

< / > 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

前端 程序员面试 算法宝典

猿媛之家 / 组编 平文 楚秦 等 / 编著

扫描二维码可免费使用
“程序员面试笔试真题库”微信刷题小程序



98%

高频知识点

本书覆盖了近**3年**程序员面试笔试中超过**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 > .

前端程序员面试算法宝典

猿媛之家 组编

平 文 楚 秦 等编著



机械工业出版社

本书是一本讲解前端程序员面试笔试真题的书籍，在写法上，除了讲解如何解答算法问题以外，还引入了示例辅以说明，让读者能够更容易理解。

本书将程序员面试笔试过程中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年来典型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，详细分析每一个题目，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，而这些内容其他同类书籍所没有的。本书根据真题所属知识点进行分门别类，结构合理、条理清晰，对于读者进行学习与检索意义重大。

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

前端程序员面试算法宝典 / 猿媛之家组编; 平文等编著. —北京: 机械工业出版社, 2018.12

ISBN 978-7-111-62539-1

I. ①前… II. ①猿… ②平… 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(2019)第072576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 尚 晨 责任编辑: 尚 晨

责任校对: 张艳霞 责任印制: 张 博

北京铭成印刷有限公司印刷

2019年6月第1版·第1次印刷

184mm×260mm·16印张·395千字

0001—3000册

标准书号: ISBN 978-7-111-62539-1

定价: 59.00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

网络服务

服务咨询热线: (010) 88361066

机工官网: www.cmpbook.com

读者购书热线: (010) 68326294

机工官博: weibo.com/cmp1952

金书网: www.golden-book.com

封面防伪标均为盗版

教育服务网: www.cmpedu.com

前言

计算机技术博大精深，日新月异，Hadoop、GPU 计算、移动互联网、模式匹配、图像识别、神经网络、蚁群算法、大数据、机器学习、人工智能、深度学习等新技术让人眼花缭乱，稍有懈怠，就会被时代所抛弃。于是，很多 IT 从业者就开始困惑了，不知道从何学起，到底什么才是计算机技术的基石。其实，其本质还是最基础的数据结构与算法知识：Hash、动态规划、分治、排序、查找等，所以，无论是世界级的大型企业还是几个人的小公司，在面试求职者的时候，往往会考察这些最基础的知识，无论你的研究方向是什么，这些基础知识还是应该熟练掌握的。

本书正是从这些基础知识点出发，讲解了求职过程中常见的数据结构与算法的真题，从而让求职者能轻松应付算法类的笔试面试题目。在写作风格上，推陈出新，对于算法的讲解，不仅使用图文并茂的方式，而且还辅以示例的讲解，目的就是为了让读者更容易理解。为了能够写出精品书籍，我们对每一个技术问题，都反复推敲，与算法精英一起反复论证可行性；对文字，我们咬文嚼字，字斟句酌，所有这些付出，只为让读者能够在读完本书后有所收获。

虽然市面上同类型书籍很多，也都写的不错，但是，我们相信，我们能够写出更适合读者需求的高质量精品书籍。为了能够在有限的篇幅里面尽可能地展现“干货”，我们在题目的选择上也下了很多功夫：首先，通过搜集近 3 年以来典型 IT 企业的面试笔试算法真题，包括已经出版的其他著作、技术博客、在线编码平台、刷题网站等，保证所选样本足够大。其次，选择题目的时候，尽可能不选择那种一眼就能知道结果的简单题，也没有选择那种怪题、偏题和很难的题，选题原则是难度适中或者看上去简单但实际容易出错的题目。通过我们的努力，力求所选出来的算法真题能够最大限度地帮助读者。在真题的解析上，采用层层递进的写法，先易后难，层层深入，将问题抽丝剥茧，使得读者能够跟随我们的思路，一步步找到问题的最优解。

写作的过程是一个自我提高、自我认识的过程，很多知识，只有深入理解与剖析后，才能领悟其中的精髓，掌握其中的技巧，程序员求职算法也不例外。本书不仅具备了其他书籍分析透彻，代码清晰合理的优点，还具备以下几个方面的优势：

第一，算法书籍分多种语言版本实现：C/C++、Java、C#、Python、PHP、Javascript、Kotlin、Go 等，这样，不管读者侧重于哪一种语言，都能够有适合自己的书。本书中如果没有特别强调，代码实现均默认使用 Javascript 语言。

