衣服。**IT 公司通常衣着比较随意，应聘者通常没有必要穿着正装，一般舒服干净的衣服都可以。
- **注意面试邀请函里的面试流程。**如果面试有好几轮，时间也很长，那么你在面试过程中可能会觉得疲劳且思维变得迟钝。比如微软对

技术职位通常有 5 轮面试，连续几小时处在高压的面试之中，人难免会变得精疲力竭。因此，应聘者可以带一些提神的饮料或者食品，在两轮面试之间提神醒脑。

- **准备几个问题。**每一轮面试的最后，面试官都会让应聘者问几个问题，应聘者可以提前准备好问题。

现场面试是整个面试流程中的重头戏。由于是坐在面试官的对面，应聘者的一举一动都看在面试官的眼里。面试官通过应聘者的语言和行动考查他的沟通能力、学习能力、编程能力等综合实力。本书接下来的章节将详细讨论各种能力。

1.3 面试的 3 个环节

通常面试官会把每一轮面试分为 3 个环节（如图 1.2 所示）：首先是行为面试，面试官参照简历了解应聘者的过往经验；其次是技术面试，这一环节很有可能会要求应聘者现场写代码；最后一个环节是应聘者问几个自己最感兴趣的问题。下面将详细讨论面试的这 3 个环节。

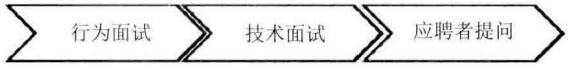

图 1.2　面试的 3 个环节

1.3.1　行为面试环节

面试开始的 5～10 分钟通常是行为面试的时间。在行为面试这个环节里，面试官会注意应聘者的性格特点，深入地了解简历中列举的项目经历。由于这一环节一般不会问技术难题，因此也是一个暖场的过程，应聘者可以利用这几分钟调整自己的情绪，进入面试的状态。

不少面试官会让应聘者做一个简短的自我介绍。由于面试官手中拿着应聘者的简历，而那里有应聘者的详细信息，因此此时的自我介绍不用花很多时间，用 30 秒到 1 分钟的时间介绍自己的主要学习、工作经历即可。

如果面试官对你的某一段经历或者参与的某一个项目很感兴趣，那么他会有针对性地提几个问题详细了解。

1. 应聘者的项目经验

应聘者自我介绍之后，面试官接着会对照应聘者的简历去详细了解他感兴趣的项目。应聘者在准备简历的时候，建议用如图 1.3 所示的 STAR 模型描述自己经历过的每一个项目。

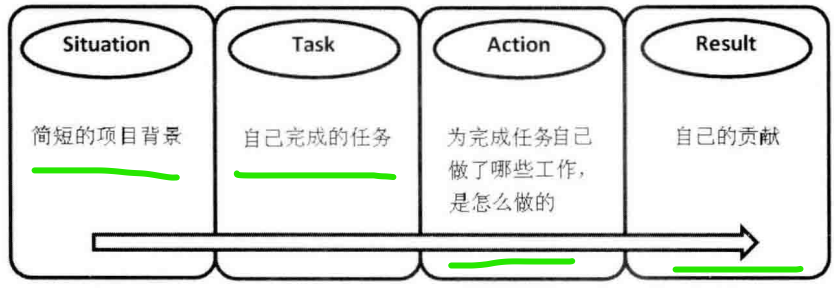

图 1.3　简历中描述项目的 STAR 模型

- **Situation：简短的项目背景。**比如项目的规模，开发的软件的功能、目标用户等。
- **Task：自己完成的任务。**这个要写详细，要让面试官对自己的工作一目了然。在用词上要注意区分“参与”和“负责”：如果只是加入某一个开发团队写了几行代码就用“负责”，那就很危险。面试官看到简历上应聘者“负责”了某个项目，他可能就会问项目的总体框架设计、核心算法、团队合作等问题。这些问题对于只是简单“参与”的人来说，是很难回答的，会让面试官认为你不诚实，印象分会减去很多。
- **Action：为完成任务自己做了哪些工作，是怎么做的。**这里可以详细介绍。做系统设计的，可以介绍系统架构的特点；做软件开发的，可以写基于什么工具在哪个平台下应用了哪些技术；做软件测试的，可以写是手工测试还是自动化测试、是白盒测试还是黑盒测试等。
- **Result：自己的贡献。**这方面的信息可以写得具体些，最好能用数字加以说明。如果是参与功能开发，则可以说按时完成了多少功能；

