写在20年初的校招面试心得与自学CS经验及找工作分享

我于大三（15年下旬）开始自学CS，并在去年（19年）参加了校招的实习与春招，很幸运地拿到了10来家公司的offer。在这里分享一下自己总结的面试心得与技巧、自学CS的方法与资料、自学CS的历程以及找工作的历程。当然，本人水平有限，且观点局限于个人经历，故有些说法难免会有不妥甚至不正确，欢迎指正。

全文接近4万字，可以根据目录各取所需。

面试心得

2019年春招和秋招，我在中国进行了多场面试，其目的是找一个暑期实习职位和找秋招的正式工作。这是我的个人心得总结。

我实习和秋招都已经面了数家国内的一线大厂（包括腾讯/阿里/头条/百度/拼多多等）和数家外企（包括Optiver/Google/Microsoft/Hulu/Airbnb/Morgan Stanley等），收到过一次拒信。经过一段时间的面试准备与几次面试经历，总结出了一些个人心得，仅供参考。本文的前半段给出了一些我认为比较通用的技巧与心得，以供参考。本文的后半段介绍了自己跨专业学CS的一些经历以及自己找工作过程的流水账，并简单讲述了与找工作期间的心路历程，也可作为自学CS与择业的一个简单参考。

在进行了比较与思考后，我最终选择的公司是Optiver，职位是Low-Latency System Developer，工作地点在上海。文末也会阐述选择的理由。

声明

所有的面试技巧都是建立在一个基础之上：面试者已经具备了相对合格的实力。2018年下半年我在一家创业公司实习，秋招时也面试过一些候选人。在我看来，面试者如果自身基础不扎实、实力不够合格，那看所谓的面经、学习所谓的技巧也意义不大：合格的面试官可以非常轻易地通过一些follow-up问题问出面试者的真实实力。面试技巧和面经固然有意义，但学习技巧和了解面经，**只能帮助有实力的面试者更大程度地发挥出自己的实力 。学习没有捷径可走，nothing replaces hard work.** 希望每一位面试者都能尽早明白这个道理。

另一方面，我身边确实有一些这样的同学：他们相当有实力，但是却因为种种原因无法在面试中展现出自己的全部实力。事实上，不同的企业有不同的面试文化，比如Google的面试官希望面试者能成为一个他愿意一起工作的同事，字节跳动的面试官也许希望面试者是一个数学、算法、coding、工程都不错的全面人才，这样的人才更可能成为一个“能解决问题的人”。但是，作为面试候选人，我们其实没必要去针对各家公司的文化对症下药：应对面试应当有一些共通的要点。在我看来，面试最关键的一点在于面试者要意识到这不仅是一场测试，**更是一次需要充满着沟通与交流的谈话，让面试官认为他/她愿意成为你的同事**，希望每一位面试者都能尽早明白这个道理。

除了上面提到的我认为至关重要的两点以外，面试还有一些其他相对通用的面试技巧和要点。我这篇文章旨在总结一些这方面的东西，希望能够帮助到这样的同学。

编程部分心得

在面试过程中，面试官常常会给出几道算法问题，需要面试者提供思路或写下代码。在大多数公司的面试中，这一部分的表现都非常重要，而对一些外企来说，这部分的表现是具有决定性的（甚至是唯一重要的表现）。对于这部分的准备，首推LeetCode等网站，这里不再赘述。再提几句话，对于一些重视算法问题的公司如Google, hulu, airbnb, 微软, 头条等，不要抱着可能撞到原题的心态去准备，很难撞到原题的，对于这些公司，你需要做的就是反复练习提升自己的能力，而且由于题目较难，需要有较多的训练量。而另一些不是很重视这类问题的公司像阿里、腾讯什么的，则刷一些常见的题目就很可能撞到原题了，而且难度一般不大。因此，根据target公司的不同，可以有不同的准备方式。下面将列举一些其他在面试中我认为比较关键的点。

一个非常简单的例子

这里先给出一个非常简单的问题，下面的关键点将结合这个问题来阐述。该问题为，*计算一棵二叉树的高度*。 简单的实现如下：

```
int getHeightOfBinaryTree(TreeNode* root) {
    if (!root) return 0;
    int left_height = getHeightOfBinaryTree(root->left);
    int right_height = getHeightOfBinaryTree(root->right);
    return max(left_height, right_height) + 1;
}
```

练习白板编程

面试的编程部分往往是白板编程：面试官要么要求在一个类似于Google Doc的地方写代码，要么就是干脆在白纸上写代码。这种情况下coding的体验与平时使用IDE的体验是完全不同的。以Google Doc为例，许多人（比如我）一开始甚至很难写出能编译的代码，更别说一遍写出bug-free的代码了。同时，没了IDE，debug的难度也会大大增加。而在白纸上写代码的难度则还要更进一步。适应白板编程的方法也很简单，只需要足量的练习即可。

问清题目

问清题目至关重要。如果你对面试官的编程问题理解得不清晰，那你应该立刻问一些能帮助你理解的问题。例如：数据范围是多少？这个数组的大小范围是多少？能不能给个样例？如果输入是这个，那输出应该是什么等等。在上面这个简单的问题中，可以问的一个问题是，二叉树的高度是什么（据我所知，高度的定义并非所有教材都一致）？

许多面试官在面试的时候，会故意先抛出一个模糊的问题。实际上，他们希望面试者能够经过一些询问理解问题。在这个过程中，面试者能够展现出自己对**问题的分析能力**以及**沟通的能力**。前者的重要性参见编程珠玑第一章：明确问题，战役就成功了90%。后者的重要性在于，问清题目的这个交流过程与面试者入职之后与同事讨论问题的形式非常类似。显而易见，一个能够很难沟通的面试者也很难成为一个很好沟通的同事。

如果没有问清题目，那会发生什么事情呢？在最坏情况下，面试者可能会花大量时间去解决一个完全错误的问题，面试结果也可想而知。或者运气好些，碰到了一个比较nice的面试官，给一些提示告诉面试者已经进入误区了，但这样不仅会浪费不少珍贵的面试时间，更会降低面试官对面试者的评价。我在面一家公司的时候，面试官给我出了一个题，这个题听上去比较困难，需要用到动态规划才能实现。我当时想，在面试开始阶段就给出一道比较困难的题，这对我来说也太不友好了！于是我询问了一句"数据的范围是什么呢？"面试官告诉我，数组的范围都是0-10的整数。这样的话，这个问题就变成了一个只需要6行代码就可以解决的贪心问题。如果我没有问清这个问题的话，面试的难度显然大大增加。

与面试官确认函数签名

确认了题目之后，我认为合理的做法是先和面试官确认函数签名，也即输入是什么参数，输出是什么参数等等。这一步的代价很低，而且相当重要。第一，这可以告诉面试官，你对函数签名的设计相当重视，而这一点在实际应用中很有价值。第二，这可以进一步帮你确认自己理解了题意。一个合理的函数签名可能就类似于LeetCode题目里的函数签名。上面代码中的签名就是一个比较合理的签名。

设计简单的测试样例

写完了函数签名之后，可以针对函数签名简单地设计一组测试样例（如果面试官之前给了样例的话，也可以直接用面试官给的样例作为测试样例）。设计测试样例地主要有三个目的：一是进一步帮助确认自己对题意的理解没有出偏差；二是告诉面试官自己对测试十分重视；三是提醒自己编码完成的时候测试自己的程序。

与面试官确认思路

在自己有了一个思路之后，一定要和面试官确认这个思路是否合理。你可以给面试官解释你的思路为什么合理，面试官可能会和你讨论其中的一些要点。这样做有几点好处。第一，在解释的过程中，你的思路也会变得更加清晰（面试官充当小黄鸭）。第二，这也展现出你对沟通的重视性。第三，可能也是最重要的一点是，如果你的思路不正确，nice的面试官会提示你甚至直接指出错误所在，这样你至少不会在一个错误的思路上耽误太多时间。**切忌有了思路之后，不与面试官交流直接写代码**。尤其需要指出的是，如果你的思路对数据有什么假设，或者需要**修改输入数据**，那一定要和面试官确认这样的做法是合理的。

如果你认为这个问题与某个经典的问题思路一致，或者可以用到某个经典的算法，那么就直接点出来。例如计算二叉树的高度，实际上是一个后序遍历，那么可以直接点出来。

抓住面试官给的提示

有的时候一道题难度比较大，候选人一时想不到最优的思路，或者目前提出的思路是错误的，那合格的面试官可能会给一些提示帮助候选人思考，这时候候选人一定要抓住面试官给的提示。以上文给的例子为例，如果候选人想不到思路，那面试官可能会提示：“你觉得一棵树的高度与它的左右子树的高度可能有什么样的关系？”这就提示候选人可以用递归的方式来解决问题。抓住提示是很重要的：一方面，面试官给的提示很可能可以帮助你想到正确/最优的思路；另一方面，这其实也是双方能够进行不错的沟通的体现。

确认边界处理

在开始写代码以前或者是写代码的过程中，一定要思考代码的边界条件。最典型的边界条件有：数据是否会溢出？指针是否可能为空？链表是不是可能存在环？数组的长度是不是零？输入的数据会不会完全不符合题意的要求？在示例中，边界条件就是当结点指针为空时，高度应该是0。当你察觉到边界条件存在时，就可以询问面试官处理方式，或者直接告诉面试官你认为什么样的处理方式是合理的。对边界条件的处理在开发软件时也异常重要。忽视了一个边界条件，就会对程序鲁棒性造成极大的影响，可能直接造成巨大经济损失甚至是人员伤亡。

代码中使用可读性高的变量名和函数名

在写代码的时候，尽量使用可读性较高的函数名和变量名。例如，要计算二叉树的深度，函数签名可以为`int getHeightOfBinaryTree(TreeNode* root)`入参就叫`root`（而非`node`）。递归时，左子树的高度的变量名可以叫`left_height`。诸如此类。这样操作的主要目的也是让面试官看到你良好的编码习惯。

写代码过程中保持与面试官交流

实现算法的过程中，切忌闷头狂写而不与面试官交流。实际上，在写一些关键代码的时候，你完全可以告诉面试官你在实现什么功能。同样如前例计算二叉树深度，那你就可以告诉面试官，`int left_height = getHeightOfBinaryTree(root->left)`是在计算左子树的高度（良好的函数名和变量名其实也让这行代码不言自明），而`int root_height= max(left_height, right_height) + 1`则是根据左子树和右子树的高度计算当前根节点的高度。

当然了，在这个简单的示例中，交流或许显得不是那么重要，但是在一些复杂的问题中交流可能会非常重要。例如，示例的follow-up是请不用递归实现同样的功能，或者更进一步，请用常数空间实现同样的功能。在这样的问题中（代码可能长达数十行），交流就至关重要了。面试官需要和你交流来理解你的思路与状态，你同样需要交流来理清思路。这种写代码过程中的交流也是正式工作时非常重要的能力。当然了，这样的交流也不必过于频繁，否则也可能影响自己的编码状态。

写完代码后主动测试

在你写完代码之后，不要急着告诉面试官你已经写完了。最好先手动跑一个/数个简单的样例。注意跑这个样例的过程要让面试官可以看见并轻易地理解，这常常是需要一些练习的。例如，我在Google Doc上跑样例的做法是，在屏幕上写出中间变量的当前取值，然后用鼠标光标告诉面试官现在程序跑到了哪一行代码，当前各个变量的取值是多少等等。主动测试的好处有很多。第一，这告诉面试官你很重视测试，而测试在实际生产中是非常非常重要的。第二，一个简单的样例常常可以找出不少类似于typo这样的小错误。第三，如果你的样例给得不错，那你甚至能够借助这个样例找到程序中的bug并纠正它，这总是要好过面试官发现并告诉你程序中存在着bug。主动测试时，你也可以确认你的程序可以很好地处理边界数据。

我自己在面一家外企的时候，主动测试的习惯就给我带了很大的回报。当时我写了一段不算复杂的程序（约20行左右），可是因为情绪紧张，程序中包含了一个相对隐蔽的bug。写完之后，我习惯性地跑了一个简单的样例，这花了我大约3分钟的时间，但却让我注意到了那个bug。我赶紧修复了这个bug。到了面试的提问环节，我问面试官本场面试中我表现最好的一点是什么。他告诉我：”是你通过一个样例发现了你的bug。实际上，在你写出了那段代码的时候我就注意到了这个bug，当时我在犹豫要不要提醒你。而你随即开始了测试并找到了这个bug。“这场面试的结果是，在面试结束半小时左右我就收到了通过面试的消息。

主动给出算法的复杂度

在写完代码之后，应当主动分析自己算法的时间与空间复杂度。一方面，这样可以展示自己扎实的算法基础。另一方面，这也可以告诉面试官自己有这方面的意识。当然了，如果复杂度分析的有误，那这个分析也可能会成为一个减分项。

讨论算法的trade-off

有些时候，题目的解法可能存在一些trade-off。最常见的就是时间-空间的trade-off，当然有时也会有一些其他的trade-off。如果意识到了这道题目存在trade-off，那么可以主动地与面试官聊trade-off，让他/她知道你的思考过程与选择。

计算机基础部分心得

面经的使用

计算机基础部分的内容包括数据结构、操作系统、编程语言、计算机网络等等。这部分的准备很大程度上是需要一些扎实的基础的，再配合一些面试公司的面经。有些同学想仅仅靠看面经就应付过去，我可以说大多数情况下是不太可能的。有经验和水平的面试官可以轻易地通过几个follow-up问题来判断出来这名候选者是不是靠面经回答出来前面的问题的。当然了，面经对于这块内容仍然是非常有价值的，但阅读面经的时候要注意，并不能仅仅看一下某道题目的答案就够了，而是要看这个题目考察的是哪一块的知识，这一块知识自己有没有遗忘的、生疏的、不扎实的，如果有的话要去做相应的准备。**面经是告诉你这家公司面试的时候喜欢问哪些知识，而不是告诉你他们喜欢问哪些特定的问题**，虽然有的时候有些高频问题确实可能在你的面试中出现。

