# Android面试整理

## 阿里巴巴

* LRUCache原理 答：[LRUCache原理](http://blog.csdn.net/qq_25806863/article/details/77548468)
* 图片加载原理
* 模块化实现（好处，原因）
* JVM
* 视频加密传输
* 统计启动时长,标准  
  答：[统计启动时长](http://blog.csdn.net/beyond_liyy/article/details/52273740)
* 如何保持应用的稳定性
* ThreadLocal 原理 答：[ThreadLocal原理](http://blog.csdn.net/imzoer/article/details/8262101)
* 谈谈classloader  答：[ClassLoader加载过程](http://blog.csdn.net/xyang81/article/details/7292380)  答：[ClassLoader加载过程](http://blog.csdn.net/javazejian/article/details/73413292)
* 动态布局
* 热修复,插件化  答：[Android插件化：从入门到放弃](http://www.infoq.com/cn/articles/android-plug-ins-from-entry-to-give-up)
* HashMap源码,SpareArray原理
* 性能优化,怎么保证应用启动不卡顿
* 怎么去除重复代码 答：[去除重复代码](http://blog.csdn.net/wangzhongshun/article/details/78738217)
* SP是进程同步的吗?有什么方法做到同步
* 介绍下SurfView
* HashMap实现原理，ConcurrentHashMap 的实现原理
* BroadcastReceiver，LocalBroadcastReceiver 区别
* Bundle 机制
* Handler 机制
* android 事件传递机制
* 线程间 操作 List
* App启动流程，从点击桌面开始
* 动态加载
* 类加载器
* OSGI
* Https请求慢的解决办法，DNS，携带数据，直接访问IP
* GC回收策略
* 画出 Android 的大体架构图
* 描述清点击 Android Studio 的 build 按钮后发生了什么
* 大体说清一个应用程序安装到手机上时发生了什么；
* 对 Dalvik、ART 虚拟机有基本的了解； 答：[Dalvik、ART 虚拟机](https://www.jianshu.com/p/58f817d176b7)
* Android 上的 Inter-Process-Communication 跨进程通信时如何工作的；
* App 是如何沙箱化，为什么要这么做；
* 权限管理系统（底层的权限是如何进行 grant 的）
* 进程和 Application 的生命周期；
* 系统启动流程 Zygote进程 –> SystemServer进程 –> 各种系统服务 –> 应用进程
* recycleview listview 的区别,性能
* 排序，快速排序的实现
* 树：B+树的介绍
* 图：有向无环图的解释
* TCP/UDP的区别 答：[TCP/UDP的区别](http://blog.csdn.net/li_ning_/article/details/52117463)
* synchronized与Lock的区别
* volatile
* Java线程池
* Java中对象的生命周期
* 类加载机制
* 双亲委派模型
* Android事件分发机制
* MVP模式
* RxJava
* 抽象类和接口的区别
* 集合 Set实现 Hash 怎么防止碰撞
* JVM 内存区域 开线程影响哪块内存
* 垃圾收集机制 对象创建，新生代与老年代
* 二叉树 深度遍历与广度遍历
* B树、B+树
* 消息机制
* 进程调度
* 进程与线程
* 死锁
* 进程状态
* JVM内存模型
* 并发集合了解哪些
* ConCurrentHashMap实现
* CAS介绍
* 开启线程的三种方式,run()和start()方法区别 答：[开启线程的三种方式](http://blog.csdn.net/longshengguoji/article/details/41126119)
* 线程池
* 常用数据结构简介
* 判断环（猜测应该是链表环）
* 排序，堆排序实现
* 链表反转 ## 腾讯
* synchronized用法
* volatile用法
* 动态权限适配方案，权限组的概念
* 网络请求缓存处理，okhttp如何处理网络缓存的
* 图片加载库相关，bitmap如何处理大图，如一张30M的大图，如何预防OOM
* 进程保活
* listview图片加载错乱的原理和解决方案
* https相关，如何验证证书的合法性，https中哪里用了对称加密，哪里用了非对称加密，对加密算法（如RSA）等是否有了解 ## 滴滴
* MVP
* 广播（动态注册和静态注册区别，有序广播和标准广播）
* service生命周期
* handler实现机制（很多细节需要关注：如线程如何建立和退出消息循环等等）
* 多线程（关于AsyncTask缺陷引发的思考）
* 数据库数据迁移问题
* 设计模式相关（例如Android中哪里使用了观察者模式，单例模式相关）
* x个苹果，一天只能吃一个、两个、或者三个，问多少天可以吃完
* TCP与UDP区别与应用（三次握手和四次挥手）涉及到部分细节（如client如何确定自己发送的消息被server收到） HTTP相关 提到过Websocket 问了WebSocket相关以及与socket的区别
* 是否熟悉Android jni开发，jni如何调用java层代码
* 进程间通信的方式
* java注解
* 计算一个view的嵌套层级
* 项目组件化的理解
* 多线程断点续传原理
* Android系统为什么会设计ContentProvider，进程共享和线程安全问题
* jvm相关
* Android相关优化（如内存优化、网络优化、布局优化、电量优化、业务优化）
* EventBus实现原理

