

< / > 在这里 / 有面试笔试常见技巧的提炼与总结;

< / > 在这里 / 有面试笔试高频算法知识点的整理与剖析;

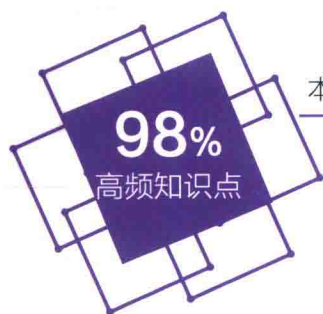
< / > 在这里 / 有面试笔试历年算法真题的解答与拓展。

PROGRAMMER

> Algorithm interview

Kotlin 程序员面试 算法宝典

猿媛之家 / 组编 孙伟 楚秦 等 / 编著



本书覆盖了近**3年**程序员Kotlin面试笔试中超过**98%**的高频算法知识点

当你细细品读完本书后，各类企业的offer将任由你挑选

一书在手 / 工作不愁 > .

Kotlin 程序员面试算法宝典

猿媛之家 组 编

孙 伟 楚 秦 等编著

机械工业出版社

本书是一本讲解 Kotlin 程序员面试笔试算法的书籍。在写法上,除了讲解如何解答算法问题以外,还引入了例子辅以说明,以便读者能够更加容易地理解。

本书将程序员面试笔试过程中的各类算法类真题一网打尽。在题目的广度上,通过各种渠道,搜集了近3年来几乎所有IT企业面试笔试算法高频题目,所选择题目均为企业招聘使用题目;在题目的深度上,本书由浅入深、庖丁解牛式地分析每一个题目,并提炼归纳,同时,引入例子与源代码、时间复杂度与空间复杂度的分析,这些内容其他同类书籍所没有的。本书结构合理,条理清晰,根据真题所属知识点进行分类,对于读者进行学习检索意义重大。

本书是一本计算机相关专业毕业生面试笔试的求职用书,也可以作为本科生、研究生学习数据结构与算法的辅导书籍,同时也适合期望在计算机软硬件行业大显身手的计算机爱好者阅读。

图书在版编目(CIP)数据

Kotlin 程序员面试算法宝典/猿媛之家组编;孙伟等编著. —北京:机械工业出版社, 2018.9

ISBN 978-7-111-61212-4

I. ①K… II. ①猿… ②孙… III. ①JAVA 语言—程序设计
IV. ①TP312.8

中国版本图书馆 CIP 数据核字 (2018) 第 243673 号

机械工业出版社(北京市百万庄大街 22 号 邮政编码 100037)

策划编辑:时 静 责任编辑:时 静

责任校对:张艳霞 责任印制:张 博

三河市宏达印刷有限公司印刷

2019 年 1 月第 1 版·第 1 次印刷

184mm×260mm·19.5 印张·477 千字

0001—3000 册

标准书号:ISBN 978-7-111-61212-4

定价:69.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换
电话服务

服务咨询热线:(010) 88361066

读者购书热线:(010) 68326294

(010) 88379203

封面无防伪标均为盗版

网络服务

机工官网:www.cmpbook.com

机工官博:weibo.com/cmp1952

教育服务网:www.cmpedu.com

金书网:www.golden-book.com



前言

Kotlin 语言在 2016 年发布了第一个正式版，而 2017 年它就成为了 Google 官方支持的 Android 开发语言，仅仅经过一年，Kotlin 就显示出了其强大的生命力，2018 年一定会是 Kotlin 爆发的一年。

为什么 Kotlin 能够成为如此热门的编程语言呢？其实，Kotlin 从诞生之初就已广受好评，特别是受到 Java 开发人员的好评。Java 语言是一个较陈旧的语言，而且更新缓慢，比起 Ruby、Python 这些开发语言，Java 语言像落后了两个年代，随着 Kotlin 的出现，它把 JVM 编程提升到了一个更高的水平，开发者可以放心地使用 Lambda、高阶函数及智能转换等特性，而不需要在项目中做太多改变，只需要下个代码文件使用 Kotlin 就可以了。