抓住面试官想问的点

有些同学被问到一些自己会的基础知识的时候会特别激动，想抓住这个机会表现自己，就会事无巨细地回答一波。我个人认为，如果是基础知识的话，其实不用回答得特别详细，说出一些面试官想问的关键要点就可以了。有时候不一定能判断出来面试官想问的要点，这也不要紧，就说一些自己认为是关键的要点，然后等着面试管继续问follow-up就可以了。这里举一个简单的例子，如果面试官问进程与线程的区别，那么简单地说线程是调度的最小单位，同一个进程的线程共享地址空间，容易有线程安全问题；进程是多数资源分配的最小单位，所以进程的地址空间都是独立的，资源安全问题相对较少。回答到这个份上就差不多够了，然后等面试官继续问follow-up，而不需要去解释为什么线程会有安全问题等。之所以建议这么做，是因为对于有些公司，面试时间是有限制的（例如Google, hulu等），所以面试时间是很宝贵的，你应该用这珍贵的时间去展示自己的优势，而不是说一些绝大部分人都懂的trivial

的知识。当然了，有经验/不nice的面试官可能会打断你，问他自己感兴趣想问的东西，但如果你运气不好恰好面试官没啥经验或者不喜欢打断人，那这样浪费宝贵的时间是很可惜的。

说出自己的insight

如果针对某个问题有自己一些独到的见解，或者是这个知识在很多教科书上可能看不到，很多同学也不一定知道，那么在回答问题的时候说出自己的这个insight，当然前提是自己的说法是有道理的。这里举一个简单的例子，比如一个面试问题是，可以用什么数据结构来实现队列。回答可以说是链表，接着可以补一句但是链表实现队列的性能不一定很好，因为链表节点的地址空间不是连续的，对cache不友好（小问题：那么如何改进这一点呢？）。这种知识其实是有一些经验的人或者基础扎实的人都知道的，不算是什么难点，但作为应届生，能直接说出这一点还是可能会让面试官觉得这个候选人基础不错。

结合自己的使用经验阐述

如果在某些基础问题上自己有一些实际经验，那么可以结合自己的经验来回答，这样会让面试官觉得这个候选人不仅基础扎实、经验丰富，而且学以致用、分析问题的能力也挺强的。

这里举一个简单的例子，比如面试官问hash table处理冲突有哪些常用的方式，各有什么优缺点。那么可以回答常用的有线性探测和拉链法两种。如果自己有相应的经验，那么就可以结合经验谈谈优缺点，例如线性探测在实际使用的时候常常需要空间开得比较大，hash table的装载因子需要维持一个一直比较小的状态（比如25%-50%这样），否则的话性能就会很差，因为查询和插入都会频繁地进行长距离的线性探测。而拉链法对空间的利用效率就会比较高。在提供足够的空间的时候，按经验线性探测会比拉链法快很多，比如之前做了个项目，在满足空间条件的时候线性探测会快7倍左右（这是在结合经验谈），原因是线性探测比拉链法对cache更友好（这是基础知识）。

类似于这样的回答方式，可以让面试官留下一个很好的印象，认为这位候选人的整体素质也非常出色。

项目部分心得

简要介绍项目背景

如果面试官是很熟悉这个领域、这类项目的人，那么你可以make some assumptions，即不需要做多少背景介绍。否则的话，还是建议简单谈一下自己项目的背景是什么。这是因为在不同的背景下，同一种功能的实现常常会有不同的选择。这样的背景介绍能帮助面试官更好地理解这个项目，以及大概理解一些实现的选择。背景主要包括场景、问题定义、需求、自己负责的部分扮演的角色等等。

介绍项目的approach

介绍完项目背景后，需要简单介绍一下自己这个项目的解决方案。解决方案主要是使用了什么技术、什么工具、怎么样的实现等等。需要注意的是，介绍解决方案的时候最好要结合场景一起说，否则会缺乏一些说服力。

这里仍然举个简单的例子。例如做深度学习的落地，深度学习框架选用的是腾讯的ncnn，那么最好说一下因为场景是嵌入式arm设备，且没有显卡，在这种场景下，ncnn做了很多指令级的优化，速度会更快。

指出项目中的困难点和解决方案

针对项目中的困难点要特别认真地谈论一下，需要介绍为什么这个点是个困难点，解决方案大致是什么样的思路，为什么要这样去设计解决方案，最终达成了一个什么样的效果。如果一个候选人能展示出准确的痛点、瓶颈分析能力，并且能提出合理的解决方案的能力，那我相信面试官对他的评价会大大提升。

这里同样举一个简单的例子。例如做数据库实现，项目中有一个问题是数据库太大，不可能放到内存里，但如果都放硬盘的话又太慢，这是项目中的一个困难点。解决困难点的关键是同时利用内存的速度优势与硬盘的容量优势，设计一个存储分层模型。做实验观察到90%的针对数据库的查询仅集中在10%的数据上。那么解决方案可以是设计一个冷热分离的模型，仅仅在内存中存储一些热（即查询频繁）的数据，而将冷（即查询频率很低）的数据存在硬盘上，同时设定一定的策略定期做冷热数据替换。经过这样的设计之后，数据库的查询速度提升了30倍。

论文部分心得

简要介绍自己research的背景

与项目不同，很多冷门的research的背景面试官往往是不了解的，所以常常需要做相对详细一些的背景介绍。

像做talk一样介绍一遍自己的论文

在面试之前，可以先自己精细地准备一下论文的介绍。假设这个面试官对这个领域不熟悉，如何才能让他在较短时间理解这个研究领域，大概明白领域的痛点，并理解你的论文的思路、解决方案与重要性呢?

模拟面试

强烈建议在面试之前找人模拟一下，并让对方给你一些反馈。这样能够大大降低紧张感，熟悉面试流程并提高面试表现。当然了，还有一个重要的方式就是多多投递，先拿一些自己不target的公司练练手，磨练自己的心态与面试技巧。

面试大忌

我也曾当过几次面试官，也参加过一些面试并了解过其他人的面试情况，这里简单说几条面试大忌，一定要避免犯的错误。

不懂装懂

对自己不懂的东西（甚至是没有十成把握的东西），一定要诚实地说出来，千万不要不懂装懂。我把这一点放在最前面，是因为我作为面试官以及平时与人讨论技术的时候，就非常讨厌别人不懂装懂。面试官的水平往往比你高很多，一下子就能判断出来你是真懂还是装懂。所以，碰到自己不懂或者没把握的问题，我建议直接告诉面试官说这个问题我没把握，不是很懂。但如果你有一些思路的话，可以接着说“虽然我不太懂，但是可以试着说一下”，这就可以变成一个展示你解决问题分析问题能力的机会了。而如果你的分析思路很合理，得出的结论也大差不差，那甚至可以很大程度地提升面试官对你的评价。

狂傲不羁

面试的时候，人要有自信，但是态度一定要平和并且尊重面试官，切不可恃才傲物、狂傲不羁。有一些公司会非常看重这一点，如果你给面试官留下了不好沟通的印象，那往往是一票否决。但面试的时候，偶尔也会碰到面试官不是很懂犯错误的情况（比如国内的一些大厂），这个时候你最好是平和地去与面试官讨论，如果他坚持不肯认错，那你也不要去较真，否则的话可能你面试就挂了。有一种情况是可以去与面试官较真的，那就是你完全不在乎这家公司的offer，这时候你可以放开了较真哈哈哈。另一方

面，当你面试一家公司或者一个组，碰到面试官不懂装懂又不肯认错的时候，你也得考虑一下这个组是不是值得你去。

远远达不到面试官对自己的期望

在面试之前，面试官往往会根据候选人的Profile而对候选人心里有一个大致的期望值。在面试的时候，面试官会根据候选人的表现评分，而这个评分实际上有很大一部分是与对候选人的期望值有关。如果一个候选人的Profile很好、简历里吹得天花乱坠，那面试官对这位候选人的期望值就会很高；结果面试的时候一问细节三不知，远远达不到面试官对候选人的期望。在这种情况下，面试官往往会给一个非常差的评价。这种case或许比较极端，但候选人面试展示的水平（以阿里的的评级为例，假设是A）与自己的Profile（假设是A+甚至阿里星）不match是相对常见的现象。这时候，面试官甚至有可能给候选人一个比候选人真实水平还差的评价（譬如B+）。这也是候选人为什么要对自己简历上的东西了如指掌，能够做到即使被狂轰滥炸也能谈笑间应对的一个原因。其实上文所提到的，当面试的时候被问到自己不会或者不确定的问题的时候要先诚实地告诉面试官，之后再靠自己的common sense、逻辑思维以及其他的一些知识来推理出一个相对合理的答案。这也是一种对面试官的期望值的管理。先诚实地告诉面试官来降低他对自己的期望值（譬如降低为B+），然后再展现自己其他的能力超出这个期望值（譬如展现出A的水平），面试官说不定反而更容易appreciate你的表现。

心态

面试总会有运气成分与偶然性，放平心态，不要因为害怕被拒就不敢投递，也不要因为患得患失而在面试的时候十分紧张。在面试中尽量让自己自然、轻松。当然，一些轻微地紧张有时是可以让自己发挥更好的，但是要适度，切不可紧张过头。面试中即使有些内容答得不好，也不要当场就心态崩盘，要沉着应付。当自己没有什么思路的时候也不要太慌，可以试着从基本的地方开始分析。例如做算法题，可以分析一些toy example，有时候能获得一些思路。回答CS基础题、system design等题目也可以从基础的地方开始分析，甚至是与面试官一起一步一步得出结果。我自己在参加一次面试的时候，一道算法题问清楚题目就花了10多分钟，然后10多分钟没有思路，同时面试官还在给我施加一定的压力。要知道面试总共就45分钟，这样的表现属于非常糟糕的了。所幸我当时稳住了心态，利用一个toy example得到了正确的思路，写出了bug-free的代码，最后还是让面试官相当满意。

最后

需要说明的是，每个人都有自己的面试风格，很多面试官也会有自己的喜好，所以没有一套universal的面试方案。本文提到的一些技巧什么的，主要是我自己总结出来适用于自己的风格与方案，读者完全可以根据自己的实际情况与面试时候的感受来调整。举个例子，本文提到的编程部分心得，主要是针对Google这样的公司的算法题部分。我自己也有过一些面试经历，面试官非常不喜欢候选人在写代码的时候与他交流，甚至会在你写代码的时候自己去做别的事情 :(这时候你最好就乖乖闭嘴，把代码写出来即可:) 因此，也希望各位因时制宜，因地制宜，结合实际情况来进行面试。最后祝大家都能有满意的offer~

番外篇：找工作的流水账与心路历程

本文的这一部分将以流水账的形式简单讲讲学CS这几年来的一些经历，以及找工作的流水账与心路历程。

背景介绍与CS学习历程

我总结的学习方式

在学习CS初期我走了很多弯路，相信了一些不合理（或至少是不适合我）的所谓"编程入门指南"。之后经过自己的摸索，到现在总结出了一套适合自己的学习方案。这里首先给出我的学习方案以供参考。使用这个学习方案还需要不错的英语水平，以后我有时间的话也许也会写一下自己学习英语的经验与心得。

一个CS领域的学习过程大致可以分为以下三个阶段。当然，有的时候不同阶段是可以迭代地进行，例如开始科研之后发现自己还缺少了某些基础知识，那可以再进行基础知识的学习。同时，在学习中，很重要的一个指导思想是要获得**监督信息与正反馈**。三个阶段如下：

1. 如果想学某门课程知识，那就找国外名校（主要是MIT/Stanford/CMU/Berkeley）有课程录像的对应课程，假装自己真的在上课一般地按照课程安排上课、看阅读材料并**完成作业。完成作业是极其重要的**，因为在这个过程中你会获得大量的监督信息，来指导你发现自己哪些地方学得不扎实；而如果仅仅看视频与阅读材料的话，常常会产生自己已经学懂了的错觉。另一方面，许多编程的作业完成之后也可以带来**成就感与正反馈，支持着自己的学习动力。**
2. 之后，找自己感兴趣的领域。根据是做开发还是做算法，做研究还是做工程，第二阶段可以分成两种不同的方式。

   a) 做工程：找一些适合练手的项目，自己实现与重构，并对照他人的实现方案。实际上如果第一阶段认真完成了好课程的作业的话，那么已经算是完成了很多练手的项目了。

   b) 做科研：了解这个领域的经典方法与最新方法，并**复现这些方法**，做实验比较结果，同时以论文的要求写报告分析结果，这个其实基本上就是国外很多课的project。**复现好的论文是极其重要的，因为复现可以获得监督信息。**在真正入门一个领域之前，人们常常会产生自己已经读懂了论文细节的错觉，实际上如果你没有能力复现一篇论文，你就不能说自己已经完全读懂了这篇论文。另外，论文中往往不会给出全部的细节，但这些细节并不trivial，需要你在复现论文的过程中自己发现体会。尤其是在系统相关的领域，**论文中一句带过的设计细节往往也蕴含着insight**，而这在你复现的时候会有更深的体会。另外，复现论文也能带来一些正反馈。复现完论文做实验的报告也是很重要的，这个整理的过程会逼迫你去进行一些更深层次的思考，整理的实验结果也可以供以后随时查阅。
3. 同样，第三阶段也可以分成两种不同的方式。

   a) 做工程：可以考虑在GitHub上找一些好的开源的项目，读源代码，并且帮助社区进行开发。有条件的话，可以去实习。

   b) 做科研：之后就是阅读论文、跟踪最新的成果、提出想法并撰写提交论文了。

需要指出的是，这套方案不一定适用于所有人，仅仅是我摸索出来的适合自己的方法。这里再多说一句，在整个学习的过程中，一定要**多提出问题**。例如刚开始阅读一些经典论文的时候，可以逼自己去提出5个与论文相关的问题，而且一定是那种自己提出来之前不知道答案的问题，再试着自己解答。中国人造词说学问学问，只有学没有问的话是一定不够的。

