面试时间：2023/3/9

地点：南京华泰证券总部楼

（这个楼很大，电梯很高级……）

开场问：自我介绍、哪个学校的、哪里人、未来想去哪个城市工作

然后问了是保研的还是考研的，本科时候的成绩

**Q1：二选一回答：在学习编程过程中印象最深的书/一次印象最深刻的debug经历**

我选了第二个问题，回答的是毕设中做游戏时z轴远近移动的问题，但其实两个都回答应该也可以？还能多磨点时间(x

**Q2：给一个未排序的数组，不能排序，如何找出第二大的数**

设置两个变量int max和int second，一开始max和second为前两个数，遍历数组找出最大的两个数即可。

**Q3：给两个已排序的长度分别为m、n的数组nums1、nums2，不能将这两个数组合并，如何找出两个数组的中位数**

两个数组总长度为m+n，由于可能为奇数也可能为偶数，所以统一处理为求下标为（m+n)/2和(m+n-1)/2的两个数的平均数。（java整数除法是向下取整，如果m+n是奇数的话这两条式子算出来是同一个下标）

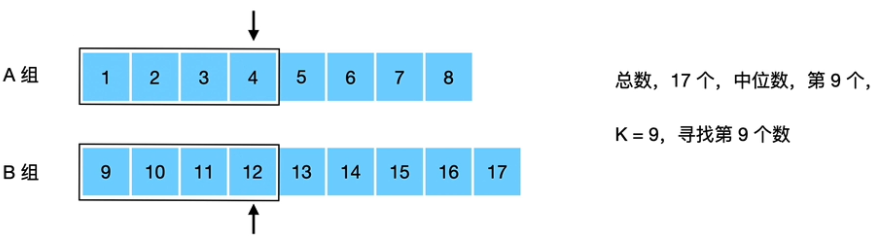
当m+n为奇数时，转化为找第(m+n+1)/2小数的问题；为偶数时，转化为找第(m+n)/2和第(m+n+1)/2小数的问题。（第几小数是从1开始的，即升序数组中下标为0的数是第1小数）

* 找**第k小数**的算法：

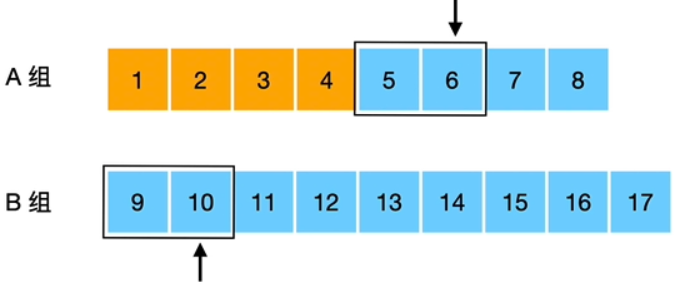
折半查找，每次删除k/2个元素，下一次删除剩余的k/2个

* 例题：A和B数组一共17个数，要找到第9小数（即总下标为8的数）

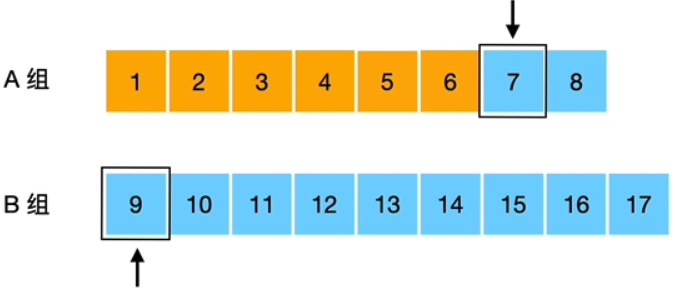
第一轮：k/2=9/2=4，两个数组都比较第四个数（即下标为3的数），A为4，B为12，A的更小，所以把A的前四个数都删除。



