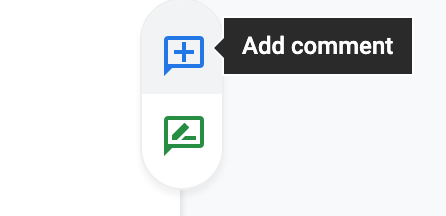
From Zero To Hero（Ongoing）

这个doc主要介绍从进入techbow学习，到最终拿offer离开，每一个阶段该干的事，会用到的资料。自上而下，按照时间顺序依次进行。但面试准备本无顺序，学习成长的过程也是螺旋式的。很多时候在准备完stage 1 - 4，进入面试环节才发现自身的缺陷，然后开始补强缺陷部分。所以，这个stages更像是将所需要的知识模块列出，需要补强时有个学习的依据。

请在不明白的地方加上comment以及自己的问题。

任何不清楚的地方皆可问。哪怕不知道如何提问也请把不知道如何提问的点写下来。感谢大家！

# General

## 校招

1. TIMING！！！！校招不需要任何工作经验，先到先得！当校招职位放出，第一时间递上简历。不需要内推，越早递送，概率越高。
2. 参考stage 1，2，3，7
3. 春招和秋招。亚麻每年一般有2次校招。此外，Google，Facebook这两个一线大厂也常有校招放出。其他二三线大厂在年初也会有校招职位。
4. 简历。简历只要适当有一些SDE工作经验即可。可以是intern经验或者短暂full time经验。
5. Online Assessment（OA）。每年亚麻OA都会不太一样，往年OA有三轮，2021年秋招只有两轮OA，一轮coding一轮Psychological test。其他大厂OA都只有coding
6. Video Onsite（VO）。

## 社招

1. 按照时间顺序从1开始到7。当stage 1刷题步骤完成，即可开始投简历，之后只要保证每天刷一点题保持手感即可。从开始海投/撩hr/内推，到有hr接触你，到oa/电面，到oniste，中间还有一部分时间足够你去深挖project，准备bq。
2. 原则上先stage 1 再stage 2，但时间紧张的同学请1，2一起进行，写完简历以后，一边刷题一边投简历并深挖项目。

## Intern

TBD

## 知识层级

算法 → OOD → System Design → Project → BQ

1. 从面试所需知识的spectrum上看，算法是OOD的基础，OOD是System Design的基础，System Design是project的基础，从左往右知识层级越高。但并不是不学好前面的就没办法学后面的，而是前面的知识了解越深，越对后面一个知识有加强作用。
2. entry level面试中，算法和project是重点考察对象。对design，无论是纯ood和system design的要求比较低，甚至不做要求。也有有一些特例出现，比如在年末，entry level headcount不多的时候，会出现sde 1和sde 2同时面，这时候面试内容也会出现design。但挂掉design并不意味着就拿不到offer。
3. 算法也就是刷题，大弓有大量的资料帮助刷题。需要提醒的是有一些很像OOD的算法，也会考察一些class之间相互关系和调用的题目。
4. OOD。正统OOD是不太写code的，更多focus在class fields，methods interface以及class之间关系的define上。
5. System Design。不写code，只画图。
6. Project。从BQ中体现Project的内容。

