8.25 就投递了，秋招没拿到面试机会。补招从笔试->一面->二面。
看到是补招，就想着面着看看。

笔试选择+4 算法+1 问答
60min 时间不够所以算法做的不是很好，但感觉算法难度不大
一面：
1、数据库
1.innodb myisam 的区别
MySQL5.5.5 以后，InnoDB 是默认引擎，因为 innodb 适合更多业务场景，原因是 i nnoDB 支持事务，支持外键。
除此之外：
锁：innodb 行锁，myisam 表锁
索引：innodb 采用聚簇索引+辅助索引，myisam 才用非聚簇索引（即：主键和非主键索引的查询速度区别不大

补充（我没答上来的）：
全文索引
MyISAM 支持 FULLTEXT 类型的全文索引
InnoDB 不支持 FULLTEXT 类型的全文索引，但是 InnoDB 可以使用 sphinx 插件支持全文索引，并且效果更好
主键
MyISAM 允许没有任何索引和主键的表存在，索引都是保存行的地址
InnoDB 如果没有设定主键或非空唯一索引，就会自动生成一个 6 字节的主键，数据是主索引的一部分，附加索引保存的是主索引的值
1.2 聚簇与非聚簇与非聚簇索引的区别
https://my.oschina.net/xiaoyoung/blog/3046779
1.3 索引覆盖是啥？
https://zhuanlan.zhihu.com/p/73204847

推荐的是我以前看过不错的文章

2、redis
redis 为什么快？（和 mysql 比）
- 它是单线程（6.0 之前），没有进程竞争，锁等设置，所以少了切换上下文的时间，相对快了很多。
- 同时，数据存储在内存中，他可以快速处理数据。
- 同时，它又是 epoll 的多路复用模式，异步的读取信息，自己要进行的逻辑处理也相对很少。并且可以涉及单机多 redis，充分利用其他 cpu 核心。

mysql 就是基于磁盘+B 加树咯

redis 常见数据结构以及使用场景
string，存储 json、照片、视频等各种各种可序列化的对象。
list， 如果对消息的可靠性没有较高的要求的话，那么就可以使用 Redis 去实现消息队列。

map，二级映射，存储对象更直观。
set，元素不重复又在内存，可以做合并数据等
zset，排行榜

推荐的以前关注的公众号博主敖丙的 redis 文章咯 ，里面挺详细的
3、设计模式
问了工厂模式和职责链模式

工厂模式：简单工厂、工厂方法、抽象工厂
动机：
工厂里提供很多方法，不同方法新建不同对象
看了很多解释我的总结有以下几点：
1、减去繁琐的 new 工作，统一让工厂创建对象
2、软件系统中经常面临对象的创建工作，由于需求的变化，这个对象可能也随之发生变化，但他却拥有比较稳定的接口。需要提供一种封装机制来隔离出这个易变对象的变化，从而保持系统中其他依赖该对象的对象不随之需求变化而变化。
其次：
三个工厂模式，各有千秋
从简单工厂模式——》工厂方法模式，解决了对产品的拓展不符合 OCT 原则的问题
从工厂方法模式——》抽象工厂模式，解决了一个过程只能生产一个产品的问题
但是反而多了一个问题，部分不符合 OCT 原则的问题，对工厂的拓展符合 OCT，但是每次要拓展一个产品，就要修改一次工厂里面的方法

职责链模式：
类似踢皮球，往职责链上一甩，谁有能力谁处理
https://www.runoob.com/design-pattern/chain-of-responsibility-pattern.html

这种直接看菜鸟教程，我这里没有问源码，涉及源码就要花时间研究了

4、一道算法，
前序遍历二叉树
相关知识点： 栈树哈希
相关知识点： 栈树哈希
，要非递归写
复制代码

```
1  class Solution {
2      List<Integer> list = new ArrayList<Integer>();
3      public List<Integer> preorderTraversal(TreeNode root) {
4          if(root == null)
5              return list;
6          Stack<TreeNode> stack= new Stack<>();
7          stack.push(root);
8          while(!stack.isEmpty()){
```

```
9           TreeNode tree = stack.pop();
10          list.add(tree.val);
11
12          if(tree.right != null){
13              stack.push(tree.right);
14          }
15          if(tree.left != null){
16              stack.push(tree.left);
17          }
18      }
19      return list;
20  }
21 }
```

注意这里有个坑，root 为空要返回一个空的 arraylist
这题是 leetcode 144

“能接受转 GO 吗”
能

一面 40 分钟，基本从项目展开就问技术

二面
自我介绍
“你自我介绍说向往我们公司，那你对我们公司有什么了解”

“你在实习干了啥”

“ab 怎么做的，你了解吗”

“聊开发项目”

“锁是怎么做的”
我一开始说操作系统的那些什么共享内存 ，对锁有死锁检测、死锁避免、死锁处理。还有四个条件什么的

他说这是进程的，线程锁怎么做

我就说 JAVA 线程的话，synchronized 信号量 阻塞队列这些已经有线程 api

“项目线程安全的理解”
项目里面的，我说了 concurrenthashmap，我太久没看代码忘了 想了很久说的不好。

“那边实习 为啥不打算留下来”

“你手里有几个 offer”

还有一个最难的题目，“一个 tcp 的 ping 命令 10ms，http 请求多少 ms”

我乱分析一通，说 ping 基于 icmp，是网络层已经是很底层了，http 是应用层，http 先是是不是就 tcp 三次握手 所以 30ms。

他说不是，后面反问环节，我问他这个问题的答案，他说思路没错，可以仔细看一下三次握手，有优化，不需要那么久。

没有撕算法