虾皮一面

40.说一下https有什么加密方式（对称、非对称，只答了非对称），其如何传输公钥保证公钥不被截获（凉）作者：mr猩猩。
41.听你刚刚说了句2.0，讲一讲新特性呗?
42.我1.X也有connect：keep-alive，怎么就不能实现服务端推送了?
43.你有一台无限内存的机器和很强的cpu，假设要写爬虫猛爬鹅厂（面试官肯定鹅厂出来的），你觉得限制在哪里
44.那反过来呢？假设你的事服务器呢?
44.送个分，来谈谈四次挥手和time_wait?
45.https了解不？加密过程说一说?
46.tcp和udp的区别
47.四次挥手TCP三次握手
48.TCP/UDP 区别
49.UDP主要应用
50.HTTPS加密具体细节
51.ping网站用什么协议
52.tcp三次握手状态
53.tcp的time wait状态
54.tcp和udp四次挥手的状态转移，为什么要四次
55.TCP有哪些措施保证可靠性
56.http状态码了解吗（不了解，一点都不了解，完全没答上来）
57.针对TCP3次握手怎么攻击？ （这个没答出来）
58.SYN攻击和DDOS攻击原理
59.TCP的传输过程是怎么样的？怎么确保有序?
60.http协议的特点
61.http状态码
62.https协议解释
63.http无状态怎么解决
64.tcp协议的特点
65.tcp的可靠性怎么保证
66.三次握手有什么隐患么？ 讲了洪泛攻击
67.洪范攻击怎么解决？ 真不知道， 猜了一个ip限制
68.http状态码 301、302
69.流量控制
70.常用不常用的说了20个左右

操作系统

1.虚拟内存
2.缺页中断
3.进程线程的通信方式，我这里被问到管道使用场景，毕竟面试官引导我答出来，有关于父子进程方面的，我顺着答出来，面试说答对了
4.虚拟内存
5.进程线程区别
6.lru实现
7.讲讲进程与线程，系统是怎么调度线程的?
8.hash冲突的解决办法有哪些?
9.介绍一下协程
10.多线程和多进程
11.socket的读写操作
12.什么是中断
13.hash表，hash冲突算法

14.进程和线程的区别
15.线程间通信（忘了问的是同步还是通信了）
16.页面置换算法 LRU 和 LFU
17.为啥我问的是cpu缓存，如何保证cpu缓存和外存缓存的一致性?
18.mysql最左匹配原则是啥?
19.tcp如何保证传输的可靠性?
20.如何保证有序性?
21.LRU算法具体实现是啥?
22.详细介绍下cpu内存?
24.进程通信方式
27.cpu内存你了解吗？（emm，我把jvm具体说了下，什么栈什么堆）
28.cpu缓存你知道吗？（我把cache的原理啥的说了一遍，面试官追问cache的缓存如何和外存的缓存保持一致性？我心想：我面得是后端吗？）
29.进程和线程和协程的区别?
30.进程与线程，以及通信方式
31.进程调度算法
32.进程间怎么通信
33.线程 进程 协程关系
34.内存置换算法有什么，说一下时钟置换算法
36.线程高速缓存讲一讲?
37.内存分配高低地址?
38.来讲一讲动态库加载机制?
42.还问了redis
43.问操作系统我有点猝不及防，后来问了为什么共享内存最快，我蒙了个减少用户态内核态切换居然对了😂. 进程通信方式
51.介绍银行家算法
52.死锁
62socket通信用的函数说一下

mysql

1.Mysql索引（B+树）
2.一个Mysql操作很慢，怎样排查原因。（设置慢查询时间，查询慢查询日志，explain查看慢的sql语句的查询情况）
3.一个联合索引(a,b,c)，where b=.. and c=....走不走索引？为什么？哪些走索引?
4.MySQL 索引类型
5.数据库索引，b+树，为什么用他
6.悲观锁，乐观锁
乐观锁实现方式
五、如何选择 我认为修改更新操作较多时，使用悲观锁。 查询远大于修改的场景下，用乐观锁可以提高效率。 在乐观锁与悲观锁的选择上面，主要看下两者的区别以及适用场景就可以了。
9.mysql引擎有哪些，有啥区别
10.关于索引的使用，哪些语句使用到了索引，具体的内容没来得及拷贝下来。
11.索引的优点与缺点，底层怎么实现的?
12.B+树的特点，与二叉树的区别
13.事务的特性，解释一下它们
14.其中的隔离性分几个级别？每个级别解释一下什么意思?
15.串行化的优缺点
16.SQL join操作，B树B+树
B树和B+树的区别：
17.数据库索引，b+树

18.最左索引匹配
19.乐观锁，悲观锁
20.数据库三大范式 有哪些反范式的设计
21.join 和 left join、right join 的区别
22.数据库事务 持久性是什么 隔离级别 幻读是什么
23.主键跟索引的区别
24.索引的特点
25.MySQL的innodb和myisam区别
26.InnoDB引擎主键为什么设成自增?
27.为什么用B+树
28.数据库索引
29.四种隔离级别
30.默认的隔离级别
31.脏读与幻读
32.事务的隔离级别?
33.数据库的索引怎么实现的？（B+树）
33.B+树和B树的区别？B+树的优点?
34.数据库的三范式是什么?
35.为什么要用索引，索引的实现，b+树和b树相比优点
36.说一下事务，事务隔离级别，怎么解决不可重复读
37.数据库innodb如何实现啥来着，我回答事务
38.innodb索引，b+树那一套
39.聚簇索引等
40.（这题我回答错了，大佬们帮我看看）写了一个sql select * from table where a>2 and b=3 问我 联合索引是a b的话 ，能不能使用到 联合索引？我说可以，因为最左匹配，他说我说反了。
41.事务隔离级别有什么
42.解释一下聚合索引？使用条件呢?
43.说一下主键和索引
44.熟悉mysql的引擎吗（不熟悉QAQ
45.数据库唯一索引和主键

linux

1.Linux操作相关
2.Linux上我怎么查看某端口被什么进程占用
3.如何修改文件的权限.
4.linux操作系统awk(这个我说我用得少),那再随便说五个命令,都是干嘛的(esay)
5.如何查看某个进程开启的socket(查看/proc/pid号/fd文件),
6.如何查看tcp链接(没太交流明白,下来发现好像就是netstat?)
7.Linux的Kill命令（-9信号的作用）
8.Linux的进程间的通信.
9.进程使用的状态怎么查看（我说的是windows的任务管理器哈哈哈哈）

redis

redis基本数据结构
redis有序集合的底层实现（ziplist和skiplist，跳跃表是什么）
Redis数据类型，有序集合实现原理Redis
分布式锁使用在了什么地方 怎么实现的 除了 Redis 还有什么方式可以实现
redis底层实现
set的使用场景
有序set的使用场景
redis缓存

- Redis持久化机制
- 你常用的数据结构、说一下原理
- redis单线程多线程、有什么优劣
- 集群、哨兵、主从
- 持久化方案
- 缓存一致性方案
- redis事务用过吗，是怎么样的?

IO

java基础

- 多态
- ArrayList 和 LinkedList 区别，使用场景
- Map 有几种，LinkedHashMap的数据结构
- 怎么实现的深拷贝、浅拷贝
- ArrayList如何自己封装成一个安全的文件
- 多线程volite关键字什么作用，追问:可见性你怎么理解的？.java如何实现并发（Thread runable）
- java哈希表
- java的treemap
- java的concurrent HashMap
- 反射 反射的弊端
- hashmap冲突解决，
- 链表遍历效率低下怎么解决
- hashmap设计及存在相同key的时候的解决方法（太紧张了 没想就答了，结果崩了）（太紧张答错了，凉）.
- 多态你怎么理解?
- HashTable原理？扩容机制？时间复杂度，空间复杂度？怎么解决冲突?
- synchronized与volatile区别?
- java的IO模型有哪些？（BIO、NIO、AIO）
- Synchronized关键字，
- 深拷贝浅拷贝
- HashMap 底层数据结构，链表长度转化、扩容
- 介绍一下主要的几种Java容器类，Collection和Map接口
- ArrayList扩容机制、和LinkedList之间的实现区别
- 有哪几种set？HashSet、TreeSet、LinkedHashSet之前的实现区别，如果遍历的话，会是怎么样一个顺序？（第一次没太听懂，再描述了下才发现意思是hash tree和link hash下存储方式的区别）
- Map呢?
- 三种map的实现
- 介绍一下hashtable、hashmap、concurrenthashmap之间的区别
- 说一下synchronize和Lock（不记得是让说区别还是只说前面那个了）
- 多线程几个问题（这里记不太清了）

算法与数据结构

- 1.怎样判断一个链表是否有环
- 2.怎样找到有环链表的环的开始节点（找到快慢指针的相遇节点，快指针改为走一步，慢指针指向head，同时走，在环开始节点相遇）
- 3.常见排序算法(快排，堆排序)
- 4.快排的时间复杂度，最坏的时候也是NlogN?（后面这个问题回答的不是很好。。。）
- 5.常用的排序算法
- 6.数组中最小的k个数?
- 7.快速排序

8.实现二进制转10进制
9.如果有n条直线，且三线无交点，问有多少种交点（0表示平行，1表示相交）
10.算法题：用数组实现栈，实现pop与push，支持扩容保证性能。
11.归并排序讲一下
12.手撕大根堆
13.堆是什么，数据结构，时间复杂度
14.排序算法有哪些 ，归并排序时间复杂度，是不是稳定的
15.链表和数组区别
16.两千万数据，取最大的一千个。
17.是找最大不重复子串长度
18.两个栈如何模拟一个队列
19.arrayList和linkList区别
20.LRU算法是如何实现的？（我说的是哈希表+链表实现，并要把具体实现的思路讲明白）
21.1000个数据，查找出现次数最多的k个数字（优先队列）还问了快排的最好最差复杂度情况
22.一千万个数找出其中最大的k个数?
23.怎么判断一个链表有没有环？怎么判断环里面几个节点?

