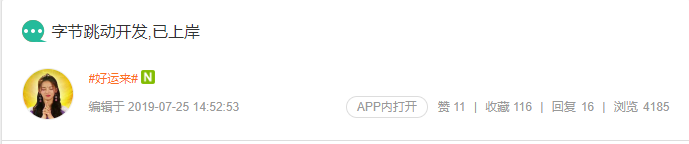
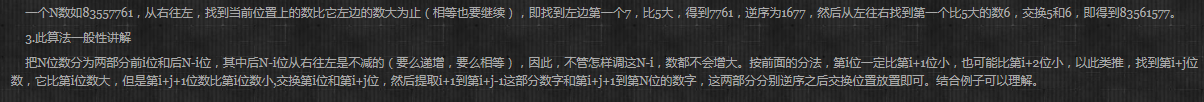
字节跳动面经总结<https://www.nowcoder.com/discuss/231362>

# 一



## 算法题，给你一个数找出下一个大的数

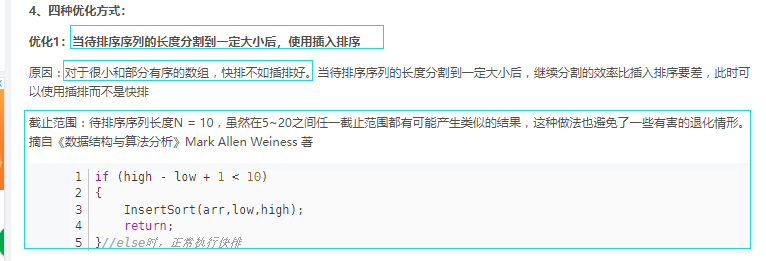
给定一个N位数字组成的数，找出比这个数大的由相同数字组成的下一个数



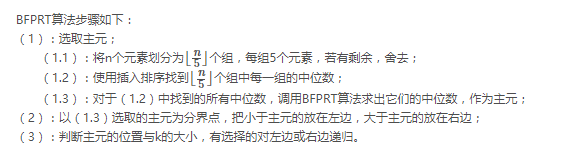
思路很重要

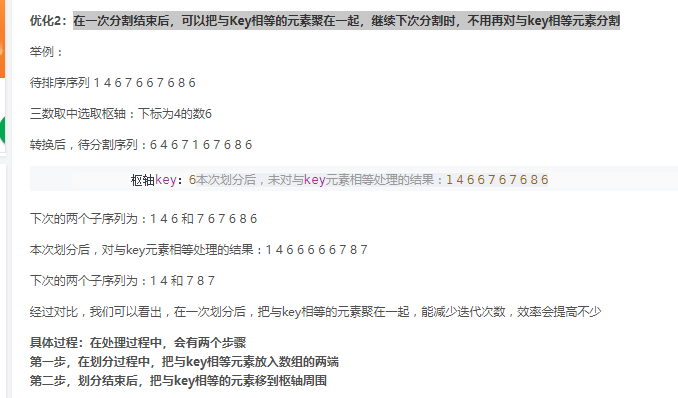
## 快排优化的方式

1. 数组小到一定量的时候用插入排序取代递归
2. 在一次分割结束后，可以把与Key相等的元素聚在一起，继续下次分割时，不用再对与key相等元素分割
3. 快排函数在函数尾部有两次递归操作，我们可以对其使用尾递归优化（是用循环避免栈溢出）
4. **优化4：使用并行或多线程处理子序列**