如果做优化，则可以说性能提高的百分比是多少；如果是维护，则可以说修改了多少个 Bug。

举个例子，笔者用下面一段话介绍自己在微软 Winforms 项目组的经历：

Winforms 是微软.NET 中的一个成熟的 UI 平台（**Situation**）。本人的工作是在添加少量新功能之外主要负责维护已有的功能（**Task**）。新的功能主要是让 Winforms 的控件风格和 Vista、Windows 7 的风格保持一致。在维护方面，对于较难的问题，我用 WinDbg 等工具进行调试（**Action**）。在过去两年中，我共修改了超过 200 个 Bug（**Result**）。

如果在应聘者的简历中上述 4 类信息还不够清晰，则面试官可能会追问相关的问题。除此之外，面试官针对项目经验最常问的问题包括如下几个类型：

- 你在该项目中碰到的最大问题是什么，你是怎么解决的？
- 从这个项目中你学到了什么？
- 什么时候会和其他团队成员（包括开发人员、测试人员、设计人员、项目经理等）有什么样的冲突，你们是怎么解决冲突的？

应聘者在准备简历的时候，针对每一个项目经历都应提前做好相应的准备。只有准备充分，应聘者在行为面试环节才可以表现得游刃有余。

面试小提示：

在介绍项目经验（包括在简历上介绍和面试时口头介绍）时，应聘者不必详述项目的背景，而要突出介绍自己完成的工作及取得的成绩。

2. 应聘者掌握的技能

除应聘者参与过的项目之外，面试官对应聘者掌握的技能也很感兴趣，他有可能针对简历上提到的技能提出问题。和描述项目时要注意“参与”和“负责”一样，描述技能掌握程度时也要注意“了解”、“熟悉”和“精通”的区别。

“了解”指对某项技术只是上过课或者看过书，但没有做过实际的项目。通常不建议在简历中列出只是肤浅地了解一点的技能，除非这项技术应聘的职位的确需要。比如某学生读本科的时候学过《计算机图形学》这门课

程，但一直没有开发过与图形绘制相关的项目，那就只能算是了解。如果他去应聘 Autodesk 公司，那么他可以在简历上提一下他了解图形学。Autodesk 是一家开发三维设计软件的公司，有很多职位或多或少会与图形学有关系，那么了解图形学的总比完全不了解的要适合一些。但如果他是去应聘 Oracle，那就没有必要提这一点了，因为开发数据库系统的 Oracle 公司大部分职位与图形学没有什么关系。

简历中我们描述技能的掌握程度大部分应该是“熟悉”。如果我们在实际项目中使用某项技术已经有较长的时间，通过查阅相关的文档可以独立解决大部分问题，那么我们就熟悉它了。对应届毕业生而言，他毕业设计所用到的技能可以用“熟悉”；对已经工作过的，在项目开发过程中所用到的技能，也可以用“熟悉”。

如果我们对一项技术使用得得心应手，在项目开发过程中，当同学或同事向我们请教这个领域的问题时，我们都有信心也有能力解决，这个时候我们就可以说自己精通了这项技术。应聘者不要试图在简历中把自己修饰成“高人”而轻易使用“精通”，除非自己能够很轻松地回答这个领域里的绝大多数问题，否则就会适得其反。通常如果应聘者在简历中说自己精通某项技术，面试官就会对他有很高的期望值，因此会挑一些比较难的问题来问。这也是越装高手就越容易露馅的原因。曾经碰到一个在简历中说自己精通 C++的应聘者，连成员变量的初始化顺序这样的问题都被问得一头雾水，那最终的结果也就可想而知了。

3. 回答“为什么跳槽”

在面试已经有工作经验的应聘者的时候，面试官总喜欢问为什么打算跳槽。每个人都有自己的跳槽动机和原因，因此面试官也不会期待一个标准答案。面试官只是想通过这个问题来了解应聘者的性格，因此应聘者可以大胆地根据自己的真实想法来回答这个问题。但是，应聘者也不要想说什么就说什么，以免给面试官留下负面的印象。