手撕算法

topN问题及其复杂度，最小n个，总共m个数， m远大于n，复杂度答了：O(M) 面试官说O(logn)
不能忽略，所以是：O(MlogN + NlogN)
（凉）有环链表的判断 和 环长度判断（凉）
说一下LRU过程
巨大文件，TOPK排序
数组和链表
哈希解决冲突的方法
还问了大文件找出排名前1000的数据
手撕代码
算法题，反转字符串
算法题，字符串中大小写字母分成前后两部分，字母顺序不变
接雨水，以前看过但是忘了。。。力扣42题，被问过不止一次。。。
给个m，求1到m的最大奇约数的和。
一个数组a，给个s，返回一组在数组里差是s的两个数。
数组链表区别
哈希表原理，常见碰撞算法
快速排序讲一讲
平衡二叉树
快排（swap的时候我写错了两个地方，面试官没有指出，面完再回去看才发现）
哈希冲突解决方法
哈希表某个桶中数据被删除怎么办
给前序中序，还原二叉树
两千万个文件找最小的一千个（答错了，应该用大顶堆，答成了小顶堆）
topk场景题（怎么统计大量帖子中的点赞数量前10）
问了快排,堆排的原理及复杂度比较
数组和链表
什么是平衡二叉树
编程：实现反转链表
编程：快速排序快排时间复杂度，什么是时候是最坏，怎么优化
先写题，找奇点，说思路，O（n）时间复杂度，然后让优化，优化找到O（logn）

给定一组非负整数，重新排列它们的顺序使之组成一个最大的整数（如给定[3,30,34,5,9]，输出9534330），结果用字符串输出....记得我看过，但是忘了，气死，后来想了一会，用了比较笨的办法

写一个函数，将输入的字符串中大写字母移动到字符串的末尾，同时需要保留出现的相对顺序, 例如输入AaBbCc 返回 abcABC, 要求不使用额外内存..... 想了一会， 用类似冒泡的方法解决

前面知识题应该都比较平和，两道算法题没答好，都是用的比较笨的方法.....2000w个数找前1000个（完全不会这种问题，扯的堆排、快排）

已知前序和中序 ， 求高度

算法1：两个栈做一个队列

算法2：数据流找最大的 n 个数

算法：4321 找比它小的字典序数字，那么是 4312，时间复杂度不能是 O(n2)

TCP UDP区别，简述一下三次握手过程

让你来做一个直播流软件，从上到下涉及协议栈？作者：essilon

大概是升序旋转数组最小值，leetcode 153，给了线性和二分两种思路，然后让手写一下二分。这里有点尴尬，二分最开始写错了下标更新，面试官看了下没让改直接继续了

jvm

1.https知道吗（只知道是加密传输）

HTTPS是一种安全地HTTP协议，端口号为443，是先用HTTP协议与SSL交互，然后通过SSL在TCP/IP基础上进行服务。 相对于HTTP，HTTPS添加了数据加密、端口认证、数据完整性保障的功能。

2.TCP timewait

三次握手：

- 首先服务器端处于LISTEN状态。
- 当客户端想要建立连接时，他将发送一个SYN包，序列号假如为u。客户端进入SYN_SENT状态。
- 当服务器端收到了这个SYN包，如果服务器同意建立连接，他将发送一个SYN，ACK包，序列号假如为v，确认号为u+1。服务器端进入SYN_RECD状态。
- 当客户端收到了服务器端的SYN,ACK包，它将再次确认，向服务器发送一个ACK包，序列号为u+1，确认号为v+1。此时客户端进入ESTABLISHED状态。
- 当服务器端收到了客户端的ACK包，也将进入ESTABLISHED状态。

四次挥手：

- 当客户端想要断开连接时，发送一个FIN数据包，序列号为u,确认号为v，客户端进入FIN_WAIT1状态。
- 当服务器端收到这个FIN包，他知道了客户端想要断开连接，它会发送一个ACK包对它进行确认，序列号为v，确认号为u+1。此时服务器进入CLOSE_WAIT状态。
- 当客户端收到了服务器端发送的ACK确认包，他知道了服务器了解了自己想要断开连接。此时客户端进入FIN_WAIT2状态。
- 当服务器端的数据都发送结束了，它将断开服务器端到客户端的连接，它向客户端发送一个FIN包，序列号假设为w，确认号依旧为u+1。此时服务器端进入LAST_ACK阶段。
- 当客户端收到了服务器端的FIN包，他知道了服务器也要断开连接，它向服务器发送一个ACK确认包，序列号为u+1,确认号为w+1,然后进入TIME_WAIT阶段等待2倍MSL，在这个期间如果没有继续收到服务器端的FIN包，就进入了CLOSED阶段。

- 服务器端收到客户端的ACK确认包，进入CLOSED阶段。

http 长连接 短连接

- http长连接指的是使用完一次http服务之后连接不断开，可以继续下一次服务使用。
- 短连接指的是每请求完一个资源，就断开连接，想要再次请求，就得再建立连接。

tcp 3次握手、4次挥手，为啥不能2次

为什么不能两次挥手呢?

两个方面:

- 防止网络中失效的连接请求到达服务器端，如果只有两次握手，服务器端将建立连接，是资源浪费。举个例子，客户端发送了一个连接请求，但是这个请求在网络中阻塞了，超过重传时间后客户端又再次发送连接请求包，建立连接，完成服务，断开连接。然后之前滞留的包又到达了服务器端，如果两次握手，连接就建立了。并且客户端不会请求关闭，资源浪费。
- 通过三次握手，让客户端和服务器端都知道了自己的发送和接收没问题。第一次握手，暂时没有什么，第二次握手，让服务器端知道了自己的接收没问题，第三次握手，让客户端知道了自己的发送和接收均无问题，当服务器端收到了客户端的最后一次确认，它也将知道自己的接收没有问题。

4.timewait.

- 如果在两倍的MSL时间内，没有收到服务器端再次发送FIN包，就可以确认服务器端收到了最后一次确认，进入CLOSED状态。
- 使此次连接中可能在网络中滞留的包都失效。

5.TCP连接如何保证数据有效性

对数据编号、确认，超时重传。

6.TCP与UDP的区别

它们都是传输层协议。

- TCP是面向连接的，发送数据前先要建立连接，UDP是面向无连接的，发送数据前不需要建立连接。
- TCP面向字节流的，UDP是面向报文的。
- TCP是可靠的，TCP可以保证数据无差错，不重复，不丢失，按需到达。UDP是不可靠的，它只能尽最大努力交付。
- TCP是点到点通信，UDP支持一对一，一对多，多对一，多对多。
- TCP首部开销大，20个字节。UDP首部开销小，8个字节。

7.TCP的优点与缺点

优点：可靠，稳定 缺点：传输数据之前先要建立连接。慢，效率低且占用资源多。

8.TCP的可靠性是通过什么来保证的?

对字节编号，确认，超时重传，差错检测，流量控制和拥塞控制

9.解释一下确定重传机制，讲讲窗口滑动

TCP对字节编号，当发送方在重传时间内没有收到期望的确认包，它就认为自己之前发送的包在网络中丢失，就会进行重传。

为了进行流量控制，在发送方和接收方都都保持一个窗口，窗口左边是已经发送且已经确认的数据。窗口右边是还未发送的数据。窗口里分为两个部分，左边是已发送未确认的部分，右边是此窗口状态下还可以发送的数据。

10.ISO七层结构，网络层和传输层的底层实现

ISO七层模型：

1. 物理层处于OSI参考模型的最低层。物理层的主要功能是利用物理传输介质为数据链路层提供物理连接，以透明地传送比特流。
2. 数据链路层在物理层提供比特流传输服务的基础上，在通信实体之间建立数据链路连接，传送以帧为单位的数据，通过差错控制、流量控制方法，变有差错的物理线路为无差错的数据链路。
3. 网络层主要任务是通过执行路由选择算法，为报文分组通过通信子网选择最适当的路径。它是OSI参考模型七层中最复杂的一层。
4. 传输层是向用户提供可靠的端到端服务，透明地传送报文 。
5. 会话层的主要目的是组织同步的两个会话用户之间的对话，并管理数据的交换。
6. 表示层主要用于处理两个通信系统间信息交换的表示方式，它包括数据格式变换、数据加密与解密、数据压缩与恢复等功能。
7. 应用层是OSI参考模型的最高层。应用层不仅要提供应用进程所需要信息交换和远程操作，而且还要作为应用进程的用户代理，完成一些为进行语义上有意义的信息交换所必须的功能。综上所述可知，ISO / OSI开放系统互连七层参考模型***能最复杂的一层是网络层。

网络层和传输层底层实现：

- TCP/IP UDP/IP?

11.TCP和UDP在程序设计的时候应该注意的点?

- TCP适用于对网络通讯质量要求高的场景，比如文件传输。
- UDP适用于对网络通讯质量要求不高，要求网络通讯速度尽可能快的场景。比如QQ语音，QQ视频。

12.TCP传输的报文基于底层什么机制?

- 为了实现TCP的可靠传输，使用校验和、序列号、确认机制、超时重传、流量控制、拥塞控制。

13.https的安全协议，如何识别新建第三方的电子证书

- 是一种通过计算机网络进行安全通信的传输协议。HTTPS经由HTTP进行通信，但利用SSL/TLS来加密数据包。是一种通过计算机网络进行安全通信的传输协议。HTTPS经由HTTP进行通信，但利用TLS来加密数据包。

14.计算机网络，一定三次握手吗?