也许有很多人认为现在会 Kotlin 语言的人还比较少，如果学会了 Kotlin，是否会很容易找到一份好的工作？我的回答是：不会。掌握 Kotlin 对找工作基本上没有决定性的帮助，这是因为 Java 开发人员转向学习 Kotlin 太简单了，只需两周左右的时间就可以上手开发，其学习难度可能都比不上一个复杂的开发框架。但是否可以说看完本书对找工作没有丝毫的帮助呢？这就大错特错了。编程语言只是一种手段，一个工具，无论是 Java 语言，还是 Kotlin 语言，无一例外，都是如此，只有算法才是核心，而算法对于程序员求职是永远有用的，特别是现在市场上人才过剩，企业在招聘的时候，对求职者的算法水平自然而然也是要求越来越高，毕竟工作年限的长短、做过项目的多少都不足以评定一个人的水平，而算法好的人，通常编程水平都不会差。用 Kotlin 写算法比起 Java 优势就很明显了：可以在网页上运行代码，可以使用交互式 shell，或者“祭出重器”IntelliJ IDEA，自动补全、优化提示、拼写检查及调试等功能，都是为 Kotlin 量身定制的。面对面试官，当面试者手写代码的时候，Kotlin 简洁的语法，一方面可以让其只需要关注问题本身，另一方面没有了 Java 语言的冗长代码，答案看起来会更加整洁。

由于 Kotlin 是一种非常优秀的开发语言，所以值得读者学习，而本书中的算法题都是精挑细选的高频面试笔试真题，更值得读者学习。编程是一个解决问题的过程，书中的算法也许不能直接解决问题，但是一定能帮助读者提升解决问题的能力。对于个人成长来说，如果想要找到一份更好的工作，算法是一块敲门砖，也许它就是面试官评定面试者能力高低的标准。如果读者能学完书中的所有算法，那么一定会有一种豁然开朗的感觉，感觉自己提升了一个层次，因为只要读者掌握了这些算法，就已经击败了 98% 的人了。

本书部分思想来源于网络上的无名英雄，无法追踪到最原始的出处，在此对这些幕后英

雄致以最崇高的敬意。没有学不好的学生，只有教不好的老师，我们希望无论是什么层次的学生，都能毫无障碍地看懂书中所讲内容。如果读者存在求职困惑或是对本书中的内容存在异议，都可以通过 yuancoder@foxmail.com 联系编者。

祝所有求职者都能找到一份满意的工作。

编 者
2018.3.18

目 录

前言

面试笔试经验技巧篇

经验技巧 1	如何巧妙地回答面试官的问题	2
经验技巧 2	如何回答技术性问题	3
经验技巧 3	如何回答非技术性问题	4
经验技巧 4	如何回答快速估算类问题	5
经验技巧 5	如何回答算法设计问题	6
经验技巧 6	如何回答系统设计问题	8
经验技巧 7	如何解决求职中的时间冲突问题	11
经验技巧 8	如果面试问题曾经遇见过，是否要告知面试官	12
经验技巧 9	被企业拒绝后是否可以再申请	12
经验技巧 10	如何应对自己不会回答的问题	13
经验技巧 11	如何应对面试官的“激将法”语言	13
经验技巧 12	如何处理与面试官持不同观点的问题	14
经验技巧 13	什么是职场暗语	15

面试笔试真题解析篇

第 1 章	链表	20
1.1	如何实现链表的逆序	20
1.2	如何从无序链表中移除重复项	25
1.3	如何计算两个单链表所代表的数之和	28
1.4	如何对链表进行重新排序	32
1.5	如何找出单链表中的倒数第 k 个元素	35
1.6	如何检测一个较大的单链表是否有环	38
1.7	如何把链表相邻元素翻转	41
1.8	如何把链表以 K 个结点为一组进行翻转	43
1.9	如何合并两个有序链表	46

1.10	如何在只给定单链表中某个结点的指针的情况下删除该结点	48
1.11	如何判断两个单链表（无环）是否交叉	50
1.12	如何展开链接列表	54
第 2 章	栈、队列与哈希	57
2.1	如何实现栈	57
2.2	如何实现队列	61
2.3	如何翻转栈的所有元素	66
2.4	如何根据入栈序列判断可能的出栈序列	69
2.5	如何用 $O(1)$ 的时间复杂度求栈中最小元素	71
2.6	如何用两个栈模拟队列操作	73
2.7	如何设计一个排序系统	74
2.8	如何实现 LRU 缓存方案	76
2.9	如何从给定的车票中找出旅程	77
2.10	如何从数组中找出满足 $a+b=c+d$ 的两个数对	79
第 3 章	二叉树	81
3.1	二叉树基础知识	81
3.2	如何把一个有序整数数组放到二叉树中	82
3.3	如何从顶部开始逐层打印二叉树结点数据	84
3.4	如何求一棵二叉树的最大子树和	86
3.5	如何判断两棵二叉树是否相等	88
3.6	如何把二叉树转换为双向链表	90
3.7	如何判断一个数组是否是二元查找树后序遍历的序列	92
3.8	如何找出排序二叉树上任意两个结点的最近共同父结点	93
3.9	如何复制二叉树	100
3.10	如何在二叉树中找出与输入整数相等的所有路径	101
3.11	如何对二叉树进行镜像反转	103
3.12	如何在二叉排序树中找出第一个大于中间值的结点	105
3.13	如何在二叉树中找出路径最大的和	108
3.14	如何实现反向 DNS 查找缓存	109
第 4 章	数组	113
4.1	如何找出数组中唯一的重复元素	113
4.2	如何查找数组中元素的最大值和最小值	119
4.3	如何找出旋转数组中的最小元素	122
4.4	如何找出数组中丢失的数	125
4.5	如何找出数组中出现奇数次的数	127
4.6	如何找出数组中第 k 小的数	130
4.7	如何求数组中两个元素的最小距离	133