## 美团

* static synchronized 方法的多线程访问和作用，同一个类里面两个synchronized方法，两个线程同时访问的问题
* 内部类和静态内部类和匿名内部类，以及项目中的应用
* handler发消息给子线程，looper怎么启动
* View事件传递
* activity栈
* 封装view的时候怎么知道view的大小
* arraylist和linkedlist的区别，以及应用场景
* 怎么启动service，service和activity怎么进行数据交互
* 下拉状态栏是不是影响activity的生命周期，如果在onStop的时候做了网络请求，onResume的时候怎么恢复
* view渲染 ## 今日头条
* 数据结构中堆的概念，堆排序
* 死锁的概念，怎么避免死锁
* ReentrantLock 、synchronized和volatile（n面）
* HashMap
* singleTask启动模式
* 用到的一些开源框架，介绍一个看过源码的，内部实现过程。
* 消息机制实现
* ReentrantLock的内部实现
* App启动崩溃异常捕捉
* 事件传递机制的介绍
* ListView的优化
* 二叉树，给出根节点和目标节点，找出从根节点到目标节点的路径
* 模式MVP，MVC介绍
* 断点续传的实现
* 集合的接口和具体实现类，介绍
* TreeMap具体实现
* synchronized与ReentrantLock
* 手写生产者/消费者模式
* 逻辑地址与物理地址，为什么使用逻辑地址
* 一个无序，不重复数组，输出N个元素，使得N个元素的和相加为M，给出时间复杂度、空间复杂度。手写算法
* .Android进程分类
* 前台切换到后台，然后再回到前台，Activity生命周期回调方法。弹出Dialog，生命值周期回调方法。
* Activity的启动模式 ## 爱奇艺
* RxJava的功能与原理实现
* RecycleView的使用，原理，RecycleView优化
* ANR的原因
* 四大组件
* Service的开启方式
* Activity与Service通信的方式
* Activity之间的通信方式
* HashMap的实现，与HashSet的区别
* JVM内存模型，内存区域
* Java中同步使用的关键字，死锁
* MVP模式
* Java设计模式，观察者模式
* Activity与Fragment之间生命周期比较
* 广播的使用场景 ## 百度
* Bitmap 使用时候注意什么？
* Oom 是否可以try catch ？
* 内存泄露如何产生？
* 适配器模式，装饰者模式，外观模式的异同？
* ANR 如何产生？
* String buffer 与string builder 的区别？
* 如何保证线程安全？
* java四中引用
* Jni 用过么？
* 多进程场景遇见过么？
* 关于handler，在任何地方new handler 都是什么线程下
* sqlite升级，增加字段的语句
* bitmap recycler 相关
* 强引用置为null，会不会被回收？
* glide 使用什么缓存？
* Glide 内存缓存如何控制大小？
* 如何保证多线程读写文件的安全？ ## 携程
* Activity启动模式
* 广播的使用方式，场景
* App中唤醒其他进程的实现方式
* AndroidManifest的作用与理解
* List,Set,Map的区别
* HashSet与HashMap怎么判断集合元素重复
* Java中内存区域与垃圾回收机制
* EventBus作用，实现方式，代替EventBus的方式
* Android中开启摄像头的主要步骤 ## 网易
* 集合
* concurrenthashmap
* volatile
* synchronized与Lock
* Java线程池
* wait/notify
* NIO
* 垃圾收集器
* Activity生命周期
* AlertDialog,popupWindow,Activity区别 ## 小米