是的，必须三次握手，两次握手可能会使网络中滞留的连接请求到达服务器端直接建立连接，浪费服务器资源。

15.tcp的可靠性

校验码、序列号、确认、超时重传、流量控制、拥塞控制等机制来保证可靠传输。

16.session和cookie

session:

- Session是服务器的会话技术，是存储在服务器的。

cookie:

- Cookie相当于服务器给浏览器的一个通行证，是一个唯一识别码，服务器发送的响应报文包含 Set-Cookie 首部字段，客户端得到响应报文后把 Cookie 内容保存到浏览器中。客户端之后对同一个服务器发送请求时，会从浏览器中取出 Cookie 信息并通过 Cookie 请求首部字段发送给服务器，服务器就可以识别是否是同一个客户。

区别：

- Cookie只能存储ASCII 码字符串，而 Session 则可以存储任何类型的数据，因此在考虑数据复杂性时首选Session。
- Cookie 存储在浏览器中，容易被恶意查看。如果非要将一些隐私数据存在 Cookie 中，可以将 Cookie 值进行加密，然后在服务器进行解密。
- 对于大型网站，如果用户所有的信息都存储在 Session 中，那么开销是非常大的，因此不建议将所有的用户信息都存储到 Session 中。

17.session放本地的方法

将seesion数据加密，然后存储在cookie中。

18.拥塞控制是什么

拥塞：有时候网络中负载过大，发送的数据可能会网络中长时间滞留。就像堵车一样。

拥塞控制：通过慢开始、拥塞避免、快重传、快恢复来拥塞控制。

- 慢开始：刚开始时，拥塞窗口为1，发送一个数据单元，经过一个RTT，拥塞窗口翻倍变成2，第二次发送两个数据单元，再经过一次RTT，拥塞窗口变成4，这样直到拥塞窗口达到拥塞窗口门限值。
- 当拥塞窗口等于门限值时，可以用拥塞避免，也可以用慢开始。
- 当拥塞窗口大于门限值时，启用拥塞避免，每经过一个RTT，拥塞窗口不再翻倍，而是加一，当拥塞发生时，门限值变成此时拥塞窗口的一半，拥塞窗口置为1，重新采用慢开始策略。
- 快重传，接收端不使用累计确认，当发送端连续收到同样的三个确认包，则直接传送此包，不需要等待超时重传。
- 快恢复：当发生快重传时，门限值和拥塞窗口都变成此时拥塞窗口的一半，并且采用拥塞避免。

19.三次握手

- 首先服务器端处于LISTEN状态；
- 当客户端想要建立连接时，发送一个SYN数据报，序列号为u，此时客户端的处于SYN-SENT状态；
- 当服务器端收到这个确认包并且同意建议连接时，向客户端发送一个SYN,ACK包，序列号为v，确认号为u+1,此时服务器端进入SYN-RECV状态。
- 当客户端收到了服务器端的SYN,ACK包，则再次对之确认，向服务器发送一个ACK包，进入ESTABLISHED状态。

- 当服务器端收到了客户端的ACK包，也进入ESTABLISHED状态。

20.HTTP 状态码有哪些

- 1XX:信息性状态码
- 2XX:成功状态码
- 3XX:重定向状态码
- 4XX:客户端错误状态码
- 5XX:服务器错误状态码

- 100 Continue:表示到目前为止都很正常，客户端可以继续发送请求或者忽略这个响应。
- 200 OK
- 204 No Content:请求已经成功处理，但是返回的响应报文不包含实体的主体部分。一半只需要从客户端往服务器端发送信息，而不需要返回数据时使用。
- 206 Partial Content：表示客户端进行了范围请求，响应报文包含由Content-Range指定范围的实体内容。
- 301 Moved Permanently：永久性重定向
- 302 Found：临时性重定向
- 303 See Other：和302有着相同的功能，但是303明确要求客户端应该采用GET方法获取资源。
- 304：Not Modified：如果请求报文首部包含一些条件，例如：If-Modified-Since，如果不满足条件，则服务器会返回304状态码。
- 307 Temporary Redirect：临时性重定向，与302的含义类似，但是307要求浏览器不会把重定向请求的POST方法改成GET方法。
- 400 Bad Request：请求报文中存在语法错误。
- 401 Unauthorized：状态码表示发送的请求需要有认证信息，如果之前已经进行过一次请求，则表示用户验证失败。
- 403 Forbidden：请求被拒绝。
- 404 Not Found：未发现资源
- 405 Method Not Allowed：方法不允许。
- 500 Internal Server Error：服务器正在执行请求时发生错误。
- 503：Server Unavailable：服务器暂时处于超负载或者正在进行停机维护，现在无法处理请求。

21.GET POST 区别 他们系统里有些 GET 请求 用了 POST，这样设计是为什么（想不出来）？

区别：

- GET主要用于向服务器获取资源，POST主要用于提交数据。
- GET和POST的请求都能使用额外的参数，但是GET的参数是以查询字符串出现在URL中，而POST参数存储在实体主体中，不能因为POST参数存储在实体主体中就认为它的安全性更高，因为可以使用一些抓包工具（Fiddler）查看。
- GET方法不会改变服务器状态是安全的，POST不是安全地。
- 如果要对响应进行缓存，需要满足请求报文的方法本身是可缓存的，PUT可以缓存，POST多数情况下不可缓存。

22.TCP，udp区别

- TCP是面向连接的，通讯之前要三次握手要建立连接。UDP是无连接的。

- TCP是面向字节流的，UDP是面向报文的。
- TCP是可靠的，它可以保证数据无差错、不重复、不丢失、按需达到接收端。UDP是不可靠的，它提供的是尽最大努力交付。
- TCP是点对点的，UDP是可以一对一，一对多，多对一，多对多。
- TCP首部开销大，有20个字节，UDP首部开销小，8个字节。

23.URL输入后，整套流程浏览器中，打开一个网页到页面出现的全过程?

- 先检查输入的URL是否合法，然后查询浏览器的缓存，如果有则直接显示。
- 通过DNS域名解析服务解析IP地址，先从浏览器缓存查询、然后是操作系统和hosts文件的缓存，如果没有查询本地服务器的缓存。
- 通过TCP的三次握手机制建立连接，建立连接后向服务器发送HTTP请求，请求数据包。
- 服务器收到浏览器的请求后，进行处理并响应。
- 浏览器收到服务器数据后，如果可以就存入缓存。
- 浏览器发送请求内嵌在HTML中的资源。
- 浏览器渲染页面并呈现给用户。

24.http无状态

协议对交互场景没有记忆能力。浏览器对服务器完成一次资源请求后，再次对服务器进行资源请求，服务器不会记得之前的行为。

25.下载一个超大文件为什么网速会越来越快,解释背后的技术

拥塞窗口机制：慢开始。

26.三次握手，四次挥手

三次握手：

- LISTEN
- SYN,u, SYN-SENT
- SYN,ACK,v,u+1,SYN-RECV
- ACK,u+1,v+1,ESTABLISHED
- ESTABLISHED

四次挥手：

- FIN,u,FIN_WAIT_1
- ACK,v,u+1,CLOSE_WAIT
- FIN_WAIT_2
- FIN,w,u+1,LAST_ACK
- ACK,u+1,w+1,TIME_WAIT
- CLOSED
- CLOSED

27.为什么四次挥手简述下tcp四次挥手?

- 当客户端想要断开连接时，他会发送一个FIN包，序列号为u，客户端进入FIN_WAIT_1状态。
- 当服务器端收到了这个FIN包，知道了客户端想要断开连接，就回应一个ACK确认包，序列号为v，确认号为u+1，此时服务器进入CLOSE_WAIT阶段。

- 当客户端收到了服务器端的ACK包，进入FIN_WAIT_2阶段。
- 此时客户端到服务器端的发送连接已经断掉。
- 当服务器端数据发送结束想要断开连接时，它向客户端发送一个FIN包，序列号为w，确认好还是u+1，进入LAST_ACK阶段。
- 当客户端收到了这个FIN包，就对它进行确认，回应一个ACK包，序列号为u+1,确认号为w+1,进入TIME_WAIT阶段，当经过二倍的MSL，没有再次收到服务器端的FIN包，就可以进入CLOSED阶段。
- 当服务器端收到了客户端最后的ACK包，则进入CLOSED阶段。
- 至此，四次挥手接收。

28.追问：tcp如何保证传输的有序性，可靠性?

校验码、编号、确认、超时重传、流量控制、拥塞控制。

29.session和cookie区别？追问：session是如何识别用户的？（emm，我说了session id，面试官又追问id存在哪儿）

区别：

- cookie只能存储ascll码字符串，session则可以存储任何类型的树，因此再考虑数据复杂性的时候优先选择session。
- cookie存储在浏览器上，容易被恶意查看。
- session存储在服务器中，对于大型网站，如果用户所有的信息都存储在session中，那么开销非常大。

通过cookie存储一个session_id，然后具体的数据则是保存在session中。如果用户已经登录，则服务器会在cookie中保存一个session_id，下次再次请求的时候，会把该session_id携带上来，服务器根据session_id在session库中获取用户的session数据。就能知道该用户到底是谁，以及之前保存的一些状态信息。

30.session 和cookie 有什么区别?

- cookie保存在浏览器，session保存在服务器。
- cookie只能使用ascll码表示，session可以存储任意数据类型，因此在考虑数据复杂性时优先选用seesion。

31.TCP UDP HTTP区别?

TCP，DUP是传输层协议。

HTTP是应用层协议，基于TCP的。

然后说一下TCP，UDP的区别：

- TCP是面向连接的，数据通信前要建立连接。UDP是无连接的。
- TCP是面向字节流的，UDP是面向报文的。
- TCP是可靠的，他能保证数据无差错、不重复、不丢失、按需达到接收端。UDP是不可靠的，它提供的是尽最大努力交付。
- TCP是点对点的，UDP可以一对一，一对多，多对一，多对多。
- TCP首部开销大，20字节，UDP首部开销小，8个字节。

32.讲讲TCP三次握手?