第二，每个题目除了循序渐进的分析以外，还对方法进行了详细阐述，针对不同方法的时间复杂度与空间复杂度，都进行了详细地分析，除此之外，为了更具说服力，每一种方法几乎都对应有示例（代码可以从 <https://github.com/pwstrick/FEG-Code> 下载）讲解辅以说明。

第三，除了对题目的讲解，还有部分触类旁通的题目供读者练习。本书不可能将所有的程序员求职类的数据结构与算法类题目囊括，但是，本书会尽可能地将一些常见的求职类算

法题、具有代表性的算法题重点讲解，将其他一些题目以练习题的形式展现在读者面前，以供读者思考与学习。

本书中有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 联系作者。

猿媛之家

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	如何回答技术性的问题	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	在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”	13
经验技巧 12	如何处理与面试官持不同观点这个问题	14
经验技巧 13	什么是职场暗语	14

面试笔试真题解析篇

第 1 章	经典算法题	19
1.1	有多少苹果用来分赃	19
1.2	哪只猴子可以当大王	20
1.3	移动多少盘子才能完成汉诺塔游戏	21
1.4	如何利用约瑟夫环来保护你与你的朋友	22
1.5	怎样才能得到阿姆斯壮数	24

1.6	如何获取规定的排列组合	25
1.7	如何实现洗牌算法	26
1.8	怎样求出斐波那契数列	27
1.9	如何实现杨辉三角	28
1.10	牛的数量有多少	29
1.11	百钱买百鸡	29
1.12	经过这个路口多少次	30
1.13	球的反弹高度有多高	31
1.14	如何找出 1000 以内的“完数”	31
1.15	猴子吃了多少桃子	32
1.16	移动最少次数的三色旗	33
第 2 章	逻辑、智力题	35
2.1	经典逻辑题	35
2.1.1	老鼠相遇的概率是多少	35
2.1.2	如何计算时钟的三针重叠	35
2.1.3	如何喝到最多瓶汽水	36
2.1.4	住旅店花了多少钱	36
2.1.5	如何判断哪个开关控制着哪盏灯	36
2.1.6	如何用烧绳来计算时间	36
2.1.7	如何用水壶获取指定的水量	37
2.1.8	怎样才能猜出另外两个人的数	37
2.1.9	卖鸡总共赚了多少钱	38
2.1.10	跳高名次是多少	38
2.1.11	如何根据银币猜盒子	39
2.1.12	马牛羊的价格是多少文钱	39
2.1.13	赔多少	39
2.1.14	海盗如何分金才能让他获得最多的金子	40
2.1.15	张老师的生日是哪一天	41
2.1.16	拿几个乒乓球	41
2.2	逻辑推理题	42
2.2.1	怎样才能推理出学生的专业	42
2.2.2	错误的判断是哪一个	43
2.2.3	最后参加紧急项目的开发人员是谁	43
2.2.4	猜的第一个数字是多少	43
2.2.5	需要多少个人测试才能判断出毒酒	44
2.2.6	地图重合点有几个	45
2.2.7	掰断多少次金条才能按要求给雇工报酬	46
2.2.8	握手次数是多少	46
2.2.9	如何判断出坏鸡蛋	47

2.3 概率与组合	48
2.3.1 抽球人数是多少	48
2.3.2 案件发生在 A 区的可能性是多少	48
2.3.3 男女比例将会是多少	48
2.3.4 对称矩阵有多少个	49
2.3.5 A、B 点有多少种走法	49
2.3.6 多少种排队方式	50
2.3.7 把球放到小桶中有多少种放法	51
2.3.8 正确描述 100 台虚拟机发生故障的是哪一个	51
2.3.9 圆桌上一共有多少种坐法	51
2.3.10 兵马俑博物馆可容纳多少人	52
2.3.11 两种策略的预期收益是多少	52
2.3.12 拾起别人帽子的概率是多少	53
2.3.13 合法表达式有多少个	53
2.3.14 会写 Java 和 C++ 程序的有多少人	54
2.3.15 乘坐甲车的概率是多少	54
2.3.16 A 到 Z 的最短路径数是多少	54
2.3.17 选取红黄白球的概率是多少	55
2.3.18 一共有多少种染色情况	55
2.3.19 肇事车是白车的概率是多少	55
2.3.20 获得冠军的情况有多少种	56
2.3.21 一红一黑的概率是多少	56
2.3.22 谁会赢	56
2.3.23 描述正确的是哪一项	57
2.4 数学计算	58
2.4.1 一共等了女神多少分钟	58
2.4.2 使用了什么进制运算	58
2.4.3 三角形有多少个	59
2.4.4 数列的规律是什么	60
2.4.5 数列使用了什么规律	60
2.4.6 余数是多少	60
2.4.7 如何才能找到最好的羽毛球员工	61
2.4.8 亮着的灯泡有多少个	61
2.4.9 工作时长是多少	62
2.4.10 最小夹角是多少度	62
2.4.11 求解到的余数是多少	62
2.4.12 如何正确计算余数	63
2.4.13 最高效的矩阵是哪个	63
2.4.14 可以实现的函数是哪个	63

