亚马逊秋招面经
亚马逊的研发岗位叫做 SDE，我面试的就是这个岗位。
一面：
1 多线程的通信，同步方式。面试官问我 volatile 和 synchronized 的区别。
我从底层原理方面讲了两者的实现方式，主要解释了 synchronized 基于 mutex lock 实现，重量级锁，需要从用户态切换到内核态。volatile 则是通过插入内存屏障的方式，保证变量读写的可见性。
2 集合类用过哪些
当时回答的是 ArrayList，linkedlist，hashmap 这类简单的集合类，于是面试官就问了 hashmap 的实现原理，以及和 chm 的比较，不同版本 JDK 的区别。也是比较老生常谈的问题了。
3 JVM 的内存模型
说完内存模型的主要情况，面试官开始问我关于内存泄漏和内存溢出相关的问题，也比较简单，另外还问了 GC 相关的问题。
4 NIO 和 BIO 的区别
主要就是讲述 BIO 的阻塞式 IO 读写，然后讲一下 NIO 的实现原理，相关类，并且说到 IO 多路复用的实现方式，底层的 epoll 实现方式等等。
5 算法题：实现一个 LRU
只要求实现超过容量时的缓存淘汰，不用处理缓存超时的问题，所以只要写一个双向链表来存节点即可。另外，不能直接使用 linkedhashmap 实现，所以直接用 hashmap 作为成员变量，另外写一个含有容量参数的构造方法即可，也是比较常见的问题了。
6 n 个有序链表合并，怎么实现。
刚开始说的是直接合并，然后优化使用多机进行。后来看了一下，可以用最小堆实现。
二面：
这轮是经理面
1 项目
2 项目的多线程问题
3 项目的架构
4 项目的数据库使用，部署方式，缓存部署方式。
5 项目的实际场景
6 讲一下 JVM 的内存分区
7 秒杀系统架构设计
秒杀系统设计也是一个比较常见的问题了。一般可以分几个方面作答。
首先，前端限制访问时间，以及同一 IP 的访问次数。
然后，第一层的服务做负载均衡，比如使用 nginx，然后服务器做集群。
接着，可以用消息队列做削峰和限流，然后做一层缓存，最后只有少量请求到达数据库。
面试官问 nginx 怎么做高可用，前一层能不能再做负载均衡，我回答的是 nginx 也做集群，前一层可以用硬件负载均衡或者 dns 服务器做第一层负载均衡。
另外面试官还问了如果有 ddos 攻击怎么办，我刚好想到通过人机验证来避免大量肉鸡的攻击，于是就说了验证码的方式，面试官也说可以。
8 Redis 的分布式部署
说了 Redis cluster 的部署方式，其实就是分片加哨兵的部署方式，另外 Redis 还可以使用 codis 这类***来做分布式。
9 MySQL 的主从部署，读写分离。
这个就是比较常见的 MySQL 部署方案了，稍微说了一下实现方式就没再问了。

10 cap 定理
讲了一下为什么三者只能选两者，这个问题还是挺绕的。
11 负载均衡怎么做
12 kafka 的作用，持久化，其他问题
kafka 之前看了一些比较好的文章，但是时间一久就忘了，于是我就说了读写性能好，以及多个副本的部署方式。
13 前端解决一些无效的请求过滤，怎么做
14 有什么 offer，想去哪里工作。
15 有什么问题想问我的
亚马逊实习生面经
一面
1 聊项目 20 分钟，难点，重构特点。
2 写题，跳台阶的递归和 dp 方式。
3 如何查找一个数组中的局部最小值。
直接遍历复杂度是 O（N），但不是最优。
使用二分查找优化，逼近局部最小值，复杂度是 O(logn)，有一些问题，并且写出死循环了。不过面试官挺好的，让我过了。
二面
1 聊聊项目中的重构，数据表设计的方式。谈到了索引。
2 使用索引和不使用索引的区别，何时使用索引的查询效率不如不使用索引的查询效率。
这个问题懵了好久，以前没有遇到过。
面试官提醒了我一下，因为索引需要 logn 次 IO 找叶节点，然后再一次 IO 去真正找数据。而不用索引每次查找都要一次 IO，所以数据行为 N 时，不使用索引要 N 次 IO，使用索引，假设索引包括 M 行数据，则需要（logn + 1）* N/M 次，当两者相等时效率相近，否则不同。
3 聚簇索引，最左前缀匹配索引，B+树。
4 算法题 版本号的大小比较
首先正则表达式过滤非法输入，并且拆成数组进行判断。
这个就不难了，就是比如 1.1 和 2.2.1 这种类型的版本号比较。
5 算法题 一个单词变成另一个单词。
一次只能变一个字母，并且中间过程需要是一个合法的单词。并且每一个位置都可以修改多次。文最少需要变化几次能变成指定单词。
刚开始在纠结判断合法的问题，后来面试官说这个可以放放。于是开始考虑 DFS 方式，巴拉巴拉说了一下，后来发现单词不是按顺序修改字母的，所以 dfs 根本行不通，想了半天只写了个大概的 dfs 模型。当然其中还有很多交流，但是最后也没写对。
现在看来应该是用 BFS 来做，每次把所有的修改情况找出来，然后分别进队，接着每次拿出一种方案继续做下去，最后成功到达结果时就是最小的变化次数了。