这里再简单说一下如何找到合适的课程。我一般直接找MIT/Stanford/CMU/Berkeley（四大）的有视频的课程。MIT的课程很多在OCW有收录，可以在OCW里直接搜索。更常用的方式大概有两种，其一是在Google上搜学校名字+课程的名字找到课程主页，然后就可以跟这门课程了。例如想学操作系统，就搜MIT Operating System即可。第二种是在Youtube上搜学校名字+课程，很多有录像的课程可以这样搜到。除了这些名校的课程以外，很多MOOC的课程也是值得一看的。这里再简单给出我对各大院校/MOOC课程质量的评价：

主要的MOOC平台有Coursera, Edx和Udacity。Coursera算是三大mooc平台做得比较成功的了，课很多，有好课也有一般的课，鱼龙混杂。课程类型偏学院风。Edx我以前看的时候是平均质量最高的，但那时候完全不商业化，课很少，学院风。Udacity课也不错，挺多的质量挺高，课程是工业风。MOOC的好处是针对自学的人有优化，但坏处是课程难度普遍较低，适合零基础入门的时候用。四大院校的真实课程的难度与质量往往会高好几个档次。其中MIT的课程质量在我看来是最高的，公开的课程也多。Berkeley的也不错，但是公开的课程相对少。CMU的课程质量也还可以。Stanford的课程的讲课质量感觉可能会低一些>_<，因为很多是PhD TA上课，感觉比很多经验丰富的Professor还是有一些差距。如

果实在找不到有视频的课程资源，那也只好不看视频，直接看课件、reading材料和写作业了。这样往往吃力一些，效果差一些，但也能学到不少东西。

CS学习历程

这段非常冗长，可以直接跳过。

我是根不正苗不红的浙大CS小硕。高考填志愿的时候，突然有一些家国情怀，选择去西北工业大学航天学院学习航天，立志航天报国，也正是因此自己的计算机基础异常不扎实:(本科的时候与CS相关的课程只有一门C++程序设计，自己学得也算是班上最认真的之一了，还刷了100来道POJ来巩固。当时任课老师劝我参加学校的ACM集训，我却中二地认为coding这种事情只不过是实现航天的工具而已，就没去参加，错过了一个亿。后来到了大三开始接触一些系里一些项目以及七七八八的事情的时候，我突然意识到他们做的航天项目和自己想的不太一样，航天系统里的人也与自己想得不太一样，遂萌生退意。

于是开始自学CS。因为周围没有认识学CS的人，就去看知乎上大家的推荐，大家都推荐从SICP和CLRS入门，于是我就去看SICP和CLRS并努力做习题，而且当时太年轻，看的还是翻译版的。看到后面怀疑自己完全就是弱智，别人的入门书我怎么就学不懂呢...其实有几个原因，一方面是这两门书挺难的，不见得适合入门；另一方面是翻译版翻译得也不是很好。可以说自己刚开始学CS的时候走了不少弯路。后来看到Coursera上有开Princeton的Algorithm课，就去把那个课跟了一遍，完成了习题。这时候才终于感觉原来自己还是能学会一些东西的:) 又把Edx上MIT的6.001x跟完了，还看了一些Harvard的CS50课程的内容，这时候终于感觉自己算是开始学CS了。

那门课跟完之后，AlphaGO的事情开始刷屏。因为自己之前对围棋有些兴趣，也自学了一些，知道李世石是什么水准的棋手，于是就关注了AlphaGO与小李的比赛。本以为小李会轻松击败AlphaGO，没想到AlphaGO把小李吊锤了一通，很是震撼又让人感到excited，有一种看到世界的大幕缓缓拉开的感觉。也是在那时听到了人工智能(AI)这个词，觉得哇好高大上又有趣，就萌生了做AI方向的想法。

然后就去找了《Artificial Intelligence: a Modern Approach》看。结果又是一本垃圾翻译书，国防科大的一些老师翻译的，我怀疑我用Google Translate都能比他们翻译得好...后来我终于开窍了，买了本影印的英文版看。这本书写得挺有意思的，容易读懂又不失深度，就是太厚了有1000多页，读了我很久，读到后面晕乎乎的。当时看了最有印象的几段有：启发式搜索A*，书上我记得还将围棋归在这一类，说AI围棋距离人类职业水平仍有较大距离，短时间内很难得到人类顶尖水平:)；用一阶逻辑从数据与规则中学习新的规则（也就是规则学习），当看到这样的程序证明了许多数学定理，有的证明甚至比原始证明更优雅的时候尤其激动与兴奋，不过后来真的开始学AI/ML之后发现这已经完全是个dead area；机器人使用强化学习来学习走迷宫和闯关游戏；用MM做NLP，HMM来做POS Tagging等等。不过要说明的是，这本书以现在的眼光来看已经有些过时了，而且里面大部分的内容与现在的研究、应用领域关系不大，不是特别建议阅读了。

看完那本书后，调研发现大家都在做机器学习(ML)，就开始自学一些ML的知识。大三的暑假先从Coursera上Andrew NG课程入门，这门课是一门挺不错的课，我跟完并完成了习题，学习的过程中觉得ML真有趣又有用。然后开始读一些相关的书，先看了Tom Mitchell的那本《机器学习》，这也是一本挺不错的教材（虽然我又看的翻译版，晕乎乎的），不过也有些过时了。看完之后注意到周志华老师有本新出的《机器学习》，还有本出名的书是李航的《统计学习方法》，我就去买了这两本看，不过也只是很快地扫了一遍 。周志华老师的书写得还是挺认真的；李航的书的话我觉得太干了，不适合用来学习，适合用来复习。草草过了一遍两本书之后，又看了张学工老师的《模式识别》，不过后来发现这本书的内容与《Pattern Classification》非常接近，还是推荐直接去看PC。之后开始看《Elements of Statistical Learning》（ESL）。但是ESL对当时的我来说有点太难了，很多都看不懂。而且我发现自己线性代数的基础似乎不够扎实，就先学了MIT的18.06，再继续去看ESL。值得一提的是，MIT的18.06是一门非常好的课，很适合作为对自己线代知识的拨乱反正，消除一些国内垃圾教材带来的坏毛病。学完之后继续看ESL，胡乱看到unsupervised learning之后，开始写research proposal申请日本的大学。

这里要说一下，我之前一直比较向往能出国学习，但是美国的master实在太贵了，家里没钱上不起，CS PhD我又不可能申得上，我就转而准备申请日本的学校。日本的学校学费很便宜，一年也就2w人民币左右，东京的话生活费一年在8w左右，但是日本的master很多能申请到奖学金，至不济也能打工赚钱。我本来想走他们的G30计划，这是一个不需要日语成绩只需要英语成绩，并提供全奖的项目，结果

那个项目那年因为日本扩充军费而没有资金了……为此我还自学了一段时间的日语去考了N1，也考了托福(110)。日本的教授在申请中往往具有绝对的权力，说要你就要你，所以陶瓷在申请日本的大学中极为重要。陶瓷的方式往往是写一个研究计划书，发给教授，教授会判断你的基础、能力、vision等等。总之，我在大三的暑假学了一些机器学习知识之后就开始写research proposal了，当时还不懂深度学习，写的是半监督学习，主要是对以前的半监督学习（self-training, co-training那套）的工作的简要介绍，并写了一些自己浅陋的看法。现在看来，那时候的水平自然是非常差劲，对机器学习并没有形成一个整体的认识，也没有自己的insight；research上完全没有上路，对领域未来发展的看法自然也是错漏百出，不值一提。暑假结束之后，9月初的时候调研了很久教授的信息，看了一些他们最近的工作，并找了三个不同大学的教授发了套磁信。不过非常幸运的是三个教授都给了比较积极的回复，进行了一些交流后，其中两个教授直接表示愿意提供funding :) 我想我当时吸引到他们的可能是本科还可以的GPA，以及出色的语言成绩?

这时候家里出了一些变故，然后我的姐姐也告诉我说自己想出国的想法给父母造成了很大的压力和负担。经过了一晚上的仔细思考，我决定放弃出国准备考研浙大，主要原因是浙大离家比较近，照顾家里的事情也会方便一些。但是我不确定自己能不能考得上研，毕竟准备时间很短只有3个月左右，我也没有什么CS基础。第二天起床之后，我借了同学的考研真题做了一下前一年（2016）的数学卷和英语卷，数学的话卷面大概120分左右吧，英语的话除掉作文翻译大概是扣了5分左右。当时就感觉3个月左右就努力突击专业课，应该有机会考上。遂写了邮件告诉日本的教授们自己因为一些原因不打算去了，并买了一些考研的书开始准备考研。

准备考研的过程是漫长且枯燥乏味的。除了中间有5天回老家参加姐姐的婚礼以外，我每天都能保证9-10小时左右的有效学习时间。我的时间分配策略很简单：大部分时间（也许70%-80%）都投入到专业课的学习上，花一些时间在政治上，数学则定期做一套真题，而英语则一点时间都不花。总得来说这个策略是合理的，当时的失误之处在于学习专业课的资料选择问题。我觉得需要针对考试准备，就使用了国内的教材进行学习，事实证明这个选择是非常错误的，我不仅学得很吃力，最后专业课考得也很低，而且计算机基础也不是很扎实，后面还是花了大量的时间用四大的课程去补。总之呢，就这样日复一日地学习与上课。值得一提的是，我们专业大四上仍然有很多会点名的专业课:) 我在课上就一直看政治的内容，也因此有些课分数考得很低，让我本科的GPA最终变得很难看。终于，考研的日子到来了，我并不很紧张地上了考场，发现我考位左右两个考计算机的同学都弃考了。考场上觉得专业课是真的难，其他的科目都普普通通。按部就班地考完之后，就开始了漫长的等待成绩的生活。在等待成绩期间，我开始刷浙大的PAT，即考研的上机考试，主要是按照胡凡学长的《算法笔记》刷。一开始我还不是特别认真，一天大概只做两三个小时的机试题吧，剩下的时间去学一些其他的东西，例如Stanford的CS229。后来考研成绩出来了，我发现自己的初试成绩排名不靠前，这样如果机试考得很低的话就有可能考不上研了。为了让自己的机试成绩尽量高，我把每天刷题的时间提高到了8-10小时左右，这个状态维持了一个月左右，把历年PAT题目（大概100多道？）来来回回刷了两三遍，这才有了一定的信心。机试的考试来临了，我在西安交通大学的考场参加考试，考场里总共也就10来个人。题目非常简单，应该是历年最简单的一次。我顺利地秒完所有的题目之后，就交卷出场了。

之后就是准备考研面试与找导师了。我并没有认识在浙大计算机学院的同学，只能在网上搜一些相关资料并咨询一些前几年考研的学长学姐，按照他们的推荐联系了几名老师。当时联系的老师们基本都给了回复，约我前去实验室面试。我有一些机器学习的基础，除了本科不是CS这一个致命伤以外其他的条件也不算很差，所以也收获了一些offer。同一个时间，考研的复试面试也来了，我并不紧张地参加完了学院的面试，并取得了大概是前三的面试成绩吧。最后经过一些时间的考虑，参考学长学姐的意见和网上的一些帖子选择了一个实验室的老师。

选导师结束之后我回到了本科学校，开始做毕设，毕设课题是在一个叫NAO的机器人上做一些与机器学习相关的事情。因为选择的实验室看上去似乎与数据库相关，我就找了Stanford的CS145课程（没有课程录像）的课件看了一下。这门课程的学习经历让我意识到课程的录像是非常重要的，在此之后我找课程学习的时候一定优先挑选有课程录像的课。学了点数据库之后，我补了点概率论的知识（Harvard Stat110），然后开始学习Stanford的CS231n课程。这门课让我首次接触到deep learning，感受到计算机视觉CV（学习）的有趣之处以及deep learning的强大与玄学。我比较认真地看完了视频并做完了课程作业之后，决定毕设做一个机器人上简单的目标检测（object detection)。内容设定为机器人通过手眼相机检测到目标之后，根据目标位置抓取目标。由于当时根本没有显卡，只有一台i3的小破笔记本，我挑选了YOLO v2 tiny作为检测的骨干网络，并且使用了Github上的一个基于TensorFlow的YOLO

项目。显然我的计算资源不支持我从头训练网络，我只能找了一个pretrain的网络finetune。当时那个YOLO项目还不支持finetune，我就自己看TensorFlow的文档瞎改代码，改出了个支持finetune的版本。然后自己收集了一些图片，标注了一些数据之后开始finetune。由于机器太差，区区百来张图片的finetune就让我费劲了力气：单组参数的finetune大约需要48h才能完成训练，而且训练过程中电脑会非常卡。我大概花了两周的时间才终于训出了一个勉强可用的网络。在此期间由于电脑不能用，我买了本《Information Theory, Inference, and Learning》开始看，也做一些里面能用笔算的习题。训完了网络之后，我在机器人做了一些实验，并录了一些demo，然后写完了毕业论文。暑假学车，利用空余的时间学了几门课，记得有Stanford的CS41 Python Language，还有MIT的6.006 Intro to Algorithms。

需要说明的是，直到这个阶段，我对学习/CS/机器学习/做研究都还没真正的入门。现在回忆起来已经有点模糊了，但我大致记得当时的学习并不"主动"，纯粹是"被动"地理解书上写的内容、公式推导和代码等等。现在我认为真正的学习应该是**非常"主动"地去进行**，看书、论文的时候要对框架和motivation有一个清晰的把握，知道/理解算法这样设计的目的是什么，并要相对频繁地去ask some questions，批判地进行阅读与消化。但可惜当时的自己还差得很远。另一方面，自己习题、coding做得太少，没有获得足够的**监督信息**，自以为自己学懂了，但其实只学了点皮毛。好在那时候还算是通过一些课程的作业获得了很多**正反馈**，支持着我的学习热情。