2.4.15	可以兑换多少瓶加多宝	64
2.4.16	共赚多少钱	64
2.4.17	实际折扣是多少	65
第 3 章	排序算法	66
3.1	如何实现冒泡排序	67
3.2	什么是插入排序	70
3.3	归并排序的原理是什么	72
3.4	快速排序使用了什么思想	73
3.5	选择排序的实现过程是怎样的	75
3.6	什么叫希尔排序	78
3.7	如何实现堆排序	79
3.8	计数排序的原理是什么	81
3.9	怎样用 JavaScript 代码实现桶排序	83
第 4 章	链表	85
4.1	如何实现链表的逆序	85
4.2	如何从无序链表中移除重复项	91
4.3	如何计算两个单链表所代表的数之和	95
4.4	如何对链表进行重新排序	99
4.5	如何找出单链表中的倒数第 k 个元素	102
4.6	如何检测一个较大的单链表是否有环	107
4.7	如何把链表相邻元素翻转	110
4.8	如何把链表以 k 个结点为一组进行翻转	112
4.9	如何合并两个有序链表	116
第 5 章	栈与队列	120
5.1	如何实现栈	120
5.2	如何实现队列	124
5.3	如何翻转栈的所有元素	128
5.4	如何根据入栈序列判断可能的出栈序列	134
5.5	如何用 $O(1)$ 的时间复杂度求栈中最小元素	138
5.6	如何用两个栈模拟队列操作	140
第 6 章	二叉树	141
6.1	二叉树基础知识	141
6.2	如何实现二叉树	143
6.3	如何把一个有序的整数数组放到二叉树中	148
6.4	如何用二叉树实现多层次分类	151
6.5	如何找到二叉树中的最大最小值	153
6.6	如何对二叉树进行遍历	155
6.7	如何判断一棵二叉树是否是二叉树	158

第 7 章	数组	161
7.1	如何找出数组中唯一的重复元素	161
7.2	如何查找数组中元素的最大值和最小值	166
7.3	如何找出旋转数组的最小元素	170
7.4	如何找出数组中丢失的数	172
7.5	如何找出数组中出现奇数次的数	174
7.6	如何找出数组中第 k 小的数	176
7.7	如何求数组中两个元素的最小距离	179
7.8	如何求解最小三元组距离	181
7.9	如何求数组连续最大和	184
7.10	如何求数组中绝对值最小的数	188
7.11	如何找出数组中出现一次的数	190
7.12	如何在不排序的情况下求数组的中位数	192
7.13	如何求集合的所有子集	194
7.14	如何对数组进行循环移位	196
7.15	如何在有规律的二维数组中进行高效的数据查找	198
7.16	如何寻找最多的覆盖点	200
7.17	如何判断请求能否在给定的存储条件下完成	201
7.18	如何按要求构造新的数组	203
7.19	如何获取最好的矩阵链相乘方法	204
7.20	如何求解迷宫问题	206
7.21	如何从三个有序数组中找出它们的公共元素	208
第 8 章	基本数字运算	210
8.1	如何判断一个自然数是否是某个数的二次方	210
8.2	如何判断一个数是否为 2 的 n 次方	212
8.3	如何不使用除法操作符实现两个正整数的除法	213
8.4	如何只使用递增运算符 ($++$) 实现加减乘除运算	218
8.5	如何根据已知随机数生成函数计算新的随机数	220
8.6	如何判断 $1024!$ 末尾有多少个 0	221
8.7	如何按要求比较两个数的大小	222
8.8	如何求有序数列的第 1500 个数的值	222
8.9	如何求二进制数中 1 的个数	224
8.10	如何计算一个数的 n 次方	225
8.11	如何在不能使用库函数的条件下计算正数 n 的算术平方根	227
8.12	如何不使用 “ $\wedge$ ” 符号实现异或运算	227
8.13	如何不使用循环输出 $1 \sim 100$	228
第 9 章	排列组合与概率	230
9.1	如何拿到最多金币	230
9.2	如何求正整数 n 所有可能的整数组合	231

