**如果需要了解了解算法以及对面试没有很大把握的同学，建议把面试时间约在一周或者两周后，期间好好准备**

**另外面试时间可以调整，至少提前一天修改时间**

**面试可以现场可以视频，如果时间不方便可以约视频面试**

**面试共计四轮**：

第一轮笔试或直接面试，重基础，细节

第二轮面试技术问题更深入一些

第三轮着重项目经验和技术广都和深度

第四轮HR面试，看稳定性，意向性，性价比，了解薪资

最后是offer沟通，定级、出薪资方案、入职时间等等

技术面试要知其然并知其所以然，每一轮面试或多或少都有算法，请做一些准备

**刷题网站：**

https://leetcode-cn.com/explore/featured/card/bytedance/

**温馨提示：**

1.如果是视频面试请准备好**网络**，不要因为网络这种因素导致您的面试效果不好。

2. 不管结果如何一定要做完，至少态度是端正的

3.头条以技术为导向、喜欢**积极、谦虚、有创业精神**的同学，因为这个也是他们的企业文化，所以最好表现出自己对于技术的热爱还有积极的心态。

4. 如果觉得有什么不合适或者是不太满意的地方，千万不要在面试的时候说，或者表现出不太喜欢的这种心情，这也是为了您之后的发展着想，他们会有**面试评价**，影响您之后在头条的所有面试，面试评价好的话也是会对您有好处的，如果对于**岗位有不满意**的我们可以再**调整**。

5 没有提薪资的问题您也不要主动提，因为之后会有专门的谈薪环节，我们都是会在目前的薪资上有一定的涨幅，这个还是看具体的情况，问了薪资您就说一下现在的薪资情况：月薪\*多少个月+什么补贴，期望能在现在的基础上有一定的涨幅，这样回答就好

**后台题库：仅做参考，不同的面试官有不同的面试风格和提问方式、内容**

**一轮**

1.二叉树镜像

2.大概是题目给出了一个例子，然后让根据这个例子，在输入的数组中找出第k大的素数，输入数组可能包含不是素数的数字

3.求无向无环题的直径

4.求二叉树的左视图看到的节点

5.二叉树求和，找出数字只和等于21的数组下边，要求复杂度要小于o(logn)

6.随机给两个单链表的数，如1->2->3->4，2->3->4，求和的代码实现

7.一个二叉排序树，自己实现其一个Iterator

8.一个顺序递增的数组，内含重复的数，现查找一个给定的数在不在数组，在的话给出最后一个出现的index

9. 堆排序相关

10. 动态规划

11. 一道算法，总共七八道题目吧(数据库，操作系统，缓存，数据结构)

算法：给定数组（数字可重复），求两数和为21的所有下标

数据库（mysql）：两种表引擎区别，索引，事务隔离级别

<https://blog.csdn.net/fath_kevin/article/details/81322565>

数据结构：redis中zset实现原理：

zset的每一个成员都有一个分数与之对应，并且分数是可以重复的。有序集合的增删改由于有啦排序，执行效率就是非常快速的

Type key可以查看key所存值得类型

操作系统：LRU原理：

**LRU**是Least Recently Used的缩写，即最近最少使用，常用于页面置换算法，是为虚拟页式存储管理服务的。

LRU 算法的设计原则是：如果一个数据在最近一段时间没有被访问到，那么在将来它被访问的可能性也很小。也就是说，当限定的空间已存满数据时，应当把最久没有被访问到的数据淘汰。

### **基于哈希表和双向链表的LRU算法实现**

如果要自己实现一个LRU算法，可以用哈希表加双向链表实现：

设计思路是，使用哈希表存储 **key**，值为链表中的节点，节点中存储值，双向链表来记录节点的顺序，头部为最近访问节点。

LRU算法中有两种基本操作：

* get(key)：查询key对应的节点，如果key存在，将节点移动至链表头部。
* set(key, value)： 设置key对应的节点的值。如果key不存在，则新建节点，置于链表开头。如果链表长度超标，则将处于尾部的最后一个节点去掉。如果节点存在，更新节点的值，同时将节点置于链表头部。