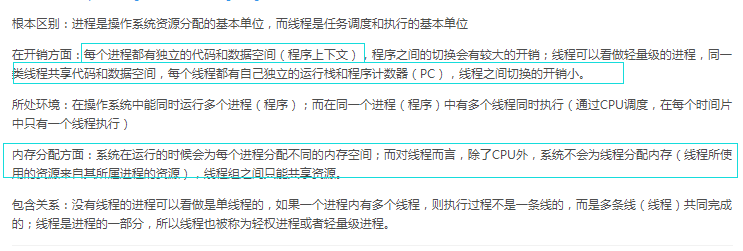


## TopK问题+BFPRT算法（o(n)）





## 进程和线程

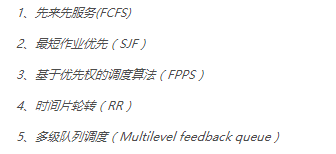


## 线程同步的几种方法

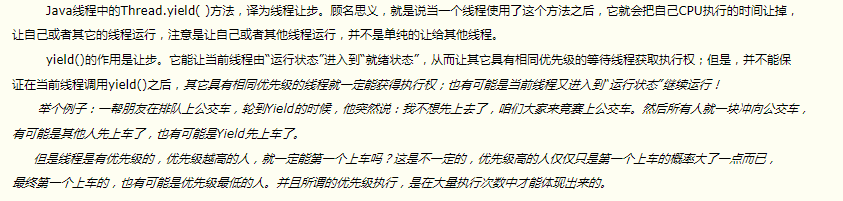


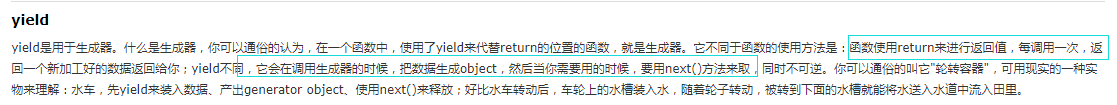
事件对象也可以通过通知操作的方式来保持线程的同步。并且可以实现不同进程中的线程同步操作。

## 线程调度的算法



## yeild方法的语义





## redis的持久化机制,淘汰机制,内存模型,哨兵

### 淘汰机制

设置了时间的选择（最近最少使用，将要过期，任意）

没有设置过期时间（最近最少，任意，禁止）



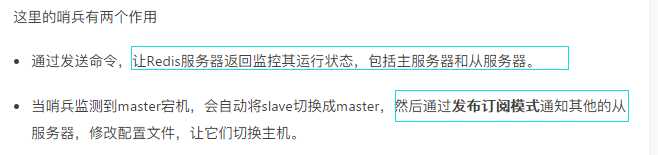
### 内存模型

used\_memory和used\_memory\_rss，前者是从Redis的角度得到的量，后者是从操作系统的角度得到的量。二者之所以有所不同，一方面是因为内存碎片和redis进程运行需要占用内存，使得前者可能比后者小，另一方面虚拟内存的存在，使得前者可能比后者大。

redis内存：数据+进程本身需要内存+缓冲内存+内存碎片

### 哨兵

哨兵通过发送命令，等待Redis服务器响应，从而监控运行的多个Redis实例。



## 数据库关于事务

事务是逻辑上的一组操作，要么都执行，要么都不执行。

（ACID+问题）

## 抓取网页时数据转换异常

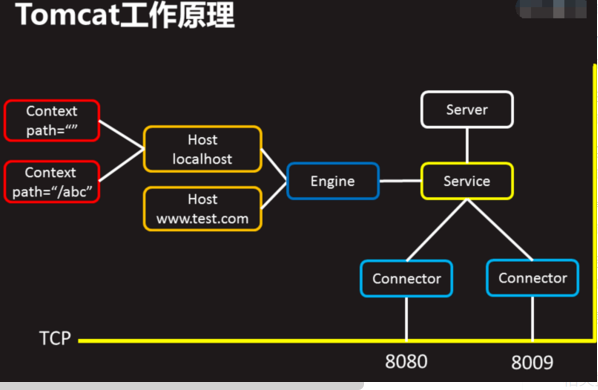
原来它服务器端采用gzip压缩传输的，客户端gunzip一下就行了

## python下载发文件时，要注意什么

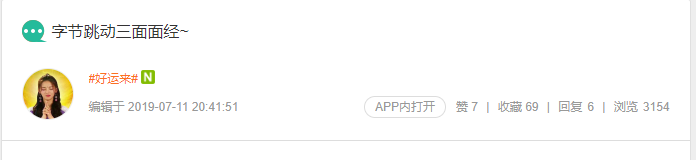
## tomcat的运行原理，一个访问如何返回

url地址被8080端口的Connector获得，Connector把该请求交给它所在的Service的Engine处理，并等待来自Engine的回应。Engine开始匹配，匹配路径。