# Stages 1 刷题

## 针对完全0基础，听不懂算法陪练课的同学

1. 听不懂才是正常的。算法陪练是综合了Algrithm，Data Structure，以及Java Fundamental，等一系列本科研究生课程的综合面试课程。除非是CS专业，或者有一定刷题基础的同学，否则很难快速了解课上提到的概念。
2. 时间不足或者人在美东的同学推荐把精力放在加速听录音课，而不是晚上跟课。原因如下，晚上跟课精力会跟不上，老师上课介绍的知识点不会给你快速回放重听弄明白的机会，这样会让上课变成一种煎熬。快速听录音，在听不懂的部分暂停，然后通过回放，或上网找知识点，或问同学，或问老师都可以让你听课效率变高，并且不会有跟课怕被刘老师点名的心理负担。
3. DFS的学习方法。每听到一个知识点，深挖，找到相关的知识点继续查。适合喜欢钻研的同学。坏处是很快就会lost，忘记自己到底在查什么，并且听课速度变得很慢。好处是能快速拓宽知识面。
4. BFS的学习方法。每听到一个知识点，Google或者wiki一下什么意思，大概理解一下，理解不了就跳过，继续听课。好处是能快速将需要的知识点听一遍，在闹钟声会有一个印象，下次再遇到就不会非常陌生。BFS学习方案的好处是目标明确，以听完课为目的。坏处是漏过的知识点可能需要回来补。
5. 第一遍“跟课”大概2个月时间，目的是熟悉算法陪练的内容。第二遍跟课请跟这刘老师写代码，作为自己听课效果的检测。

## 初阶刷题，跟算法陪练课

1. 上课的目标  
   记住刘老师说的解题思路为目的。所谓思路是一道题解题的flow。刷题的时候一般也是先有思路，然后根据思路去实现算法陪练大部分的题。  
   比如bfs中，需要一个数据结构去储存层级的数据，然后loop这个储存做层级遍历以及propagation，那么根据这个思路，就需要建一个Queue，然后用一个while loop去遍历，在while loop里面对poll出的data做propagation。
2. 有关刷题  
   在听课学有余力的时候，按照topic/章节刷题。  
   刷题的目的是把课上记住的思路，用代码去实现，巩固记住的思路，帮助熟练掌握数据结构的API，以及提升刷题手感。  
   换topic之前，回顾复习总结一下当前topic刷过的题，看看哪些code在这个topic下面出现频率最高，这些code一般可以复用，可以认为是代码模版，有哪些是不一样的，不一样在什么地方，下次遇到类似情况能不能使用这个解题方案/code。
3. 刷题能力的提高体现在能handle的code越来越长，结构越来越复杂  
   算法陪练综合演练章节之前出现的题目，大部分是implementation并不长的code，一般10行左右。当这些题目熟练掌握之后，会成为其他implementation较长，难度较高题目的基石和砖块。每一道记住/熟练掌握的题目会成为你今后写code的基石/建筑材料。
4. 保证数量和质量  
   陪练大概有250题左右，只要把这250题刷完，并且能融汇贯通其中30%左右的题，就已经基本达到可以开始面试的要求了。原则上算法陪练的题刷完即可进入下一个stage，开始投简历了。[老刘算法陪练参考答案](https://docs.google.com/document/d/1SnQoeeUrpUdTAR7JiC6ge-Pto6sFlLSTGfv_kmmeKUI/edit?usp=sharing)
5. 资料  
   大弓可用的刷题列表，[老刘算法陪练参考答案](https://docs.google.com/document/d/1SnQoeeUrpUdTAR7JiC6ge-Pto6sFlLSTGfv_kmmeKUI/edit?usp=sharing) by 方脑壳，或者按照课程的流程刷[拿offer全靠勇哥](https://docs.google.com/document/d/1AkjYLZsrhJ4g6b0kR2_F0KUfMrLwBsqojqwiTOf-X20/edit?usp=sharing) by Bo哥，或者用[笨笨学姐的checklist](https://docs.google.com/spreadsheets/d/1jJHTcSBRMdbzQs5L9hdQ0bW-Qv9x-HEiRSNBTNzGadI/edit?usp=sharing)，checklist先复制一份到你的drive再使用。

注册leetcode账号。有一些题目是被lock的，只有会员才能看，并且注册会员可以使用debug的功能，类似于工作时使用IDE打断点debug。leetcode-cn中文，会员更便宜。

1. （optional）用github和ide每天交作业。ide的自动补全功能可以帮助初学者快速学习各种api，github和git是一个工作中每天使用的工具，一遍刷题一遍练习git一段时间就会让工作平顺很多。