<https://zhuanlan.zhihu.com/p/34133067>

另外还有2,3个，一个是大文件放在有限内存排序，一个是负载均衡相关的，具体忘记了

12. mysql 两种存储引擎的区别

13.where a = x and b > y ,,, where a >x,,,,, where b >y 三种情况如何加索引。

---没找到答案。

索引原则**1**.索引越少越好

原因：主要在修改数据时，第个索引都要进行更新，降低写速度。**2**.最窄的字段放在键的左边**3**.避免file sort排序，临时表和表扫描.

<https://www.cnblogs.com/liangyueyuan/p/9757330.html>

14.四种事务隔离的区别

15.操作系统的页面置换算法有哪几种。

16.N 个集群节点 一致性哈希的长问道的问题。

17.大文件内存排序问题。

18.100 G文件 8G内存 如何排序。思路：先分段排序,存储到临时文件中,然后合并.

使用10000个整数来模拟大数据,每次读取100个到内存中.

100G 数据，按照 100M（具体的可用内存大小） 内存拆分，然后排序有序的数据，然后写入到 file1,file2…file100。

多路归并。

第一回合：

从 file1,file2,file3……file100.取出第一个数。及最小的。所有的初始指针都是第一行。

min1=min(min1=min(fil1,file2,file3…..file100)；file2,file3…..file100)；min1 写入到大数据文件。大数据行数指针+1。min1对应的行数指针+1。第二回合：从file1,file2,file3……file100.取出第一个数。及最小的。所有的初始指针都是第一行。min1对应的行数指针+1。第二回合：从file1,file2,file3……file100.取出第一个数。及最小的。所有的初始指针都是第一行。min2 = min(fil1,fil1,file2,file3…..file100)；

min2写入到大数据文件。大数据行数指针+1。min2写入到大数据文件。大数据行数指针+1。min2 对应的行数指针+1。

多路归并排序：https://blog.csdn.net/clevercode/article/details/81743736

19. 两个单链表求和

20. qps限制怎么做

<https://blog.csdn.net/u012104435/article/details/50963698>

21. 分布式锁的内容

22. 两个链表分别表示两个数，对这两个链表加和，产生一个新的链表

23. 给定一个价格数字列表，计算哪天买入，哪天卖出利润最大

24. http状态码/header/请求方法, 哪些请求方法是幂等的

25. https加密原理

26. . python协程原理/缺点

27.. python GIL, 为什么有GIL 还需要threading

28 . python gevent协程调度原理/缺点

29.. 设计一个文本编辑器实现插入/删除/查询

30.. 头条文章向用户推送避免重复推送的问题

31.. 工作中遇到的印象深刻的问题

32. 两种存储引擎

33. http的返回码

34. mysql 索引

35..链表操作(在线写代码)

3 6. http状态码

37. ->UDP / TCP 协议

38. ->链表操作 [手写代码]