9.3	如何用一个随机函数得到另外一个随机函数	233
9.4	如何等概率地从大小为 n 的数组中选取 m 个整数	234
9.5	如何计算 1、2、5 这三个数使其和为 100 的组合个数	235
9.6	如何判断有几盏灯泡还亮着	236
第 10 章	海量数据处理	238
10.1	如何从大量的 url 中找出相同的 url	238
10.2	如何从大量数据中找出高频词	239
10.3	如何找出某一天访问百度网站最多的 IP	240
10.4	如何在大量的数据中找出不重复的整数	240
10.5	如何在大量的数据中判断一个数是否存在	241
10.6	如何查询最热门的查询串	241
10.7	如何统计不同电话号码的个数	243
10.8	如何从 5 亿个数中找出中位数	243
10.9	如何按照 query 的频度排序	245

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？如何才能让自己的答案令面试官满意呢？

谈话是一门艺术，回答问题也是一门艺术。同样的话，不同的回答方式，往往也会产生出不同的效果，甚至是截然相反的效果。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，可以如下回答：我设计的这种查找算法，对于 80%或以上的情况，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”和“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，但有一个人却不太好相处，每次我们小组讨论问题时，他都不太爱说话，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厉害了，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，有根有据，让面试官一目了然。

回答问题的技巧，是一门大学问。求职者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，自然就得得心应手了。

经验技巧2 如何回答技术性的问题

程序员面试中，面试官会经常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。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的问题呢？编者建议：会做的一定要拿满分，不会做的一定要拿部分分。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（编者认识的一个朋友曾把《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熟，找工作时遇到该类问题解决得非常轻松）；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的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去正确地解决问题。所以，对于不会的问题，面试官很有可能会循序渐进地启发求职者去思考，通过这个过程，让面试官更加了解求职者。

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，因此，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，影响面试成绩，相反还对面试结果产生积极的影响：一方面，提问可以让面试官知道求职者在思考，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。求职者可能没有头绪，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？此时，求职者大可以将自己的疑问提出来，问题清楚了，解决方案也自然就出来了。

（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，完成基本功能最多只能算及格水平，要想达到优秀水平，至少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，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

以首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。

(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 20 分钟。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，速度快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，认真检查一些边界情况、异常情况及极限情况等，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，除了字迹要工整外，最好是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

任何软件都有 bug，但不能因为如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，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多了解求职者，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，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重要了。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，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,部分与公务员考查相似,主要包括行政职业能力测验(简称“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。由于题目的难易不同,答题要分清轻重缓急,最好的做法是不按顺序答题。“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,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。

2) 注意时间的把握。由于题量一般都比较大会,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,则马上做后面的题。在做行测题目时,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、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,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,则很可能导致测试前后不符,这样很容易让企业发现求职者是个不诚实的人,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,一时很难想到解决的方案。而且,这些题目乍一看确实是毫无头绪,无从下手,其实求职者只要冷静下来,稍加分析,就能找到答案。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。

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

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，每层长约 100m，宽约 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%，则年销售额为 $4.38 \text{ 亿元} \times 5 = 21.9 \text{ 亿元}$ 。计算平均日销售额为 $21.9 \text{ 亿元} / 365 = 600 \text{ 万元}$ 。

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

此类题目涉及面比较广，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 5 分钟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，不是哪道题记住答案就可以应付得了的。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掌握得是否扎实，理解得是否深入，只要面试前买本《程序员面试笔试宝典》，应付此类题目完全没有问题。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更好地区分开来，面试题会年年推陈出新，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，不再是以前的问题了，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一样，必须但不是最重要的，它只是众多考核方面中的一个方面而已。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呢？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，一是由于内容过多，篇幅有限，二是也没必要，今年考过了，以后一般就不会再考了，不然还是没有区分度。编者认为，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和良好的运用能力，因为这些能力需要求职者“十年磨一剑”，但“授之以鱼不如授之以渔”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。

(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