想上岸 ⚔ 找上岸

>>>>这次上岸拿微软做了个实验…

一直以来上岸都强调拒绝“哑巴”coding。并提出在现在的IT面试中，算法能力的比重正在逐年降低，取而代之的是更多的交流，以及表达能力的考察。毕竟所有的组都是在找一名合适的同事而不是code monkey。

为了证明我们的观点，这次轮到了微软…我们安排人员参加了近期的hiring event, target 63 64。

这次面试为了**测试**我们的观点，我们遵循以下原则：

- 能不说话尽量不说话。
- 除了BQ，对于算法部分的交流，面试官有任何问题我们尽量少答。
- 要做Clarification，但是做的比较机械，流程化。
- 描述思路，但从不画图。

- Coding部分必须optimal solution，并且在最短时间内写完，bug free。
- 最后的测试部分我们主动跑代码和用例，但尽量不解释为什么跑这些用例。

那么就开始我们这一次的流程吧！

面试是早上8点开始直到12点完成，四轮，每轮1小时。

因为根据NDA，我们将不会透露原题，此次面试几乎所有题目都是原题变种。**我们会在最后总结现在面试变化的趋势。**

01

第一轮　Design File System

面试官是一个印度小哥，口语一般，上来给了一道3秒钟就被我们识破的原题变种：Design File System。

面试官花了约10分钟问了基本的BQ，然后我们就开始正常的讨论题目了。

在开始之前我们问了如下问题:

- Input/ output 类型
- 是否所有的input 都是valid path
- 支持的方法

然后这类题目肯定是难不倒上岸的老司机的，**我们首先抛出使用hashmap。**然后面试官表示这能work。在交流过程中我们没有画任何的case，只是单纯简单描述，尽量做到描述的英语as naïve as possible。

然后我们抛出了使用Trie。然后面试官说能实现一下吗？在接下来的约15-20分钟我们单纯用正常速度写题，最后检查了一遍typo。然后面试官要求测试一下。

当时我们使用的env**自带测试环境，他需要我们自己倒包和进行测试，**我们写了几个简单的用例后，面试官问我们有什么问题，官方的回答了他的废话以后我们

继续了下一轮。

02

第二轮　Reverse Integer

面试官是一个白人小哥，这人和上一个人属于同一组，强烈暗示他们组缺前端。这一轮的BQ比较多，约20分钟，还顺便问我们喜欢做什么。我们说我们前后都能做，无所谓，只要项目有impact就行。于是开始了coding部分。

题目很简单，　reverse 一个数字：Reverse Integer。

我们问了如下问题：

- Input/output
- Input range/size
- Positive/negative

我们知道有overflow问题但没有问，打算在code一次解决。

然后**这类题目在面试的时候肯定属于是挖坑题的**。题目本身简单，但是往深了想，绝对不会只是一道Easy题。

代码时间当然很简单，因为input是Integer，所以注意正负，以及Integer类的溢出问题。五分钟就写完了，大概跑了几个case。可能也没什么好聊的了，面试官要求我们写unit test。我们先写了几个简单的system print然后证明我们是对的，但面试官必须严格要求我们有Junit，所以之后写了一些Assertions。

我们给了全面的test case，正负，溢出，零等。但是注意，我们在之前的交流过程中从来没有交流过如果溢出怎么处理，只是在代码上全部返回0。

这一轮就这么结束了，代码肯定是无懈可击的。

03

第三轮　jump game II

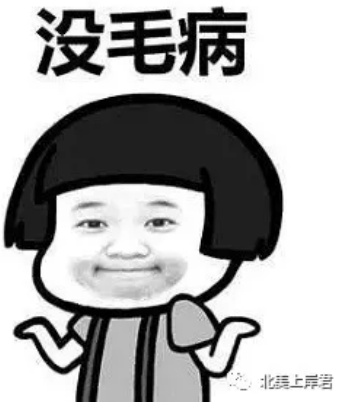
面试官是一个白人小哥，这次换了一个组的后端。这一轮的BQ正常，约10分钟。然后贴心的给我们描述了怎么用coding env。于是开始了coding部分。

题目很简单，上来说有个数组，然后有个起始点，第一感觉就是jump game。然后变了一下其实是jump game II 的变种。

我们问了如下问题：

- Input/output
- 正负数, value range
- 能否回跳
- Array size
- 能否一定跳出

上去我们先说用greedy。然后就大概五分钟写完了。然后这一轮是最快的，跑了三个case，感觉面试官也想下班吃饭了，所以就问了三个问题，大概25分钟就结束面试了。



04

第四轮 minesweeper

面试官是一个印度小哥，是第三位的老板。这一轮的BQ正常，约10分钟。然后就开始了迷之描述。反正就是越短的题目越难。题目是扫雷，但是没有给任何题干，就是我想让你实现一个扫雷：minesweeper。