我一开始对能够进入浙大计算机学院是很有一些激动的心情的，也破天荒地去积极认识了很多同学，还认真制定了课程计划等等。但开学之后，我发现研究生的课程质量比我想象中差了很多。好在蔡登老师的机器学习课程让我感觉非常好，我认为蔡老师的授课水平和这门课的作业都是world-class的。可惜我运气不好，没能选上这门课，而且蔡老师怕TA压力太大，不给增加课程容量。我一开始就只能蹭课听，坐在最前面，学得很认真，课上蔡老师的问题也总是积极地第一个回答，最后竟然让蔡老师破例地为我签了条子，选上了这门课:) 托这门课的福，我的机器学习基础扎实了很多，这门课也取得了不错的成绩。值得一提的是，这件事情也成了后来我成为蔡老师学生的契机。我还担任了这门课接下来的三任TA，在此期间我与另一位TA一起将之前的MATLAB作业改写成了Python的作业。后来这门课要扩充一倍的新内容，我还与蔡老师一起设计了新的课程内容，并狗尾续貂地再出了四份对应新内容的作业。只做了这么一点微小的工作，非常惭愧。

其他还有两门课给自己留下了一些印象。一门是潘纲老师与章国锋老师合上的《计算机视觉》。课程project我做的是preserve information in style transfer，大概就是说image style transfer中会丢失很多信息，于是就引入了一些正则项来保留部分原始图片中一些可能有用的信息。这个project是与室友一起做的，一开始做得很挣扎，主要原因是自己的Laptop上transfer一张图片就要一个晚上，所以做实验非常缓慢。后来得知实验室有一张1080Ti，就与实验室同学轮流使用。这也是我第一次亲眼见识到显卡的强大：竟然能在几分钟内就完成一张图片的style transfer，太强了！后来课程做pre的时候，两位老师建议我们可以把工作整理一下投一篇paper，但我俩觉得肯定中不了好的会议，就没去写。第二门课是钱徽老师的《凸优化引论》，这门课印象深刻的主要是教材：Nesterov 的《Introductory Lectures on Convex Optimization》。这本"Introductory"的教材让我久违地感受到了被支配的恐惧，里面各种反（我的）直觉的推导看得我头大，也没有任何motivation说明，而是那种苏联式教材的典型风格：就是硬推。钱徽老师说这本学懂之后，看paper会比较轻松。我抱着对他的信任一顿硬啃，后来竟然渐渐学懂了一点。课程作业是挑几篇paper看，我挑了几篇Zeyuan Zhu的paper看，发现竟然还真能比较轻松地看懂>_< paper大概是说SGD对数据梯度的估计会有一些variance，这些variance在训练后期会导致训练的不稳定，并推导了一些方法降低variance，感觉也挺有意思的。不过后来自己还是因为数理基础不扎实，没敢继续做理论方向。

后来开始接触了一些自己实验室的项目之后，感觉不是很感兴趣，就与实验室下另一个组的一位同学cc一起合作搞一些超分辨率(SR)相关的工作，经常跑到他那边去合作。后来我和那边的老师商量，想换到那边的组去。我想大家都是一个实验室名下的，应该可以成功。那边的导师去找我原来的导师聊，结果失败了。我原来的导师也很不高兴我去自己实验室的频率太低，把我训斥了一顿。后来我只能放弃与那个组的合作，在自己组里认真干活。但最终因为方向不感兴趣、缺少显卡等原因，我选择了转出原来的实验室，这里也非常感谢原导师对我意愿的理解与尊重。后来很幸运地，蔡登老师愿意接收我作为他的学生，也许是认为我在他的课上表现还可以吧哈哈。

我在实习之前的学习经历就到此为止。前期因为没人指导，自己也没有很好地判断力，很愚蠢地相信了很多知乎上的答案，后来才慢慢有点上路了，总结出了一些适合自己的学习方案。这个学习方案在前问已经给出来了，此处不再赘述。

最后在这里小小地宣传一下我们组。组里的蔡登老师与何晓飞老师都是学术顶尖、人品很好的老师。何晓飞老师现在创业做无人驾驶了，我与他接触不多；但我与蔡老师交流很多，有一起讨论review过大概几十篇paper吧。蔡老师学术水平极高，读博期间就发表过非常多很有影响力的论文（虽然现在那些topic已经不是很热门了），也经常能一眼看出组里同学许久也看不出来的问题要点；有很好的学术准则，绝对不会抢夺学生的成果，也坚决不参与学术圈一些拉帮结派的事情；人品非常好，很为学生考虑，对我的请求从来都是有求必应>_<；还是一位很照顾家庭的好男人和好父亲。总之，蔡老师不仅是我的学术导师，更是我的人生榜样；在我眼中他是一位真正的计算机科学家，也是一位脱离了低级趣味的高尚老师。能成为他的学生是我的幸运。组里的同学们也都基础扎实，且很努力，产出也不少。组里2019年大概有发了15篇左右的CCF A类的paper，学术实力有目共睹，且蔡老师要求组里每篇发表的paper原则上都要公开代码在组里的Github账号zjulearning下，并开放issue欢迎随时challenge，这也说明组里的工作至少都是扎实可复现的工作。

找工作之前的准备

刷题

我在找实习和找工作之前刷了一些LeetCode题目以做准备。找实习之前大约是刷了400道左右，到找工作之前大约是刷了700多道，具体可以移步我的知乎回答。不过由于我不是竞赛背景出身，不够有天赋，刷题也不是全都认真地独立完成而常常参考discussion，我直到最后也没能达到可以轻松做出绝大多数hard题目的水平（周围的很多朋友都可以轻松秒杀），这也给我面试一些对算法题要求很高的公司（例如hulu等）带来了一些不确定性。

面经与面试技巧

我面经倒是看得不多，基本上只有看过Google和Optiver的面经。原因是我运气比较好，在找实习初期（2月底3月初）就拿到了实习dream offer Google，而在秋招初期（9月初）就拿到了秋招dream offer Optiver，游戏开局就爆了神器；所以之后的面试都比较无所谓，主要是抱着聊天的心态参加面试的。至于面试技巧，主要是根据自己的经历总结出来的。当时并没有找到这样的文章供我参考>_<

模拟面试

找实习之前幸运地获得了Google官方提供的模拟面试(mock)机会，通过mock直观地感受到了Google面试的形式与风格等，对后面拿到Google的实习offer有很大的帮助。后来在Google的实习的时候，公司很贴心地给每个实习生都安排了四次mock，这四次mock也让我学到了很多东西。除此以外就没有进行过mock了 >_<

日常实习

加入蔡老师的组之后，我一开始是在何老师的无人驾驶公司飞步实习。实习期间受宠若惊地担任了不少重任，例如一开始独自一人地做一个产品的某个算法模块（包括数据采集（与标注）、算法选型、模型训练调整、开发代码、测试、部署等）。值得一提的是，为了降低自己标注数据的负担（大概标了几万张分类图片），也自己想了一些合适的采集数据的方式，并设计了一些半自动标注的算法，大概能够降低95%的工作量吧；又开发了一些简单的标注工具，才终于让我survive the labeling task。后来还担任了某个产品的开发主力（甚至很多时候是唯一开发人员>_<），同样担任了数据收集、算法选型、模型训练调整、开发代码、测试、部署等等。在这段为期5个月的实习中我学/锻炼了非常多的能力，为我后面找工作也奠定了一些基础。

不过实习期间，我发现自己应用机器学习的水平并不很好：我虽然了解大多数常用的算法的原理，但当deep learning模型不work的时候我往往会不知从何下手debug。尤其是做第一个项目的时候，模型在测试集上能够达到很好的精度，但算法上线之后效果很差，会有很高的虚警率。我当然知道这是因为自己收集的训练集与测试集过于单一，均无法很好地代表线上场景，但当时确实不具备收集更完善的数据集的条件。无奈之下，只能选择利用类似迁移学习的方式从一些有一定相似性的数据集中学习一些信息，效果虽有提升但仍然很差。后来尝试调参数、模型，但都没什么效果（我也知道不可能有效果>_<，你永远不可能解决数据上的问题，调参只是为了给leader一个交待）。最后为了用户体验，只能从阈值与告警策略着手，调整了很久的阈值与告警策略，并做了尝试自适应的阈值、bad case的判断与特殊优化等等。做第二个项目的时候，模型的训练倒是相对顺利，因为是已经相对成熟的应用领域了，倒是大部分的工作是写文档开发测试部署等等。总之，经过这段时间的实习，我发现自己其实不太喜欢收集数据、标注数据、调参调阈值调模型这样的工作，更重要的是我很惶恐于那种我不知道这个deep model为什么不work，不知道从何入手debug的感觉；也惶恐于不知道这个deep model怎么就work了，到底是因为什么而work的感觉。与之形成对比的是，我发现自己做开发的工作感觉还挺有意思的。这也是我第一次萌生转行做开发的念头。

印象比较深的还有将算法deploy到塞林斯的板子上的时候碰到的两个bug。我是组里最晚往板子上deploy算法的人，所以被leader催得比较厉害。deploy完，我测了一下算法的结果是否正确，结果发现结果是完全错的，就开始了debug。leader催我在两天内搞定，我能力不足，只能加班来弥补。第一天debug的晚上发现应该是塞林斯官方的交叉编译器的浮点运算有bug，就汇报给了leader，leader不相信。后来另一位围观的小伙伴帮忙弄了一个minimal复现的程序(记得是sin(pi/2)=0.08之类的)，才终于说服了leader。最后我们给塞林斯官方写了邮件，官方回复承认了bug的存在，并说一时半会儿解决不了。然后小伙伴Z找了个开源的交叉编译器解决了问题。这个bug告诉了我一个道理：要勇于challenge，有的时候编译器真的会有bug。但是浮点运算正确之后，程序的结果还是不对。第二天又debug到晚上1点多，发现bug是OpenCV版本带来的问题。具体地说，服务器上当时的OpenCV是2.4.9，而deploy环境的OpenCV是2.4.8。我在2.4.9版本里使用了一个操作似乎是scalar times matrix，这个操作在2.4.9里能给出正确结果，然而2.4.8的OpenCV只支持matrix times scalar（不确定，有可能把这两个记反了，总之是2.4.8只支持一个顺序，而我的代码在2.4.9里开发的，写的是另一个顺序）。更过分的是，OpenCV的2.4.8还重载了scalar times matrix这个操作，然后返回了一个错误的结果（似乎是直接return 零矩阵）。最后还是小伙伴Z过来帮我用gdb一步一步看汇编代码发现的。Z把我们一顿喷："为什么不用最新的版本？小版本的更新说明之前的版本一定是有bug的！"然而我早就提议用更新的版本了>_< 这个bug告诉我一个道理：广泛使用的库中也可能一些很弱智的bug，一定要用最新版的。（实际上我碰到了OpenCV 2.4.x版本的很多bug）

在飞步实习期间要特别感谢我的实验室同学Z。Z是我身边技术最强的朋友之一，无论是技术深度还是广度都是我生平罕见得强大。Z虽然并没有在公司实习，但在我实习期间他偶尔会来公司carry我一把。Z给我的帮助非常非常大，例如帮我一起解决了那个编译器的bug，带我一起完成了第二个项目的系统方案设计等等。后来Z也选择吃了我的安利，与我一起加入Optiver，成为浙大第三个应届拿到Optiver offer的人>_<

这里再说一句题外话。我在实习期间因为要用C++，但自己以前用的基本上都是C++中的C，所以又找了些资料学了一些C++。特别推荐一个C++的学习资料，Stanford的C++课程CS106L的Course Reader。这本书讲的C++虽然有些过时，主要是C++11之前的一些东西，但仍然能够从里面学到非常多重要的C++思想（如封装、继承、抽象、多态、const的重要性等等）。对我来说，C++是一门不容易学好的语言，许多人推荐的C++ Primer和C++ Primer Plus会将我淹没在语法细节的海洋中。但是这本Course Reader不一样，作者会从很多motivation/design的角度来讲述C++，并给出了很多motivation example，况且作者的写作水平也很好，写得很有趣>_< 如果你和我一样没什么学C++的天赋，发现其他的书学起来很困难，那么不妨试试这本。稍稍遗憾的是，这本书自10年之后就没有更新了。

最后总结一下，在这段实习让我学到了挺多东西的，对我后面找实习和工作都起到了很大的帮助。我之前从来没有在公司待过，更别说做工程产品什么的了，这次实习让我体会了一把工业界的感觉。刚开始写产品开发代码的时候是诚惶诚恐的，就我这垃圾代码也能当产品？做个demo还差不多。但后来也算是适应了这种惶恐的心理。另一方面，我一开始是独立负责产品线上一个大的算法模块的所有内容，后来有一段时间更是几乎独立开发了一个产品，因此对于算法工程师所必须的一些能力有了比较好的锻

炼。第二个项目的产品最后还成功交付出去了，成为了公司的（第？）一笔营收，甚至听说现在那块业务还成为了公司的一项重要业务。

做research

交付了第二个项目之后，我开始想做一些research，就选择回了实验室专心做research。这里主要有两个原因：其一是我想体验一下做research的生活，来帮助决定以后是否读博走学术道路；其二是听说算法岗的job market现在已经是神仙打架，甚至知乎上有人说"没有顶会paper可能就直接表刷了"。当然，根据我后来自己找工作的经验，我觉得这种话纯粹是危言耸听的。就我的经历而言，大家会更看重coding的能力、基础是否扎实、是否具有解决问题的能力以及是否具有不错的沟通能力等等。相比较而言，在job market上，顶会paper很多时候只是能justify个人能力而已，况且以现在ML会议的现状而言，能不能（短时间内）中也是挺看运气的。如果真的有某家公司招硕士算法岗要用顶会paper来表刷人，那我觉得可以考虑一下这家公司是不是真的值得投递了。