- LISTEN
- SYN,U,SYN-SENT
- SYN,ACK,V,U+1,SYN-RECV
- ACK,U+1,V+1,ESTABLISHED
- ESTABLISHED

33.http1和http2

http2的新特性：

- 对头部进行压缩。
- 二进制帧层
- 请求和响应多路复用
- 服务端推送

34.tcp udp区别

- tcp面向连接、面向字节流，可靠的，点对点，头部开销大
- udp面向无连接，面向报文，不可靠，尽最大努力交付，支持一对一，一对多，多对一，多对多，头部开销小。

35.tcp 如何保证可靠性

校验码、编号、确认、超时重传、流量控制、拥塞控制。

36.三次握手

http请求方法：get post put delete head patch

37.握手为什么是3次，挥手4次?

因为第二次握手，服务器端同时发送了SYN,ACK

38.TCP相比于UDP在什么方面保证了其可靠性?

通过编号、确认、超时重传、流量控制，拥塞控制。

39.说一下cookie和session

- cookie保存在客户端，容易被恶意查看。
- session保存在服务器端，保存太多的话容易对服务器造成压力。
- cookie只能使用ascll码表示，session可以采用任意数据类型，在考虑数据复杂性的场景下，优先选择seesion。

40.说一下https有什么加密方式（对称、非对称，只答了非对称），其如何传输公钥保证公钥不被截获（凉）作者：mr猩猩。

对称加密和非对称加密混合方式。 采用非对称加密传输对称加密的密钥。 然后使用对称加密的密钥进行数据传输。

http header里有啥讲一讲？ 5.5.服务器怎么知道body里有什么玩意?

41.听你刚刚说了句2.0，讲一讲新特性呗?

- 对头部进行压缩
- 二进制帧层
- 请求和响应多路复用
- 服务器端推送

42.我1.X也有connect：keep-alive，怎么就不能实现服务端推送了？

HTTP/1.1协议里没有包含server push功能，都是一去（request）一回（response），或者多去多回（pipelining）。

所以后续的HTTP协议，比如SPDY，HTTP/2.0，都增加了主动push。

43.你有一台无限内存的机器和很强的cpu，假设要写爬虫猛爬鹅厂（面试官肯定鹅厂出来的），你觉得限制在哪里

44.那反过来呢？假设你的事服务器呢？

44.送个分，来谈谈四次挥手和time_wait？

- FIN,U,FIN_WAIT_1
- ACK,V,U+1,CLOSE_WAIT
- FIN_WAIT_2
- FIN,W,U+1,LAST_ACK
- ACK,U+1,W+1,TIME_WAIT,2MSL----CLOSED
- CLOSED

1. 在两倍MSL之间如果没有再次收到FIN包，则客户端进入CLOSED阶段。
2. 保证网络中此次连接范围内发送的数据报都消失。

45.https了解不？加密过程说一说？

https是一种安全地http协议，它由http和ssl/tsl实现。 非对称加密和对称加密结合。

46.tcp和udp的区别

47.四次挥手TCP三次握手

48.TCP/UDP 区别

49.UDP主要应用

QQ语音，视频，即时性要求高，准确性要求相对不高。

50.HTTPS加密具体细节

51.ping网站用什么协议

ICMP

ICMP请求报文和ICMP回答报文。

Ping 是 ICMP 的一个重要应用，主要用来测试两台主机之间的连通性。 Ping 的原理是通过向目的主机发送 ICMP Echo 请求报文，目的主机收到之后会发送 Echo 回答报文。Ping 会根据时间和成功响应的次数估算出数据包往返时间以及丢包率。

52.tcp三次握手状态

- LISTEN
- SYN,U,SYN-SENT
- SYN,ACK,V,U+1,SYN-RECV
- ESTABLISHED
- ESTABLISHED

53.tcp的time wait状态

2倍MSL

- 防止服务器没接收到ACK包
- 使本次连接在网络中滞留的数据过期。

54.tcp和udp四次挥手的状态转移，为什么要四次

- FIN,U,FIN_WAIT_1
- ACK,V,U+1,CLOSE_WAIT
- FIN_WAIT_2
- FIN,W,U+1,LAST_ACK
- ACK,U+1,W+1,TIME_WAIT----2MSL-----CLOSED
- CLOSED

因为是全双工通道，服务器到客户端断开连接要一次通知一次确认，客户端到服务器端也要一次通知一次确认，所以要四次。

55.TCP有哪些措施保证可靠性

校验码、编号、确认、超时重传、流量控制、拥塞控制

56.http状态码了解吗（不了解，一点都不了解，完全没答上来）

- 1XX：信息性状态码
- 2XX: 成功状态码
- 3XX: 重定向状态码
- 4XX: 客户端错误状态码
- 5XX：服务器端错误状态码
- 100：continue，表示到目前位置都很正常，可以继续请求或者忽略这个回应。
- 200：OK
- 204：No content：请求已经成功处理，但是响应不包含主体部分。
- 206：partial content：返回请求的范围部分
- 301：move permanently：永久性重定向
- 302：found：临时性重定向
- 303：see other:和302类似，但是明确要求用GET
- 304：not modified；请求首部包含一些条件，不满足则返回304
- 307：和302类似，但是明确要求浏览器不将POST变为GET

- 400：bad request:请求报文中存在语法错误
- 401：unauthorized:表示请求需要有认证信息，如果已经请求过一次了，则说明认证失败。
- 403：forbidden:请求被拒绝
- 404：not found
- 405: method not allowed:方法不允许
- 500: Internal serval error:服务器处理请求时发生错误
- 503: server unavarilable：服务器负载过大或者正在维护。

57.针对TCP3次握手怎么攻击？（这个没答出来）

SYN-洪水攻击

- 假设一个用户向服务器发送了SYN报文后突然死机或掉线，那么服务器在发出SYN+ACK应答报文后是无法收到客户端的ACK报文的（第三次握手无法完成），
- 这种情况下服务器端一般会重试（再次发送SYN+ACK给客户端）并等待一段时间后丢弃这个未完成的连接。 这段时间的长度我们称为SYN Timeout，一般来说这个时间是分钟的数量级（大约为30秒-2分钟）。一个用户出现异常导致服务器的一个线程等待1分钟并不是什么很大的问题，
- 但如果有一个恶意的攻击者大量模拟这种情况，服务器端将为了维护一个非常大的半连接列表而消耗非常多的资源----数以万计的半连接，即使是简单的保存并遍历也会消耗非常多的CPU时间和内存，何况还要不断对这个列表中的IP进行SYN+ACK的重试。
- 实际上如果服务器的TCP/IP栈不够强大，最后的结果往往是堆栈溢出崩溃---即使服务器端的系统足够强大，服务器端也将忙于处理攻击者伪造的TCP连接请求而无暇理睬客户的正常请求（毕竟客户端的正常请求比率非常之小）。 此时从正常客户的角度看来，服务器失去响应， 这种情况我们称作：服务器端受到了SYN Flood攻击（SYN洪水攻击）。

58.SYN攻击和DDOS攻击原理

59.TCP的传输过程是怎么样的？怎么确保有序？

1. 主机每次发送数据时，TCP就给每个数据包分配一个序列号并且在一个特定的时间内等待接收主机对分配的这个序列号进行确认，
2. 如果发送主机在一个特定时间内没有收到接收主机的确认，则发送主机会重传此数据包。
3. 接收主机利用序列号对接收的数据进行确认，以便检测对方发送的数据是否有丢失或者乱序等，
4. 接收主机一旦收到已经顺序化的数据，它就将这些数据按正确的顺序重组成数据流并传递到高层进行处理。 有点生疏，但还是答出来了7788

60.http协议的特点

1. 支持客户/服务器模式。
2. 简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。
3. 灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type（Content-Type是HTTP包中用来表示内容类型的标识）加以标记。
4. 无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。
5. 无状态：HTTP协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

61.http状态码

- 100 continue
- 200 OK
- 204 NO CONTENT
- 206 PARTIAL CONTENT
- 301 move permenantly
- 302 found
- 303 see other
- 304 not modified
- 307 和302类似，要求不允许浏览器将POST变为GET
- 400 bad request
- 401 unauthorized
- 403 forbidden
- 404 not found
- 405 method not allowed
- 500 internal servel error
- 503 servel unavailable

62.https协议解释

http+SSL/TSL 数据加密，端点身份验证，数据完整性保证

63.http无状态怎么解决

采用cookie和session

64.tcp协议的特点

面向连接，面向字节流，可靠，点对点。

65.tcp的可靠性怎么保证

校验码、编号、确认、超时重传、流量控制、拥塞控制。

66.三次握手有什么隐患么？ 讲了洪泛攻击

dos,ddos

67.洪范攻击怎么解决？ 真不知道， 猜了一个ip限制

减少服务器半连接保持的时间。用COOKIE配合

68.http状态码 301、302

- 301 move permenantly :永久性重定向
- 302 found：临时性重定向

69.流量控制

滑动窗口机制，通过接收端接收窗口的大小控制发送端发送的速度。

70.常用不常用的说了20个左右

- 100 continue
- 200 ok
- 204 no content
- 206 partial content
- 301 move permenantly
- 302 found
- 303 see other
- 304 not modified
- 307 和302类似
- 400 bad request
- 401 unauthorized
- 403 forbidden
- 404 not found
- 405 nethod not request
- 500 internal server error
- 503 server unavailable

操作系统

1.虚拟内存

虚拟内存是计算机系统内存管理的一种技术。它使得应用程序认为它拥有连续可用的内存（一个连续完整的地址空间），而实际上，它通常是被分隔成多个物理内存碎片，还有部分暂时存储在外部磁盘存储器上，在需要时进行数据交换。

2.缺页中断

- 在请求分页系统中，每当所要访问的页面不存在时，便产生一个缺页中断，请求操作系统将所缺失的页调入内存。此时将缺页的进程阻塞，如果内存中有空闲块，则分配一个一个块，将要调入的页装入该块，并修改页表中相应页表项，若此时内存中没有空闲块，则要淘汰某页（如果淘汰页在内存期间被修改过，则要将其写回外存）。