对于servlet请求，会构造HttpServletRequest对象和HttpServletResponse对象，作为参数调用JspServlet的doGet或doPost方法，Context把执行完了之后的HttpServletResponse对象返回给Host；Host把HttpServletResponse对象返回给Engine，Engine把HttpServletResponse对象返回给Connector，Connector把HttpServletResponse对象返回给客户browser。



# 二



动态规划算法是通过拆分问题，定义问题状态和状态之间的关系，使得问题能够以递推（或者说分治）的方式去解决。

动态规划算法的基本思想与分治法类似，也是将待求解的问题分解为若干个子问题（阶段），按顺序求解子阶段，前一子问题的解，为后一子问题的求解提供了有用的信息。在求解任一子问题时，列出各种可能的局部解，通过决策保留那些有可能达到最优的局部解，丢弃其他局部解。依次解决各子问题，最后一个子问题就是初始问题的解。

## 进程通信的方式

管道，消息队列,信号量（锁），共享内存，信号（信号是一种比较复杂的通信方式，用于通知接收进程某个事件已经发生），共享内存，套接字（套解口也是一种进程间通信机制，与其他通信机制不同的是，它可用于不同机器间的进程通信。）

## Spring注解配置和xml之间的比较

注释配置相对于 XML 配置具有很多的优势：

它可以充分利用 Java 的反射机制获取类结构信息，这些信息可以有效减少配置的工作。

XML的优点：

1. XML配置方式进一步降低了耦合，使得应用更加容易扩展，即使对配置文件进一步修改也不需要工程进行修改和重新编译。

2. 在处理大的业务量的时候，用XML配置应该更加好一些。因为XML更加清晰的表明了各个对象之间的关系，各个业务类之间的调用。同时spring的相关配置也能一目了然。