这里再简单讲讲我做research的一些经历。需要说明的是，我做research的能力并不强，也没什么拿得出手的成果，所以仅供参考。最早想做research的时候是本科，那时候非常中二，希望自己能花多年时间在一个领域上，以达到专家的水平，甚至能推动领域的进步。在学了CS231n的课程之后就开始思考做research的idea，主要的想法是找一些目前还没应用deep learning但可能可以应用的领域。硕士入学期间（2017年5月-9月）总共想了三四个idea，现在看来回顾起来还是有一些有趣的。第一个idea是想用deep RL来做启发式搜索的评估函数，不过这个没有设备支持只能放弃。后来看paper感觉AlphaGo Zero的思路与这个有些类似。第二个idea是在学6.006的课的hash部分时候想到的。Prof说hash function往往是一个比较complex的function，我就想那能不能用deep learning来做hash呢？但自己闭门造车，一直想不到合适的监督信息，就放弃了。过了一年才知道有个领域叫deep hash，是用deep learning来代替LSH中的hash function，来做近似最近邻检索的，而早在16年就已经有deep hash的paper了。第三个idea可能是我最接近能做出来的一个idea了。当时的idea是想用神经网络来近似B-tree。我做了一些实验，感觉效果不错，但是一直困扰在一个点上：神经网络的输出的上下限是不确定的，万一query的数据的输出超出了训练时候的上下限的话就不知道怎么处理了。后来2017年12月的时候看到Jeff Dean发了一篇paper《The Case for Learned Index Structures》，我看了之后，发现他们的assumption是query的数据都在train中出现过，这样就不会出现上下限的不确定性问题了。而如果要去update这个神经网络模拟的B-tree，他们就选择重新训练整个模型。应该说他们的想法是更加合理的，因为直到现在似乎也没有人做出不需要retrain的learned B-tree。当时看到这篇paper的时候我非常激动，觉得有人和自己想到一块去了，而且还是一位大牛，这说明了自己的idea还是有一定靠谱性的。于是我还很兴奋地在数据库的课上分享了这篇paper>_<

应该说，我在硕士之前做research的路子是很有一些问题的，这个问题主要在于我的野心太大，常常想做一些以自己的能力/资源不大可能做出来的research，简直就像很多民科想证1+1=2或者P != NP的人一样可笑。当时也没有人来指导自己的research，就拿一台i3小破笔记本自己一个人瞎想瞎折腾。后来有一位朋友和我说，他认为PhD需要培养两个重要的能力：其一是能够**大致判断自己能否做出一项工作**；另一点则是能够**大致判断工作的impact**。PhD应该**在自己能够做出来的工作中，挑选impact最大的去做**。而我当时显然就不具备判断自己能否做出某项工作的能力。

研二的时候回到实验室开始做research，一开始和导师商量，打算做deep learning与manifold learning的结合，算是semi-supervised learning的一种吧。当时觉得这个工作最大的困难点在于如何用mini-batch去做manifold learning中graph的更新。一些已有的相关工作（如cvpr16的《Joint Unsupervised Learning of Deep Representations and Image Clusters》）基本上是都用一些iterative的方式来训练和迭代的。后来想了一段时间，觉得还是先从相关领域的paper复现开始做起吧。然后就开始复现semi-supervised learning的paper。大概有尝试复现了7、8篇吧，发现都无法复现出他们的效果，反而复现的baseline能比他们的baseline高5-10个点左右。终于在复现一篇领域大牛（无人不知的那种）的paper的时候复现出了paper claim的效果，结果复现他们的baseline比他们paper claim的自己的效果还要好>_<

经历了一些失败的复现之后，向一位刚中了AAAI的小伙伴C请教了一些经验。C安利我做他们的领域，我听了他们的工作之后，当天晚上突然想了一个比较偏data mining的idea。第二天就写了代码实现一下，发现效果特别好。当时非常激动，觉得随便一弄效果就这么好，这要是认真调调岂不是要上天，结果那天的结果就基本上是最后调出来最好的结果了>_<然后匆匆写了一篇paper，submit到了IJCAI上。结果被reject掉了。AC评价说觉得是一个简单但是有效的工作，但是漏引了两篇10多年前idea相似的paper，我一看才知道原来10多年前就有人做过了相似的工作了，被拒得挺没有脾气的。后来这篇paper又submit到了AAAI上，有一位reviewer给了非常低的分，理由是认为data mining这类approach早就out-dated了。我后来也觉得这篇paper确实很难投中，就扔掉不管了。总得来说算是一段有些失败的research经验吧。

除了自己做的research工作以外，我还与导师一起讨论review了很多paper，注意到有些会议里的review真是招呼满天飞，不禁感到有些失望。就开始思考自己究竟是否是真正想做research的人。最后得出结论，我想做的是真正实用的顶级的research，但我显然远没有那个能力，只得作罢。硕士期间的research经历基本就到此为止了。

最后在这里安利一个沈向洋老师在华中科技大学给的talk: "How to do research"，里面分享了很多有用的方法。不过里面有些点我也不是特别同意（虽然我完全无法与沈老师相比），譬如他说做研究，应该有3年做不出来的觉悟。我觉得对大多数普通的PhD来说，这么长时间拿不到正反馈可能是会压力很大且非常痛苦的>_<

找实习

先写一些简单的总结体会吧：

1. 能找内推尽量找内推。一方面是有些公司可以免掉笔试避免自己翻车，另一方面是有内推人的话在很多公司可以帮忙查询进度什么的，比较方便。
2. 多看公司的面经了解风格。不同公司考察的重点、风格可以有很大差异，一定要提前了解公司的面试。
3. 简历上的内容一定要非常熟悉。这个没啥可说的了。
4. 面试官真的是一家公司的门面，极大地影响我对公司的印象。
5. 剩下的要点都在前面tips部分提过了，这里不再赘述。

实际上，我实习的第一个offer就是自己的dream offer Google，本来打算就此结束找实习了，把时间用来补CS基础上。陪妹子参加了一场拼多多的面试，与面试官聊了聊他们在做的事情以及难点与痛点的时候，竟然受到了一些启发，有了个idea（虽然这个fancy的idea最后没有work）。于是决定多投递多面试，与各家公司的面试官多聊聊。每家公司的面试官往往是公司里水平比较不错的人，尤其是最后一面技术面的面试官更是技术leader甚至是技术Boss，平时想要与他们聊天的机会可不容易获得。但是面试这样的事情能够让他们自动送上门来聊天，何乐而不为呢?

在这样的想法下，我投递了阿里，腾讯，头条，百度，微软，hulu，airbnb，摩根士丹利。需要特别提一下的是Optiver。对Optiver的投递其实是比较巧合的。三年前ZJUCS有一位硕士学姐应届去了Optiver，我认识那位学姐的几个学弟，学弟们对学姐是极其推崇，各种膜拜。我想，这么厉害的学姐会选择的公司，那肯定也非常厉害，于是就这样投递了Optiver。当然要说明的是我并没有对Google以外的公司有做什么面试准备，也没继续刷题，因为我早就确定了要去Google实习了。而其他公司给我发了offer之后我也是立刻拒掉，避免耽误其他同学的机会。下面按时间顺序简单谈谈各家公司的面试。

Google

流程：两轮电面，主要是问算法题。

过程：找实习时候的dream company就是Google了。平时有空的时候会做一下Google的kickstart比赛，并因此拿到了去Google参观的机会，参观完更想去Google了。于是投递了Google的实习。当时自己想去Google上海做开发SWE，同时Google的算法岗ML SWE也只在冬令营中产生，也没办法投递。运气不好的是当时Google的面试与IJCAI会议的ddl基本上重合，所以当时是做research赶paper与刷题准备面试同时进行。Google的面试安排相对较早，大约是在2月底吧。有趣的是，由于我在填Google的

表的时候写了中英文面试均可，结果HR小姐姐就给我安排了两场英文面试Orz 而且都是晚上11点半或者早上7点这样的事件Orz 第一面似乎是位印度的Googler，结果因为一些原因鸽了我。第二次一面是一位新加坡的Googler，结果又因为被浙大的邮箱坑了进不去Google的视频会议。第三次一面试终于顺利完成，面试官是一位纽约的小哥哥，题目并不难，我很轻松地写完之后我们闲聊了一会儿。二面是一位上海的Googler，题目挺难的，而且给的压力不小，差点就崩了，好在我还是稳住了心态顺利地做了出来。面试完半小时左右就收到HR小姐姐的消息通过了面试，此时我还没从面试的地方走回宿舍……后来过了Google的Hiring Committee，进入Team Match后竟然也很快（不到半天）地就被捞了起来，还有点受宠若惊。捞我的人（也就是我后来的Host）希望我能去北京做ML SWE，我想应该是看中了我的ML背景吧。我问HR小姐姐我的项目内容，小姐姐给我发了俩，说不确定是哪个，其中一个内容大概是自动化地针对手机设备压缩网络，我比较感兴趣；另一个项目看上去是与TensorFlow相关的，感觉也不错。同时注意到自己的Host是Berkeley的PhD和美国一所大学的professor，还发现那个组的创建人是李飞飞。在与妹子商量后，就接了这个offer。可以说第一个offer就是自己的dream offer，非常幸运和激动。

拼多多

流程：笔试+HR面 + 两轮onsite技术面

过程：投递的是算法岗。拼多多在3月中旬的时候在浙大附近租了个酒店，在那里进行了现场的面试。HR通知我们早早地过去，然后在那边等着排队，记得自己等了挺久的。先进行了HR面，主要是问我对公司的工作时间什么看法，我当然是回答996不在话下，不然我可能就要当场被赶出去了？之后一面技术面问了一道数据结构的coding题。一面的面试官有些傲慢，一开始还弄错了一个地方怼了我一会儿，后来我只能耐心纠正他的错误。之后问了些机器学习基础与项目的问题。这位面试官让我对拼多多的印象非常糟糕，后来也就直接拒了offer，秋招也没有投递。二面技术面倒是感觉不错，问了一道很难写的链表题目，我比较轻松地写出来之后同样问了机器学习基础与项目的问题。面试完过了一两周吧，晚上10点半接到offer call，直接拒掉了。

摩根士丹利

流程：全英文，笔试+电面+两轮onsite技术面

过程：投递的是C++开发岗。3月底大摩突袭打电话电面，全程英文面试，问了很多数据结构、操作系统与计算机网络的基础问题，最后问了一道system design的题。之后约了4月下旬的onsite面试，我与妹子一块参加。结果因为一些原因（起晚了）到达现场迟到了，非常不好意思。到达现场之后尴尬地发现除了我和妹子之外的所有候选人都穿着正装Orz 现场先做了一点笔试题，交了之后hr小姐姐就带人去面试。面试也是全程英文面。两轮面试问的都是数据结构、操作系统、计算机网络与system design的题。两位面试官给我留下的印象都不错，谦逊和气，水准也不错。一位面试官当时问我有了哪些offer，我就诚实地回答了一下，面试官非常惊讶地问我"Then why do you come to Morgan for an interview?"我说自己想稍微了解一下金融的情况。大摩也是在面试完大约一两周后来了offer call，我拒绝之后并说明了自己的去向，HR说那我们继续保持联系。

头条

流程：笔试（内推可免)+3轮视频面+HR面

过程：内推的是头条产品的推荐算法岗。4月初进行了前两轮技术面。头条很喜欢考算法题，两位面试官都问了很难的算法题，当时比较困状态不好，艰难地写出来之后又问了一些数学基础、机器学习基础和项目的问题。二面结束5分钟后接到电话约三面。过了两天进行了三面，这次的题倒不是很难，写了两道之后面试官问了一些操作系统、编程语言的问题，最后问了我一个场景设计题。面完5分钟后收到offer call。最后还是拒绝了。

阿里

流程：笔试（内推可免)+3轮电面+1轮交叉面+HR面

过程：内推的是阿里搜索推荐部门的算法岗。4月初的一个晚上一面，面试官是内推我的师兄，我觉得可能怕被骂防水有点矫枉过正，问了足足70分钟的问题，主要是算法题（很简单而且不用写代码）、机器学习基础、深度学习基础与项目问题。过了两天进行了二面，也是算法题（很简单而且不用写代码）、机器学习基础、深度学习基础与项目问题，面试官有问一些开放性的问题，还蛮有意思的。过了一周左右进行了三面，基本上只问了项目问题和开放性问题，后来知道那个开放性问题竟然是他们投kdd的paper……然后过了挺久的，在安徽参加valse的时候突然接到电话，原来是阿里的交叉面（即另一个部门的人来面试我），问了机器学习基础和项目的问题，还问了深度学习框架的一些实现细节（不知道为什么问我这个，但还是回答了出来）。总得来说技术面试体验还可以。又过了一周，晚上9点半，接到阿里HR的电话，阿里HR还真是如传闻一般……后来拒了offer了。

腾讯

流程：笔试（内推可免）+3轮电面+HR面

过程：投的是数据挖掘岗位。比较尴尬的是，当时接到了优图Lab的电话，说他们要求实习4个月以上，问我能不能满足要求，我没经过思考就说了那肯定不行，和导师说好了3个月。然后就面试终止了。我真傻，我本来面试的目的就不是去实习啊，直接答应下来说可以4个月就好了…

微软

流程：笔试（内推可免）+3轮onsite面

过程：找学长内推了微软苏州的SDE，与妹子一起去苏州onsite。一面问了项目和一个算法题，又问了两个设计模式的题。面试官给我的印象非常差，看不懂Range For就challenge我，没用过priority_queue也challenge我。二面的面试官比较nice，先问了项目。之后问了一道比较简单的算法题，然后是难一些follow-up，都相对轻松地解决了。最后还剩下10分钟，面试官就开始和我闲聊，问了我已经拿的offer，并问我有没有肉翻的打算，我说至少要等妹子也有肉翻能力了再一起吧，暂时不考虑。三面面试官是一个leader，先问了一道system design，并讨论了一些面向对象的设计思想，然后问了我开放性的题目，比较难。总得来说，除了一面面试官给我印象非常差外，另外两个面试官都很nice，尤其三面的面试官水平感觉也很好。

