知乎一面
职位： 后端开发实习生
时间：2020.8.11
结果： 一面通过
更好的阅读体验 ：
秋招的一点总结 秋招进展
自我介绍，项目介绍
巴拉巴拉巴拉
算法题

合并两个有序数组

就是归并排序的一部分，直接写算法模板

复制代码

```
while(i  mid && j  r){  if(q[i]  q[j]){  t[k ++] = q[i ++];  } else {  t[k
1++] = q[j ++];  }  } while(i  mid){  t[k ++] = q[i ++]; } while(j  r){  t[k ++] = q[j
++]; }  for(int i = l, j = 0; i  r; i ++, j ++){  q[i] = t[j]; }
```

说一下归并排序的过程

复制代码

```
1\. 确定分界点 mid = (l + r) / 2 [l, mid] [mid + 1, r] 2\. 递归排序左右 3\. 合并两个
1有序数组 时间复杂度 O(NlogN) 因为会递归 logN 层 每层都是 O(N) 的 是稳定
的
```

说一下快排的过程

复制代码

```
1\. 确定分界点 x 2\. 调整区间 是得左边区间都小于等于 x 右边区间都大于等于
1
x 3\. 递归排序左右区间
```

说一下快排的最坏情况是什么，快排稳定嘛

复制代码

1不稳定 时间复杂度 O（NlogN） 最坏情况是每次取的都是最大或者最小元素

说一下哈希表这种数据结构
哈希表由逻辑上一系列可以存放词条的单元组成，这些单元被称为桶，底层可以由数组实现。
散列函数是将关键码映射到数组地址空间的函数。
装填因子是非空桶的数目与桶单元总数的比值
哈希冲突： 1. 开散列 2. 闭散列
了解哈希表扩容的过程吗
当装填因子达到了阈值之后，会扩容。redis 中是会开辟另一个两倍大小的哈希表，然后将原哈希表中第一个非空元素插入到第二个哈希表中，之后每次插入一个元素的时候，都将原表中的第一个非空元素插入到第二个表中。
线程和进程的区别

进程是具有一定独立功能的程序在一个数据集合上的一次动态执行过程

线程是进程的一部分，描述指令流的状态，是进程中指令流的最小单位，是 CPU 调度的基本单位

区别

线程进程都是 CPU 工作时间段的描述，不过颗粒大小不同

线程依赖于进程存在，一个进程至少有一个线程

从拥有资源看 进程有独立的地址空间，线程共享所属进程的地址空间。进程独立拥有系统资源，如内存 IO CPU

通信 进程通信需要 IPC， 线程可以访问全局变量

系统开销 进程切换系统开销大，涉及整个 CPU 环境的设置和新进程的 CPU 环境，而线程只需要保存设置少量的寄存器内容，并不涉及存储器方面的操作，进程切换开销远大于线程切换开销

鲁棒性 多线程程序只要有一个线程崩溃，整个程序就崩溃了，但多进程程序中一个进程崩溃并不会对其它进程造成影响，因为进程有自己的独立地址空间，因此多进程更加健壮

了解协程吗
协程是一种用户态的轻量级线程，协程的调度完全由用户控制。协程拥有自己的寄存器上下文和栈。协程调度切换时，将寄存器上下文和栈保存到其他地方，在切回来的时候，恢复先前保存的寄存器上下文和栈，直接操作栈则基本没有内核切换的开销，可以不加锁的访问全局变量，所以上下文的切换非常快。
协程和线程的关系

一个线程可以拥有多个协程，一个进程也可以单独拥有多个协程，这样 python 中则能使用多核 CPU。

线程进程都是同步机制，而协程则是异步

协程能保留上一次调用时的状态，每次过程重入时，就相当于进入上一次调用的状态

Python GIL 了解吗
就知道会让整个程序同一时间只有一个线程在运行
对 CPU 密集型的程序和 IO 密集型的程序的影响
会导致 CPU 密集型的多线程程序并不能有很好的性能，但是对 IO 密集型的程序影响不大
Python multiprocess 库了解吗
不了解。。。
并发并行区别