在回答这个问题时不要抱怨，也不要流露出负面的情绪。负面的情绪通常是能够传染的，当应聘者总是在抱怨的时候，面试官就会担心如果把他招进来，那么他将成为团队负面情绪的传染源，从而影响整个团队的士气。应聘者应尽量避免以下 4 个原因：

- **老板太苛刻**。如果面试官就是当前招聘的职位的老板，那么当他听

到应聘者抱怨现在的老板苛刻时，他肯定会想要是把这个人招进来，接下来他就会抱怨我也苛刻了。

- **同事太难相处**。如果应聘者说他周围有很多很难相处的同事，则面试官很有可能会觉得这个人本身就很难相处。
- **加班太频繁**。对于大部分 IT 企业来说，加班是家常便饭。如果正在面试的公司也需要经常加班，那等于应聘者说他不想进这家公司。
- **工资太低**。现在的工资太低的确是大部分人跳槽的真实原因，但不建议在面试的时候对面试官抱怨。面试的目的是拿到 Offer，我们要尽量给面试官留下好印象。现在假设你是面试官，有两个人来面试：一个人一开口就说现在工资太低了，希望新工作能加多少多少工资；另一个说我只管努力干活，工资公司看着给，相信公司不会亏待勤奋的员工。你更喜欢哪个？这里不是说工资不重要，但我们要清楚面试不是谈工资的时候。等完成技术面试之后谈 Offer 的时候，再和 HR 谈工资也不迟。通过面试之后我们就掌握主动权了，想怎么谈就怎么谈，如果工资真的开高了，那么 HR 会和你很客气地商量。

笔者在面试的时候通常给出的答案是：现在的工作做了一段时间，已经没有太多的激情了，因此希望寻找一份更有挑战的工作。然后具体论述为什么有些厌倦现在的职位，以及面试的职位我为什么会有兴趣。笔者自己跳过几次槽，第一次从 Autodesk 跳槽到微软，第二次从微软跳槽到思科，后来又从思科回到了微软。从面试的结果来看，这样的回答都让面试官很满意，最终也都拿到了 Offer。

当时在微软面试被问到为什么要跳槽时，笔者的回答是：我在 Autodesk 开发的软件 Civil 3D 是一款面向土木行业的设计软件。如果我想在现在的职位上得到提升，就必须加强土木行业的学习，可我对诸如计算土方量、道路设计等没有太多兴趣，因此出来寻找机会。

在微软工作两年半之后去思科面试的时候，笔者的回答是：我在微软的主要工作是开发和维护.NET 的 UI 平台 Winforms。由于 Winforms 已经非常成熟，不需要添加多少新功能，因此我的大部分工作是维护和修改 Bug。两年下来，调试的能力得到了很大的提高，但长期如此，自己的软件开发和设计能力将不能得到提高，因此想出来寻找可以设计和开发系统的职位。

同时，我在过去几年里的工作都是开发桌面软件，对网络了解甚少，因此希望下一个工作能与网络相关。众所周知，思科是一家网络公司，这里的软件和系统或多或少都离不开网络，因此我对思科的职位很感兴趣。

1.3.2 技术面试环节

面试官在通过简历及行为面试大致了解应聘者的背景之后，接下来就要开始技术面试了。一轮一小时的面试，通常技术面试会占据 40～50 分钟。这是面试的重头戏，对面试的结果起决定性作用。虽然不同公司不同面试官的背景、性格各不相同，但总体来说他们都会关注应聘者的 5 种素质：扎实的基础知识、能写高质量的代码、分析问题时思路清晰、能优化时间效率和空间效率，以及学习沟通等各方面的能力（如图 1.4 所示）。