4.8	如何求解最小三元组距离	135
4.9	如何求数组中绝对值最小的数	140
4.10	如何求数组连续最大和	143
4.11	如何找出数组中出现 1 次的数	147
4.12	如何对数组旋转	149
4.13	如何在不排序的情况下求数组中的中位数	151
4.14	如何求集合的所有子集	152
4.15	如何对数组进行循环移位	155
4.16	如何在有规律的二维数组中进行高效的数据查找	158
4.17	如何寻找最多的覆盖点	160
4.18	如何判断请求能否在给定的存储条件下完成	161
4.19	如何按要求构造新的数组	163
4.20	如何获取最好的矩阵链相乘方法	165
4.21	如何求解迷宫问题	167
4.22	如何从三个有序数组中找出它们的公共元素	169
4.23	如何求两个有序集合的交集	171
4.24	如何对有大量重复数字的数组排序	174
4.25	如何对任务进行调度	178
4.26	如何对磁盘分区	179
第 5 章	字符串	181
5.1	如何求一个字符串的所有排列	181
5.2	如何求两个字符串的最长公共子串	187
5.3	如何对字符串进行反转	190
5.4	如何判断两个字符串是否为换位字符串	193
5.5	如何判断两个字符串的包含关系	195
5.6	如何对由大小写字母组成的字符数组排序	197
5.7	如何消除字符串的内嵌括号	198
5.8	如何判断字符串是否是整数	200
5.9	如何实现字符串的匹配	203
5.10	如何求字符串里的最长回文子串	206
5.11	如何按照给定的字母序列对字符数组排序	213
5.12	如何判断一个字符串是否包含重复字符	215
5.13	如何找到由其他单词组成的最长单词	217
5.14	如何统计字符串中连续的重复字符个数	219
5.15	如何求最长递增子序列的长度	220
5.16	求一个串中出现的第一个最长重复子串	222
5.17	如何求解字符串中字典序最大的子序列	223
5.18	如何判断一个字符串是否由另外一个字符串旋转得到	226

5.19	如何求字符串的编辑距离	227
5.20	如何在二维数组中寻找最短路线	230
5.21	如何截取包含中文的字符串	232
5.22	如何求相对路径	234
5.23	如何查找到达目标词的最短链长度	235
第 6 章	基本数字运算	238
6.1	如何判断一个自然数是否是某个数的平方	238
6.2	如何判断一个数是否为 2 的 n 次方	240
6.3	如何不使用除法操作符实现两个正整数的除法	241
6.4	如何只使用++操作符实现加减乘除运算	246
6.5	如何根据已知随机数生成函数计算新的随机数	248
6.6	如何判断 $1024!$ 末尾有多少个 0	250
6.7	如何按要求比较两个数的大小	251
6.8	如何求有序数列的第 1500 个数的值	252
6.9	如何把十进制数 (long 型) 分别以二进制和十六进制形式输出	253
6.10	如何求二进制数中 1 的个数	254
6.11	如何找最小的不重复数	256
6.12	如何计算一个数的 n 次方	259
6.13	如何在不能使用库函数的条件下计算 n 的平方根	262
6.14	如何不使用^操作实现异或运算	262
6.15	如何不使用循环输出 1~100	264
第 7 章	排列组合与概率	265
7.1	如何求数字的组合	265
7.2	如何拿到最多金币	267
7.3	如何求正整数 n 所有可能的整数组组合	269
7.4	如何用一个随机函数得到另外一个随机函数	271
7.5	如何等概率地从大小为 n 的数组中选取 m 个整数	272
7.6	如何组合 1、2 和 5 这三个数使其和为 100	273
7.7	如何判断有几盏灯泡还亮着	275
第 8 章	排序	277
8.1	如何进行选择排序	277
8.2	如何进行插入排序	278
8.3	如何进行冒泡排序	280
8.4	如何进行归并排序	282
8.5	如何进行快速排序	285
8.6	如何进行希尔排序	287
8.7	如何进行堆排序	289