3、利用 xml 配置能使软件更具扩展性。例如 Spring 将 class 间的依赖配置在 xml 中，最大限度地提升应用的可扩展性。

4.具有成熟的验证机制确保程序正确性。利用 Schema 或 DTD 可以对 xml 的正确性进行验证，避免了非法的配置导致应用程序出错。

缺点：

需要解析工具或类库的支持。

解析 xml 势必会影响应用程序性能，占用系统资源。

配置文件过多导致管理变得困难。

编译期无法对其配置项的正确性进行验证，或要查错只能在运行期。

查错变得困难。往往配置的一个手误导致莫名其妙的错误。

开发人员不得不同时维护代码和配置文件，开发效率变得低下。

Annotation的优点：

1. 在class文件中，可以降低维护成本，annotation的配置非常简单。

2. 不需要第三方的解析工具，利用java反射技术就可以轻松获取been。

3. 编辑器会实时校验错误，如果注解错误会有提醒。

4. 不用维护xml配置文件，提高了开发效率。

缺点：

如果需要对于annotation进行修改，那么要重新编译整个工程。

在程序中Annotation比较多，直接影响代码质量，对于代码的简洁度有一定的影响。

## 泛型中的伪泛型

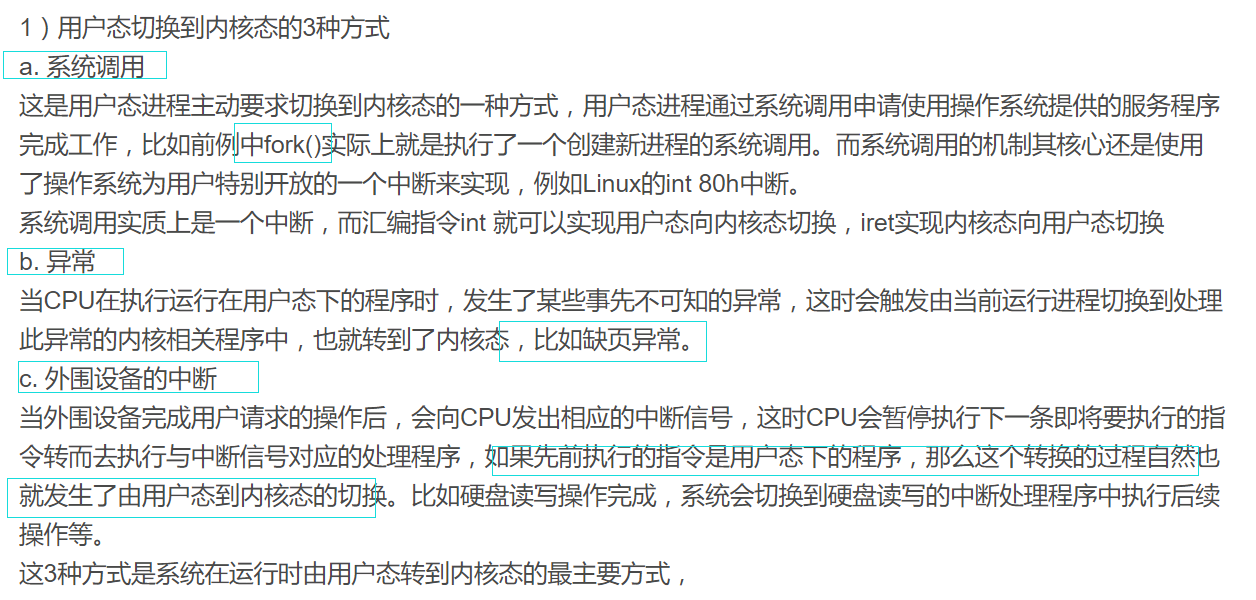


## 多个进程如何读取数据

## 内存分页，内存页满问题，每次读取的空间

## 内核态用户态相互转换那一套

用户态切换到内核态的三种方式：系统调用，异常，外围设备的中断



## 实现开启指定的线程

用rentrantlock的condition来做

## 平面上三个点组成三角形，判断另一个点是否在三角形中

利用面积法，如上图所示，如果点P在三角形ABC的内部，则三个小三角形PAB, PBC, PAC的面积之和 = ABC的面积，反之则不相等。

## 算法打赏问题

## 递归路径总数

dfs

## 场景题如何在一个地图中查到某块地图的玩家

dfs

# 三

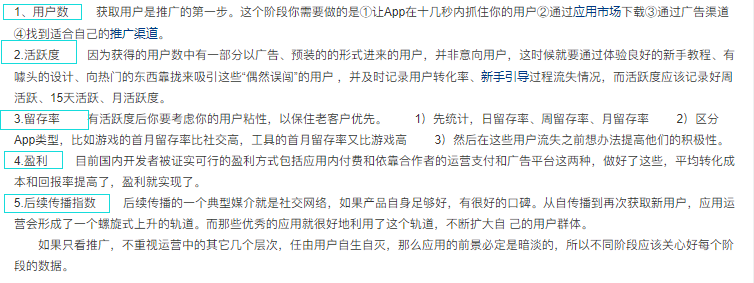


## 对于一个app说一个你最关注的指标 为什么是这个指标而不是其他的

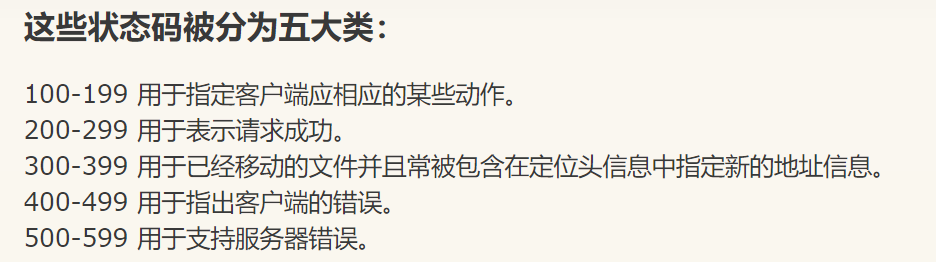
1、保留率

2、流失率

3、LTV（用户终身价值）



## http状态码



200，300，301，302，401，403，404，

**400 - Bad Request 错误请求；**

**401 - Unauthorized 未授权；**

**402 - 需要付款**

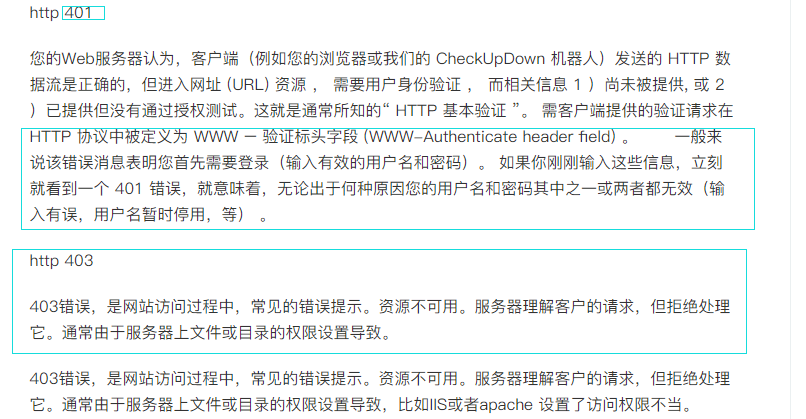
