< / > 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

PROGRAMER

Algorithm interview

Java 程序员面试 算法宝典

猿媛之家 /组编

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 > .

机械工业出版社
CHINA MACHINE PRESS

Java 程序员面试算法宝典

猿媛之家　组编

机械工业出版社

本书是一本讲解 Java 程序员面试算法的书籍，在写法上，除了讲解如何解答算法问题外，还引入了实例辅以说明，让读者能够更好地理解本书内容。

本书将 Java 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，本书收集了近三年来几乎所有 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，引入实例与源代码、时间复杂度与空间复杂度的分析，而这些内容是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，力图做到结构合理、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，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

Java 程序员面试算法宝典 / 猿媛之家组编. —北京：机械工业出版社，2018.7
ISBN 978-7-111-60395-5

Ⅰ. ①J… Ⅱ. ①猿… 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 149932 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　王　荣
责任校对：张艳霞　责任印制：李　昂

北京京师印务有限公司印刷

2018 年 7 月第 1 版 • 第 1 次印刷
184mm×260mm • 18.75 印张 • 456 千字
0001－3000册
标准书号：ISBN 978-7-111-60395-5
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

网络服务
机 工 官 网：www.cmpbook.com
机 工 官 博：weibo.com/cmp1952
金　书　网：www.golden-book.com
教育服务网：www.cmpedu.com

前　言

程序员，一类听起来神秘看上去却普普通通的人群，写着那些密密麻麻的常人看不懂的代码，领着一份让很多人都羡慕的薪水。也许每个人心中对程序员的理解都不尽相同，也许每个程序员的生活状态各有千秋，但毋庸置疑，在程序员是否是一个高大上的职业上，他们还是惊人的一致：是。

在信息技术高度发达的今天，很难想象，离开了程序员，这个社会会是什么样子。从手机、计算机、电视机、洗衣机等日常生活用品到宇宙飞船、空间站、飞机、坦克、大炮，都离不开计算机程序。计算机程序已然渗透到生活中的方方面面。虽然计算机程序是一个软实体，看不见摸不着，人们也感知不到它的存在，但是你们是否知道，没有了操作系统与应用程序，手机就是一个废物；没有了计算机程序，大炮、飞机、航空母舰就是一个铁疙瘩；没有了计算机程序，洗衣机、空调就是一个摆设。

计算机程序有多么重要，也许上面说的还不清楚，在此举两个简单例子就可以说明。当前，智能手机高度发展，但是手机的 Android 操作系统是由软件巨头谷歌（Google）研发的，所有使用 Android 系统的手机厂商都受制于 Google。诺基亚（Nokia）手机曾经是世界上销量非常好的手机，由于没能跟上 Android 的步伐，最终走向了覆灭，被另一家软件巨头微软（Microsoft）公司收购了。苹果（Apple）智能设备之所以很受欢迎，除了其良好的外形设计以外，另一个重要原因就是其苹果操作系统（iOS）的独一无二。这些都是软件打败硬件的例子。

计算机技术博大精深，而且日新月异，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，稍有不慎，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，究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 IT 企业还是小公司，在面试求职者时，往往都会考察这些最基础的知识，无论研究方向是什么，这些基础知识都是必须熟练掌握的。

本书在写作风格上推陈出新，对于算法的讲解，不仅文字描述，更以示例佐证，让读者能够更好地读懂本书内容。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，只为让读者能够对书中技术点放心，文字描述舒心。

市面上同类型书籍很多，但是，我们相信，我们能够写出更适合读者的高质量精品书籍。为了能够在有限的篇幅里面尽可能地全是“干货”，作者在选择题目上下了很大的工夫。首先，我们通过搜集近三年来几乎所有 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，保证所选样本足够全面。其次，在选择题目时，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、难题。我们的原则是选择那些难度适中或者看上去简单但实际容易中陷阱的题目。我们力求遴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，将问题抽丝剥茧，使得读者能够跟随作者的思路，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、代码清晰合理等优点，还具备以下几个方面独有的优势。

第一，本书分多种语言版本实现：C/C++、Java、C#等，不管读者侧重于哪一种语言，都能够从书中找到适合自己的内容。后续可能还有 PHP 等其他语言描述的图书出现。**本书中如果没有特别强调，则代码实现均默认使用 Java 语言。**

第二，每个题目除了有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的分析。除此之外，为了更具说服力，每一种方法几乎都对应有示例讲解辅以说明。

第三，代码较为规范，完全参照华为编程规范、Google 编程规范和编码规范。程序员要想在团队中大展拳脚，就离不开合作，而合作的基础就是共同遵循统一的编码规范。不仅如此，规范化的编码往往有助于读者理解代码。

第四，除了题目讲解，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，但是会尽可能地将一些常见的求知类算法题、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。