8.8	各种排序算法有什么优劣.....	291
第 9 章	大数据	292
9.1	如何从大量的 url 中找出相同的 url	292
9.2	如何从大量数据中找出高频词	293
9.3	如何找出访问百度最多的 IP	294
9.4	如何在大量的数据中找出不重复的整数	294
9.5	如何在大量的数据中判断一个数是否存在	295
9.6	如何查询最热门的查询串	296
9.7	如何统计不同电话号码的个数	297
9.8	如何从 5 亿个数中找出中位数	298
9.9	如何按照 query 的频度排序	299
9.10	如何找出排名前 500 的数	300

面试笔试经验技巧篇

想找到一份程序员的工作，一点技术都没有显然是不行的，但是，只有技术也是不够的。面试笔试经验技巧篇主要针对程序员面试笔试中遇到的 13 个常见问题进行深度解析，并且结合实际情景，给出了一个较为合理的参考答案以供读者学习与应用，掌握这 13 个问题的解答精髓，对于求职者大有裨益。

经验技巧 1 如何巧妙地回答面试官的问题

所谓“来者不善，善者不来”，程序员面试中，求职者不可避免地需要回答面试官各种刁钻、犀利的问题，回答面试官的问题千万不能简单地回答“是”或者“不是”，而应该具体分析“是”或者“不是”的理由。

回答面试官的问题是一门很深的学问。那么，面对面试官提出的各类问题，如何才能条理清晰地回答呢？如何才能让自己的回答不至于撞上枪口呢？如何才能让自己的回答结果令面试官满意呢？

谈话是一种艺术，回答问题也是一种艺术，同样的话，不同的回答方式，往往会产生不同的效果，甚至是截然不同的效果。在此，编者提出以下几点建议，供读者参考。首先回答问题务必谦虚谨慎，既不能让面试官觉得自己很自卑，唯唯诺诺，也不能让面试官觉得自己清高自负，而应该通过问题的回答表现出自己自信从容、不卑不亢的一面。例如，当面试官提出“你在项目中起到了什么作用”的问题时，如果求职者回答：我完成了团队中最难的工作，此时就会给面试官一种居功自傲的感觉，而如果回答：我完成了文件系统的构建工作，这个工作被认为是整个项目中最具有挑战性的一部分内容，因为它几乎无法重用以前的框架，需要重新设计。这种回答不仅不傲慢，反而有理有据，更能打动面试官。

其次，回答面试官的问题时，不要什么都说，要适当地留有悬念。人一般都有猎奇的心理，面试官自然也不例外，而且，人们往往对好奇的事情更有兴趣、更加偏爱，也更加记忆深刻。所以，在回答面试官问题时，切记说关键点而非细节，说重点而非和盘托出，通过关键点，吸引面试官的注意力，等待他们继续“刨根问底”。例如，当面试官对你的简历中一个算法问题有兴趣，希望了解时，可以这样回答：我设计的这种查找算法，对于 80% 以上的情况，都可以将时间复杂度从 $O(n)$ 降低到 $O(\log n)$ ，如果您有兴趣，我可以详细给您分析具体的细节。

最后，回答问题要条理清晰、简单明了，最好使用“三段式”方式。所谓“三段式”，有点类似于中学作文中的写作风格，包括“场景/任务”“行动”“结果”三部分内容。以面试官提的问题“你在团队建设中，遇到的最大挑战是什么”为例，第一步，分析场景/任务：在我参与的一个 ERP 项目中，我们团队一共四个人，除了我以外的其他三个人中，两个人能力很强，人也比较好相处，但有一个人不太好相处，每次我们小组讨论问题的时候，他都不太爱说话，也很少发言，分配给他的任务也很难完成。第二步，分析行动：为了提高团队的综合实力，我决定找个时间和他好好单独谈一谈。于是我利用周末时间，约他一起吃饭，吃饭的时候，顺便讨论了一下我们的项目，我询问了一些项目中他遇到的问题，通过他的回答，我发现他并不懒，也不糊涂，只是对项目不太了解，缺乏经验，缺乏自信而已，所以越来越孤立，越来越不愿意讨论问题。为了解决这个问题，我尝试着把问题细化到他可以完成的程度，从而建立起他的自信心。第三步，分析结果：他是小组中水平最弱的人，但是，慢慢地，他的技术变得越来越强了，也能够按时完成安排给他的工作了，人也越来越自信了，也越来越喜欢参与我们的讨论，并发表自己的看法，我们也都愿意与他一起合作了。“三段式”回答的一个最明显的好处就是条理清晰，既有描述，也有结果，有理有据，让面试官一目了然。