注意事项

1. 算法学习的时间不要拉的过长。强烈建议在尽量短的time span内集中完成学习。拖的越久越疲劳。越容易学了后面忘记前面。
2. 学习Java这门语言是为了刷题，请勿本末倒置。Java本身可以通过大量刷题去学习。
3. 半小时原则。每道道题目做题时间不超过半小时，10分钟想不出思路，直接看答案，有思路但写不完或者花费很久才能写完，直接看答案。
4. 难题跳过。遇到太难，看不太明白的的题直接跳过。也许这道题超出了你的能力水平，即便花很多时间去刷，最终很有可能还是记不住写不出来。但你的刷题水平提高之后回头再看这些“难”题也许就不那么难了。

## 进阶刷题

1. 目的  
   面对一道较为复杂的题目，学会如何构建code的结构。听算法加强时候，算法哥会把每一道题目high level的解题步骤先讲出来，一般稍长一点的题目至少有3，4个步骤。构建这种结构性的思维模式是进阶刷题要达到的最主要目的。有了这种思维模式，就开始有自己的coding style了。

拓宽知识面。很多题目的类型，解题方法，算法陪练并不会涉及，但在面试中经常出现。

1. 如何开始

算法加强，建议听完算法陪练，或者能在没有任何外界辅助的情况下，做出LC 200 Number of Islands，BFS，DFS两种方法的code，不要求bug free，可以有corner case和typo的问题，但必须要求思路清晰，结构清晰，就可以开始。算法加强给你的是内功，主要增加算法基本功以及对题目结构上的理解和提升。如果能够融汇贯通算法加强，可一战Google。

算法冲刺，建议听完陪练可以开始。实战风格，会讲大量面试真题，面试常见的特种算法，算法加强中不会cover的题目。

## 面试算法轮flow

老刘0 - 7的面试流程

1. 读题 & 理解题意 & clarification。这一步主要是通过交流，搞清楚面试官的意图。  
   交流很需要技巧。可以通过mock，自身面试，看别人面试积累经验，形成自己的一套交流方法。  
   在刷leetcode的题目时候是通过阅读搞清楚题意。面试的时候可以通过猜想印证的方法去得到想要的答案。所谓猜想印证是指在面试官给出题以后，给出一些sample并给出答案，询问面试官正确与否的方式，在back and forth几次之后理解面试官意图。
2. 解题 / 展现思路。主要是向面试官展现大体的解题思路，时间尽量短，只要让面试官认可即可。  
   这部分主要需要展现2个部分，第一，data structure的选择，如果题里面已经给出了抽象的数据结构，比如Tree，Graph，Linked List，Array，那这一步可以跳过，第二，method / algorithm的选择，例如bfs，dfs，linear sweep，sort。之后需要给面试官解释如何使用这个方法把需要的答案找到。  
   例如在number of islands这道题里面，第一步是loop整个matrix，当遇到1的时候，用BFS/DFS的方法搜索和当前1相邻的1，并mark visited或者flip成0，在完成一整个BFS/DFS之后，result +1，最终把result return。
3. implementation。这一步骤主要通过刷题去练习，在此不去赘述。
4. check the code。自己检查一下是否有typo，是否有bug，corner case，逻辑对不对，是否达到面试官的要求等等。
5. demo。可以自己找一个简单的case，自己给自己demo一下。顺便也可以检查code有没有问题。
6. time & space complexity。