数据结构与算法知识博大精深，非一本或是几本著作就能将其讲解透彻的。尽管我们力求将所有程序员求职过程中出现的面试、笔试题一网打尽，试图做到知识覆盖面广，内容知识全，但仍然无法做到面面俱到，百分之百的读者满意率是本书以及后续改版奋斗与追求的目标，希望读者能够体谅。有兴趣的读者可以参阅《算法导论》《编程珠玑》等国外知名专家编写的专著进行知识的扩展与延伸。

其实，本书不仅可以作为程序员求职的应试类书籍，还可以作为数据结构与算法的教辅书籍。书中的很多思想和方法对于提高对数据结构与算法的理解是大有裨益的，不管是本科生还是研究生，不管是低年级学生还是高年级学生，不管对计算机底层知识或是当前的计算机前沿知识是否了解，都不影响你学好本书。

本书是作者历经四年时间打造的一本技术类精品图书，尽管我们尽最大努力希望做到百分之百的准确性，但百密一疏，书中不足之处在所难免，在恳请读者原谅的同时，也希望读者们能够将这些问题进行反馈，以便于未来继续改进与提高，为读者提供更加优秀的作品。

本书中引用的部分内容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

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 20min。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、对各种不合规范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，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，从而首先予以筛除。

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 40×30000×365 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 亿元×5=21.9 亿元。计算平均日销售额为 21.9 亿元/365=600 万元。

5）促销时的日销售额一般是平时的 10 倍，所以约为 600 万元×10=6000 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

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，也不可能对所有的情况进行列举，所以得出的结论只是一种猜测，还需要进行证明。

（2）相似法

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，如改变数据类型、空间大小等，然后尝试着将简化后的问题解决，一旦有了一个算法或者思路可以解决这个问题，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×××网站次数最多的那个 IP。由于数据量巨大，直接进行排序显然不可行，但如果数据规模不大时，采用直接排序是一种好的解决方法。那么如何将问题规模缩小呢？这时可以使用 Hash 法，Hash 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，在寻求全排列时，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…n]中的最后一个字母，生成所有 s[1…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。而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，而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。

分治法是程序员面试常考的算法之一，一般适用于二分查找、大整数相乘、求最大子数组和、找出伪币、金块问题、矩阵乘法、残缺棋盘、归并排序、快速排序、距离最近的点对、

导线与开关等。

（6）Hash 法

很多面试、笔试题目，都要求求职者给出的算法尽可能高效。什么样的算法是高效的？一般而言，时间复杂度越低的算法越高效。而要想达到时间复杂度的高效，很多时候就必须在空间上有所牺牲，用空间来换时间。而用空间换时间最有效的方式就是 Hash 法、大数组和位图法。当然，有时，面试官也会对空间大小进行限制，那么此时求职者只能再去思考其他的方法了。

其实，凡是涉及大规模数据处理的算法设计中，Hash 法就是最好的方法之一。

（7）轮询法

在设计每道面试、笔试题时，往往会有一个载体，这个载体便是数据结构，如数组、链表、二叉树或图等，当载体确定后，可用的算法自然而然地就会显现出来。可问题是很多时候并不确定这个载体是什么，当无法确定这个载体时，一般也就很难想到合适的方法了。

编者建议，此时，求职者可以采用最原始的思考问题的方法——轮询法。常考的数据结构与算法一共就几种（见表 0-1），即使不完全一样，也是由此衍生出来的或者相似的。

表 0-1　最常考的数据结构与算法知识点

数据结构	算　法	概　念
链表	广度（深度）优先搜索	位操作
数组	递归	设计模式
二叉树	二分查找	内存管理（堆、栈等）
树	排序（归并排序、快速排序等）	—
堆（大顶堆、小顶堆）	树的插入/删除/查找/遍历等	—
栈	图论	—
队列	Hash 法	—
向量	分治法	—
Hash 表	动态规划	—

此种方法看似笨拙，却很实用，只要求职者对常见的数据结构与算法烂熟于心，一点都没有问题。

为了更好地理解这些方法，求职者可以在平时的准备过程中，应用此类方法去答题，做题多了，自然对各种方法也就熟能生巧了，面试时再遇到此类问题，也就能够得心应手了。当然，千万不要相信能够在一夜之间练成“绝世神功”。算法设计的功底就是平时一点一滴的付出和思维的磨炼。方法与技巧只能锦上添花，却不会让自己变得从容自信，真正的功力还是需要一个长期的积累过程的。

经验技巧 6　如何回答系统设计题

应届生在面试时，偶尔也会遇到一些系统设计题，而这些题目往往只是测试求职者的知识面，或者测试求职者对系统架构方面的了解，一般不会涉及具体的编码工作。虽然如此，