回答问题的技巧，是一门大的学问。求职者完全可以在平时的生活中加以练习，提高自己与人沟通的技能，等到面试时，自然就得心应手了。

经验技巧 2 如何回答技术性问题

程序员面试中，面试官经常会询问一些技术性的问题，有的问题可能比较简单，都是历年的面试笔试真题，求职者在平时的复习中会经常遇到，应对自然不在话下。但有的题目可能比较难，来源于 Google、Microsoft 等大企业的题库或是企业自己为了招聘需要设计的题库，求职者可能从来没见过或者从来都不能完整地、独立地想到解决方案，而这些题目往往又是企业比较关注的。

如何能够回答好这些技术性问题？编者建议：会做的题目一定要拿满分，不会做的题目一定要拿部分分。即对于简单的题目，求职者要努力做到完全正确，毕竟这些题目，只要复习得当，完全回答正确一点问题都没有（编者的一个朋友据说把《编程之美》《编程珠玑》《程序员面试笔试宝典》上面的技术性题目与答案全都背得滚瓜烂熟，后来找工作无往不利；对于难度比较大的题目，不要惊慌，也不要害怕，即使无法完全做出来，也要努力思考问题，哪怕是半成品也要写出来，至少要把自己的思路表达给面试官，让面试官知道你的想法，而不是完全回答不会或者放弃，因为面试官很多时候除了关注求职者独立思考问题的能力以外，还会关注求职者技术能力的可塑性，观察求职者是否能够在别人的引导下正确地解决问题，所以，对于你不会的问题，他们很有可能会循序渐进地启发你去思考，通过这个过程，让他们更加了解你。

一般而言，在回答技术性问题时，求职者大可不必胆战心惊，除非是没学过的新知识，否则，一般都可以采用以下六个步骤来分析解决。

（1）勇于提问

面试官提出的问题，有时候可能过于抽象，让求职者不知所措，或者无从下手，所以，对于面试中的疑惑，求职者要勇敢地提出来，多向面试官提问，把不明确或二义性的情况都问清楚。不用担心你的问题会让面试官烦恼，影响你的面试成绩，相反这样做还会对面试结果产生积极影响：一方面，提问可以让面试官知道你在思考，也可以给面试官一个心思缜密的好印象；另一方面，方便后续自己对问题的解答。

例如，面试官提出一个问题：设计一个高效的排序算法。求职者可能丈二和尚摸不到头脑，排序对象是链表还是数组？数据类型是整型、浮点型、字符型还是结构体类型？数据基本有序还是杂乱无序？数据量有多大，1000 以内还是百万以上个数？此时，求职者可以将自己的疑问提出来，问题清楚了，解决方案自然也就出来了。

（2）高效设计

对于技术性问题，如何才能打动面试官？完成基本功能是必需的，仅此而已吗？显然不是，完成基本功能顶多只能算及格水平，要想达到优秀水平，还应该考虑更多的内容，以排序算法为例：时间是否高效？空间是否高效？数据量不大时也许没有问题，如果是海量数据呢？是否考虑了相关环节，例如数据的“增删改查”？是否考虑了代码的可扩展性、安全性、完整性以及鲁棒性？如果是网站设计，是否考虑了大规模数据访问的情况？是否需要考虑分布式系统架构？是否考虑了开源框架的使用？