39. ->乐观锁 / 悲观锁

40 ->三次握手

41. ->Socket

42. ->Hbase 底层原理

43. ->Map 底层原理

44. ->数据库分表 / 分库 / 集群

总结:比较注重CS修养、底层源码实现

1. redis内存不足，如何解决

<https://blog.csdn.net/Fly_as_tadpole/article/details/85681520>

1，增加内存；

****2，使用内存淘汰策略。（LRU）****

****3，Redis集群。****

****第2点：****

我们知道，redis设置配置文件的****maxmemory****参数，可以控制其最大可用内存大小（字节）。

### **那么当所需内存，超过**maxmemory**怎么办？**

这个时候就该配置文件中的****maxmemory-policy****出场了。

其默认值是****noeviction****。



****redis中并不会准确的删除所有键中最近最少使用的键，而是随机抽取3个键，删除这三个键中最近最少使用的键。****

****那么3这个数字也是可以设置的，对应位置是配置文件中的maxmeory-samples.****

****3.集群怎么做****

Redis仅支持单实例，内存一般最多10~20GB。****对于内存动辄100~200GB的系统，就需要通过集群来支持了。****

Redis集群有三种方式：****客户端分片、代理分片、RedisCluster****

46. mysql索引的使用和原理

47. . epoll和poll,select的区别，说的是三种IO多路复用的用法。

select，poll，epoll都是IO多路复用的机制。I/O多路复用就通过一种机制，可以监视多个描述符，一旦某个描述符就绪（一般是读就绪或者写就绪），能够通知程序进行相应的读写操作。****但select，poll，epoll本质上都是同步I/O，因为他们都需要在读写事件就绪后自己负责进行读写，也就是说这个读写过程是阻塞的****，而异步I/O则无需自己负责进行读写，异步I/O的实现会负责把数据从内核拷贝到用户空间。

epoll跟select都能提供多路I/O复用的解决方案。在现在的Linux内核里有都能够支持，其中epoll是Linux所特有，而select则应该是POSIX所规定，一般操作系统均有实现。

<https://www.cnblogs.com/aspirant/p/9166944.html>

48. 给定一个二叉树，返回二叉树每层的最左的值。

49. 手里有一副扑克牌。按照下列规则吧他堆放桌上。一，拿出最上面的一张牌，放桌上，然后把接下来的一张牌放在扑克牌的最下面。循环，直到没有手牌。现在已知桌上牌的顺序。求原手牌的顺序。

50. 实现一个有顺序的LinkedList，实现deleteAll(int val)方法删除所有值为val的节点，和save(int val)方法添加一个值为val的节点。