3.进程线程的通信方式，我这里被问到管道使用场景，毕竟面试官引导我答出来，有关于父子进程方面的，我顺着答出来，面试说答对了

进程通信方式，可以大致分为8种：

1. 共享数据结构，例如信号量机制，传输较少的数据，效率低，属于低级通信。
2. 共享存储区，允许多个进程共享一个给定的存储区，可以从中申请缓存区，因为不需要在进程中复制数据，这是最快的通信方式。但是要对多进程访问存储区进行同步控制。
3. 无名管道，半双工，速度慢、容量有限，只能用于父子，兄弟进程。
4. 命名管道，可以用于任意进程之间通信。速度慢，支持半双工。
5. 消息缓冲队列，使用OS提供的原语，隐藏通信细节，但是必须知道发送进程和接收进程的ID。
6. 消息队列，通过共享信箱或队列完成通信，可实现实时和非实时通信，两种消息传递系统容量都容易受到系统限制。

7. socket，可以用于本机进程和其他主机进程之间的通信。
8. rpc，远程方法调用，无需额外编程，隐藏通信细节。

4.虚拟内存

虚拟内存是计算机进行内存管理的一项技术，它使得应用程序认为自己拥有连续可用的内存（一个连续完整的地址空间），而实际上，它通常是被分割成多个物理碎片，还有部分存储在外部磁盘存储器上，在需要的时候和内存进行数据交换。

5.进程线程区别

- 线程拥有许多传统进程的特征，他被称为轻量级进程，在引入了线程的操作系统中一个进程通常有多个线程，最少也有一个线程。
- 线程和进程最根本的区别就是，进程是操作系统进行资源分配的基本单位，而线程是处理器调度和执行的基本单位。
- 从资源分配角度，每一个进程都有独立的代码和数据空间，进程之间相互隔离，进程切换开销较大。而同一个进程内的线程则共享这个进程的代码和数据地址空间，线程本身也有自己的程序计数器和运行栈，线程之间切换开销较小。
- 从互相的影响来看，因为进程相互隔离，一个进程崩溃了，在保护模式下不会对其他进程造成影响，而一个进程内的线程崩溃了，整个进程都会停止。多进程更健壮。
- 每个进程都有自己的程序入口，顺序执行代码和程序出口，而线程只能依赖于进程执行。

6.lru实现

最近最少使用

数据结构：双向链表和HashMap

put：加入时先看看有没有相同的key，有的话就直接更新，删掉链表中原来节点，加入新节点到链表头。如果没有则加入，再判断是否超过容量，超过了的话，删去尾节点。

get:访问其中一个元素，直接通过map在O(1)时间内访问到，如果有则访问到并返回该值，同时将双向链表中对应该项移到链表头部。如果没有则返回-1代表不存在。

7.讲讲进程与线程，系统是怎么调度线程的?

- 用户程序使用系统调用进入操作系统内核
- 硬件中断。硬件中断处理程序由操作系统提供，所以当硬件发生中断时，就会执行操作系统代码。硬件中断有个特别重要的时钟中断，这是操作系统能够发起抢占调度的基础。

8.hash冲突的解决办法有哪些?

- 开放定址法
- 再哈希法
- 链地址法
- 建立公共溢出区

9.介绍一下协程

- 协程既不是进程也不是线程，协程仅仅是一个特殊的函数，协程它进程和进程不是一个维度的。
- 一个进程可以包含多个线程，一个线程可以包含多个协程。
- 一个线程内的多个协程虽然可以切换，但是多个协程是串行执行的，只能在一个线程内运行，没

法利用CPU多核能力。
- 协程与进程一样，切换是存在上下文切换问题的。

10.多线程和多进程

11.socket的读写操作

12.什么是中断

- 中断，也称为外中断，指来自CPU执行指令以外的事件的发生。如IO中断，时钟中断。
- 异常，也称内中断，指来自CPU执行指令内部的事件，如缺页中断，或者专门的陷入指令。

13.hash表，hash冲突算法

- 开放定址法
- 再哈希法
- 链地址法
- 建立公共溢出区

14.进程和线程的区别

15.线程间通信（忘了问的是同步还是通信了）

主要通过共享内存和消息传递两种机制

- volatile
- synchronized+wait+notify/notifyAll
- Lock+await+condition+signal/signalAll
- juc的工具类，cyclicbarrier,countdownlatch,exchanger
- yield,sleep,join

16.页面置换算法 LRU 和 LFU

- 最近最久未使用LRU:双向链表+HashMap
- 最近最少使用LFU：小顶堆+哈希表

17.为啥我问的是cpu缓存，如何保证cpu缓存和外存缓存的一致性?

- 全写法：同时写入cache和主存
- 写回法：只修改cache，对cache设置一个脏位表示被修改，当这个块被换出的时候才写回主存。

18.mysql最左匹配原则是啥?

按索引顺序向右匹配，直到遇到范围查询停止。

19.tcp如何保证传输的可靠性?

校验码、编号、确认、超时重传、流量控制、拥塞控制。

20.如何保证有序性?

编号，确认，重传

21.LRU算法具体实现是啥?

双向链表+HashMap

22.详细介绍下cpu内存?

24.进程通信方式

- 共享数据结构，例如信号量机制
- 共享存储区
- 无名管道
- 命名管道
- 消息缓冲队列
- 消息队列
- socket
- rpc

27.cpu内存你了解吗？（emm，我把jvm具体说了下，什么栈什么堆）

类似JVM内存模型

JVM内存模型可分为程序计数器，Java虚拟机栈，本地方法栈，堆，本地方法区，运行时常量池，直接内存。 其中程序计数器、Java虚拟机栈、本地方法栈是线程独有的。

28.cpu缓存你知道吗？（我把cache的原理啥的说了一遍，面试官追问cache的缓存如何和外存的缓存保持一致性？我心想：我面得是后端吗？）

局部性原理

29.进程和线程和协程的区别?

30.进程与线程，以及通信方式

31.进程调度算法

- 先来先服务
- 短作业优先
- 最短剩余时间调度算法
- 优先级调度算法
- 高相应比优先调度算法
- 时间片轮转调度算法
- 多级反馈队列调度算法

32.进程间怎么通信

- 共享数据结构
- 共享存储区
- 无名管道
- 命名管道
- 消息缓冲队列
- 消息队列

- socket
- rpc

33.线程 进程 协程关系

- 一个进程可以有多个线程，一个线程可以有多个协程
- 进程，线程的切换需要切换到内核，协程的切换在用户态。
- 进程、线程可以并行执行，协程只能一个一个执行。无法利用多核CPU。

34.内存置换算法有什么，说一下时钟置换算法

- 最佳置换算法
- 先进先出置换算法
- 最近最久未使用算法
- 最近最少使用算法
- 时钟置换算法
- 给内存中的每一帧加一个使用位标记，当需要进行内存置换时，遍历内存中的帧，找到第一个使用位为0的帧替换掉，每走过一个位置，将帧的使用为置零，如果走了一圈都没有找到，则从头开始，此时所有的帧使用位均为0，第一个就可以置换。

36.线程高速缓存讲一讲?

cache 局部性原理，将一些使用频繁的数据放到访问速度很快的cache中。

37.内存分配高低地址?

大端模式和小端模式?

- 小段模式：低字节对应低位
- 大端模式：高字节对应地位

38.来讲一讲动态库加载机制?

42.还问了redis

43.问操作系统我有点猝不及防，后来问了为什么共享内存最快，我蒙了个减少用户态内核态切换居然对了😂. 进程通信方式

51.介绍银行家算法

当进程首次申请资源时，先测该进程需求的最大资源数是否能被满足，不满足就推迟分配。当进程在执行中继续申请资源时，测试该进程占有的资源加上申请的资源是否大于系统最大资源，大于则推迟分配，不大于则可以分配。

- 可利用资源
- 最大需求矩阵
- 分配矩阵
- 需求矩阵

52.死锁

多个进程因为资源竞争而造成的一种僵局（互相等待），如果没有外力，这些进程都无法向前推进。

- 互斥
- 请求并保持
- 不可剥夺
- 循环等待

62socket通信用的函数说一下

mysql

1.Mysql索引（B+树）

答：可以进行范围查询，并且查询速度较快。 对于其他数据结构做索引的优势（哈希表，二叉树，B树）

2.一个Mysql操作很慢，怎样排查原因。（设置慢查询时间，查询慢查询日志，explain查看慢的sql语句的查询情况）

3.一个联合索引(a,b,c)，where b=.. and c=....走不走索引？为什么？哪些走索引?