（3）伪代码先行

有时候实际代码会比较复杂，上手就写很有可能会漏洞百出、条理混乱，所以，求职者

可以先征求面试官的同意，在编写实际代码前，写一个伪代码或者画好流程图，这样做往往会让思路更加清晰明了。

切记在写伪代码前要先告诉面试官，否则他们很有可能对你产生误解，认为你只会纸上谈兵，实际编码能力却不行。只有征得了他们的允许，方可先写伪代码。

(4) 控制节奏

如果是算法设计题，面试官都会给求职者一个时间限制用以完成设计，一般为 20min 左右。完成得太慢，会给面试官留下能力不行的印象，但完成得太快，如果不能保证百分之百正确，也会给面试官留下毛手毛脚的印象，速度快当然是好事情，但只有速度，没有质量，根本不会给面试加分。所以，编者建议，回答问题的节奏最好不要太慢，也不要太快，如果实在是完成得比较快，也不要急于提交给面试官，最好能够利用剩余的时间，认真仔细地检查一些边界情况、异常情况以及极性情况等，看是否也能满足要求。

(5) 规范编码

回答技术性问题时，多数都是纸上写代码，离开了编译器的帮助，求职者要想让面试官对自己的代码一看即懂，除了要字迹工整，不能龙飞凤舞以外，最好能够严格遵循编码规范：函数变量命名、换行缩进、语句嵌套和代码布局等。同时，代码设计应该具有完整性，保证代码能够完成基本功能、输入边界值能够得到正确的输出、对各种不合规的非法输入能够做出合理的错误处理，否则，写出的代码即使无比高效，面试官也不一定看得懂或者看起来非常费劲，这些对面试成功都是非常不利的。

(6) 精心测试

在软件行业，有一个事实：任何软件都有缺陷（bug）。但不能因此就纵容自己的代码，允许错误百出。尤其是在面试过程中，实现功能也许并不十分困难，困难的是在有限的时间内设计出的算法，各种异常是否都得到了有效的处理，各种边界值是否都在算法设计的范围内。

测试代码是让代码变得完备的高效方式之一，也是一名优秀程序员必备的素质之一。所以，在编写代码前，求职者最好能够了解一些基本的测试知识，做一些基本的单元测试、功能测试、边界测试以及异常测试。

在回答技术性问题时，注意在思考问题的时候，千万别一句话都不说，面试官面试的时间是有限的，他们希望在有限的时间内尽可能地去了解求职者，如果求职者坐在那里一句话不说，不仅会让面试官觉得求职者技术水平不行，还会认为求职者思考问题能力以及沟通能力可能都存在问题。

其实，在面试时，求职者往往会存在一种思想误区，把技术性面试的结果看得太过重要。面试过程中的技术性问题，结果固然重要，但也并非最重要的内容，因为面试官看重的不仅仅是最终的结果，还包括求职者在解决问题的过程中体现出来的逻辑思维能力以及分析问题的能力。所以，求职者在与面试官的博弈中，要适当地提问，通过提问获取面试官的反馈信息，并抓住这些有用的信息进行辅助思考，从而博得面试官的认可，进而提高面试的成功率。

经验技巧 3 如何回答非技术性问题

评价一个人的能力，除了专业能力，还有一些非专业能力，如智力、沟通能力和反应能力等，所以在 IT 企业招聘过程的笔试面试环节中，并非所有的笔试内容都是 C/C++/Java、数

据结构与算法及操作系统等专业知识，也包括其他一些非技术类的知识，如智力题、推理题和作文题等。技术水平测试可以考查一个求职者的专业素养，而非技术类测试则更加强调求职者的综合素质，包括数学分析能力、反应能力、临场应变能力、思维灵活性、文字表达能力和性格特征等内容。考查的形式多种多样，但与公务员考查相似，主要包括行测（占大多数）、性格测试（大部分都有）、应用文和开放问题等内容。

每个人都有自己的答题技巧，答题方式也各不相同，以下是一些相对比较好的答题技巧（以行测为例）：

1) 合理有效的时间管理。由于题目的难易程度不同，所以不要对所有题目都“绝对的公平”、都“一刀切”，要有轻重缓急，最好的做法是不按顺序回答。行测中有各种题型，如数量关系、图形推理、应用题、资料分析和文字逻辑等，而不同的人擅长的题型是不一样的，因此应该首先回答自己最擅长的问题。例如，如果对数字比较敏感，那么就先答数量关系题。

2) 注意时间的把握。由于题量一般都比较大，可以先按照总时间/题数来计算每道题的平均答题时间，如 10s，如果看到某一道题 5s 后还没思路，则马上放弃。在做行测题目的时候，以在最短的时间内拿到最多分为目标。