# Stage 2 写简历

## 构建自己的项目

1. 粗听项目课。可从飞哥，宋扬老师，小熊，或者别的项目课程中任意一个project课开始学习，尽量找有简历模版的课。了解整个项目的架构框图，use case或者叫适用范围。项目课给大家提供的是项目中的骨骼。  
   例如，小熊老师讲过一个near real-time data pipeline的项目。你需要粗略了解有几个service从inject service，rabbitmq，processor service，最终到DB。需要了解这个项目的适用范围，这个project可以，其目的也是是承接大量的，至少上千的qps。之所以称为near real-time是因为承接的数据不是立刻处理返回给用户，而是有可能是给data scientist作为后续处理的data source。
2. 马甲。马甲本身来源于生活，可以是你的爱好，你自己转码前的项目抑或是其他你熟悉的东西，只要这个小的领域里面某些部分可以通过搭建automation来替代一部分功能，达到省钱或者赚钱的目的，这个马甲就是可行的。如果有已知的产品在market上会更好。  
   比如上面提到的这个near real-time data pipeline的项目适用于一些IoT的场景。比如在用户端会有很多sensor监控读取用户端的数据，这些数据由near real-time data pipeline负责，根据use case，存进对应的database或者file system，以待日后使用。
3. 粗略估算自己公司，项目规模。这一步决定了有些项目课中听到的技术栈在你的项目中适不适用。
   1. 你们公司有多少人
   2. 你们组有多少人
   3. 项目上线了没有
   4. 流量，QPS/TPS有多少
   5. 有多少用户
   6. etc

如果你的项目是个大项目，traffic比较高，那么很多技术栈都是有应用场景的。但是如果流量不大，用户较少，那么过大的技术栈，比如rabbitMQ，Kafka这些就很难有适用场景。解决方案是适用最基本的service框架，比如spring，然后简历内容focus在产品的use case，feature或者叫功能上。

## 准备简历

1. 不犯错。一份好简历首先需要不犯低级错误，比如typo，缺少关键信息，gramma mistake，这些错误会让hr觉得这个candidate不合格。将hr或者hiring manager需要的信息清晰给出即可。
2. 对于一个candidate，hr只关心一下内容，姓名，联系方式，疫情前需要城市信息安排面试住宿，感兴趣岗位，工作年限。
3. 选择简历模版时，要求结构清晰简单明了，可以有色调但不要过于花哨。
4. 内容，社招。（[Template](https://docs.google.com/document/d/1Qqe8F1cFfPMDc51jkXUlRkB7EzAX5dt2Vy6c3XLwHwE/edit?usp=sharing)）
   1. 电话，姓名，邮箱必须有，地址写到city和邮编即可。
   2. 尽量在显著的地方告诉HR你找的职位是SDE，偏backend。
   3. 第一个section为keywords，主要介绍tech stack。
   4. 第二个section是working experience，写清楚公司名称，title以及工作时间。
   5. 第三个section可以是projects，如果没有projects则略去
   6. 第四个section为education。
5. 内容，校招。
   1. 电话，姓名，邮箱必须有，地址写到city和邮编即可。
   2. 尽量在显著的地方告诉HR你找的职位是SDE，偏backend。
   3. 第一个section为keywords，主要介绍tech stack。
   4. 第二个section为education。如果还没毕业，把毕业时间后面加上expected
   5. 第三个section是working experience或者是projects，working experience写清楚公司名称，title以及工作时间。
6. 简历的风格。  
   当我们工作经验不足，或对组里的contribution不多的时候，流水账是一个比较常见的方式。比如有一种方式，简单罗列出有哪些service，并介绍每个service的功能，作为简历中的bullet point。  
   进阶一些的写法可以以feature为stand point，以完成过哪些feature，功能是什么，做这个feature的意义是什么，achievement是什么，你的contribution是哪个部分。  
   再高级一些，以project为主，launch过哪些project，lead过哪些project，带过多少人的团队等等。

## 关于小号

视频地址：techbow.com -> 我的求职 -> TPMP Special【必听】-> Oct 06 7:00 PM - 10:00 PM NO. 4 答疑 by 博哥

TPMP Special: <http://techbow.com/app/course/interviewtips>

音频: <http://techbow.com/app/lesson/61615539b256295443fcc5fc>

# Stage 3 投简历

## 投简历

1. 用LinkedIn撩hr。主动出击，在LinkedIn上通过加hr，找到活跃的，手上有headcount的hr。这个方案对投Amazon的和有一段工作经验投MS的candidate极为有效。对毕业时间比较短，工作时间不够长的投MS的candidate不太有效。