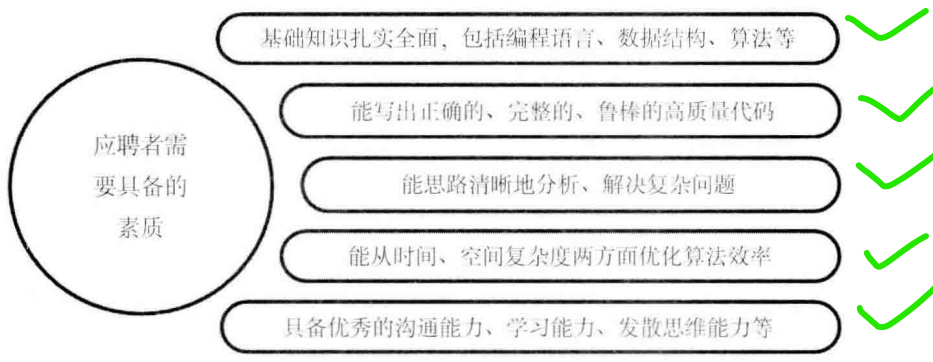

图 1.4 应聘者需要具备的素质

应聘者在面试之前需要做足准备，对编程语言、数据结构和算法等基础知识有全面的了解。面试的时候如果遇到简单的问题，则应聘者一定要注重细节，写出完整、鲁棒的代码。如果遇到复杂的问题，则应聘者可以通过画图、举具体例子分析和分解复杂问题等方法先厘清思路再动手编程。除此之外，应聘者还应该不断优化时间效率和空间效率，力求找到最优的解法。在面试过程中，应聘者还应该主动提问，以弄清楚题目的要求，表现自己的沟通能力。当面试官前后问的两个问题有相关性的时候，尽量把解决前面问题的思路迁移到后面的问题中去，展示自己良好的学习能力。如果能做到这几点，那么通过面试获得心仪的职位将是水到渠成的事情。

1. 扎实的基础知识

扎实的基本功是成为优秀程序员的前提条件，因此面试官首要关注的应聘者素质就是是否具备扎实的基础知识。通常基本功在编程面试环节体现在 3 个方面：编程语言、数据结构和算法。

首先，每个程序员至少要掌握一两门编程语言。面试官从应聘者在面试过程中写的代码及跟进的提问中能看出其编程语言掌握的熟练程度。以很多公司面试要求的 C++举例。如果写的函数需要传入一个指针，则面试官可能会问是否需要为该指针加上 const、把 const 加在指针不同的位置是否有区别；如果写的函数需要传入的参数是一个复杂类型的实例，则面试官可能会问传入值参数和传入引用参数有什么区别、什么时候需要为传入的引用参数加上 const。

其次，数据结构通常是编程面试过程中考查的重点。在参加面试之前，应聘者需要熟练掌握链表、树、栈、队列和哈希表等数据结构，以及它们的操作。如果我们留意各大公司的面试题，就会发现与链表和二叉树相关的问题是很多面试官喜欢问的问题。这方面的问题看似比较简单，但要真正掌握也不容易，特别适合在这么短的面试时间内检验应聘者的基本功。如果应聘者事先对链表的插入和删除节点了如指掌，对二叉树的各种遍历方法的循环和递归写法都烂熟于胸，那么真正到了面试的时候也就游刃有余了。

最后，大部分公司都会注重考查查找、排序等算法。应聘者可以在了解各种查找和排序算法的基础上，重点掌握二分查找、归并排序和快速排序，因为很多面试题都只是这些算法的变体而已。比如面试题 11“旋转数组的最小数字”和面试题 53“在排序数组中查找数字”的本质是考查二分查找，而面试题 51“数组中的逆序对”实际上是考查归并排序。少数对算法很重视的公司如谷歌或者百度，还会要求应聘者熟练掌握动态规划和贪婪算法。如果应聘者对动态规划算法很熟悉，那么他就能很轻松地解决面试题 14“剪绳子”。