3) 平时多关注图表类题目，培养迅速抓住图表中各个数字要素间相互逻辑关系的能力。

4) 做题要集中精力，只有集中精力、全神贯注，才能将自己的水平最大限度地发挥出来。

5) 学会关键字查找，通过关键字查找，能够提高做题效率。

6) 提高估算能力，有很多时候，估算能够极大地提高做题速度，同时保证正确率。

除了行测以外，一些企业非常相信个人性格对入职匹配的影响，所以都会引入相关的性格测试题用于测试求职者的性格特性，看其是否适合所投递的职位。大多数情况下，只要按照自己的真实想法选择就行了，不要弄巧成拙，因为测试是为了得出正确的结果，所以大多测试题前后都有相互验证的题目。如果求职者自作聪明，选择该职位可能要求的性格选项，则很可能导致测试前后不符，这样很容易让企业发现你是个不诚实的人，从而首先予以筛选。

经验技巧 4 如何回答快速估算类问题

有些大企业的面试官，总喜欢出一些快速估算类问题。对他们而言，这些问题只是手段，不是目的，能够得到一个满意的结果固然是他们所需要的，但更重要的是通过这些题目他们可以考查求职者的快速反应能力以及逻辑思维能力。由于求职者平时准备的时候可能对此类问题有所遗漏，一时很难想起解决的方案。而且，这些题目乍一看确实是毫无头绪，无从下手，其实求职者只要从惊慌失措中冷静下来，稍加分析，就会发现也就那么回事。因为此类题目比较灵活，属于开放性试题，一般没有标准答案，只要弄清楚回答要点，分析合理到位，具有说服力，能够自圆其说，就是正确答案，一点都不困难。

例如，面试官可能会问这样一个问题：“请你估算一下一家商场在促销时一天的营业额”，求职者不是统计局官员，如何能够得出一个准确的数据？求职者家又不是开商场的，如何能够得出一个准确的数据？即使求职者是商场的经理，也不可能弄得清清楚楚明明白白。

难道此题就无解了吗？其实不然，本题只要能够分析出一个概数就行了，不一定要精确数据，而分析概数的前提就是做出各种假设。以该问题为例，可以尝试从以下思路入手：从商场规模、商铺规模入手，通过每平方米的租金，估算出商场的日租金，再根据商铺的成本

构成，得到全商场日均交易额，再考虑促销时的销售额与平时销售额的倍数关系，乘以倍数，即可得到促销时一天的营业额。具体而言，包括以下估计数值：

1) 以一家较大规模商场为例，商场一般按 6 层计算，每层大约长 100m，宽 100m，合计 60000m^2 的面积。

2) 商铺规模占商场规模的一半左右，合计 30000m^2 。

3) 商铺租金约为 40 元/ m^2 ，估算出年租金为 $40 \times 30000 \times 365$ 元 = 4.38 亿元。

4) 对商户而言，租金一般占销售额 20% 左右，则年销售额为 $4.38 \text{ 亿元} \div 20\% = 21.9 \text{ 亿元}$ 。计算平均日销售额为 $21.9 \text{ 亿} / 365 = 600 \text{ 万}$ 。

5) 促销时的日销售额一般是平时的 10 倍，所以大约为 $600 \text{ 万元} \times 10 = 6000 \text{ 万元}$ 。

此类题目涉及面比较广，例如：估算一下北京小吃店的数量，估算一下中国在过去一年方便面的市场销售额是多少，估算一下长江的水的质量，估算一下一个行进在小雨中的人 5min 内身上淋到的雨的质量，估算一下东方明珠电视塔的质量，估算一下中国去年一年一共用掉了多少块尿布，估算一下杭州的轮胎数量。但一般都是即兴发挥，不是死记硬背就可以应付得了的。遇到此类问题，一步步抽丝剥茧，才是解决之道。

经验技巧 5 如何回答算法设计问题

程序员面试中的很多算法设计问题，都是历年来各家企业的“炒现饭”，不管求职者以前对算法知识学习得是否扎实，理解得是否深入，只要面试前买本《程序员面试笔试宝典》（编者早前编写的一本书，由机械工业出版社出版），学习上一段时间，牢记于心，应付此类题目就完全没有问题，但遗憾的是，很多世界级知名企业也深知这一点，如果纯粹是出一些毫无技术含量的题目，对于考前“突击手”而言，可能会占尽便宜，但对于那些技术好的人而言是非常不公平的。所以，为了把优秀的求职者与一般的求职者能够更好地区分开来，企业会推陈出新，越来越倾向于出一些有技术含量的“新”题，这些题目以及答案，不再是以前的样子了，而是经过精心设计的好题。