第一步建LinkedIn账号。

* 1. LinkedIn账号最主要的作用是用来主动reach out hr。所以内容可以不用很多很全。你可以把Linkedin当成聊天工具。
  2. LinkedIn账号的邮箱电话需要和简历匹配，方便管理。
  3. **不要**把简历放在LinkedIn里。首先，简历放进LinkedIn会有各种莫名其妙的人给你打电话。其次，比如在投亚麻的时候，有多个hr联系你，如果hr在页面能找到你的简历，会直接把你lock住，失去自主选择权。
  4. 打开open to work选项。

第二步使用Linked Helper。视频地址：techbow.com -> 我的求职 -> TPMP Special【必听】(<http://techbow.com/app/course/interviewtips>)

* 1. 【通用】如何写简历，如何撩HR，如何答BQ (<http://techbow.com/app/lesson/601b2eac426cfd2f559ee91b>)
  2. 【拿面试】如何在Linkedin上撩HR (<http://techbow.com/app/lesson/601b2ce5426cfd2f559ee916>)

1. 海投。在LinkedIn上或者在公司官网上找到你感兴趣的职位，直接投简历。尽量找和自己工作年限匹配的职位。如果是Google，FB，LinkedIn这三家需要严格匹配，JD上要求2年就必须有2年经验，并且尽量找朋友内推。其他公司如eBay，PayPal等，需要尽量匹配，年限不够其实也可以尝试。
2. 内推。找到朋友，在内部的系统找到合适的职位进行内推。如果方便，让内推的朋友帮你给hiring manager写一份邮件，简单介绍你的履历，同时附上你的简历。
3. 大厂 vs 小厂。  
   大弓的套路不适合小厂。小厂偏实操，轻面试套路，在算法层面上考察不重，而是对实际完成project能力更加看中。大厂更看中你是否聪明，是否有潜力，因为大厂用的infrastructure大概率是自己搭建的，进公司以后需要自主学习使用。
4. 中国公司。如何看待腾讯、字节、阿里、小鹏等中国公司。

# Stage 4 完善项目

project课程只能给你的项目提供基础知识，具体的项目马甲，细节以及项目相关的BQ需要自己去完善。

需要任何帮助可以预约老师或助教的1对1

## 深挖project

1. 准备初步的System design的知识。看一下宋扬老师System design前几节课，以及飞哥的project课，初步了解一些基本的concepts，例如DAU，QPS，SLA，online/near-realtime/offline service，functional requirements，non-functional requirements，etc，以及这些概念在project中如何体现
   1. DAU，任何一个产品都会有人在用。也许你的产品不一定有明确的DAU的数字，但至少你要知道客户群体是谁
   2. QPS，根据DAU可以粗略估计QPS，每个service的QPS都是不一样的，你说参与的项目的QPS都是多少？
   3. SLA，这是一个商业agreement，online service的SLA会以秒记
   4. API / Interface。service会收到request，有的会return一个带有body的response，有的则是只有200，400 + error info这些基本信息的response。
   5. DB schema。
   6. business logic。每个service的内部logic主要用来处理以及运送data。比如一个aggregation service，会make很多call去不同的data source，把所有需要的data聚合到一起，传到下游。而call的这这个步骤就是你需要了解的部分。
2. 不要Overkill。所谓overkill是指，一个非常小的产品，用了一个scale非常大的架构。一个只服务几十一百个人，一个月也没几次会被visit的产品，却有着能承载上百万qps，服务几十万人的架构，这本身就不合理，这种不合理在小公司，初创公司的背景下会更加不合理。
3. 给自己准备人设
   1. 进组的时候，组里的产品开发到了一个什么程度？是你从0开始和组员一起开发的，还是已经有一定基础了，你在已有的service上添砖加瓦？一般情况后者居多。
   2. 你自己有多少工作经验？你在这个组工作了多久？一般一个entry level engineer花在ramp up上的时间至少3个月，这3个月里面，你可能对整个组没有任何的contribution。3个月之后开始能够完成一些简单的task。
   3. 你对这个组的贡献是什么？开发了什么feature？
4. 了解工作日常
   1. Sprint。sprint以2周为一个unit，开始的第一天会给组员分配任务，最后一天会做一个sprint demo，展示你在这两周的工作成果。
   2. Daily stand up。汇报前一天干了什么，有没有blocker，今天的plan是什么。
   3. 一个产品从无到有的life cycle。每个公司的dev cycle是不一样的。  
      scope -> design -> implement -> test -> deploy / rollout / on production -> monitor -> A/B test -> improve the product
5. 准备和工作时间相匹配的tasks/features/projects
   1. 你们组的services和business logic。在项目课上，你可以认为老师是tech lead，在给你介绍你们组的架构以及这些service的用途也就是business logic。
   2. 自己负责的或者做过的部分。比如你负责写了一个ingest service，aggregation service，etc. 或者是这些service中的一部分，而你写的这些feature主要完成了什么功能。每一个功能就是一个feature，每一个feature又可以拆成好多task。在面试的时候是以feature或者projects为单位给面试官介绍的。