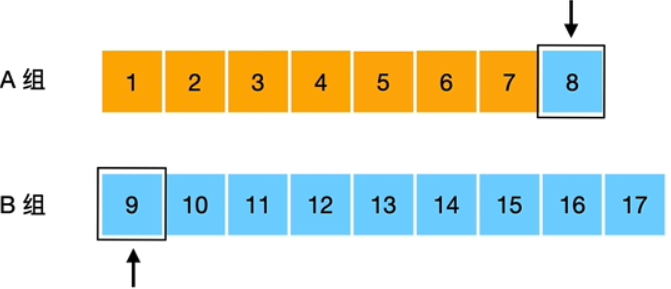
第二轮：删掉4个数后，k=5，k/2=5/2=2，两个数组的剩余部分比较第二个数，A为6，B为10，A小，所以把A的这两个数也删了。



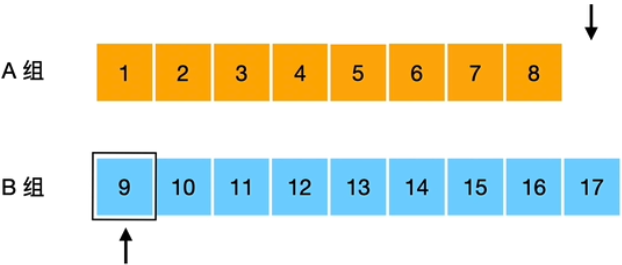
第三轮：k=3，k/2=3/2=1，两个数组的剩余部分比较第一个数，还是A小，把A的这个元素删了



第四轮：k=2，k/2=2/2=1，还是两个数组的剩余部分比较第一个数，还是删A的这个元素



第五轮：k=1，这时候数组A的元素已经删完了，所以直接返回B中的第一小数即可



因为要考虑到偶数个的情况，所以一律转为求**第(m+n+1)/2小数**和**第(m+n+2)/2小数**，再求平均

如果有两个数相同的话，那**任意删一个**

如果k/2大于某个数组的长度，则**取这个数组的最后一个数**，如果这个数比另一个数组的对应数小的话就把这个数组全部删完，下一轮就找另一个数组剩余部分的第k小数即可

当k=1时且两个数组均有剩余元素时，返回两个数组第一个元素的**小者**

* 时间复杂度：O(log (m+n))
* 代码见findTwoArrayMedian

**Q4：排序算法的稳定/不稳定是什么意思**

没答上来，但其实很简单。

排序算法稳定性：意思就是说大小相同的两个值在排序之前和排序之后的先后顺序不变，这就是稳定的。

稳定的：冒泡排序、插入排序、归并排序、基数排序

不稳定的：选择排序、快速排序、shell排序、堆排序

参考博客：https://blog.csdn.net/xjm1018/article/details/88547231

**Q5：synchronzied和volatile的区别、**

背景知识：java中，整个程序有一个主内存，每个线程也有自己的工作内存，线程在自己的工作内存里对共享变量进行修改，然后将修改后的值更新到主内存中。

* 线程对共享变量的操作遵循以下两条规则：

1. 线程对共享变量的所有操作都必须在自己的工作内存中进行，不能直接从主内存中读写。
2. 不同线程之间无法直接访问其他线程工作内存中的变量，线程间变量值的传递需要通过主内存来完成。

* 要实现共享变量的可见性，必须保证两点：

1.线程修改后的共享变量值能够及时从工作内存刷新到主内存中

2.其他线程能够及时把共享变量的最新值从主内存更新到自己的工作内存中

synchronized和volatile就是实现共享变量可见性的两种方式。

* synchronized

**符合线程安全的三个要素：**

* **原子性（同步）**：做某个操作的时候一旦开始就会运行到结束，中间不会被其他进程或线程打断。

像i++这种操作分三步：从主内存读取i的值、在工作内存将i值+1、将新值赋予主内存 的i，如果要将其变为原子操作就要加上synchronnized关键字。

同一时间只能有一个线程去执行synchronized 中的代码块。

* **可见性**：指一个线程对共享变量的修改，另外一个线程能够立刻看到。

sychronized底层是通过monitorenter指令来进行加锁（具有load屏障作用）、通过monitorexit指令来释放锁（具有store屏障作用）。

* **有序性：**指程序在执行的时候，程序的代码执行顺序和语句的顺序是一致的，当前面的语句执行完后才能执行后一条。原子性和有序性都是通过一段时间只能一个线程访问来实现。