Optiver

流程：全英文，笔试+电面+两轮onsite面

过程：

终于要讲到自己最终选择的雇主Optiver了。我投递的是C++ developer。之前提到，投递Optiver的主要原因是相信大神S学姐的眼光。在校期间听说了不少S学姐的传说，据说她读书期间写System代码就从来是bug-free的，秋招的时候腾讯问她“你开个价吧”，但照样被她拒绝。还听说她当时是拒了一家叫Jane Street的公司的offer去了Optiver。不过S学姐的那些传说我也没有向她考证过。后来我才知道S学姐是浙大第一个应届进Optiver的人，而我则是第二个。

总之我就这样投递了。过了几天收到笔试通知，笔试题有两套，一套是在一个平台上做两道C++偏数据处理的题目。另外一套是一些数学智力题，很多题目难度非常大，做得我头晕眼花。做完之后第二天接到HR小姐姐的电话，开始了HR面。HR的口语非常好，我听上去与native speaker没有什么差别，问了许多比较有趣的问题，我也一一回答。面试完过了两小时，HR给我打电话问我愿不愿意做一下公司另一个岗位Application Engineer (AE)的笔试题。我一向是相当flexible的，就答应了下来。这个岗位的笔试题主要是一些与Linux操作相关的问题。第二天HR给我打电话，说我AE的题做的分非常高，建议我先和这个Team的人交流下，看看感不感兴趣，我反问难道我C++ Developer的分不高吗。HR说了两点。1. HR觉得我的沟通能力很不错，而AE非常需要沟通能力，她觉得AE说不定会挺适合我的。2. 我投递得比较晚，Developer他们已经有一些不错的候选人了，不一定能排得到我。HR问我为什么想做C++ Developer，我诚实地说因为周围有一些大神朋友是做System Developer的，我和他们交流觉得挺有意思的，而且我觉得System Developer的技术很硬核，具有很好的技术护城河，能够让我在多年后仍然保持很强的竞争力。HR于是建议我与两个team的人都聊聊，看看自己对哪一块更感兴趣。我接受了HR

的建议，毕竟我早就决定去Google实习了:) 与越多越的人交流我越开心（请Optiver的同事们看到这段不要打我>_<）。

去大摩参加onsite面试的高铁上我接到了Optiver面试电话，稍微想了一下，我决定就接受了面试，因为我其实不在意拿不拿得到offer，面试环境差一些也不要紧。于是就这样在嘈杂的高铁上开始了AE的面试（听说我开始电面之后周围的人都瞬间不说话了，感谢有爱的乘客们）。面试的内容很多，考察了数据库、操作系统、Linux常用操作、运维场景问题、计算机网络、版本管理与system design等内容。这场电面是给我印象最深刻的电面之一，内容很多，且很多问题是很有意思的一些design问题，电面的面试官甚至还有两位>_< 无奈我自己其实不太懂运维的内容，很多问题都答不上来，只能靠猜测（如前面面试心得里所述，我是先claim了自己要猜测的），在一些design方面与面试官进行了比较多的讨论。面试官的脾气也非常好，并没因高铁上时常响起的播报声而恼怒，而是一直说never mind, that's ok，让我感到非常非常不好意思。如果早知道高铁上环境如此嘈杂，我就推迟面试时间了……

然而，这次面试让我意识到自己和公司的AE也即运维岗的技术栈很不match，发现自己对运维也不感兴趣，后来HR联系我的时候，我就请她还是继续安排我走system developer的流程，HR尊重了我的意愿。然而大约在4月22日的时候，HR打电话告诉我说公司的员工们大都去休年假了，这样可以与五一假期连一块，休一个比较长的假期（我怀疑这是公司的PR，告诉我公司年假很多>_<），因此面试可能得安排到5月之后，我自然是欣然同意。可惜到了5月多的时候，HR联系我说他们已经有一些合适的候选人了，就不继续我的System Developer的面试了，并说如果我对AE有兴趣的话可以给我AE的offer。这一次的Optiver实习面试旅程就到此为止了。虽然没有去Optiver实习，但这次实习面试之旅为我后来秋招选择Optiver埋下了伏笔。

百度

流程：笔试+三次技术面

过程：过了百度笔试之后，大约在4月中旬接到百度的面试通知。三次技术面试都有考算法题、机器学习基础和项目等。百度的算法题不算难，大都是LeetCode medium难度吧。百度虽然这几年发展不好，名声口碑也一般，但是说实话面试官水平还是挺不错的，给我印象挺好。4月底的时候收到offer然后拒掉了，但是不知道为什么后面有一位百度的员工加了我微信，说我去实习的时候他负责带我Orz

Airbnb

流程：笔试+笔试确认面+两次技术面+文化面

过程：投递的是全栈工程师。投递的原因主要是听说Airbnb的package挺不错的，不加班，而且前景也还可以。大概是4月中旬接到笔试通知，笔试题同样是在一个平台上完成，算是一道稍微有点麻烦的模拟题吧，但也不算很难。做完笔试题的第二天有一位面试打电话来和我约笔试确认面。笔试确认面时间很短，大概是15分钟，面试官非常客气地问了我design的motivation是什么，并让我分析了一下复杂度，个人感觉只要代码是自己写的，这一面不会有任何问题。过了一两周之后安排了两次技术面试。技术面试的内容主要就是写算法题，不过Airbnb的算法题风格与其他家还不太一样，他们的题目思维难度不大，但常常较难实现，会有一些corner case什么的。一面的时候我在实验室的学生休息室面，结果附近在修路，非常吵，十分艰难地才把代码写出来，面试官不太满意。二面的时候换了个安静的地方，相对轻松地搞定了题目，面试官还比较满意。但是应该是由于我一面表现不好，Airbnb让我在备胎池里待了很久……总得来说Airbnb的面试难度也挺高的，尤其是这种只做很难的算法/模拟题的风格，受运气/状态影响不小，一不小心就会挂。

Hulu

流程：笔试（内推可免）+三次技术面

过程：投递的是软件开发工程师（SDE）。在5月的时候Hulu来浙大办了宣讲会，并于接下来几天在浙大安排了现场面试。这种送上门来的面试机会我当然不会错过。没有投递Research SDE（即算法岗）的原因似乎是Hulu不打算从浙大招算法实习生？总之是明确说了只招SDE。猜测可能是只收清北的吗hhh 总之就这样上场面了Hulu。一面的面试官看了我写满了机器学习项目的履历之后，决定问我分布

式系统的问题，因为他在公司是做分布式系统的。尽管我一再强调自己完全没接触也没学过分布式系统，但还是被迫回答了两个问题。之后就开始写算法题了，第一个算法题是道LeetCode medium难度的题，大概花了5分钟秒杀。然后面试官给我出了一道hard难度的题，想了10多分钟，在面试官的提示下我终于给出了最优解，然后面试官没让我写代码，又问了我一道hard难度的题，然后又是想了10多分钟，这次终于没能给出最优解。当时我的做题能力就是medium难度基本秒杀，hard难度的要么不会，要么就得做很久才能做出来:) 总之面完之后面试官和我说了一句你辛苦了，然后就把我挂掉了，终于吃到了第一个也是唯一的一个拒信>_< 总得来说还是实力不济，Hulu的面试题大都是hard难度的，自己当时确实做不出来，而且3个月没刷题了手也生。

至此，实习面试就全部结束了。总得来说各家公司的面试各有一定的特点，这可以通过看面经来了解。另一方面，这些面试也多多少少有一些共同点，考察的能力与一些相应的技巧我在本文前半部分也已经介绍过了，此处不再赘述。

实习经历

Google的实习经历对我的影响、帮助都挺大，再加上Google可能是很多同学心目中的dream company，这里就简单介绍一下我在Google的实习经历，以供参考。当然，其中也夹杂了许多我的个人经历，那些绝对是个例现象，与Google无关。

入职

Google的入职活动（onboarding day）还是挺有意思的，上午是一位技术同事志愿服务地带我们逛了北京Office的各个地方，并给了一些介绍，然后给我们分了Laptop。然后HR小姐姐Y带我们参加了一个活动，是整个大中华区当天的入职的Googler（Google员工）一起开一个会，介绍Google的文化、基本信息等等，有一位非常Senior的领导会一起聊一聊。我记得当时应该是台北那边的一位VP参加了会议。印象最深的是提问环节，我看到没有人有问题，我就问了一个问题：“你对新入职的Googler有什么建议吗？”VP回答：“Try to confident. Your products and efforts change the world.”

之后与我的host见面聊了聊。他是一位非常nice的host，水平也很高，也很会带人，在实习期间教了我很多东西。host与我聊项目，是一个OCR的项目，我发现这个项目似乎和之前我得到的信息不太一样，就问他为什么变了。host告诉我说后来招了两个清华的本科EP实习生，打算让她们做之前安排我的项目。我还是比较flexible的，就愉快地接受了新项目。

项目初期进展

新项目一开始做得很顺利，主要是需要我对Google一个很有影响力的开源项目(30k+ star)做一些改动，增加一些新的功能。确切地说，原始的项目的自动生成训练数据的训练脚本只支持字体文件作为输入，而我们期望它能够支持image-label这样的输入。这需要我阅读项目的代码结构（大概几十万或者几百万行代码），理解系统的设计、数据结构以及内部使用的算法，并找到自己需要改动的地方，再做修改。困难点在于这个项目并没有什么完善的文档，也没有（在我看来）非常完善的注释，需要直接阅读源代码。我大概花了一周大致看懂了实现原理（略去了一些技术细节），又花了一周左右理清思路并实现了需要的功能。但是我的噩梦才刚刚开始>_< 我开发的功能结合原始的训练代码训练出来的模型效果不合理得差。进行了一周多的debug，以及与host非常多的讨论，终于找到了bug在哪里，原来是原始的训练代码对于输入的数据是有一定的assumption的，但是在文档和注释中并没有说明。项目的训练数据均是由他们的代码自动生成的，这些数据会是符合这个assumption的，但是我新增的功能生成的数据则不符合这个assumption，这样会导致训练过程中调用的聚类算法中出现矩阵不可逆的问题，使得生成的训练模型效果很差。然而即使在修复了这个bug之后，模型的性能还是不能完全令人满意，又做了一周的实验慢慢提升性能。最终的效果虽然还不完全令人满意，但已经勉强可以使用。至此，我已在Google实习了一个月。

这里顺带一提，在北京office的实习生们大多数都是清北的，而且很多同学都有本科甚至是初高中的竞赛背景。考虑到我的CS背景与他们的差距，这多多少少给了我一些peer pressure。又因为妹子在杭州实习，我周末就没什么安排，而我的房子租得又离公司非常近（步行5分钟），故我整个实习期间基本都是一周去7天公司，且在第一个月每天都是早上9点到晚上10点以后走的。顺带一提，Google确实相当注重work-life balance，我晚上的时候时常会闲逛一圈，极少看到有除了实习生之外的同事加班，周末则只能看到少数一些实习生。

虽然第一个月的前两周我每周都会100个小时左右的时间在公司，但实际上我大概只有一半左右的时间花在自己的实习项目上，还有一半的时间则在如饥似渴地阅读Google内部的一些课程、资料。进Google之前我就听说公司有非常好的学习资源，而我可能在Google的时间只有这么3个月，我希望能够尽量地多学一些东西。不过遗憾的是我觉得我所看的学习资源比四大的课程基本都有明显差距，但想来也是，四大毕竟是最顶级的学校，且专门focus在教育上，要是还比不过Google的课程的话那也太没天理了。这两周我虽然学习/工作的时间很长，但项目进展顺利，也学到了不少东西，自己感觉非常得充实与满意。第一个月的后两周因为模型的效果一直很差，我给自己的压力很大，原因主要是觉得自己项目做得这么挣扎，一定是自己能力不足，要多花时间赶上，不能拖host的后腿。实际上我host没给我多少压力，对我也一直非常nice，但正因为如此，我更不想拖他的后腿（但我后来发现，项目做得挣扎是一个实习生尤其是ML实习生的常态:））。我一直希望能够尽快地让模型work，每天都会debug到11点多才离开公司，而且晚上睡觉睡得也不好，基本上每天都会梦到自己在debug模型，而且debug不出来。这段时间的经历也让我意识到自己的ML水平其实也不怎么高，并不能以一种很systematic的方式去逐步debug模型的问题，而且也对debug的进度没有很好的把握，并不能合理地预估让模型work的工期。算法岗与开发岗不太一样，后者的工作我往往可以相对合理地预估工期。例如我前两周的工作就更偏向于开发，我每天都能有一些合理的进度，并且我能够相对清晰地预估自己需要多久阅读理解代码/实现功能/debug等等。另一方面，我做的算法岗的工作很多时候的工作内容就是调参数/调模型/调特征，会相对无聊，且我往往没有很好的insight（尤其是deep learning相关的），这让我感觉不到很好的技术沉淀与积累。总之，这段时间的经历让我又开始思考算法岗/开发岗的选择问题，并且对我最后做出选择起到了关键的作用。

顺带一提的是关于实习生转正的事情。由于妹子表示在阿里的组非常好，师兄们带她都很尽心，人也很好，晚上下班也挺早的，故我此时的秋招打算就是去阿里做推荐。当然Google的转正还是要争取一下，可以用来compete嘛。