## BQ

1. bq故事核心。故事内核分2个部分
   1. 发生的事件主干，比如miss deadline里面，可以是合作的队友生病，你去cover队友的部分导致almost miss deadlien。
   2. 对应的principle。principle源自Amazon Leadership Principles。这个决定了你这个故事能够回答的问题。比如a里的miss deadline至少可以cover ownership和delivery这个两个principles
2. Trade off。比如在1中提到的故事，其核心trade off其实是perfect product vs delivery on time。这个trade off的两个选项不应该有对错，只应该在某些情况下面更倾向于其中一个选项。在这个故事中我们选择了delivery on time，可以给出一些原因比如这个项目是在一个更大的cross team的框架下的项目，你们的delay会让整个项目delay。但在某些情况，比如你做的feature是一个customer face的产品，你需要考虑产品的使用体验。贸然rollout出去一个未完成的feature很可能会导致客户流失。
3. 亚麻16条Leadership Principles。在1这个故事里面我们至少可以cover ownership和delivery这两个principles。ownership体现在，这个情况本不是你的问题，但你却take the ownership，把整个项目看成是你自己的项目。delivery本身就是这个故事的核心
4. STAR原则。面试的是一个交流的过程，最终目的是让面试官认可你。而使用一个结构去承载你的bq会让面试官更容易理解你的意思，让整个面试变得更加平顺。以1中miss deadline为例子展开STAR
   1. Situation，故事的背景，1句话说完。在完成XXX project的时候，队友因为XXX原因离开的project，导致项目可能delay的状态
   2. Task，任务，1句话说完。需要按时delivery整个项目。
   3. Action，你的move，一般2，3个action即可。  
      1. 我立刻找到manager和techlead第一时间开了一个紧急会议。在会议中，我汇报了项目进度，已完成模块，正在完成的模块和未完成模块。  
      2. 我提出为了按时完成项目，把我手中其他一些不重要的项目暂时后延，把当前项目重新评估，去掉了一些并不是非常关键的模块，等到第二期去完成。
   4. Result。在deadline之前完成任务。
   5. What you learn。下次遇到这种大项目的时候要给更多的buffer time。
   6. If you do again what you can improve。
5. Sample BQ。[Link](https://docs.google.com/document/d/1Hwib2ORf6txT_FayDuxkEYtsgu_-icj59JZwa3DNylk/edit?usp=sharing)

# Stage 5 OOD & SYSTEM DESIGN

## 学习

OOD和System Design的学习不像coding，需要一个短期集中时间的训练。这两个偏design的topic更像是一种底蕴的积累。在有一定的工作经验和阅读一些design的资料之后的一种解决实际问题的方案或者叫套路。

这里主要focus在纯design，一些需要写algorithm的面试在这个topic里面不会cover。

### OOD

1. OOD陪练。刘老师的OOD陪练更多的是告诉你什么是OOD，会以一种接近coding/算法的方式呈现给你一个不同于算法的东西。
2. OOD加强。算法哥着重教的是如何design。
3. Grokking OOD interviews。非常detail的一个OOD课程。需要链接找Bo哥。