并发条件下的重排序很容易破坏有序性，synchronized的monitorenter、monitorexit指令嵌入了volatile的那四个禁止重排序的内存屏障指令，能够禁止重排序。

关于synchronized的两条规定：

* 线程释放锁前，必须把共享变量的最新值刷新到主内存中
* 线程获得锁时，将清空工作内存中共享变量的值，从而使用共享变量时，需要从主内存中重新读取最新的值(注意：加锁与解锁需要是同一把锁)

**线程执行互斥代码的过程：**

1. 获得互斥锁（monitorenter指令）
2. 清空工作内存
3. 从主内存拷贝变量的最新副本到工作的内存
4. 执行代码
5. 将更改后的共享变量的值刷新到主内存
6. 释放互斥锁（monitorexit指令）

**使用场景：**

* 可以修饰代码块、方法、类、静态方法

× 不能修饰变量

* volatile

× 原子性

* 可见性：通过内存屏障保证可见性。对volatile变量执行读操作时，会在读操作前加入一条load屏障指令，保证volatile变量每次读取数据的时候都强制从主内存读取；对volatile变量执行写操作时，会在写操作后加入一条store屏障指令，保证每次volatile修改之后强制将数据刷新会主内存。
* 有序性：对volatile变量进行操作时使用的内存屏障指令（LoadLoad屏障、StoreStore屏障、LoadStore屏障、StoreLoad屏障）能够禁止重排序。

线程写volatile变量的过程：

1. 改变线程工作内存中volatile变量副本的值
2. 将改变后的副本的值从工作内存刷新到主内存

线程读volatile变量的过程：

1. 从主内存中读取volatile变量的最新值到线程的工作内存中
2. 从工作内存中读取volatile变量的副本

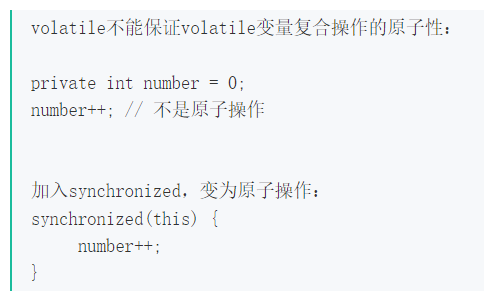
使用场景：

* 修饰变量，所有线程使用这个变量的时候都有用

× 不能修饰方法

总结：

1. synchronized加了互斥锁，会阻塞其他线程；volatile没有加互斥锁
2. synchronized能保证原子性、可见性、有序性；volatile只能保证可见性和有序性。
3. synchronized用来修饰方法或代码块，volatile用来修饰变量。



参考博客：https://blog.csdn.net/chenzengnian123/article/details/122686371

**Q6：java中如何排查和定位OutOfMemory异常（简称OOM）**

可以通过在发生OOM时生成dump文件分析，dump是jvm运行时内存的一份快照，利用它可以分析是否存在内存浪费，可以检查内存管理是否合理，当发生OOM的时候，可以找出问题的原因。生成dump文件有以下几种方法：

1. 运行时手动输命令生成

获取这个运行时的java进程的pid，通过以下jmap命令来导出dump文件：

jmap -dump:file=javaDump.dump,format=b 25535

以上命令针对pid为25535的java进程，获取到一个名为javaDump.dump的文件。

1. 设置程序发生OOM时自动生成

设置jvm启动参数：

-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=./

以上命令设置在发生OOM时，将dump文件输出到./（根目录）下。

如何分析dump文件：

1. 使用jdk自带工具jhat

通过jhat xx.dump/hprpf命令能够分析xx.dump文件，然后本地访问http://localhost:7000，即可在浏览器看到访问效果。



1. 使用jdk自带工具jvisualvm

找到$JAVA\_HOME/bin/jvisualvm.exe，双击即可启动，启动后点击左上角文件->装入，装入想要分析的dump文件即可



**Q7：你最熟悉的技术栈是什么**

我说我最近做的比较多的是小程序，面试官说不行啊小程序都屏蔽了很多底层细节，估计觉得太low了，无语。。。

最后面试官问我是不是今年第一次面试，我说是的，他说看出来了，因为问的东西你肯定学过但一看就是忘了，你下次好好准备吧！