算法面试指南

为什么要学习算法和数据结构？尽管原因有很多，比如锻炼逻辑思维能力、编码能力、阅读源码的能力等等，但我想对于大多数人来说，最务实、最简单的原因是应付算法面试。

现在，随着IT就业人员越来越多，内卷越来越严重，公司的招聘门槛也越来越高。之前很多公司的面试重视框架、语言、项目经历等层面的考察，现在，为了拔高招聘门槛，很多公司开始越来越重视基本功的考察，特别是数据结构和算法。

除此之外，对于很多人心仪的大厂，比如BAT、今日头条、拼多多，算法更是面试中的必考项目。不仅如此，近两年，一些二三线公司、优质创业公司，也都开始重视算法面试。所以，不管是校招还是社招，只要是应聘一线开发，不管是工程师、架构师，还是开发Leader，都很难逃过算法面试这一环。

能轻松应对算法面试，并不是件临时抱佛脚就能搞定的事情，需要长期的学习和积累。不过，本篇文章不打算细致入微的讲解如何学习算法，而是只针对算法面试这一点，就我自己面试和被面试的经历，给你指明应对算法面试的一个大的努力方向和学习框架，让你可以有章可循，事半功倍。

01

算法面试的目的和题型

为什么大厂都喜欢考算法呢？仅仅是为了拔高招聘门槛吗？当然不是。面试是为了过滤、选拔具有优秀开发能力的人才，所有面试题目的设计都围绕着这个目的。了解了算法面试的目的，你才能更加“投其所好”地表现。

通过算法面试，我们可以很好的了解候选人对数据结构和算法的掌握程度、逻辑思维是否清晰、代码编写是否规范，甚至可以考察出候选人的学习能力、理解能力、知识迁移能力、举一反三能力、沟通能力等等，而上面列举到的这些所有能力，都会直接体现到候选人在今后工作中的表现。

不管你工作多久，三年、五年，还是十年，如果在面试中被要求写一段代码，千万别以为面试官只是考察你会不会写代码而已。比如有一道经典的算法面试题目是求平方根问题，你可能会说，这个题目直接调用Math.sqrt()函数不就解决了吗？为啥还要多此一举自己实现？显然面试官要考察的并不是问题本身。实际上，写代码只是一个考察形式，面试官希望你展示的是写出好代码背后所需要具备的能力。

算法面试题目一般分为两类，一类是偏实战的题目，另一类是偏编码的题目。

对于偏实战的题目，一般面试官会给你一个接近真实项目场景的问题，然后让你结合具体的场景，思考解决方案。这类题目往往比较开放，需要你做需求的挖掘、分析、假设，合理恰当地应用数据结构和算法是答题的关键。这类问题一般代码实现都比较繁琐，所以，不需要你写代码实现，只需要给出思路即可。

对于偏编码的题目，一般面试官会给你一个类似笔试一样的题目，有确切的问题描述、输入和输出格式，让你给出算法思路并且编码实现，这就有点类似做LeetCode上的题

目。这类题目的编码实现也是考察的重点，不过，一般代码都不会很长，10行~30行的样子，最多也不会超过50行。

一般来讲，偏实战的算法面试题目，相对来说要少一些，因为要贴近实战，所以不好出题。而偏编码的算法面试题目，相对来说就是算法面试的主要形式。一场时长1个小时左右的面试，一般会有两道题目。第一道题目会比较简单，相当于一个暖场和摸底，缓和候选人的紧张情绪，试探一下候选人的水平。第二道题目就是“正餐”了，一般会比第一道题目难不少。不过，整体上来说，算法面试并不是竞赛，考察的是基本功，所以，难度一般只相当于在LeetCode上“简单”、“中等”题目的难度。

02

算法面试前的准备工作

首先，我们需要全面掌握经典的数据结构和算法。对于经典的数据结构和算法，我们又分为基础的和高级的两类。

基础的数据结构有数组、链表、栈、队列、散列表、二叉树、堆、图的定义等，基础的算法有：排序、二分查找、二叉树上的操作（遍历、查找、插入、删除等）、图的深度广度优先搜索、字符串朴素匹配算法，以及递归、分治、贪心、回溯、动态规划等基础算法思想。

高级的数据结构和算法有跳表、并查集、线段树、树状数组、BM算法、KMP算法、AC自动机、红黑树、B+树、图的一些高级算法（比如最大流、二分匹配、Dijkstra、Floyd算法）等。

对于基础的数据结构和算法，我们需要掌握原理、熟练代码实现、复杂度分析等，毕竟它们是很多算法问题解决的基础，需要掌握牢固。比如经典的求平方根问题，实际上就可以转化成简单的二分查找，再比如经典的求逆序对个数问题，实际上就是归并排序算

法的改进。如果你连二分查找、归并排序都没有写熟练，对于这些问题的解答，显然就已经输在了起跑线上。

对于高级的数据结构和算法，我们只需要理解算法原理、掌握应用场景，对于代码实现，基本上不做要求，更不需要像对待基础数据结构和算法那样，要牢记原理和实现。学了之后忘记了，也没有关系。