在程序员面试中，算法的地位就如同是 GRE 或托福考试在出国留学中的地位，必需但不是最重要的，它只是众多考核方面中的一个而已，不一定就能决定求职者的成败。虽然如此，但并非说就不用去准备算法知识了，因为算法知识回答得好，必然会成为面试的加分项，对于求职成功，有百利而无一害。那么如何应对此类题目，很显然，编者不可能将此类题目都在《程序员面试笔试宝典》中一一解答，一来由于内容众多，篇幅有限，二来也没必要，今年考过了，以后一般就不会再考了，不然还是没有区分度。编者以为，靠死记硬背肯定是行不通的，解答此类算法设计问题，需要求职者具有扎实的基本功以及良好的运用能力。编者无法左右求职者的个人基本功以及运用能力，因为这些能力需要求职者“十年磨一剑”地苦学，但编者可以提供一些比较好的答题方法和解题思路，以供求职者在面试时应对此类算法设计问题。“授之以鱼不如授之以渔”，岂不是更好？

(1) 归纳法

此方法通过写出问题的一些特定的例子，分析总结其中一般的规律。具体而言就是通过列举少量的特殊情况，经过分析，最后找出一般的关系。例如，某人有一对兔子饲养在围墙中，如果它们每个月生一对兔子，且新生的兔子在第二个月后也是每个月生一对兔子，问一

年后围墙中共有多少对兔子？

使用归纳法解答此题，首先想到的就是第一个月有多少对兔子，第一个月的时候，最初的一对兔子生下一对兔子，此时围墙内共有两对兔子。第二个月仍是最初的一对兔子生下一对兔子，共有 3 对兔子。到第三个月除最初的兔子新生一对兔子外，第一个月生的兔子也开始生兔子，因此共有 5 对兔子。通过举例，可以看出，从第二个月开始，每一个月兔子总数都是前两个月兔子总数之和， $Un+1=Un+Un-1$ ，一年后，围墙中的兔子总数为 377 对。

此种方法比较抽象，也不可能对所有情况进行列举，所以，得出的结论只是一种猜测，还需要进行证明。

(2) 相似法

此方法考虑解决问题的算法是相似的。如果面试官提出的问题与求职者以前用某个算法解决过的问题相似，此时此刻就可以触类旁通，尝试改进原有算法来解决这个新问题。而通常情况下，此种方法都会比较奏效。

例如，实现字符串的逆序打印，也许求职者从来就没遇到过此问题，但将字符串逆序肯定在求职准备的过程中是见过的。将字符串逆序的算法稍加处理，即可实现字符串的逆序打印。

(3) 简化法

此方法首先将问题简单化，例如改变一下数据类型、空间大小等，然后尝试着将简化后的问题解决，一旦有了算法或者思路可以解决这个被简化过的问题，再将问题还原，尝试着用此类方法解决原有问题。

例如，在海量日志数据中提取出某日访问某网站次数最多的那个 IP 地址。很显然，由于数据量巨大，直接进行排序不可行，但如果数据规模不大时，采用直接排序不失为一种好的解决方法。那么如何将问题规模缩小呢？于是想到了哈希（Hash）法，Hash 法往往可以缩小问题规模，然后在简化过的数据里面使用常规排序算法即可找出此问题的答案。

(4) 递归法

为了降低问题的复杂度，很多时候都会将问题逐层分解，最后归结为一些最简单的问题，这就是递归。此种方法，首先要能够解决最基本的情况，然后以此为基础，解决接下来的问题。

例如，在寻求全排列的时候，可能会感觉无从下手，但仔细推敲，会发现后一种排列组合往往是在前一种排列组合的基础上进行的重新排列，只要知道了前一种排列组合的各类组合情况，只需将最后一个元素插入到前面各种组合的排列里面，就实现了目标：即先截去字符串 $s[1\cdots n]$ 中的最后一个字母，生成所有 $s[1\cdots n-1]$ 的全排列，然后再将最后一个字母插入到每一个可插入的位置。

(5) 分治法

任何一个可以用计算机求解的问题所需的计算时间都与其规模有关。问题的规模越小，越容易直接求解，解题所需的计算时间也越少。分治法正是充分考虑到这一内容，将一个难以直接解决的大问题，分割成一些规模较小的相同问题，以便各个击破，分而治之。分治法一般包含以下三个步骤：

- 1) 将问题的实例划分为几个较小的实例，最好具有相等的规模。
- 2) 对这些较小的实例求解，最常见的方法一般是递归。
- 3) 如果有必要，合并这些较小问题的解，以得到原始问题的解。