我们问了如下问题：

- 我们做了很多assumption，比如二维数组里面什么字母代表雷什么字母代表空，什么表示revealed格子等等。
- 我们确定了用户行为，就是用户通过点击一个block能获得什么。
- 我们确定了返回是一个结果的二维数组。
- 确定了一些别的细节用户行为
- 同样我们只描述不画画 (啊，真是非常累！)

题目本身也很简单，做过的同学都知道BFS DFS都行，反正就是模拟一次点击效果然后去周围算一下结果然后标记blocks。

因为这题描述比较长，所以最后写完可能还剩下20分钟。然后面试官看了我们的代码以后没有问额外的问题，只是说，你的input默认有一个2d dashbaord，能不能random生成一下。

我们写了五行代码搞定自动生成dashboard。当然在写之前面试官问我们想怎么生成，我们说别废话看我写:)。写完以后面试官说他懂了。

这一轮也在欢声笑语中最后问了一些问题结束了。

微笑中透露着疲倦

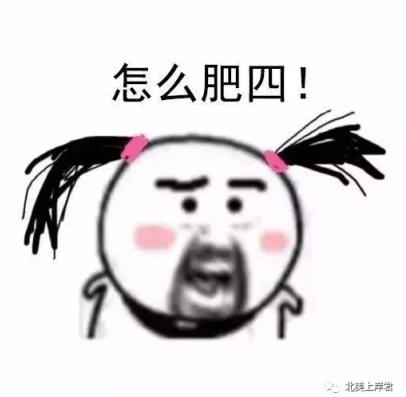
这四轮的流程如果摆到论坛上应该算是一个干货贴了，但在上岸看来这只是起步。接下来我们说说结果。

你们先猜一猜，我们过了几轮，挂了几轮???

En…很好，我们挂了三轮 （solid no x 3 L），另一轮面试官不给feedback。(我猜是第三轮)

我们再申明一下所有代码我们保证bug free和optimal solution…

那为什么我们挂的这么惨？



就在前几周我们和HR交流了一下。HR人非常Nice,交流中我们显然恢复了我们灵活的嘴巴，然后把feedback都套了出来。其中三轮都非常详细，另一轮没有数据。

我们先来看看面试官是怎么说的，我总结了一下下面几个拒点：

- **代码写的work，但不完全是我想要的。**
- **每次当他停顿的时候我试图和他交流，但是我不知道怎么去和他交流，我也不知道他在想什么，最后的代码是work的，但感觉就像是默写出来的，我完全不知道他是怎么思考的。**
- **我问了他一题扫雷，他给的答案就像是leetcode上抄的，写的非常快，也没有思路上的停顿和解释。**
- **我让他写unit test，他只是给了一些简单的system out print， 但是我想要看他写assertion. 虽然最后写了但是我总觉得在交流上我们有**

disconnect。

- **他的代码写的很快很好，就像做过的一样。**
- **我让他翻转integer，他给了一个特别好的答案就像是知道的一样，但这并不是我最初想的。**

Well，　大家可以看出，很大的问题就是交流。但很多人会说，看我们的面经，我们也做了clarification，该说的话也说了，为什么评价这么差呢？

没错，答案就是你的描述没有完整的逻辑思路链。

锁得死内。

如果你仔细思考下人类在碰到未知问题的时候，解决方案一定是从一个方案慢慢优化，或者能够比较快的指出一些核心问题，然后各个击破，最后达到完美的solution。但这次面试过程中，仿佛一切都来得太快。

你熟悉coding很好，但是每一个细节其实都应该跟面试官交流确认，在写代码的过程中，每写一段完整的逻辑其实就应该跟面试官阐述交流并且确定你在right track。

但凡突兀的逻辑，在面试的时候，从面试官的角度看，都意味着你做过这一题或者在现在的VO中，意味着你在抄答案。

敲黑板了！

另外我们最后总结一下最近的面试趋势：

1. 首先，北美IT行业最近肯定在扩招，不论哪家公司，现在都是近3年来最好的机会，不论new grad还是在职跳槽。

2. 因为VO的存在，所以大家都有很大的机会作弊，而且市场上肯定存在这样的现象。就我们的老师在日常面试中也有不少类似发现。所以**题目变种越来越多。**

3. 交流，交流，交流！不要作哑巴coder！必须要think loud and think out of box!

4. 这类面试能力不是一个网课能解决的，必须在**每天**的**实战**中，**让面试官告诉你你哪里错了**，该怎么改进，我们的上岸算法小班就是为此而生。

5. 不论是什么级别的面试，**交流永远是第一位**，从交流中找到方向和答案，哪怕这题你是做过的，也必须要遵守这个逻辑流程。

6. 最后我们想要说将来的面试，能写题已经是一个基本能力了，越来越多的要求会从一个人的综合能力考查。如果还抱着一个算法视频觉得就能上岸，那只能是韭菜中的韭菜了。

认真学习你就是最棒的！