其次，光学不练也不行，我们还需要进行刻意的刷题训练。毕竟算法面试一般不会直接问你二分查找如何来写、堆如何来实现，所以，除了要掌握理论知识之外，还要锻炼将知识应用来解题的能力，这就包括知识迁移能力、举一反三的能力、抽象建模能力、组合数据结构和算法解决问题的能力等等，而这些能力完全可以通过刷题来锻炼。而且，在类似LeetCode这样的网站上刷题，也算是比较接近真实面试的一种模拟演练。

所以，在基本掌握了经典的数据结构和算法之后，你还要用比较长的一段时间（比如半年）来刷题强化训练。目前，最著名的刷题网站非LeetCode莫属了。其中，网站对题目有难易程度和类型的分类，你可以针对每种类型，选择一些题目刻意训练。特别是对于比较高频的一些问题，比如链表、二叉树、动态规划、递归回溯相关的问题，以及一些经典的问题，比如归并求逆序对、借助快排求TOP K等问题，要反复练习。毕竟常用的算法和解题套路都是有限的，大部分面试题目也都是基于现有的题目的改造、变形或组合，又或者换一个新的问题背景重新来问，所以，高频题、经典题要多练习，做到看到题目脑袋里能立刻反应出解决方法，并且能熟练写出没有bug的代码实现。

长期的积累和刷题很重要，但也不能忽略短期的突击。在面试前，我们需要重新回顾一遍所有的数据结构和算法，并且从网上找一些目标公司的面试题，真刀真枪地模拟演练一把，热热身。不仅如此，根据网上的面经，我们还能了解目标公司面试题的难易程度和面试官的喜好，有针对性的准备。比如有的公司喜欢面试动态规划，面试前就去多刷下这类题目，有的公司的算法面试比较简单，就不要花太多时间刷难题了。

除此之外，我们平时都是在IDE中写代码。IDE本身有自动提示功能，并且可以随意删删改改，而算法面试一般要求手写代码，对整洁度的要求就要高很多，如果写得乱七八

糟，面试官会觉得你思路混乱。所以，在面试前，你一定要在纸上多练习一下，做到脑袋里想好算法之后，能一气呵成地写出代码。总之，要让平时的训练无比接近真实的面试场景，这样才能在面试中不会因为环境的改变而紧张，才能像平时训练一样正常发挥。

03

算法面试中的解题技巧

实际上，算法面试也有固定的答题套路。

首先，跟面试官沟通确认问题，包括数据规模、输入输出要求，以及其他一些不清楚的地方，一定要确认没有理解误差之后，再进行答题。

答题的过程，先思考最简单的解决方案，说给面试官听，然后再行优化，思考更加好的解决方案。这样做的目的是，一方面能缓和自己紧张的心情，不至于大脑放空、卡壳，另一方面，一开始就思考最优解法，可能要闷头想很久，面试官很难知道你的思考进度，也无法基于你现有的思考进度做提示。

不管题目的难易，建议每个题目的思考时间都不要超过10分钟。10分钟还想不出解法，更多的时间可能也无济于事了，而且10分钟是面试官可以忍受的沉默时间的极限。所以，如果10分钟还没有思路，建议跟面试官沟通，以求给与提示。

在想到最优解决思路之后，也不要着急写代码，先要跟面试官沟通，看是否满足面试官的要求。在得到面试官的肯定之后，再进行编码实现，以免进入思维误区，想出来的解决办法有漏洞，并非正确或最优解，急匆匆地写代码，写完才发现有问题，最后也没有时间去改正或优化了。

编码也是一个非常重要的环节。很多算法问题，即便有了解决思路，编码实现也并不简单。比如在O(1)空间复杂度内判断存储在链表中的字符串是否是回文串这样一个题目，实际上，它就是反转链表和链表求中间结点这两个问题的组合，算法并不难，但要正确、快速地用代码实现，并不简单，需要处理很多细节，稍有不慎就会引入bug，非常考验候选人的编码能力。

除此之外，在编写代码的过程中，一定要注意编码规范，保证代码整洁、可读，切忌使用i、j、k这样的字母来命名重要的变量。一目了然的命名，清晰的代码结构，会反映出候选人良好的编程习惯，从而赢得面试官的好感。

在写完代码之后，一定要进行单元测试。列举完备的测试用例，特别是一些极端测试用例，比如输入为0、null等，看程序是否运行正确。一方面能验证自己写的代码的正确性，另一方面还能发现一些边界条件处理不到位的情况，再者还能让面试官觉得你思考问题比较全面、细心。

一般情况下，面试官会自己阅读你的代码来判断是否编写正确，但也有些面试官会要求候选人逐行解释代码。为了方面解释，特别是针对链表或树相关的一些复杂问题，我们可以通过具体的例子或者画图来辅助讲解。

算法面试并非笔试，并不只看最终答案，考察的重点是候选人在面试的过程中体现出来的沟通能力、逻辑思维能力、分析问题能力、优秀的编码开发能力等等。所以，有的时候即便你没有给出最优解决思路，也有可能会被录用，而有的时候，你觉得回答的很不错，给出了最优解，也未必会被录用。