**403 - Forbidden 禁止访问；**

**501 - Not Implemented 没有实现；**

**502 - Bad Gateway 错误的网关；**

**503 - Service Unavailable 服务不可用；**

**504 - Gateway Timeout 网关超时；**



## \*\*\*\*\*\*\*\*spring生命周期



第一步就是对实例化bean，调用构造函数来创建实例，第二步是根据配置，进行相应属性的设置，依赖注入就是在这一步完成的。

第三步和第四步是让spring去了解咱们的spring容器，第五步和第八步可以针对指定的Bean进行功能增强，这时一般是采用的动态代理，（两种动态代理方式：jdk动态代理和cglib动态代理）。第六步和第十步是通过实现指定的接口来完成init（初始化）和destory（销毁）操作。但是我们在通常情况下不会使用这两步，因为我们可以通过第七步和第十一步，在配置文件中设置相应的初始化和销毁方法。

## 输入url到spring后端整个流程是怎么样的

1. 去域名服务器找ip地址
2. 找到ip地址后建立tcp连接
3. 建好连接发送http请求
4. 服务器接受请求后，根据路径参数，经过后端的一些处理生成html返回给浏览器
5. 浏览器根据拿到的资源进行渲染

## SpringBoot和Spring的区别



今日头条点赞功能  
再做个数据库的题目吧，设计一个表实现今日头条点赞功能。我不得不说掉陷阱了，以后是个数据库的题目，加外健什么的。他也没说什么，也没提示我，最后一起说你这样设计，完全不好，然后我才反应过来他的意思，我说用redis做缓存，再用MQ做削峰，异步，他说你怎么不早说呢？我？what？

**Mysql设计**

这一块和写入写mysql是一样的，毕竟是要落地存储的。

所以还是同样的需要post\_like, user\_like\_post这两表存储文章被点赞的个数(等统计), 用户对那些文章点了赞(取消赞)。

这两表分别通过post\_id, user\_id进行关联。

**redis设计部分:**

post\_set

在redis中弄一个set存放所有被点赞的文章

post\_user\_like\_set\_{$post\_id}

对每个post以post\_id作为key, 搞一个set存放所有对该post点赞的用户;

post\_user\_like\_{$post\_id}\_{$user\_id}

将每个用户对每个post的点赞情况放到一个hash里面去， hash的字段就

随意跟进需求来处理就行了。

**为啥用hash**

只所以用hash是因为完全可以用hash来存储一个点赞的对象， 对应数据库的一行记录。

当然有同学会说用key, value也可以， 将所有的数据序列化(json\_encode等)