### System Design

1. System Design课程 by 宋扬。以Designing Data Intensive Application为基础，主要介绍了System Design里面的一些high level的知识。有一定难度，但是highly recommend。
2. Grokking system design interviews。一本case by case的system design的书。主要提供了一些design的sample，知道generally system design到底在干什么。

## 面试

正统design套路。一般正统system design / ood，都是需求面试者能够主动，自发地BB 45 min ～ 1 h，将需要design的内容systematically展现给面试官。

面试时候搞清楚到底是OOD还是System Design非常重要。OOD是不需要处理scalability的问题的，所以如果面试时候当你提到non-functional requirement或者在聊DAU，traffic这些话题的时候，面试官表现出不感兴趣或者明确让你不用担心这些，请把所有精力放在functional requirements，把这个面试当成OOD去聊。

1. clarification。内容涉及预估traffic，预估DAU，预估permanent storage所需要的空间；给出预想的functional requirements以及询问extra functional requirements；给出合理的non-functional requirements。
2. 画图。根据basic functional requirements画出service design draft。可能是层级结构，例如可以是 client tier -> routing tier -> frontend -> backend -> permanent storage tier，也可以是star的结构。在这一步需要define有哪些service，具体的功能是什么，有哪些database / filesystem / cache，具体作用是什么。
3. design service interface以及db schema。db table具体schema长什么样，有哪些field，table和table如何关联。service interface主要define一个service有几个api，这些api的request和response长什么样，high level的说一下这些request如何被handle。
4. scalability & further functional requirement。如果聊到这一步基本上你的system design就已经过关了。至于需要解决的问题是scalability，high traffic带来的non-functional requirement，还是面试官提出的其他的functional requirement，需要你的面试官沟通，因为一般到这一步的时候，时间往往不多了。

# Stage 6 面试

## 面试

1. **Practice makes perfect。**多面才能提高面试技能，多面才能提高拿offer的几率。
2. **复盘！！！**面试是个不断练习的过程，无论失败多少次，只要最终成功1次，就上岸了。所以每次面挂，请找老师复盘，补强自己的不足，迎接下一次面试。
3. Thank you letter。面试完记得给面试官（如果面试官给了或者能简单找到的话）和hr发一封thank you letter。不用很长，但记得一定要customize，尽量不要照抄模版。面试官的letter可以加入一部分面试当场发生的事情，问的问题，etc.。

## Amazon

1. 题。刘老师算法陪练，算法哥经典章节，黄金110。
2. BQ。

## MS

1. 题。刘老师算法陪练，算法哥经典章节，黄金110。
2. BQ。

## LinkedIn

1. 大号微软。
2. 题。难度与Amazon类似。
3. System Design。必有一轮
4. Technical Talk
5. BQ。

## Google

刷题。算法哥的视频看完并把算法哥的题刷完。这样才有一战Google的资本。如果算法哥的题刷熟了，可以按照topic刷Google tag。

## Facebook

1. 题。FB有题库，量不大。把FB的题库刷熟。
2. BQ。除了基本的BQ以外，需要准备一些FB情怀相关的BQ，比如Why FB。

# Stage 7 准备入职

## 谈package

1. argue package最基本条件是需要和hr对等。有“你不着急我们就耗着的态度”的时候才有谈判的资本。一般建议如果有大厂背景的可以利用自己大厂背景，如果没有的话可以用其他家offer去compete。
2. 无论如何不要透底，不要先开口给具体数字。

## 背调

一般公司背调都有多种方案，常见的是一下两种方案2选1

1. payroll，W2。不会对当前工作产生影响，能直接证明雇佣关系。
2. 找supervisor做背书。找自己的上司，manager或以上都可以。

## 准备入职

1. 学习git相关概念 （[大弓交作业wiki how](https://docs.google.com/document/d/1E-OuqDvlKeLTCWUvfRx8TTqcBtFPWFX9pY7m6KhZ5uE/edit)）
2. 了解project dev的cycle
3. 学习如何看code
4. 如何和老板1v1

## 