




我的人缘 0

2 主题 | 17 帖子 | 2499 积分


发消




分享帖子到朋友圈


 intelliui 2018-7-26 14:25:58 | 只看该作者 | 只看大图 ▶




1楼 电梯直达 

本楼:  97% (1)

2% (1) 

全局:  98% (781)

1% (8) 

## 前言

前几天写了个关于面试官心理活动和侧重点的帖子, 见 [谈谈面试官在面试coding题目时的考察终点与心理活动](#), 求大米, 首先感谢大家的捧场与大米, 也请大家原谅我的一些错别字与语无伦次 (为了快速求大米写的粗糙了一点), 反应超出我的预期。也让我有了动力接着写这一篇帖子。

在上一篇帖子中提到了coding的一般步骤与考察点, 在开贴讨论design, behavior, culture fit前, 继续深入的讨论一下coding的考察范围, 也以此做为对一些同学, 特别是new grad的问题的统一回复。我就不排版和检查拼写了, 大家继续凑合读。

## coding 种类与应对策略

大致上, 面试官在开始面试前, 会收到一封email, 里面回大致说明每个人需要侧重于考察面试者的哪个方面。对于coding来说, 一般有三类问题, 每个面试官会被分配到一类问题。

### 1. solid coding

这类问题说白了, 谁都知道怎么做, 纯粹就是考察coding是不是扎实, 平时自己写code多不多, 能不能快速的把自己的idea转化为code。对于面试者来说属于必考种类, new grad 一般会有两轮甚至三轮这样的题目, 有很多工作经验的人可能就只有1轮了。这类题目不过关, 很可能电面死掉或者前几轮突然死亡。

solid coding又一般可以分成两个小类:

1.1 考察你对算法的基本理解以及边界条件的运用, 比如findkth largest integer, search in rotate array, bit manipulation 等等。

1.2 考察你对基本数据结构以及复杂度的理解。比如binary search tree, linkedlist vs array, stack, tree dfs, tree bfs 等等。

按难度来分, easy的比如3 sum, tree level order iterator. medium 难度的比如 reverse linked list from index m to index n, course schedule, string multiplication, hard 难度的比如valid number, 复杂的calculator等。

### 应对策略:

1.1 类型, 如果是简单和medium的没得说了, 就是希望你又快又好, 除了勤奋和熟练, 没有什么好策略。对于像merge sort, partition这类的算法, 如果7-8分钟还写不出bug free我估计就没戏了。easy问题请多多注意边界条件, int 溢出, nullpointer, 负数, 非法输入等。

hard 1.1的请参考1.3

1.2 类型, 简单和medium请在写代码前多阐明复杂度, 这类数据结构的问题往往也可以在coding前画图来表示运行状态, 图画的清楚也是个重要的加分项。hard请参考1.3

1.3 hard类型的coding题目。这往往是考察你的solid coding的能力, 即我在前文中提到的, 你做事的方式和你思考问题的方法。即给你一个coding任务, 你如何从白板开始, 一步步的做出bug free的程序。这类问题的过程重于结果。比如valid number, 你能确保每实现一个模块, 都没有regressgion, 都没有bug, 比你一下子实现所有的feature但是有很多bug要好很多。一般来说面试官看你是否能够一步步的分隔出小的coding模块 (method), 你如何设计test case, 你如何能够确保这些test case能cover所有的scenario, 你是不是和面试官提前做了足够的沟通并且限定了coding范围。从这个角度来说, valid number其实是个很不错的solid coding面试题。限于篇幅, 我就不展开来说了。

### 2. problem resolving

这类问题对于new grad是关键, 也是能帮你differentiate的关键。说白了, 计算机并不是只有算法, 我们还需要数据库, 操作系统, 网络, 安全等方面的知识。new grad这些方面要弱一些, 所以面试官希望new grad能展现出思维敏捷, 多思考, 快速反应的能力。problem resolving就为了考察这个能力而诞生的。

problem resolving也可以分成四个小类型。

2.1 API design. 这类问题是为了更深入的考察你对数据结构的理解与运用。例如LRU cache, insert delete getRandom ALL O(1), design twitter等等。

2.2 Abstraction. 这类问题是考察你能不能把一个相对抽象的问题规约到你熟悉的问题上面。比如skyline problem, int stream find median, cleaning robot等等。

2.3 计算机小程序, 例如thread pool, 爬虫, 日志merge等, random generator等。

2.4 dynamic programming问题。这类问题有点像solid problem resolving. 主要考察你是不是有systematic的方法来降低一个brute force程序的复杂度

这类问题一般都不是很easy的问题, 根据面试官心情, 可能走的很深很难。也可能最后演变成bar raiser.

### 应对策略:

2.1 主要考察你对数据结构的深层次认识。首先请同时确保你理解了题目的意思, 最好能问清条件 例如immutable array max subarray sum, 那数组将来会变吗? 问清这类的问题有助于你写代码前做好重构和测试的准备。其次, 如果你能证明你选择的算法的复杂度, 甚至证明这就是最佳复杂度, 那是一个大大的加分项, 如果不能, 至少你也问问面试官是不是已经满意了再开始写代码。

2.2 这个我自己也头疼, 说实话如果第一次遇见了skyline, 我也不知道能不能搞定。大家有好办法请回复有什么好办法能系统化的解决这类的问题, 我个人觉得很多时候靠灵光一闪。

2.3 这类问题主要看你平时积累, 也是一大类不能通过leetcode练习的问题。临时抱佛脚的话, 我个人推荐java concurrency in practice这本书。

2.4 动态规划, 我不知道为什么很多人害怕动态规划。面试中的动态规划大致分为单向递归 (首或者尾), O(n2) 或者 O(n3) 距离递归, O(mn)递归, 有限定条件的NP (背包)。每种类型听几节课, 懂了基本原理即可。至于贪心和带状态的dp(走道铺砖)一类的dp, 至少我没在面试中遇到过, 因为很难临时造出一道这样的题目, 面试官一般也没这个能力和时间来思考题目是不是严谨。贪心准备下加油站, 迪杰斯特拉, 最小生成树就足够了。

### 3. bar raiser

这类的问题只有当onsite应聘者的数量远远大于head count的时候，或者你前几轮明显超出了电面对你的定位才会发生。其目的是帮助公司选择最优秀的人。对应聘者来说，坏消息是要度过痛苦一小时，好消息是你充分了解这家公司厉害的人有多厉害，能充分展示你的能力，甚至被越级录取也不是不可能。  
bar raiser也是三小类。

23: 25, 喊我去睡觉了, 留下这类问题下次有空再说。大家赏大米啊