在本书的第 2 章“面试需要的基础知识”中，我们将详细介绍应聘者需要熟练掌握的基础知识。

2. 高质量的代码

只有注重质量的程序员，才能写出鲁棒、稳定的大型软件。在面试过程中，面试官总会格外关注边界条件、特殊输入等看似细枝末节但实则至

关重要的地方，以考查应聘者是否注重代码质量。很多时候，面试官发现应聘者写出来的代码只能完成最基本的功能，一旦输入特殊的边界条件参数，就会错误百出甚至程序崩溃。

总有些应聘者很困惑：面试的时候觉得题目很简单，感觉自己都做出来了，可最后为什么被拒了呢？面试被拒有很多种可能，比如面试官认为你性格不适合、态度不够诚恳等。但在技术面试过程中，这些都不是最重要的。技术面试的面试官一般都是程序员，程序员通常没有那么多想法。他们只认一个理：题目做对、做完整了，就让你通过面试；否则失败。所以遇到简单题目却被拒的情况，应聘者应认真反思在思路或者代码中存在哪些漏洞。

以微软面试开发工程师时最常用的一个问题为例：把一个字符串转换成整数。这个题目很简单，很多人都能在 3 分钟之内写出如下不到 10 行的代码：

```
int StrToInt(char* string)
{
    int number = 0;
    while(*string != 0)
    {
        number = number * 10 + *string - '0';
        ++string;
    }

    return number;
}
```

看了上面的代码，你是不是觉得微软面试很容易？如果你真的这么想，那你可能又要被拒了。

通常越是简单的问题，面试官的期望值就会越高。如果题目很简单，面试官就会期待应聘者能够很完整地解决问题，除完成基本功能之外，还要考虑到边界条件、错误处理等各个方面。比如这道题，面试官不仅仅期待你能完成把字符串转换成整数这个最起码的要求，而且希望你能考虑到各种特殊的输入。面试官至少会期待应聘者能够在不需要提示的情况下，考虑到输入的字符串中有非数字字符和正负号，要考虑到最大的正整数和最小的负整数以及溢出。同时面试官还期待应聘者能够考虑到当输入的字符串不能转换成整数时，应该如何做错误处理。当把这个问题的方方面面都考虑到的时候，我们就不会再认为这道题简单了。

除问题考虑不全面之外，还有一个面试官不能容忍的错误就是程序不

够鲁棒。以前面的那段代码为例，只要输入一个空指针，程序立即崩溃。这样的代码如果加入软件当中，那么将是灾难。因此，当面试官看到代码中对空指针没有判断并加以特殊处理的时候，通常他连往下看的兴趣都没有。

当然，不是所有与鲁棒性相关的问题都和前面的代码那样明显。再举一个很多人被面试过的问题：求链表中的倒数第 k 个节点。有不少人在面试之前在网上看过这个题目，因此知道思路是用两个指针，第一个指针先走 $k-1$ 步，然后两个指针一起走。当第一个指针走到尾节点的时候，第二个指针指向的就是倒数第 k 个节点。于是他大笔一挥，写下了下面的代码：

```
ListNode* FindKthToTail(ListNode* pListHead, unsigned int k)
{
    if(pListHead == nullptr)
        return nullptr;

    ListNode *pAhead = pListHead;
    ListNode *pBehind = nullptr;

    for(unsigned int i = 0; i < k - 1; ++ i)
    {
        pAhead = pAhead->m_pNext;
    }

    pBehind = pListHead;

    while(pAhead->m_pNext != nullptr)
    {
        pAhead = pAhead->m_pNext;
        pBehind = pBehind->m_pNext;
    }

    return pBehind;
}
```

写完之后，应聘者看到自己已经判断了输入的指针是不是空指针并进行了特殊处理，于是以为这次面试必定能顺利通过。可是他没有想到的是这段代码中仍然有很严重的问题：当链表中的节点总数小于 k 的时候，程序还是会崩溃；另外，当输入的 k 为 0 时，同样也会引起程序崩溃。因此，几天之后他收到的仍然不是 Offer 而是拒信。

要想很好地解决前面的问题，最好的办法是在动手写代码之前想好测试用例。只有把各种可能的输入事先都想好了，才能在写代码的时候把各种情况都进行相应的处理。写完代码之后，也不要立刻给面试官检查，而是先在心里默默地运行。当输入之前想好的所有测试用例都能得到合理的