答：不会，最左匹配原则

4.MySQL 索引类型

主键索引、唯一索引、普通索引，全文索引 主键索引：不允许重复，不允许为null值，一个表只能有一个主键 唯一索引：不允许重复，允许为null值，一个表允许多个列创建唯一索引 ALTER TABLE table_name ADD UNIQUE (column) 普通索引：基本的索引类型，没有唯一性的限制，可以为null值 ALTER TABLE table_name ADD INDEX (column) 全文索引： ALTER TABLE table_name ADD FULLTEXT (column)

5.数据库索引，b+树，为什么用他

答：数据库索引有主键索引、唯一索引、普通索引、全文索引。 索引的数据结构有b+树，b树，哈希 我们一般选用b+树。 b+树的优缺点： 优点：相对于其他两种，b+树不仅可以进行等值查询，还可以进行范围查询。 相对于b树，b+树的非叶节点只包含索引不包含数据，同样的块可以保存更多的数据项，可以有效降低树高，减少磁盘IO次数。 缺点：必须查询到叶结点才能查到，而哈希只需要O(1)时间复杂度，b树可能在非叶节点就查询到。这也同样是b+树的优点，查询性能稳定。

6.悲观锁，乐观锁

常说的并发控制，一般都和数据库管理系统（DBMS）有关。在DBMS中的并发控制的任务，是确保在多个事务同时存取数据库中同一数据时，不破坏事务的隔离性和统一性以及数据库的统一性。 实现并发控制的主要手段大致可以分为乐观并发控制和悲观并发控制两种。

首先要明确：无论是悲观锁还是乐观锁，都是人们定义出来的概念，可以认为是一种思想。其实不仅仅是关系型数据库系统中有乐观锁和悲观锁的概念，像hibernate、tair、memcache等都有类似的概念。所以，不应该拿乐观锁、悲观锁和其他的数据库锁等进行对比。

悲观锁：对数据的修改抱有悲观态度的并发控制方式。我们一般认为数据被并发修改的概率比较大， 所以需要在修改之前先加锁。更安全但是效率低，有可能会造成死锁。

有读锁和写锁。

悲观锁实现方式：悲观锁的实现，往往依靠数据库提供的锁机制。在数据库中，悲观锁的流程如下：

在对记录进行修改前，先尝试为该记录加上排他锁（exclusive locking）。

如果加锁失败，说明该记录正在被修改，那么当前查询可能要等待或者抛出异常。具体响应方式由开发者根据实际需要决定。

如果成功加锁，那么就可以对记录做修改，事务完成后就会解锁了。

期间如果有其他对该记录做修改或加排他锁的操作，都会等待解锁或直接抛出异常。

拿比较常用的MySql Innodb引擎举例，来说明一下在SQL中如何使用悲观锁。

要使用悲观锁，必须关闭MySQL数据库的自动提交属性。因为MySQL默认使用autocommit模式，也就是说，当执行一个更新操作后，MySQL会立刻将结果进行提交。（sql语句：set autocommit=0）

以淘宝下单过程中扣减库存的需求说明一下悲观锁的使用：

悲观锁使用 以上，在对id = 1的记录修改前，先通过for update的方式进行加锁，然后再进行修改。这就是比较典型的悲观锁策略。

```
// 0.开启事务

begin;

// 1.查询出商品库存信息

select quantity from items where id=1 for update;

// 2.修改商品库存为2

update items set quantity=2 where id=1;

// 3.提交事务

commit;
```

如果以上修改库存的代码发生并发，同一时间只有一个线程可以开启事务并获得id=1的锁，其它的事务必须等本次事务提交之后才能执行。这样可以保证当前的数据不会被其它事务修改。

上面提到，使用select...for update会把数据给锁住，不过需要注意一些锁的级别，MySQL InnoDB默认行级锁。行级锁都是基于索引的，如果一条SQL语句用不到索引是不会使用行级锁的，会使用表级锁把整张表锁住，这点需要注意。

乐观锁：乐观锁是相对悲观锁而言的，乐观锁假设数据一般情况下不会造成冲突，所以在数据进行提交更新的时候，才会正式对数据的冲突与否进行检测，如果发现冲突了，则返回给用户错误的信息，让用户决定如何去做。

相对于悲观锁，在对数据库进行处理的时候，乐观锁并不会使用数据库提供的锁机制。一般的实现乐观锁的方式就是记录数据版本。

乐观锁实现方式

使用乐观锁就不需要借助数据库的锁机制了。

乐观锁的概念中其实已经阐述了它的具体实现细节。主要就是两个步骤：冲突检测和数据更新。其实现方式有一种比较典型的就是CAS(Compare and Swap)。

CAS是项乐观锁技术，当多个线程尝试使用CAS同时更新同一个变量时，只有其中一个线程能更新变量的值，而其它线程都失败，失败的线程并不会被挂起，而是被告知这次竞争中失败，并可以再次尝试。

比如前面的扣减库存问题，通过乐观锁可以实现如下：

乐观锁使用 以上，在更新之前，先查询一下库存表中当前库存数（quantity），然后在做update的时候，以库存数作为一个修改条件。当提交更新的时候，判断数据库表对应记录的当前库存数与第一次取出来的库存数进行比对，如果数据库表当前库存数与第一次取出来的库存数相等，则予以更新，否则认为是过期数据。

以上更新语句存在一个比较重要的问题，即传说中的ABA问题。

比如说一个线程one从数据库中取出库存数3，这时候另一个线程two也从数据库中取出库存数3，并且two进行了一些操作变成了2，然后two又将库存数变成3，这时候线程one进行CAS操作发现数据库中仍然是3，然后one操作成功。尽管线程one的CAS操作成功，但是不代表这个过程就是没有问题的。

ABA 有一个比较好的办法可以解决ABA问题，那就是通过一个单独的可以顺序递增的version字段。改为以下方式即可：

ABA的解决 乐观锁每次在执行数据的修改操作时，都会带上一个版本号，一旦版本号和数据的版本号一致就可以执行修改操作并对版本号执行+1操作，否则就执行失败。因为每次操作的版本号都会随之增加，所以不会出现ABA问题，因为版本号只会增加不会减少。

除了version以外，还可以使用时间戳，因为时间戳天然具有顺序递增性。

以上SQL其实还是有一定的问题的，就是一旦遇上高并发的时候，就只有一个线程可以修改成功，那么就会存在大量的失败。对于像淘宝这样的电商网站，高并发是常有的事，总让用户感知到失败显然是不合理的。所以，还是要想办法减少乐观锁的粒度的。有一条比较好的建议，可以减小乐观锁力度，最大程度的提升吞吐率，提高并发能力！如下：

优化 以上SQL语句中，如果用户下单数为1，则通过quantity - 1 > 0的方式进行乐观锁控制。

以上update语句，在执行过程中，会在一次原子操作中自己查询一遍quantity的值，并将其扣减掉1。

高并发环境下锁粒度把控是一门重要的学问，选择一个好的锁，在保证数据安全的情况下，可以大大提升吞吐率，进而提升性能。

五、如何选择 我认为修改更新操作较多时，使用悲观锁。 查询远大于修改的场景下，用乐观锁可以提高效率。 在乐观锁与悲观锁的选择上面，主要看下两者的区别以及适用场景就可以了。

乐观锁并未真正加锁，效率高。一旦锁的粒度掌握不好，更新失败的概率就会比较高，容易发生业务失败。 悲观锁依赖数据库锁，效率低。更新失败的概率比较低。 随着互联网三高架构（高并发、高性能、高可用）的提出，悲观锁已经越来越少的被应用到生产环境中了，尤其是并发量比较大的业务场景。

9.mysql引擎有哪些，有啥区别

存储引擎Storage engine：MySQL中的数据、索引以及其他对象是如何存储的，是一套文件系统的实现。

引擎有myisam引擎和inoodb引擎

区别：

myisam是非聚簇索引，支持表级锁，不支持事务，不支持外键，支持全文索引，保存了表的总数据行个数，查询总行数更快，加了查询条件则两者没有区别。

inoodb是聚簇索引，支持表级锁和行级锁（必须在使用索引的情况），支持事务，支持外键，不支持全文索引，未保存表的总数据行。

适用场景：

MyISAM:以读写插入为主的应用程序，比如博客系统、新闻门户网站。

Innodb:更新（删除）操作频率也高，或者要保证数据的完整性；并发量高，支持事务和外键。比如OA自动化办公系统。

10.关于索引的使用，哪些语句使用到了索引，具体的内容没来得及拷贝下来。

具体参考最左匹配原则

11.索引的优点与缺点，底层怎么实现的?

索引是什么?

索引是一种数据结构。数据库索引，是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。索引的实现通常使用B树及其变种B+树。

更通俗的说，索引就相当于目录。为了方便查找书中的内容，通过对内容建立索引形成目录。索引是一个文件，它是要占据物理空间的。

索引的优点：

加快查询速度。 索引的缺点：

创建、维护索引时间开销，对数据库进行插入，更新，删除操作时，也要同时对索引维护。

占据磁盘空间。

底层：可以用b+树，b树，哈希等数据结构。（转向介绍b树等对比优缺点）

12.B+树的特点，与二叉树的区别

b+树的特点：

有n棵子树的节点中含有n个关键字。

非叶节点只包含索引，不包含数据，所有的数据都存储在叶节点，并且按照从小到大的顺序从左至右链接起来。

通常B+树含有两个头指针，一个指向根节点，一个指向关键字最小的叶子节点。

与二叉树的区别：

b+树是多叉平衡查找树，子树一般大于二，二叉树只有左右子树，所以b+树树高小。

b+树的叶子节点全部连接起来。

13.事务的特性，解释一下它们

事务：

事务是满足ACID特性的一组操作，可以通过commit提交一个事务，也可以通过rollback回滚一个事务。

ACID：

原子性：一组操作要么全部成功，要么全部失败。undo

一致性：数据库在事务执行前后保持一致性状态。在一致性状态下，所有事务对同一个事务的读取结果都是一致的。（不是很理解)

隔离性：在一个事务未提交之前，它所做的修改对其他事务不可见。

持久性：一旦事务提交，数据将永久保存才数据库。redo

14.其中的隔离性分几个级别？每个级别解释一下什么意思？

隔离级别有四种：

未提交读：一个事务做了修改，还没有提交，修改的数据就可以被其他事务读取。可能导致脏读、不可重复度、幻读；

提交读：只有当一个事务提交了，它所做的修改才能被其他食物看见。解决了脏读，未解决不可重复读和幻读。

可重复读：在一个事务中，对一个数据无论读多少次都是一致的。可以解决脏读、不可重复读，未解决幻读。

可串行化：仿佛像是一个一个一次串行执行一样，结局了脏读、不可重复读、幻读问题。

15.串行化的优缺点

优点：满足了事务的ACID特性，更安全。

缺点：满足这些特性的代价是大量加锁，降低了执行效率。

16.SQL join操作，B树B+树

inner join:在两张表进行连接查询时，只保留两张表中匹配的项。

left join:在两张表连接查询时，会返回左表中所有的行，即使右表中没有。

right join:在两张表连接查询时，会返回右表中所有的行，即使左表中没有。

full join:在两张表连接查询时，会返回左右表中所有的行。

B树和B+树的区别：

B+树的非叶节点只保存索引不保存数据。同样大小的块可以保存更多的项，树高更低，减少磁盘IO。

B+树的所有叶子节点按大小顺序从左至右连接，形成链表，b树没有。

17.数据库索引，b+树

主键索引：不可重复，无null值，只能有一个

唯一索引：不可重复，可以有null值，可以有多个

普通索引：可重复，可为null

全文索引

哈希，b树，b+树

18.最左索引匹配

mysql索引会一直向右匹配，直到遇到范围查询停止。

19.乐观锁，悲观锁

乐观锁：对发生并发冲突的情况保持乐观态度，在读取数据的时候，直接读取，认为不会产生并发冲突。在修改数据的时候会进行冲突检测，数据提交。

悲观锁：对并发冲突的发生保持悲观，认为总是会发生冲突，读写数据时都会先加锁再操作。

20.数据库三大范式 有哪些反范式的设计

第一范式：属性不可分

第二范式：第一范式基础上，非主键属性必须完全函数依赖于键码，不能部份依赖。

第三范式：第二范式基础上，非主属性不传递函数依赖于键码。

哪些反范式的设计

21.join 和 left join、right join 的区别

两表进行连接查询时，。。。。。

22.数据库事务 持久性是什么 隔离级别 幻读是什么

ACID

23.主键跟索引的区别

1:主键是为了标识数据库记录唯一性,不允许记录重复,且键值不能为空,主键也是一个特殊索引.