后全部放到value里面去。 反复序列化也是一个很大的开销不是， hash可以很

方便的修改某个字段， 而序列化和反序列化的操作。

post\_{$post\_id}\_counter

对每个post维护一个计数器， 用来记录当前在redis中的点赞数，

这里我们只用counter记录尚未同步到mysql中的点赞数(可以为负)， 每次

刷回mysql中时将counter中的数据和数据库已有的赞数相加即可。

**用户点赞/取消赞**

获取user\_id, post\_id， 查询该用户是否已经点过赞， 已点过则不允许再次点赞，

或者设计为前端允许用户点， 只是后台不重复计算;

这里需要注意的是用户点赞的记录可能在数据库中， 也可能在缓存中， 所以查询的时候

缓存和数据库都要查询， 缓存没有再查询数据库。

将用户的点赞/取消赞的情况记录在redis中， 具体为:

1、写入post\_set

将post\_id写入post\_set

2、写入post\_user\_like\_set\_{$post\_id}

将user\_id写入post\_user\_like\_set\_{$post\_id}

3、写入post\_user\_like\_{$post\_id}\_{$user\_id}

将用户点赞数据， 例如赞状态, post\_id, user\_id, ctime(操作时间), mtime(修改时间)写入post\_user\_like\_{$post\_id}\_{$user\_id}中

4、更新post\_{$post\_id}\_counter

更新post\_{$post\_id}\_counter, 这里的更新稍晚复杂一点， 需要和前面一样先获取当前用户是否对这个post点过赞

如果点过， 并且本次是取消赞， counter减一， 如果没点过， 本次是点赞， counter加一。

如果原来是取消赞的情况， 本次是点赞， counter加一。

**同步刷回数据库**

循环从post\_set中pop出来一个post\_id至到空

    根据{$post\_id} , 每次从post\_user\_like\_set\_{$post\_id}中pop出来一个user\_id直到空

        根据post\_id, user\_id, 直接获取对应的hash表的内容(post\_user\_like\_{$post\_id}\_{$user\_id}

        将hash表中的数据写入user\_like\_post表中

        将post\_{$post\_id}\_counter中的数据和post\_like中的数据相加， 将结果写入到post\_like表中

**页面展示**

1、查询用户点赞情况

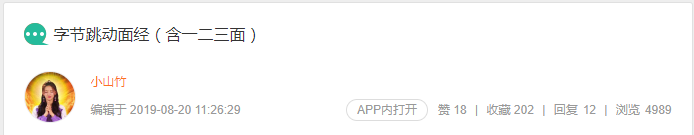
前面已经说过， 需要同时查询redis和mysql

2、查询post点赞统计

同样需要查询redis中的post\_{$post\_id}\_counter和mysql的post\_like表, 并将两者相加

得到的结果才是正确的结果

# 四



## http的头部字段

请求方式

htpp头部可以包含二进制不

TCP拥塞控制

http与https的区别

https的安全外壳是怎么实现的

怎么查看linux下那个进程打开了那些文件

epoll的底层实现

linux的阻塞和非阻塞是怎么实现的

多个有序数组的合并求时间复杂度

## 逻辑推理

地球是一个二维圆，12点位置是一个飞机场，飞机必须从飞机场起飞，必须从飞机场降落，飞机加满油可以绕地球半圈，飞机可以相互加油，最少需要出动多少次飞机，使得有一架飞机可以绕地球一圈

项目文件传输的时候是怎么限速的

# 五

## 解释XSS 和 CSRF，它们之间有什么区别，以及如何防范。

### XSS

Cross Site Script,跨站脚本攻击。是指攻击者在网站上注入恶意客户端代码，通过恶意脚本对客户端网页进行篡改，从而在用户浏览网页时，对用户浏览器进行控制或者获取用户隐私数据的一种攻击方式。