输出时，再把代码交给面试官。做到了这一步，通过面试拿到 Offer 就是顺理成章的事情了。

在本书的第 3 章“高质量的代码”中，我们将详细讨论提高代码质量的方法。

 **面试小提示：**

面试官除了希望应聘者的代码能够完成基本的功能，还会关注应聘者是否考虑了边界条件、特殊输入（如 nullptr 指针、空字符串等）及错误处理。

3. 清晰的思路

只有思路清晰，应聘者才有可能在面试过程中解决复杂的问题。有时候面试官会有意出一些比较复杂的问题，以考查应聘者能否在短时间内形成清晰的思路并解决问题。对于确实很复杂的问题，面试官甚至不期待应聘者能在面试不到一小时的时间里给出完整的答案，他更看重的可能还是应聘者是否有清晰的思路。面试官通常不喜欢应聘者在没有形成清晰思路之前就草率地开始写代码，这样写出来的代码容易逻辑混乱、错误百出。

应聘者可以用几个简单的方法帮助自己形成清晰的思路。首先，举几个简单的具体例子让自己理解问题。当我们一眼看不出问题中隐藏的规律的时候，可以试着用一两个具体的例子模拟操作的过程，这样说不定就能通过具体的例子找到抽象的规律。其次，可以试着用图形表示抽象的数据结构。像分析与链表、二叉树相关的题目，我们都可以画出它们的结构来简化题目。最后，可以试着把复杂的问题分解成若干简单的子问题，再一一解决。很多基于递归的思路，包括分治法和动态规划，都是把复杂的问题分解成一个或者多个简单的子问题。

比如把二叉搜索树转换成排序的双向链表这个问题就很复杂。遇到这个问题，我们不妨先画出一两棵具体的二叉搜索树，直观地感受二叉搜索树和排序的双向链表有哪些联系。如果一下子找不出转换的规律，则可以把整棵二叉树看成 3 个部分：根节点、左子树和右子树。当我们递归地把转换左右子树这两个子问题解决之后，再把转换左右子树得到的链表和根节点链接起来，整个问题也就解决了（详见面试题 36“二叉搜索树与双向链表”）。

在本书的第 4 章“解决面试题的思路”中，我们将详细讨论遇到复杂问题时如何采用画图、举例和分解问题等方法帮助我们解决问题。

面试小提示：

如果在面试的时候遇到难题，我们有 3 种办法分析、解决复杂的问题：画图能使抽象问题形象化，举例使抽象问题具体化，分解使复杂问题简单化。

4. 优化效率的能力

优秀的程序员对时间和内存的消耗锱铢必较，他们很有激情地不断优化自己的代码。当面试官出的题目有多种解法的时候，通常他会期待应聘者最终能够找到最优解。当面试官提示还有更好的解法的时候，应聘者不能放弃思考，而应该努力寻找在时间消耗或者空间消耗上可以优化的地方。

要想优化时间或者空间效率，首先要知道如何分析效率。即使同一种算法，用不同方法实现的效率可能也会大不相同，我们要能够分析出算法及其代码实现的效率。例如，求斐波那契数列，很多人喜欢用递归公式 $f(n)=f(n-1)+f(n-2)$ 求解。如果分析它的递归调用树，我们就会发现有大量的计算是重复的，时间复杂度以 n 的指数增加。但如果我们先求 $f(1)$ 和 $f(2)$，再根据 $f(1)$ 和 $f(2)$ 求出 $f(3)$，接下来根据 $f(2)$ 和 $f(3)$ 求出 $f(4)$，并以此类推用一个循环求出 $f(n)$，这种计算方法的时间效率就只有 $O(n)$，比前面递归的方法要好得多。

要想优化代码的效率，我们还要熟知各种数据结构的优缺点，并能选择合适的数据结构解决问题。我们在数组中根据下标可以用 $O(1)$ 时间完成查找。数组的这个特征可以实现用简单的哈希表解决很多问题，如面试题 50“第一个只出现一次的字符”。为了解决面试题 40“最小的 k 个数”，我们需要一个数据容器来存储 k 个数字。在这个数据容器中，我们希望能够快速地找到最大值，并且能快速地替换其中的数字。经过权衡，我们发现二叉树如最大堆或者红黑树都是实现这个数据容器的不错选择。