51. 线程池的工作流程。线程的几种状态，什么情况会进入阻塞

52. JDK基础 集合 锁

53. Mysql的索引结构 聚簇索引

54. 限流。限流算法。令牌桶的不足

55. GC CMS，CMS的参数

56. JVM的一些排查工具jstat, jstack ,jmap

57. 工作中遇到的问题如何排查

58. LRU的实现

59. MySQL的索引实现方法

60. MySQL的存储引擎的区别

61. 一排盒子，每个盒子上标了能走的最大步数，从第一个盒子开始，判断能否走到最后的盒子

62. 栈，增加一个max方法；

63. 自己设计hash

64. 微博限定用户每次输入最多140个字符，用户如果传字符串很长的链接，怎么办

65. 两个线程，一个只能存有数组1、2、3和另一个存有a、b、c，然后通过调度，最终结果输出1a2b3c

66. 输入一个int数组，返回一个数组，其中奇数都在左边，偶数都在右边 附加条件 不使用额外的空间

67. 数据库如何建索引，和如何分库分表

68. 找到一个文件里出现次数最多的的数字，文件大小远大于内存容量

69. 给一个数组和一个整数，输出数组中所有和为给定整数的元素下标。

70. MySQL两种存储引擎的差异；

71. where a>1 and b>1;where a = 1; where b = 2,如何为这种条件语句建立索引；

72. LRU的实现。。

73. 介绍一个项目中遇到的问题。

74. 一个100G的文件，内存只有8G，如何给文件排序，文件内容都是按行存储的URL

**二轮**

1、面试的是java程序员哈，直接来就问我hashmap底层原理，同步和锁的区别，数据库这些，还给了一道情景题，如何限制每分钟每个手机号短信发送数量

头条的面试跟其他公司不是一个套路，一般就问几个问题，几个里面还有一半是手写算法实现

2、mysql索引原理，联合索引

项目里的技术hbase原理

二叉搜索树的打印

二维码登陆设计

HashMap实现

发短信业务，1分钟内一个号一个业务1000条

3.   场景设计：设计一个直播中答题获百万奖金的设计。分析场景中的难点

4. 场景设计：设计一个IP数据库，表中包括IP数据和地址。用户可以通过IP地址查询到地点

5. oracle和mysql

6. 聚簇索引和非聚簇索引

7. B+树和B树的区别

8. 同步和锁的概念

9. volatile

10. 算法题leetcode 670

11. 介绍经历、java线程池基础知识、自己实现一个线程池，然后是二叉树的bfs，最大公共子串的长度，最大不重复子串的长度

三轮

1、设计一个订单查询加缓存优化的

2、leetcode 670，然后系统设计题，设计群消息已读功能

3、都是结合简历问的 项目经历、实现的方法、遇到的难点，穿插着所用的技术。代码是考察设计一下搜索模块

候选人经验

面试4轮， 其中技术面三轮， HR一轮

一轮技术（45min）：

1. 自我介绍， 我30s说完。

2. 算法题 jump game II，分别用了dfs和dp解答。

3. 数据库优化，日期该如何建索引（这个我没想到聚合索引，他说我对mysql不熟），B+树的特性

4. 实现一个短链接服务，并实现永久保存功能。

二轮技术（1hr）：

算法题，LRU Cache的几个基本操作，主要是写数据结构，代码没时间会让你说说想法， 并问了linux page cache的原理。

设计一个微博系统， 按时序排列用户评论并实现分页功能， 写出redis/mysql 数据结构

设计一个文件下载服务器，各方面进行优化。 当时有从CDN，Nginx集群，磁盘集群，下载任务队列， 更换内存分配器，网卡驱动（RSS和NAPI）， 多路I/O， mmap快速加载文件，tcp优化，文件碎片分级缓存（SSD SATA）等各方面进行描述，每一处都会被问细节。

和面试官聊天， 聊团队和技术， 并问我对哪方面技术感兴趣（这个很关键，决定你下一轮面试的方向，我答得是底层，内核什么的）

三轮技术（45min）

先问为什么对底层感兴趣，然后问非阻塞I/O为什么比线程更加适合网络， 线程切换开销在哪， 那些玩意是线程私有的，如何自己用内核函数实现一个线程。

如何进行交叉编译和调试， 答得是编译成目标bin拷贝过去，起一个server用ptrace debug子进程，并用socket传debug命令， 同时提到了JTAG是通过硬件trap调试进程。面试官问了错误信息如何跟踪， 我答得是core dump， 又问core dump如何调出堆栈， 我答得是通过解析function frame（讲了esp， ebp， ret）

问了项目，问如何实现任务派发，我说是通过bash询问HPC自带的任务队列和资源管理进程来寻找最优节点实现的，面试官让我自己实现一个。 我说用redis 队列记录当前任务， 每次选最小的那一个。 想了一下又补充队列可以用zset按任务时间排序，能快速完成的排在前面。

如何实现一个set，我回答了redis的实现方式，大的用散列，小的用int\_set（忘记说bitmap了， 因为他给了一个小范围的条件）。 他又问我bloom filter有没有听说过， 我跟他把优缺点以及原理全部说了一边。

用户如何找到最近CDN， 这个比较狠，其实是ISA帮你配置的用户根本就不用管，我答得是通过发ping包测试查看ttl， ttl越高越近。

针对项目，问视频播放器存在延迟该如何解决， 我说了TCP缓存减小和跳帧， 他问packet是否能够丢弃，我说packet不能丢， 解码成frame才能跳。

问他问题，我问了项目需求，入职培训这些网上对他们比较负面的评价的地方。 同时问能不能不参加团建，我不太喜欢社交。

java 内存模型

运行过程中声明一个类是存在哪里

java oom 排查

kafka为什么要存在分区这个概念

redis的数据类型：String，hash，List，set，zset有序集合

redis持久化 单线程如何持久化

Prometheus收集监控数据原理

海量url统计访问排名前1000的url

sso单点登录+springcloud登录验证

从数字里剔除k位保证结果数字最小 算法实现

etcd zk 为什么不用kv形式存储数据

1.二叉树镜像

2.大概是题目给出了一个例子，然后让根据这个例子，在输入的数组中找出第k大的素数，输入数组可能包含不是素数的数字

3.求无向无环 图？树的直径

4.求二叉树的左视图看到的节点

5.二叉树求和，找出数字之和等于21的数组下边，要求复杂度要小于o(logn)