并发是两个程序在极短的时间内，先运行 A 再运行 B 再运行 A，如此切换

并行是两个程序真的同时跑

说下同步异步阻塞非阻塞的概念

同步 所谓同步就是一个任务的完成需要依赖另外一个任务时，只有等待被依赖的任务完成后，依赖的任务才能算完成，这是一种可靠的任务序列

异步 所谓异步是不需要等待被依赖的任务完成，只是通知被依赖的任务要完成什么工作，依赖的任务也立即执行，只要自己完成了整个任务就算完成了，所以它是不可靠的任务序列

阻塞 阻塞调用是指调用结果返回之前，当前线程会被挂起，一直处于等待消息通知，不能够执行其他业务

同步阻塞 用户线程发起 IO 读/写操作之后，线程阻塞，直到可以开始处理数据；对 CPU 资

源的利用率不够

同步非阻塞 发起 IO 请求之后可以立即返回，如果没有就绪的数据，需要不断地发起 IO 请求直到数据就绪；不断重复请求消耗了大量的 CPU 资源

异步 用户线程发出 IO 请求之后，继续执行，由内核进行数据的读取并放在用户指定的缓冲区内，在 IO 完成之后通知用户线程直接使用

给定一个 list ，用列表推导式取出里面大于 3 的元素
复制代码
1res = [n for n in arr if n > 3]
说一下装饰器和生成器
动态语言和静态语言的区别
静态语言编译时就知道变量的类型，动态语言运行时才做类型检查。
动态语言因为编译前类型不确定，所以比较难维护。
TCP UDP 区别

TCP 面向连接， UDP 无连接

TCP 可靠，UDP 不可靠

TCP 只支持点对点通信，UDP 支持一对一，一对多

TCP 面向字节流，UDP 面向报文流

TCP 有拥塞控制，流量控制，UDP 没有

TCP 首部开销比 UDP 大

UDP 主机不需要维护复杂的连接状态表

说一下 TCP 三次握手
直接看图
SYN 泛红攻击怎么解决

SYN COOKIES 策略

操作系统会维护两个队列，一个时 SYN 队列， 一个时 ACCEPT 队列，当服务端收到一个 SYN 之后，会将其假如 SYN 队列，收到了 ACK 之后，会假如 ACCEPT 队列。如果 SYN 队列满，则将 SYNACK 的 seq 设置为 cookies ， 之后如果收到了 ACK 并且 ack=cookies + 1, 则直接假如 ACCEPT 队列

TCP 慢启动过程
老生常谈了
列举几个应用层协议

FTP 21 端口

SSH 22 端口

TELNET 23 端口

SMTP 25 端口

HTTP 80 端口

DNS 53 端口

HTTP HTTPS 的区别

端口不同：HTTP 使用的是 80 端口，HTTPS 使用 443 端口；

HTTP（超文本传输协议）信息是明文传输，HTTPS 运行在 SSL(Secure Socket Layer)之上，添加了加密和认证机制，更加安全；

HTTPS 由于加密解密会带来更大的 CPU 和内存开销；

HTTPS 通信需要证书，一般需要向证书颁发机构（CA）购买

数据库事务的特征

原子

隔离

一致

持久

事务隔离级别

并发一致性问题

脏写 一个事务修改了另一未提交事务的修改的数据

一个事务读取了另一未提交事务修改的数据

一个未提交事务读到了另一已经提交的事务修改过的数据，每次其他事务提交后，该事务都能读到修改值

一个事务先根据某些条件查询出一些记录，之后另一事务又向表中插入了符合这些条件的记录，原先事务再次按照该条件查询时，能把另一个事务插入的记录也读出来

隔离级别

未提交读

已提交读

可重复读

可串行化

索引什么数据结构，特点是什么，好处是什么

B+ 树

特点

优点

IO 次数少：B+树的中间结点只存放索引，数据都存在叶结点中，因此中间结点可以存更多的数据，让索引树更加矮胖；

范围查询效率更高：B 树需要中序遍历整个树，只 B+树需要遍历叶结点中的链表；

查询效率更加稳定：每次查询都需要从根结点到叶结点，路径长度相同，所以每次查询的效率都差不多

场景题
有三个业务场景，请设计一下数据库表结构，并给出 SQL 语句

判断两个人之间是否有关注关系

判断某个人的粉丝数和关注数

展示关注的人的列表和粉丝列表

| user 表 |||
id
name
| follow 表 |||
id
followed_id
follower_id

判断 用户 a 是否关注了用户 b

复制代码

```
1 SELECT COUNT(*) FROM follow WHERE followed_id=(SELECT id FROM user WHERE name="a") AND follower_id=(SELECT id FROM user WHERE name="b")
```

查询 用户 a 的粉丝数和关注数

关注数

复制代码

```
1 SELECT COUNT(*) FROM follow INNER JOIN user ON follower_id=user.id WHERE
  user.name="a"
```

粉丝数

复制代码

```
1 SELECT COUNT(*) FROM follow INNER JOIN user ON followed_id=user.id WHERE
  user.name="a"
```

查询 用户 a 的粉丝列表和关注列表

同 2