2:数据表中只允许有一个主键,但是可以有多个索引.

3.使用主键会数据库会自动创建主索引,也可以在非主键上创建索引,方便查询效率.

4:索引可以提高查询速度,它就相当于字典的目录,可以通过它很快查询到想要的结果,而不需要进行全表扫描.

5:主键索引外索引的值可以为空.

6:主键也可以由多个字段组成,组成复合主键,同时主键肯定也是唯一索引.

7:唯一索引则表示该索引值唯一,可以由一个或几个字段组成,一个表可以有多个唯一索引.

24.索引的特点

第一，通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

第二，可以大大加快数据的检索速度，这也是创建索引的最主要的原因。

第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。

第四，在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。

第五，通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

25.MySQL的innodb和myisam区别

inoodb:非聚簇索引，不支持事务、外键、支持全文索引、保存了全部数据行个数，查询总行数时速度快。适用于读写插入比较频繁的场景。

myisam:聚簇索引，支持事务、外键，不支持全文索引，查询总行数个数慢。适用于删除更新也比较频繁的场景。

26.InnoDB引擎主键为什么设成自增?

因为在InnoDB存储引擎中，主键索引是作为聚簇索引存在的，也就是说，主键索引的B+树叶子节点上存储了主键索引以及全部的数据(按照顺序)，如果主键索引是自增ID，那么只需要不断向后排列即可，如果是UUID，由于到来的ID与原来的大小不确定，会造成非常多的数据插入，数据移动，然后导致产生很多的内存碎片，进而造成插入性能的下降。

总之，在数据量大一些的情况下，用自增主键性能会好一些。

关于主键是聚簇索引，如果没有主键，InnoDB会选择一个不含null值得唯一索引来作为聚簇索引，如果没有唯一键，会生成一个隐式的主键。

27.为什么用B+树

28.数据库索引

主键索引、唯一索引、普通索引、全文索引

哈希，b树，b+树

29.四种隔离级别

未提交读、提交读、可重复读、可串行化

30.默认的隔离级别

这里需要注意的是：Mysql 默认采用的 REPEATABLE_READ隔离级别 Oracle 默认采用的 READ_COMMITTED隔离级别

事务隔离机制的实现基于锁机制和并发调度。其中并发调度使用的是MVVC（多版本并发控制），通过保存修改的旧版本信息来支持并发一致性读和回滚等特性。

因为隔离级别越低，事务请求的锁越少，所以大部分数据库系统的隔离级别都是READ-COMMITTED(读取提交内容):，但是你要知道的是InnoDB 存储引擎默认使用 **REPEATABLE-READ（可重读）**并不会有任何性能损失。

InnoDB 存储引擎在 分布式事务 的情况下一般会用到**SERIALIZABLE(可串行化)**隔离级别。

31.脏读与幻读

脏读：比如一个数据i开始为1，事务1对它进行修改使i=2但是未提交，这时事务2读取i=2，事务1此时回滚，i又变成了1，事务2此时读取到的就是脏数据。

幻读：在同一事物中都两次范围值不一致，比如事务1第一次读取表中年龄在20-30间得人数为5，此时事务二又添加了5个年龄在20-30之间的人，事务一再次读取就变成了10，前后不一致。这就是幻读。

32.事务的隔离级别?

未提交读、提交读、可重复读、可串行化

33.数据库的索引怎么实现的？（B+树）

b+树，b树，哈希

33.B+树和B树的区别？B+树的优点？

B+树得非叶子节点只保存索引不保存数据，同样的内存块可以包含更多的项，降低树高，减少IO次数。

B+树得叶子节点按递增顺序从左至右依次连接。

B+树优点：

可以进行等值查询，也可以进行区间查询。

树高低，较少磁盘IO次数。

缺点是必须找到叶结点才能查找完成，也是优点，查询稳定。

34.数据库的三范式是什么？

第一范式：属性不可分

第二范式：在第一范式基础上，非主属性完全依赖于键码。不可部份依赖。

第三范式：在第二范式基础上，非主属性不可传递函数依赖于键码。

35.为什么要用索引，索引的实现，b+树和b树相比优点

加快查询速度。

b+树，b树，哈希

树高低，区间查询。

36.说一下事务，事务隔离级别，怎么解决不可重复读

事务是满足ACID特性的一组操作。

未提交读、提交读、可重复读、可串行化。

37.数据库innodb如何实现啥来着，我回答事务

(1)读未提交：select不加锁，可能出现读脏；

(2)读提交(RC)：普通select快照读，锁select /update /delete 会使用记录锁，可能出现不可重复读；

(3)可重复读(RR)：普通select快照读，锁select /update /delete 根据查询条件情况，会选择记录锁，或者间隙锁/临键锁，以防止读取到幻影记录；

(4)串行化：select隐式转化为select ... in share mode，会被update与delete互斥；

InnoDB默认的隔离级别是RR，用得最多的隔离级别是RC

38.innodb索引，b+树那一套

b+树索引和哈希索引，用的更多的是b+树索引。

39.聚簇索引等

将数据存储与索引放到了一块，找到索引也就找到了数据.

40.（这题我回答错了，大佬们帮我看看）写了一个sql select * from table where a>2 and b=3 问我 联合索引是a b的话 ，能不能使用到 联合索引？我说可以，因为最左匹配，他说我说反了。

41.事务隔离级别有什么

未提交读、提交读、可重复读、可串行化。

可重复读级别下，两个事务并发读取一个 i=1，并且i++，最终结果是什么（凉）

2,3

42.解释一下聚合索引？使用条件呢?

数据存储和索引放到了一起，找到了索引就找了数据。

43.说一下主键和索引

44.熟悉mysql的引擎吗（不熟悉QAQ

innodb,myisam

myisam:非聚簇索引，不支持事务，外键，支持全文索引，支持表锁，查找全表个数快。

inoodb:聚簇索引，支持事务，外键。不支持全文索引，支持表锁、行锁。

45.数据库唯一索引和主键

主键索引不可重复、不可为null,只能有一个

唯一索引不可重复、可为null，可以有多个

b+树索引和哈希索引的应用场景，区别MySQL复合索引

MySQL引擎MyISAM和InnoDB有什么区别

redo与undo作者：偏远山区的高ping战士 链接：https://www.nowcoder.com/discuss/404960?type=post&order=create&pos=&page=1&channel=666&source_id=search_post 来源：牛客网

数据库主键和索引的区别（扯到了回表查询） 5.回表查询是什么 6.所有存储引擎都是这样吗？（就是myisam用的非聚集，扯一下两个区别） 7.解释脏读、不可重复读、幻影读，举例子说明（幻影读没说好，还要学一学） 8.隔离级别作者：GalaxyMoon 链接：https://www.nowcoder.com/discuss/406812?type=post&order=create&pos=&page=1&channel=666&source_id=search_post 来源：牛客网

说一下知道的DB引擎、区别--innodb myisam memory等自由发挥 索引以及优化，联合索引问题（这里记不太清了，只记得有索引的问题） 说一下MySQL隔离级别--RU RC RR Serializable 每种隔离级别解决的问题，以及innodb的幻读解决方案，MVCC 说一下几种锁和实现--乐观锁、悲观锁、共享锁、互斥锁、行锁、表锁

linux

1.Linux操作相关

2.Linux上我怎么查看某端口被什么进程占用

- losf -i 端口号
- netstat -tunlp|grep 端口号

3.如何修改文件的权限.

- chmod 三个数字 文件名

4.linux操作系统awk(这个我说我用得少),那再随便说五个命令,都是干嘛的(esay)

- awk是一个报告生成器，拥有强大的文本格式化能力。

5.如何查看某个进程开启的socket(查看/proc/pid号/fd文件),

6.如何查看tcp链接(没太交流明白,下来发现好像就是netstat?)

7.Linux的Kill命令（-9信号的作用）

8.Linux的进程间的通信.

9.进程使用的状态怎么查看（我说的是windows的任务管理器哈哈哈哈）

redis

redis基本数据结构

String,list,hash,set,zset

redis有序集合的底层实现（ziplist和skiplist，跳跃表是什么）

Redis数据类型，有序集合实现原理Redis

分布式锁使用在了什么地方 怎么实现的 除了 Redis 还有什么方式可以实现

redis底层实现

set的使用场景

有序set的使用场景

redis缓存

Redis持久化机制

你常用的数据结构、说一下原理

redis单线程多线程、有什么优劣

集群、哨兵、主从

持久化方案

缓存一致性方案

redis事务用过吗，是怎么样的?