？？给定一棵二叉树，和一个数值。求二叉树的路径和等于给定值的所有路径，是这个意思？必须包含根节点？

6.随机给两个单链表的数，如1->2->3->4，2->3->4，求和的代码实现??每个链表表示一个数，即算出4321+432=4753，输出3->5->7->4吗

7.一个二叉排序树，自己实现其一个Iterator

8.一个顺序递增的数组，内含重复的数，现查找一个给定的数在不在数组，在的话给出最后一个出现的index

<https://www.cnblogs.com/A--G/p/5128440.html>

jvm的双亲委派机制类加载过程：

* 主要有三种类加载器：**启动（Bootstrap）类加载器**：是用本地代码实现的类装入器，它负责将 <Java\_Runtime\_Home>/lib下面的类库加载到内存中（比如rt.jar）。由于引导类加载器涉及到虚拟机本地实现细节，开发者无法直接获取到启动类加载器的引用，所以不允许直接通过引用进行操作。
* **标准扩展（Extension）类加载器**：是由 Sun 的 ExtClassLoader（sun.misc.Launcher$ExtClassLoader）实现的。它负责将< Java\_Runtime\_Home >/lib/ext或者由系统变量 java.ext.dir指定位置中的类库加载到内存中。开发者可以直接使用标准扩展类加载器。
* **系统（System）类加载器**：是由 Sun 的 AppClassLoader（sun.misc.Launcher$AppClassLoader）实现的。它负责将系统类路径（CLASSPATH）中指定的类库加载到内存中。开发者可以直接使用系统类加载器。

除了以上列举的三种类加载器，还有一种比较特殊的类型 — 线程上下文类加载器。

双亲委派机制描述   
某个特定的类加载器在接到加载类的请求时，首先将加载任务委托给父类加载器，**依次递归**，如果父类加载器可以完成类加载任务，就成功返回；只有父类加载器无法完成此加载任务时，才自己去加载。

意义在于防止内存中出现多份相同的字节码。

tcp协议的三次握手四次挥手  
一棵树有多个值，每条路径上的数连起来，如何求出最后的结果  
网络中从客户端到服务端一条消息走过的所有层，网络的几层等等吧  
长连接是如何找到从客户端到服务端的连接的路径的  
网络连接拥堵有什么解决方案

hashmap的实现

java中有几种list，这两种有什么区别（这个区别是要求把实现原理都说出来的） <https://blog.csdn.net/weixin_42468526/article/details/81178698>

* 1.Linkedlist基于链表的动态数组，数据添加删除效率高，只需要改变指针指向即可，但是访问数据的平均效率低，需要对链表进行遍历。

2,。Arraylist：底层是基于动态数组，根据下表随机访问数组元素的效率高，向数组尾部添加元素的效率高；但是，删除数组中的数据以及向数组中间添加数据效率低，因为需要移动数组。例如最坏的情况是删除第一个数组元素，则需要将第2至第n个数组元素各向前移动一位。而之所以称为动态数组，是因为Arraylist在数组元素超过其容量大，Arraylist可以进行扩容（针对JDK1.8  数组扩容后的容量是扩容前的1.5倍），Arraylist源码中最大的数组容量是Integer.MAX\_VALUE-8，对于空出的8位，目前解释是 ：①存储Headerwords；②避免一些机器内存溢出，减少出错几率，所以少分配③最大还是能支持到Integer.MAX\_VALUE（当Integer.MAX\_VALUE-8依旧无法满足需求时）。

原文链接：<https://blog.csdn.net/weixin_42468526/article/details/81178698>