要想优化代码的效率，我们也要熟练掌握常用的算法。面试中最常用的算法是查找和排序。如果从头到尾顺序扫描一个数组，那么我们需要 $O(n)$

时间才能完成查找操作。但如果数组是排序的，应用二分查找算法就能把时间复杂度降低到 $O(\log n)$（详见面试题 11“旋转数组的最小数字”和面试题 53“在排序数组中查找数字”）。排序算法除了能够给数组排序，还能用来解决其他问题。比如快速排序算法中的 Partition 函数能够用来在 n 个数里查找第 k 大的数字，从而解决面试题 39“数组中出现次数超过一半的数字”和面试题 40“最小的 k 个数”。归并排序算法能够实现在 $O(n\log n)$时间统计 n 个数字中的逆序对数目（详见面试题 51“数组中的逆序对”）。

在本书的第 5 章“优化时间和空间效率”中，我们将详细讨论如何从时间效率和空间效率两方面进行优化。

5. 优秀的综合能力

在面试过程中，应聘者除了展示自己的编程能力和技术功底，还需要展示自己的软技能（Soft Skills），诸如自己的沟通能力和学习能力。随着软件系统的规模越来越大，软件开发已经告别了单打独斗的年代，程序员与他人的沟通变得越来越重要。在面试过程中，面试官会观察应聘者在介绍项目经验或者算法思路时是否观点明确、逻辑清晰，并以此判断其沟通能力的强弱。另外，面试官也会从应聘者说话的神态和语气来判断他是否有团队合作的意识。通常面试官不会喜欢高傲或者轻视合作者的人。

IT 行业知识更新很快，因此程序员只有具备很好的学习能力才能跟上知识更替的步伐。通常面试官有两种办法考查应聘者的学习能力。第一种方法是询问应聘者最近在看什么书、从中学到了哪些新技术。面试官可以用这个问题了解应聘者的学习愿望和学习能力。第二种方法是抛出一个新概念，接下来他会观察应聘者能不能在较短的时间内理解这个新概念并解决相关的问题。比如面试官要求应聘者计算第 1500 个丑数。很多人都没有听说过丑数这个概念。这时候面试官就会观察应聘者面对丑数这个新概念时，能不能经过提问、思考、再提问的过程，最终找出丑数的规律，从而找到解决方案（详见面试题 49“丑数”）。

知识迁移能力是一种特殊的学习能力。如果我们能够把已经掌握的知识迁移到其他领域，那么学习新技术或者解决新问题就会变得容易。面试官经常会先问一个简单的问题，再问一个很复杂但和前面的简单问题相关的问题。这时候面试官期待应聘者能够从简单问题中得到启示，从而找到解决复杂问题的窍门。比如面试官先要求应聘者写一个函数求斐波那契数

列，再问一个青蛙跳台阶的问题：一只青蛙一次可以跳上 1 级台阶，也可以跳上 2 级台阶。请问这只青蛙跳上 n 级台阶总共有多少种跳法。应聘者如果具有较强的知识迁移能力，就能分析出青蛙跳台阶问题实质上只是斐波那契数列的一个应用（详见面试题 10“斐波那契数列”）。

还有不少面试官喜欢考查应聘者的抽象建模能力和发散思维能力。面试官从日常生活中提炼出问题，比如面试题 61“扑克牌的顺子”，考查应聘者能不能把问题抽象出来用合理的数据结构表示，并找到其中的规律解决这个问题。面试官也可以限制应聘者不得使用常规方法，这要求应聘者具备创新精神，能够打开思路从多角度去分析、解决问题。比如在面试题 65“不用加减乘除做加法”中，面试官期待应聘者能够打开思路，用位运算实现整数的加法。

我们将在本书的第 6 章“面试中的各项能力”中用具体的面试题详细讨论上述能力在面试中的重要作用。

1.3.3　应聘者提问环节

在结束面试前的 5～10 分钟，面试官会给应聘者机会问几个问题，应聘者的问题质量对面试的结果也有一定的影响。有些人的沟通能力很强，马上就能想到有意思的问题。但对于大多数人而言，在经受了面试官将近一小时的拷问之后可能已经精疲力竭，再迅速想出几个问题难度很大。因此建议应聘者不妨在面试之前做些功课，为每一轮面试准备 2～3 个问题，这样到提问环节的时候就游刃有余了。