IO

同步io和异步io区别 5.异步io.在java中调用什么api，我忘了哭了，然后面试官说没关系，你没用过我理解，我只是想了解一下而已😂😂异步IO和同步IO的区别？同步和阻塞的区别？多路IO模型 NIO异步io是什么

- 阻塞IO
- 非阻塞IO
- 多路复用IO
- 信号驱动IO
- 异步IO

java基础

多态

多态就是同一个接口或父类指向不同的实例，执行不同操作。

多态必须满足三个条件：

- 继承
- 重写
- 父类或接口指向不同子类

ArrayList 和 LinkedList 区别，使用场景

array 和 list区别

Map 有几种，LinkedHashMap的数据结构

- TreeMap:基于红黑树实现
- HashMap:基于哈希表
- Hashtable:类似于HashMap，线程安全。
- LinkedHashMap:使用双向链表来维护元素的顺序，顺序为插入顺序或者LRU（最近最少使用）顺序。

怎么实现的深拷贝、浅拷贝

ArrayList如何自己封装成一个安全的文件

多线程volite关键字什么作用，追问:可见性你怎么理解的？.java如何实现并发（Thread runable）

可见性是指，当一个线程修改一个共享变量时，其他线程能够看到这个修改。

java哈希表

java的treemap

java的concurrent HashMap

反射 反射的弊端

hashmap冲突解决，

链表遍历效率低下怎么解决

##jdbc

hashmap设计及存在相同key的时候的解决方法（太紧张了 没想就答了，结果崩了）（太紧张答错了，凉）.

多态你怎么理解?

HashTable原理？扩容机制？时间复杂度，空间复杂度？怎么解决冲突?

synchronized与volatile区别?

- volatile本质是在告诉jvm当前变量在寄存器（工作内存）中的值是不确定的，需要从主存中读取；

- synchronized则是锁定当前变量，只有当前线程可以访问该变量，其他线程被阻塞住。
- volatile仅能使用在变量级别；synchronized则可以使用在变量、方法、和类级别的
- volatile仅能实现变量的修改可见性，不能保证原子性；而synchronized则可以保证变量的修改可见性和原子性
- volatile不会造成线程的阻塞；synchronized可能会造成线程的阻塞。
- volatile标记的变量不会被编译器优化；synchronized标记的变量可以被编译器优化。

java的IO模型有哪些？（BIO、NIO、AIO）

Synchronized关键字，

说到有序性的时候突然卡了，不知道怎么解释。。按照自己理解说了## 下哪些情况线程会被阻塞

深拷贝浅拷贝

HashMap 底层数据结构，链表长度转化、扩容

介绍一下主要的几种Java容器类，Collection和Map接口

ArrayList扩容机制、和LinkedList之间的实现区别

有哪几种set？HashSet、TreeSet、LinkedHashSet之前的实现区别，如果遍历的话，会是怎么样一个顺序？（第一次没太听懂，再描述了下才发现意思是hash tree和link hash下存储方式的区别）

Map呢？

三种map的实现

介绍一下hashtable、hashmap、concurrenthashmap之间的区别

说一下synchronize和Lock（不记得是让说区别还是只说前面那个了）

- synchronized是JVM层面实现的，Lock是JDK实现的。

- reentrantlock可以中断，synchronized不行。
- synchronized不是公平锁，reentrantlock可以公平非公平均可。
- reentranlock可以绑定多个条件，synchronized只有一个。
- synchronized不用手动释放锁，reentranlock要手动释放锁。

多线程几个问题（这里记不太清了）

算法与数据结构

1.怎样判断一个链表是否有环

用快慢指针，快慢指针从表头开始遍历，快指针每次走两步，慢指针每次走一步，如果存在环，它们必然会在环中相遇。否则无环。

2.怎样找到有环链表的环的开始节点（找到快慢指针的相遇节点，快指针改为走一步，慢指针指向head，同时走，在环开始节点相遇）

是的，先用快慢指针找到相遇节点，然后慢指针指向head，快慢指针同时开始遍历，同样的速度。最终相遇节点就是环的开始节点。

3.常见排序算法(快排，堆排序)

- 冒泡排序
- 插入排序
- 希尔排序
- 快速排序
- 堆排序
- 归并排序

4.快排的时间复杂度，最坏的时候也是NlogN?（后面这个问题回答的不是很好。。。）

最好的时间复杂度是O(NlogN)，最坏也是O(N2)

5.常用的排序算法

- 冒泡排序
- 快速排序
- 插入排序
- 希尔排序
- 简单选择排序
- 堆排序
- 归并排序

6.数组中最小的k个数?

快速选择

7.快速排序

8.实现二进制转10进制

，5分钟内写完

9.如果有n条直线，且三线无交点，问有多少种交点（0表示平行，1表示相交）

dp

10.算法题：用数组实现栈，实现pop与push，支持扩容保证性能。

11.归并排序讲一下

12.手撕大根堆

13.堆是什么，数据结构，时间复杂度

可以把堆看成一颗完全二叉树，这棵完全二叉树满足：任何一个非页节点都不大于（或者不小于）它的叶子节点，若父亲大孩子小则是大顶堆，若父亲小孩子大则是小顶堆。

完全二叉树

最好是O(nlogn),最坏也是O(nlogn)。

14.排序算法有哪些，归并排序时间复杂度，是不是稳定的

- 冒泡排序
- 快速排序
- 插入排序
- 希尔排序
- 堆排序
- 对并排序
- 简单选择排序

15.链表和数组区别

- 链表对元素的插入、删除较为简单，对查找要整个遍历，较为复杂。
- 数组对于元素的插入、删除较为复杂，要移动对应元素后面的好多元素，但是对于根据索引查询来说较为简单。

16.两千万数据，取最大的一千个。

小顶堆

17.是找最大不重复子串长度

18.两个栈如何模拟一个队列

19.arrayList和linkList区别

20.LRU算法是如何实现的？（我说的是哈希表+链表实现，并要把具体实现的思路讲明白）

21.1000个数据，查找出现次数最多的k个数字（优先队列）还问了快排的最好最差复杂度情况

22.一千万个数找出其中最大的k个数?

23.怎么判断一个链表有没有环？怎么判断环里面几个节点?

手撕算法

把输入的AaBbCc字符串 按照输出小写字母在前大写字母在后的顺序输出，比如AaBbCc输出abcABC（最好不借助其他空间） 我比较菜，借助了StringBuilder 然后两次for循环

topN问题及其复杂度，最小n个，总共m个数， m远大于n，复杂度答了：O(M) 面试官说O(logn)不能忽略，所以是：O(MlogN + NlogN)

（凉）有环链表的判断 和 环长度判断（凉）

说一下LRU过程

巨大文件，TOPK排序

数组和链表

哈希解决冲突的方法

还问了大文件找出排名前1000的数据

手撕代码

最长不重复子序列 然后写完之后问怎么优化到O（n）

算法题，反转字符串

算法题，字符串中大小写字母分成前后两部分，字母顺序不变

接雨水，以前看过但是忘了。。。力扣42题，被问过不止一次。。。

给个m，求1到m的最大奇约数的和。

一个数组a，给个s，返回一组在数组里差是s的两个数。

数组链表区别

哈希表原理，常见碰撞算法

快速排序讲一讲

平衡二叉树

快排（swap的时候我写错了两个地方，面试官没有指出，面完再回去看才发现）

最好最差时间复杂度

哈希冲突解决方法

哈希表某个桶中数据被删除怎么办

给前序中序，还原二叉树

两千万个文件找最小的一千个（答错了，应该用大顶堆，答成了小顶堆）

就是要用大顶堆，每次找出最大的数丢弃，最后保留的就是最小的1000个数。

topk场景题（怎么统计大量帖子中的点赞数量前10）

如果用堆，内存装不下怎么办介绍分治算法

问了快排,堆排的原理及复杂度比较

数组和链表

什么是平衡二叉树

编程：实现反转链表

编程：快速排序快排时间复杂度，什么是时候是最坏，怎么优化

先写题，找奇点，说思路，O（n）时间复杂度，然后让优化，优化找到O（logn）

然后手写给8分钟

给定一组非负整数，重新排列它们的顺序使之组成一个最大的整数（如给定[3,30,34,5,9]，输出9534330），结果用字符串输出....记得我看过，但是忘了，气死，后来想了一会，用了比较笨的办法

写一个函数，将输入的字符串中大写字母移动到字符串的末尾，同时需要保留出现的相对顺序, 例如输入AaBbCc 返回 abcABC, 要求不使用额外内存..... 想了一会， 用类似冒泡的方法解决

前面知识题应该都比较平和，两道算法题没答好，都是用的比较笨的方法.....2000w个数找前1000个（完全不会这种问题，扯的堆排、快排）

已知前序和中序 ，求高度

算法1：两个栈做一个队列

算法2：数据流找最大的 n 个数

算法：4321 找比它小的字典序数字，那么是 4312，时间复杂度不能是 O(n2)

TCP UDP区别，简述一下三次握手过程

让你来做一个直播流软件，从上到下涉及协议栈？作者：essilon

当时只是简单的回答了一下，分情况高质量的话考虑用TCP/RTMP长连接，或者弱网用UDP/QUIC方案，然后用CDN做内容分发，直播场景比较多，当时面试官问的是实现zoom怎么选择协议栈，zoom因为涉及到双向直播的话，考虑延迟建议用后者UDP/QUIC方案，传输层往下就没咋说了。我也只是略知皮毛这个 topk相关问题以及解决上来一道算法题、

大概是升序旋转数组最小值，leetcode 153，给了线性和二分两种思路，然后让手写一下二分。这里有点尴尬，二分最开始写错了下标更新，面试官看了下没让改直接继续了

jvm

Java gcjvm 垃圾回收Java 类加载器说一下 Java内存分为哪几部分 这几个内存部分哪里会出现内存溢出的情况，只说了堆和栈的jvm 新生代老年代的 GC