XSS的避免方式

CSP本质上是建立白名单，规定了浏览器只能执行特定来源的代码。

HttpOnly阻止Cookie劫持攻击（通过JavaScript的 Document.cookie API无法访问带有 HttpOnly 标记的Cookie）

### csrf



## 场景题

假设你在宿舍玩游戏，突然发现特别卡，原来你舍友在用迅雷下载东西，为什么迅雷会占用网络带宽导致你玩游戏特别卡，而游戏竞争不过迅雷。

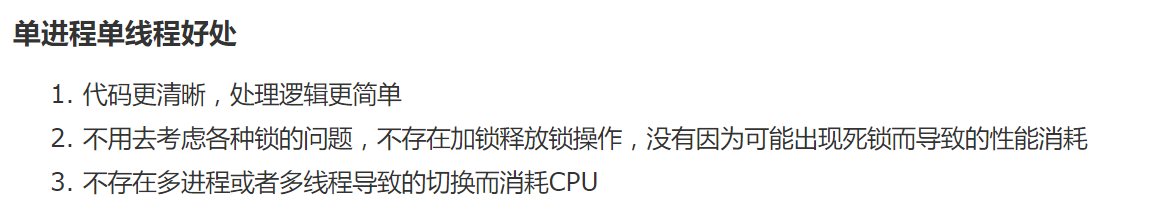
迅雷对带宽占用很多而已，游戏占用不了那么多，其实迅雷下载东西时一样很卡，只不过下载东西可以缓存，游戏缓存的时候我们叫它“卡”

## 大数据量处理题

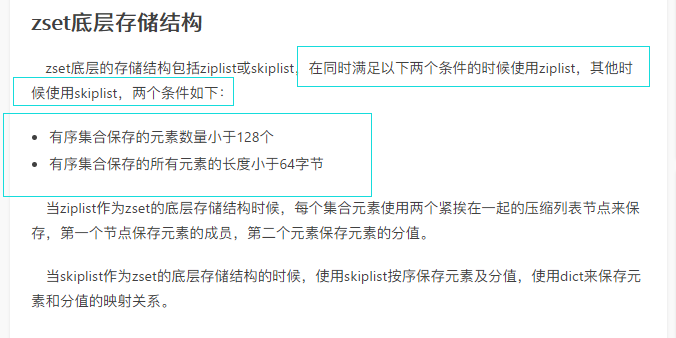
假如你在网上爬取了用户访问的网址存储在文件中，有4个G那么大，但是你目前只有一台电脑，而且内存是1个G的，请问怎么找出用户访问次数最多的网址。（差不多是这个意思）

分治+堆排序

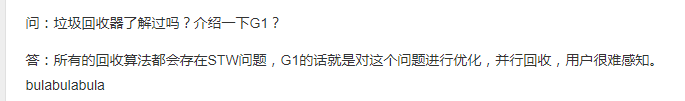
## redis中的网络IO有了解过吗，它是单线程的还是多线程的，为什么要用单线程



## zset的底层是用什么数据结构实现的



# 综合

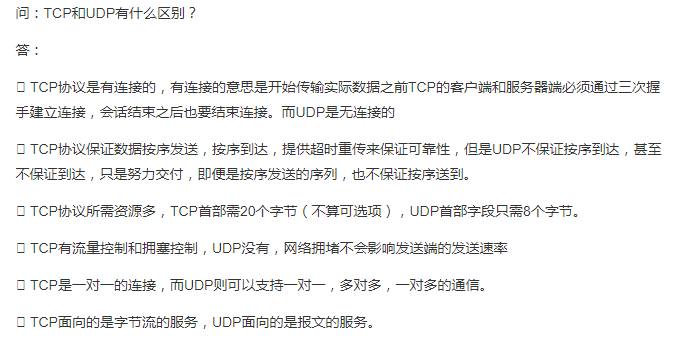


## java导包的过程是怎么样的

导入的是.class字节码文件  
java是一种解释型的语言  
.class文件是解释运行在JVM上的

## java的编译是怎么一个过程呢？

java编译器的话经过四个步骤，词义分析，语义分析，语法分析和代码生成



## AQS

AQS则实现了对**同步状态的管理，以及对阻塞线程进行排队，等待通知**等等一些底层的实现处理。

## 本地方法栈和虚拟机栈的区别



Socket编程底层如何实现

## Redis服务端有20GB内存，现在要缓存200GB数据