进抢救室

（下面将描述完全与CS无关的事情了，完全可以跳过>_<）

然而，更严重的问题出现了。我多年来一直肠胃不好，也去医院检查医治多次（但没做过胃肠镜），但都没什么效果。也许是因为实习的第三四周给自己的压力太大、工作时间太长，也许是因为还没习惯北京的饮食与生活，也许只是凑巧，总之我大概在第四周末尾开始出现胃溃疡引起的胃出血。但是我一个人生活，对生活也向来不太注意，竟然没留意到这个现象。大约胃出血了3、4天之后，某天晚上我上厕所的时候竟然感觉到一阵眩晕，眩晕得我无法站稳而摔了一跤。我意识到了不对，大概查询了一下，猜测自己是得了胃溃疡并且导致了胃出血。于是我决定第二天去附近的北医三院检查一下。我试图在网上挂号，但发现号子已经没了。

但是第二天有一场会议，我不想耽误他人的进度，故第二天起来挣扎地去了公司。路上我已经察觉到不对：我甚至已经没法正常走路了，我保持站立或者走路姿势大概两分钟左右就会脑供血不足，而不得不蹲一会儿让大脑供上血再继续前行。开完会后，我和host说明了情况，host表示非常担心，让我赶紧去医院检查一下，我就打了个的独自前往医院。实际上我对当时自己的危险情况并没有一个清楚的认识，正确的做法应该是找一位朋友一起，并拨打120去医院。

车子快要开到医院的时候开始堵了起来，原来北医三院门口的道路非常狭窄，但前来看病的病人非常多。堵了10多分钟，我开始有点焦急，出租车师傅建议我下车步行过去。思考了下，我下了车开始往医院前行。实际上这条道路只有区区几百米，但却绝对是我人生走过的最漫长的几百米。我每走大概10多米就要蹲一下，让脑子供上血，再继续前行。我运气还算不错，如果中间不幸晕了过去，各位可能就看不到这篇文章了。在路上我给妹子发了消息说了一下情况，并不敢告诉自己的家里人，怕他们担心。我终于还是这样磨到了医院，先去自助挂号机上挂号，同样显示肠胃科没号了，我想可能人工窗口有额外

的号呢？于是我去人工窗口排队，排队的时候直接坐在了地上，前面排队的人还发出了轻蔑的嘲笑。终于排到我的时候，我直接说挂肠胃科，工作人员告诉我没号了。她问我现在什么情况，我解释了下自己的症状，说我觉得应该是胃溃疡导致的胃出血，现在脑子供血不足，工作人员说那要不给你挂个脑科吧:(我苦笑了一下，问急诊科在哪里，她告诉我在地下一层，我于是继续挣扎着向急诊科磨去。

总算磨到了急诊科，科室要先分诊，我直接上去描述了症状，分诊的医生让我先去边上测血压。我于是只好先去边上测血压，但是可能是因为出血过多血压太低，血压计一直测不出来。大概测了10来次吧，总算是拿到了一个结果，我赶紧拿着结果找分诊医生。此时我的状态已经非常非常糟糕，几乎走不了路，说话也非常费劲。医生看到我的状态说你怎么样啦，我感觉你是出血过多了，来我带你测一下血压，然后把我扶到血压计边上继续测，但怎么也测不出来。我非常虚弱地说了一句这大概是测不出来了，突然感觉想吐，就说您让开一下，然后吐了一地的血>_<，吐完说了句不好意思。医生赶紧让人去叫抢救室的人过来，随后我就躺上了抢救室的那种移动病床。躺下之后，我感觉自己的状态明显好转了，就和旁边的医生开玩笑说要不要这么夸张，这就送我进抢救室了？医生大概是怕我害怕，以一种相对轻松的态度告诉我说你刚刚吐了些血，我们要检查一下。把我送到抢救室的时候，医生问我有同伴一起来吗，我说没有。他们让我赶紧喊家人，我说自己一个人在外地实习，他们就说那你赶紧喊个朋友啊老师啊过来吧，没有一个人不行的。我于是拨打了同样在Google实习的另一个同学H，但他在睡午觉，没接到电话。然后不得已打了在北京实习另一家实习的女性好朋友L的电话。我与L关系很好，但因为怕妹子吃醋，所以万不得已不想让L过来。L接了电话之后，我简单描述了下情况，然后说你别慌我现在没啥事，很正常，你方便的话就赶紧过来吧。L二话不说直接答应，随即打车过来。

之后医生给我做了很多检查，所幸检查结果都是正常，只有血压和血色素等指标很低。我说我现在感觉差不多好啦，一定要在抢救室待着吗，这里除了我以外的病人最年轻应该也有70岁吧。医生安慰我说应该没啥事，大概率是胃溃疡导致的出血，但是要做一下胃镜检查。我说那今天就能做吗。医生告诉我北医三院的肠胃科非常有名，预约胃镜通常都是要等一个月甚至3个月以上的，“但是我们急诊科去排应该可以快一点，今天或者明天就可以了。”我说那确认是胃溃疡的就能出院吗，医生笑了下应该很快的。这时候我给妹子打了个电话说了一下情况，她想要赶紧请假从杭州过来，我说再观望一下吧。

L到了之后，医生让她去帮我买了很多生活用品，我意识到似乎就算是胃溃疡也会是一场相对持久的战役了。不过我自己感觉状态还不错，看她面如死灰的样子，就和她说点玩笑话缓和一下气氛。后来医生让L到急诊室外面去等。H这时候回了我电话，问我怎么了，我描述了一下情况，请他在下班之后来接替L的班，因为抢救室的病人是需要24小时有人陪护的，L一个人肯定是扛不住的。H也二话不说答应了。考虑到这样的24小时陪护不知道还要多久，我并不想太耽误这两位朋友，终于还是让妹子过来了。另一位老家在北京的浙大同学C知道了这个事情，也表示要一起过来。于是C和妹子一起买了高铁票出发过来。

我又给host说明了一下情况，host非常担心，和Google的hr小姐姐们说了一下这个事情。hr小姐姐们也非常担心，赶紧问了一下我的情况，还有两位比较熟的小姐姐直接出发来公司看望了我，让我再次感受到了公司的温暖>_< 我和她们笑着聊了会儿天，她们发现我还挺能说的，状态也不错。我说不用担心，没啥事的，就是个小意外，等我出院吧。

到了晚上C和妹子到了。后面的事情就省略说了。第二天做了胃镜，检查出来确实有两处严重的溃疡，高水准的医生用两个钛夹夹了一下。后来就转到肠胃科住院区去，并告诉了家里，我父亲之后来了北京。我住了一周的院，住院期间主要是妹子，我父亲和C陪护我，妹子主要负责照顾我的生活。期间还有一些朋友同学过来看我。巧合的是，我住院的主治医生是Google一位HR的闺蜜，定期向HR们汇报我的情况。经过了一周的治疗后我出院了。带妹子玩了两天之后送她回了杭州，然后休息了一天，就去公司学习去了。

恢复实习

第一天去的晚上我host突然出现，和我说要不quit实习回家休养吧，还给我说了他Berkeley的导师给他说的一句话：“学业与事业是个橡胶球，挤一挤，变形了也总会恢复；身体与爱情是个玻璃球，挤一挤碎了就回不去了。”我解释说医生表示我现在其实状态已经相对正常了，只要不过分劳累、注意饮食、不要运动即可，是可以工作的。当时我心里有个很中二的想法，认为这一切的阻碍都是台阶，越是这样，我越是要迈过这些台阶，完成我的实习。另一个重要原因是我希望能用Google的offer来compete阿里的A+，因为我的profile直接面阿里应该是拿不到A+的。

由于项目的算法部分已经勉强能用，后面的任务重心就转向了产品的前端与后台开发。我之前没有任何相关经验，只能从头学起。在一位朋友的指导下，我一周就连学带做搞了个简陋的雏形。后来就与host他们讨论界面的优化、功能的加强等等，就这么迭代地进行项目的改进。

印象比较深的是我花了大量的时间在试图把我的服务部署到Google Cloud上，使得项目的用户可以接公司的WIFI即可访问到。然而Google的一些隐私相关的policy让这件看上去简单的事情变得异常麻烦，我试了很多办法都没能成功。最后按照host的建议，我向Google Cloud申请了一段时间的WiFi IP访问特权，结果帮忙查出了个Google防火墙的bug Orz 美国负责防火墙的同事表示这个bug他要上报领导解决。这使得我不得不放弃了部署到云上的打算，转而在我的台式机上部署并让手机端可以访问，但这件事情也花了很大的力气才搞定:) 这些时间花得是让我非常frustrated的，因为这些事情枯燥乏味，没有技术含量，如果不是policy的问题我只需要10分钟就可以搞定Orz

在写前端、改进用户交互页面的时候，我意识到自己对这种工作内容是不感兴趣的；另外一方面，我的产品的impact也相对很小，潜在的用户大概是个位数的。这两者都让我对写项目的前端不是很提得起兴趣。不过我还是希望自己能把这个实习项目，还是很认真地改进项目，最后也算是做出了一个简陋的产品出来，勉强完成了项目的初期目标。在这段时间，我也有去接触一些其他在Google的同事，了解他们在做的事情，这些都引起了我对自己未来职业生涯的思考。

后来公司的HR小姐姐问我要不要extend几周来补上住院的两周。我也直接拒绝了：我已迫不及待想回杭州了。答辩完，整理完所有项目的文档与代码，并与我的host以及两位co-host还有众多小伙伴告别后，我checkout了，并于第二天回到了杭州。Google的实习到此结束。

总结

首先感谢我的诸位家人、朋友、同事、领导、师长等对我在这个阶段的支持与照顾，非常非常感谢。

总得来说，由于种种原因，我的Google实习经历对自己来说不打算是特别成功的一次经历。当然，这与Google的关系不大，Google是一家非常好的公司：work-life balance，卓越的同事和leader，世界级的vision，认可度很高的履历，人性化的制度、管理与福利等等。我的同事们都很优秀又nice，给了我很多帮助；我的host和co-host们更是水平极高又认真负责，教了我很多东西。只是在Google中国工作似乎不完全是我现在所期望的生活。然而，往抢救室走了一趟之后，我又有些害怕阿里的工作时间与压力。这时候的我有了挺多迷茫的地方：似乎自己期望的工作内容与work-life balance不可兼得？我又对算法岗的工作有一些concern，有心转开发，但阿里的实习offer已经转成了秋招直通终面了，换转组或者岗在阿里又都是极其忌讳的事情。我究竟该何去何从？

秋招

在实习部分已经总结过各家公司面试的特点了，这里就不再赘述了，只以简单的流水账形式说一下自己的秋招经历与心路历程。

百度

我的秋招从百度的广告部门的算法岗开始，是实习过了面试之后自动给我转了秋招提前批。当时还在北京实习，被要求去公司参加onsite。总共有3轮技术面试，面试官问了一些coding、机器学习基础、项目和系统设计的问题。百度的面试官感觉实力还挺不错的。印象比较深的是百度面试官真的很喜欢考核各种设计问题，尤其是二面面试官，一定要我独立地设计一个系统，还得事无巨细地给出系统每个组成部分的算法、实现细节等等，即使我已经多次声称对那块知识不熟悉。

腾讯WXG

第二家的面试是腾讯的WXG看一看团队的推荐算法岗，同样也是要求去公司参加onsite。可能是运气不好，这次面试给我留下了非常糟糕的印象。首先，我到了office之后，安保人员不让我进入office，需要在门口站着等面试官出来接，出示任何信息都没用。其次，面试官迟迟没有出现，而面试官联系我的电话又是总机电话Orz，我没法联系他。无奈之下，我只能按照邮件里说的拨打了腾讯深圳总部的相关电话，请他们帮忙催面试官，结果他们告诉我只能给面试官发消息，让我等面试官回复。在门口大约站了

半小时后，腾讯深圳总部的校招热线告诉我说面试官回消息了，说马上就来。结果这一个马上又让我等了半个小时还不见人影。因为实习最后一周还有很多东西要整理，我没有时间继续浪费下去了，就选择离开，结果快回到Google的时候接到面试官的电话问我走了吗。Orz你们是认为我会在门口站两个小时等你们吗。面试官问我还回去参加面试吗，我说那行吧，就回去继续参加面试。总得来说面试不算很难。第三面是一位漂亮的小姐姐，面试完已经6点多了。我问还有第四面吗，小姐姐一笑，问我饿了吗。此时腾讯的员工们已经在领盒饭准备吃饭了，饭菜的香味飘来，我也确实感到了饥饿，就说是的。本以为她要拿一盒盒饭给我，没想到她再次温柔一笑，说我也饿了，我先去吃饭了，你在这里等下一位面试官吧>_<。四面面试官看上去挺年轻的，技术也很强，一直问我愿不愿意秋招去北京工作，我说我申请的是上海only啊，他说我们在上海没有组Orz 后来我强调说我只接受上海，我们只能不欢而散，他们把我放回了池子里。

阿里

第三家的面试是阿里，同样是实习offer转秋招直通车，只需要一次面试。当时我内心的第一选择是秋招加入阿里，就很认真地准备了一下。面试官春招的时候已经面过我一次了，也没问什么特别的问题，只是特别问了我的意愿，我就照实说第一选择是阿里。面试官告诉我说等HR的面试吧。后来接了Optiver的offer之后我就拒掉了阿里了>_<

Optiver

在即将结束实习的时候，Optiver的HR小姐姐给我打了电话问我秋招愿意考虑Optiver的机会吗，因为我有实习offer，就可以直通onsite面试了。我说当然愿意啦。HR与我聊了挺久的，问了我一些对公司的concern，并一一为我解答。最后HR说那我们约你下周travel来上海面试吧，我说不用不用，下周我就在杭州了。于是HR给我订了两张杭州去上海的商务座，约了实习结束的周一去上海onsite面试。HR告诉我说到时候会有两个技术团队的team lead一起面我，一个是公司的data team，工作内容会与machine learning相关，另一个是公司的system team，工作内容主要是system programming。我可以根据自己的喜好以及给我offer的team，选择一个加入，我欣然同意了。