面试官让应聘者问几个问题，主要是想了解他最关心的问题有哪些，因此应聘者至少要问一两个问题，否则面试官就会觉得你对我们公司、职位等都不感兴趣，那你来面试做什么？但也不是什么问题都可以在这个时候问。如果问题问得比较合适，则对应聘者来说是个加分的好机会；但如果问的问题不太合适，则面试官对他的印象就会大打折扣。

有些问题是不适合在技术面试这个环节里问的。首先，不要问和自己的职位没有关系的问题，比如问“公司未来五年的发展战略是什么”。如果应聘的职位是 CTO，而面试官是 CEO，那么这倒是个合适的问题。如果应聘的只是在一线开发的职位，那这个问题离我们就太远了，与我们的切身利益没有多少关系。另外，坐在对面的面试官很有可能也只是一个在一线开发的程序员，他该怎么回答这个关系公司发展战略的问题呢？

其次，不要问薪水。技术面试不是谈薪水的时候，要谈工资要等通过面试之后和 HR 谈。而且这会让面试官觉得你最关心的问题就是薪水，给面试官留下的印象也不好。

再次，不要立即打听面试结果，比如问“您觉得我能拿到 Offer 吗”之类的问题。现在大部分公司的面试都有好几轮，最终决定应聘者能不能通过面试，是要把所有面试官的评价综合起来的。问这个问题相当于白问，因为问了面试官也不可能告诉应聘者结果，还会让面试官觉得你没有自我评估的能力。

最后，推荐问的问题是与应聘的职位或者项目相关的问题。如果这种类型的问题问得很到位，那么面试官就会觉得你对应聘的职位很有兴趣。不过要问好这种类型的问题也不容易，因为首先要对应聘的职位或者项目的背景有一定的了解。我们可以从两方面去了解相关的信息：一是面试前做足功课，到网上去搜集一些相关的信息，做到对公司成立时间、主要业务、职位要求等都了然于胸；二是面试过程中留心面试官说过的话。有不少面试官在面试之前会简单介绍与招聘职位相关的项目，其中会包含从其他渠道无法得到的信息，比如项目进展情况等。应聘者可以从中找出一两个点，然后向面试官提问。

下面的例子是笔者去思科面试时问的几个问题。一个面试官介绍项目时说这次招聘是项目组第一次在中国招人，目前这个项目所有人员都在美国总部。这轮面试笔者最后问的问题是：这个项目所有的老员工都在美国，那怎么对中国这一批新员工进行培训？中国的新员工有没有机会去美国总部学习？最后一轮面试是老板面试，她介绍说正在招聘的项目组负责开发一个测试系统，思科用它来测试供应商生产的网络设备。这轮面试笔者问的几个问题是：这个组是做系统测试的，那这个组的人员是不是也要参与网络设备的测试？是不是需要学习与硬件测试相关的知识？因为我们测试的对象是网络设备，那么这个职位对网络硬件的掌握程度有没有要求？

1.4 本章小结

本章重点介绍了面试的流程。通常面试是从电话面试开始的。接下来可能有一两轮共享桌面远程面试，面试官通过桌面共享软件远程考查应聘

者的编程和调试能力。如果应聘者的表现足够优秀，那么公司将邀请他到公司去接受现场面试。

一般每一轮面试都有 3 个环节。首先是行为面试环节，面试官在这一环节中对照简历询问应聘者的项目经验和掌握的技能。接下来就是技术面试环节，这是面试的重头戏。在这一环节里，面试官除了关注应聘者的编程能力和技术功底，还会注意考查他的沟通能力和学习能力。在面试的最后，通常面试官会留几分钟时间让应聘者问几个他感兴趣的问题。

1.3.2 节是全书的大纲，介绍了面试官关注应聘者 5 个方面的素质：基础知识是否扎实、能否写出高质量的代码、思路是否清晰、是否有优化效率的能力，以及包括学习能力、沟通能力在内的综合素质是否优秀。在接下来的第 2～6 章中，我们将一一深入探讨这 5 个方面的素质。

done!

done!

done!

done!