说实话，此时我还没有把Optiver放到内心的第一位置，当时的想法还是与妹子一起去阿里。不过我还是很担心自己能否承受得住阿里的工作强度，就与阿里的师兄交流了一下，他当时表示说应该还是可以的。

实习checkout之后，我迫不及待地回到了杭州。妹子向我吐槽了很久她暑假的一些没有告诉我的遭遇，并且说了一些她家里比较棘手的事情。说完她就和我提了分手，转身回去，我只能一脸懵逼地回宿舍，收拾心情准备第二天的面试。我想那些棘手的事情可能会需要比较多的钱，而之前就听说Optiver的待遇丰厚，那么这个offer一定要拿下来。于是第二天就晕乎乎地坐上了高铁，坐上高铁之后给妹子发了个消息，说等我回来我们再好好聊聊。

我带着少有的必胜的决心进了Optiver的上海office。面试安排在下午两点左右，我到得比较早，大约是下午一点，所以HR小姐姐先和我聊了会儿天。她先和我聊了一下自己的方向选择问题，她有些疑惑我本科毕业就大跨度地从航天转了CS，现在竟然又要从算法岗转开发岗。我大概解释了一下。我们又聊了一些公司culture什么的。HR大概告诉了我几点。其中有几点特别吸引了我的关注：

1. 公司注重work-life balance。例如公司不提供晚饭，不鼓励大家晚上还继续加班。
2. 公司的福利也是相当不错。有很多年假带薪病假等，也有一些免费的餐饮、咖啡厅、大额商业保险、健身房等等。
3. 高频交易是一个竞争激烈的行业，公司希望大家都能够**self-motivated地，独当一面地去做出有impact的工作**。
4. 公司招聘宁缺毋滥，highly selective。例如我实习面的AE岗位，最后公司也没有勉强招不合适的人。

技术面试终于开始了，两位team lead进入面试房间后先很客气地与我握手并做了自我介绍。随后他们给了我一道相当复杂的系统设计的问题。在提问以确认自己理解了题目之后，我进行了一些思考，并给出了第一个设计方案。两位面试官非常细致地问了我这个方案的各个细节之处、性能分析、trade-off等等，这些问题涉及到数据结构、算法、操作系统、体系结构与计算机网络的各种知识。好在我在说出自

己的解决方案之前就已经对这些细节进行了思考，所以回答得还算流利。面试官接下来让我在白板上实现自己的代码，在这个过程中同时也考察了我的抽象能力、模块设计能力等等。写完代码并证明它是bug-free之后，面试官突然狡黠一笑，问我能不能再改进这个方案。与面试官进行了大量的讨论之后，我终于成功地给出了第二版的设计方案，当我给出来的时候一位面试官非常激动地说了一句“Exactly！"然而两位面试官并不打算就这么简单地放过我，又继续问了大量的follow-up问题，直到确认我每个细节都考虑到了之后才终于流露出了满意的笑容。随后，他们让我问了一些问题。我最好奇的一个问题是，明明有一位面试官是data组的team lead，为什么并没有问我任何机器学习相关的问题，而是问了这么一个在我看来更偏system的问题。面试官告诉我说，这个问题在他们看来是problem solving ability的问题，而Optiver认为基础扎实、沟通能力出色并且有很好的problem solving ability的人可以胜任公司的任何一个IT岗位。我还问了一个问题是，他们选择optiver的最重要的原因是什么。他们告诉我是**exciting work and impact**。

终于这轮面试结束了，时间大概过了两个小时左右。我休息了几分钟去上了个卫生间。这里有个小插曲，我突然收到一条来自腾讯的短信约我15分钟后面试，问我是否能参加，我回复了否。后面在Optiver head of IT面我的时候，腾讯的面试官突然打电话过来问我什么时候有空。好在Optiver面试官没有生气，还让我接了电话商量了和腾讯的面试时间。

回到office里的时候注意到公司在为一位同事庆生，记得是一位毕业于MIT的小哥，我也过去分了点蛋糕吃。随后开始第二轮面试，第二轮的面试是Head of IT面我，也就是Optiver IT部门的老大。面试官问了一些项目相关的问题、我对一些技术的深入理解与思考等。之后就轮到我问了面试官一些问题。大约到了5点出头，面试官看了一眼时间，我估计面试官可能在思考下班的事情了，就只再问了一个问题，结束了面试。随后HR小姐姐送我离开了公司。

毫无疑问，Optiver的面试在我参加的所有面试中最具有特点。这并不完全是说面试的难度很高或者压力很大（当然难度也确实很高），还包括面试的形式、内容、考察的能力等等。通过面试，需要具有扎实的基础、严谨的思维、优秀的沟通能力、出色的抽象问题并解决问题的能力等等。更重要的是，在与面试官讨论设计方案时，我可以说是首次在面试中感受到了exciting的感觉。

面试的第二天下午，我接到了HR小姐姐的电话，通知我通过了两个部门的面试，并问我要选择哪一个部门。根据我的Profile，公司似乎更希望我去做ML相关的项目，但是我坚持要了System部门的offer。HR小姐姐告诉我如果是接System的offer的话，那第二天上午还需要再接受一场电话面试，考察一些CS基础，我欣然同意。

电面如期而至，这次是更加细致地考察了CS的基础知识。又过了一天，HR的offer call到了，薪酬比其他公司开的要高不少。我后来才知道，这是Optiver秋招唯一的一个System Developer的offer。HR给了我一些时间考虑Offer的选择，并建议我与尽量多的朋友交流。

腾讯数据库内核

从Optiver onsite面试回来后，我发现了妹子的事情似乎比自己想象中更加复杂一些。在了解到了更多的真相之后，我的第一选择就不再是阿里了。我去喝了点酒>_< 也是过去几年内第一次喝酒，然后就引发了酒精过敏。过敏当天腾讯的电面过来了。为了避免上次被Location不合适的部门打捞的问题，这次特地找人内推了上海的数据库内核团队，并且投递的是开发岗。面试官主要问了一些操作系统与数据结构的知识，还问了一点算法题和计算机网络。由于我简历上都是机器学习的项目，面试官就没有问，面试也很快就结束了。第二天大团队在深圳的leader面我，考核范围与之前相同，但问了很多Linux系统相关的东西。面完问我愿意去深圳吗，我说我肯定不考虑上海/杭州以外的office，面试官说那行，给你上海的offer吧。10分钟后是腾讯的HR面，面完告诉我说等着接收预录取的邮件。总得来说这次面试的体验挺不错的，面试官们水平都不错，也很和气，HR也非常和气。

过了两周左右，腾讯给我开了价格，让我稍微有些惊讶地开了个很高的价格。据我所知，国内的公司向来非常看重项目经历与岗位匹配度，因此我有点惊讶像我这样背景与岗位完全不匹配的候选人竟然能拿到这样的offer。看来腾讯的面试官们对我的潜力还挺看好的>_<不过我当时已经接了Optiver的offer了，因此也就直接拒绝了。不过让我有些没想到的是之后腾讯又联系了我好几次（4次以上？），非常有诚意，也感谢各位腾讯面试官的欣赏。其实这个offer也是一个挺不错的选择，工作内容有意思又核心，待遇也挺高，部门加班也不算多，如果没有Optiver的offer，我可能也会考虑这个offer的。

Google

面完腾讯之后我又作死地去喝了一次酒>_<这次终于引起了相当严重的酒精过敏问题，让我一直发烧，连续几天都睡不了觉。喝完酒的第二天刚好是中秋节，回了趟家，并且因为不敢让父母知道自己喝酒的事情，没有去医院解决酒精过敏问题，只好终日在床上躺着。

Google的转正面试在中秋节结束后的第一天。很遗憾我在去上海前并没能成功地睡着。因此大概是72小时没睡着地去了上海。在上海的时候碰到了一起来面试的NLP大佬工友F，与F一块去吃饭的时候进行了一些关于text style transfer的学术讨论，感觉挺有意思。也许是因为与人讨论学术问题带来的愉悦感，也许只是因为酒精过敏终于好转了些，当天晚上终于成功地睡着了一会儿>_<

第二天到了Google的office进行面试。第一位面试官是毕业于Stanford的资深Googler，问了比较多的算法题。第二面是一位相对年轻一些的Googler，也考了两道算法题。虽然因为连续多天的缺乏睡眠而有些疲惫，但我还是顺利地把算法都写了出来。面完之后HR小姐姐问我想不想去国外的Office，我说我只考虑国内的Office>_<

Offer选择

在与许多老师、朋友沟通后，我最终做出了选择Optiver上海的决定。这里还要特别感谢在Optiver的H学长，学长在收到我咨询的第二天竟然直接来了浙大与我面谈Orz 至于我最终做出这个选择的理由此处就不说了。这里引用一位浙大学长求职经历帖的话："如果你跟我足够熟的话，私下问我我会告诉你的。或者你就当我是选择了给钱给得多的offer就好了。"

这里简单谈谈我从算法岗转到开发岗的理由。这些理由在前文中也多多少少地提到了，此处也就不再赘述。其中最大的一个concern就是我对算法岗的工作内容有些不感兴趣。我担心自己对Optiver的data scientist岗位也不感兴趣，这才坚持选择了System Developer。不过公司在知道我的concern之后表示，我可以到公司之后看一看data scientist在做的事情，如果感兴趣的话也可以随时转岗，甚至两个岗位的项目一起做>_< 另一方面，我也觉得自己的工程能力还是太弱了一些，也想借着从事System Developer的机会把自己的工程能力培养起来。

最后再简单谈谈为什么我会选择金融行业。这个问题的答案其实非常简单，因为Optiver是金融行业的公司>_<我选择了Optiver，所以也就这样选择了金融行业。

确认选择Optiver之后，我的秋招到此算是结束了。虽然我后面还面了几家公司，但主要也是奔着与面试官交流去的。

一些学习资料推荐

这里简单推荐一些我看过的（有些看完了，有些看了部分）适合自学的学习资料，以供参考。因为我太懒了，所以覆盖的内容不全面，自学CS的同学也可以参考MIT的课表看看要学哪些内容。再次强调，**上课不做题，等于白上课。**

数学

MIT 18.01，MIT 18.02：微积分。

MIT 18.06：线性代数

Harvard Stat110：概率论。印象深刻的有一句话，"Random variable is a function"。

MIT 6.042J：Mathematics for Computer Science. 在我看来最有趣的数学课，如果TA能少上几节课就好了:(

CS导论

MIT 6.001：Introduction to Computer Science and Programming in Python

数据结构与算法

MIT 6.006：Introduction to Algorithms，教材是CLRS。

MIT 6.046：Design and Analysis of Algorithms. 进阶版，但其实前面那门课就已经内容不少了。

算法笔记：胡凡著。一位浙大学长写的使用c++实现基础数据结构与算法的书，写得挺清晰的，里面的代码也很实用。我现在面试之前还会翻一下这本书上一些算法的实现。

操作系统

CMU 15-213：教材是CSAPP。

MIT 6.824：分布式系统。不久前终于感人地放出视频了。

MIT 6.828：其实还没看，小伙伴都说好。

MIT 6.004：Computer Structure。偏硬件，老师讲得很好。

Modern Operating System

编程语言

Stanford CS41: Python Language Programming. 可惜没有视频，作业挺有意思的。其实我觉得学语言比较好的一种方式就是学了语法之后做一些练手的项目、作业，这样会掌握得比较快。

Stanford CS106L: C++ Language Programming. 前文也介绍过了，我觉得那本Full Course Reader写得真的很好，推荐一下~

还有一些领域内很知名的书，这里就不再专门推荐啦，这些课程里也会有介绍/推荐。这里再啰嗦一句，对大多数人来说，往往只需要学会某种编程语言中20%不到的常用特性，而这20%的常用特性往往占了实际使用这门编程语言中的90%。所以我不是很建议一开始弄本厚厚大大的书看，那样又吃力又缺少反馈，而且可能会花大量精力在自己根本用不到的地方。我其他的编程语言似乎都是用官方tutorial入门的，这里也就不推荐啦。

机器学习

浙大的同学来蹭蔡登老师的课>_<

Coursera Andrew NG的Machine Learning：机器学习之路从这里开始。

Stanford CS229：Machine Learning. Youtube上有视频。

CMU 15-701：Intro to Machine Learning. 就找到这么一年有视频的。

Coursera PGM：知名课程。

CMU 15-708：Probabilistic Graphical Model 我只看过两个Lecture。

下面有几本书，不过还是推荐和课程一起看。

Pattern Classification (PC)：比较老的书了，但是内容还是很有意义。

Elements of Statistical Learning：insight很多。

Pattern Recognition and Machine Learning (PRML)

Machine Learning: a Probabilistic Approach (MLAPP)

Information Theory, Inference, and Learning：可惜David J. C. MacKay英年早逝。

深度学习

Stanford CS231n：Deep learning for CV. 推荐深度学习从这里开始

Stanford CS224n：Deep learning for NLP.

Berkeley CS285：Deep RL. 我只挑着看了几个Lecture，小伙伴说好。

Deep Learning Book：花书。

尾声

本来这篇的主要目的是总结一些自己的面试心得供大家参考，后来又加入了自己自学CS历程的介绍、心得等，又介绍了自己找工作的历程和思考，反而使得本文的后半部分显得有些喧宾夺主了>_< 此时突然想到一位师兄博客里讨论的话题："如果能回到十年前，你会对过去的自己说什么？"如果我能回到开始CS的旅程之前，我也许会对过去的自己说："你会经历一段难以置信、跌宕起伏的旅程。你会遇到很多志同道合的好友，以及你十分尊敬的师长。虽然你也会走很多弯路，碰到很多困难与挫折，但是不要担心，你一直兢兢业业、勤勉刻苦，你的努力最终都会得到回报。"

总之，我希望这篇文章里所写的自己的一些经历、思考、心得、总结等能对后来者起到一些帮助。如果你在阅读了这篇文章之后觉得有一些收获，那本文的目的也就达到了。

各位同学，我们江湖上见。