* String 为什么要设计成不可变的？
* fragment 各种情况下的生命周期
* Activity 上有 Dialog 的时候按 home 键时的生命周期
* 横竖屏切换的时候，Activity 各种情况下的生命周期
* Application 和 Activity 的 context 对象的区别
* 序列化的作用，以及 Android 两种序列化的区别。
* List 和 Map 的实现方式以及存储方式。
* 静态内部类的设计意图。
* 线程如何关闭，以及如何防止线程的内存泄漏 ## 360
* 软引用、弱引用区别
* 垃圾回收
* 多线程：怎么用、有什么问题要注意；Android线程有没有上限，然后提到线程池的上限
* JVM
* 锁
* OOM，内存泄漏
* ANR怎么分析解决
* LinearLayout、RelativeLayout、FrameLayout的特性、使用场景
* 如何实现Fragment的滑动
* ViewPager使用细节，如何设置成每次只初始化当前的Fragment，其他的不初始化
* ListView重用的是什么
* 进程间通信的机制
* AIDL机制
* AsyncTask机制
* 如何取消AsyncTask
* 序列化
* Android为什么引入Parcelable
* 有没有尝试简化Parcelable的使用
* AIDL机制
* 项目：拉活怎么做的
* 应用安装过程 ## 某海外直播公司
* 线程和进程的区别？
* 为什么要有线程，而不是仅仅用进程？
* 算法判断单链表成环与否？
* 如何实现线程同步？
* hashmap数据结构？
* arraylist 与 linkedlist 异同？
* object类的equal 和hashcode 方法重写，为什么？
* hashmap如何put数据（从hashmap源码角度讲解）？
* 简述IPC？
* fragment之间传递数据的方式？
* 简述tcp四次挥手?
* threadlocal原理
* 内存泄漏的可能原因？
* 用IDE如何分析内存泄漏？
* OOM的可能原因？
* 线程死锁的4个条件？
* 差值器&估值器
* 简述消息机制相关
* 进程间通信方式？
* Binder相关？
* 触摸事件的分发？
* 简述Activity启动全部过程？
* okhttp源码？
* RxJava简介及其源码解读？
* 性能优化如何分析systrace？
* 广播的分类？
* 点击事件被拦截，但是相传到下面的view，如何操作？
* Glide源码？
* ActicityThread相关？
* volatile的原理
* synchronize的原理
* lock原理
* 翻转一个单项链表
* string to integer
* 合并多个单有序链表（假设都是递增的） ## 其他公司
* 四大组件
* Android中数据存储方式
* 微信主页面的实现方式
* 微信上消息小红点的原理
* 两个不重复的数组集合中，求共同的元素。
* 上一问扩展，海量数据，内存中放不下，怎么求出。
* Java中String的了解。
* ArrayList与LinkedList区别
* 堆排序过程，时间复杂度，空间复杂度
* 快速排序的时间复杂度，空间复杂度
* RxJava的作用，与平时使用的异步操作来比，优势
* Android消息机制原理
* Binder机制介绍
* 为什么不能在子线程更新UI
* JVM内存模型
* Android中进程内存的分配，能不能自己分配定额内存
* 垃圾回收机制与调用System.gc()区别
* Android事件分发机制
* 断点续传的实现
* RxJava的作用，优缺点

## Android基础

1、什么是ANR 如何避免它？

2、View的绘制流程；自定义View如何考虑机型适配；自定义View的事件

3、分发机制；View和ViewGroup分别有哪些事件分发相关的回调方法；自定义View如何提供获取View属性的接口；

4、Art和Dalvik对比；虚拟机原理，如何自己设计一个虚拟机(内存管理，类加载，双亲委派)；JVM内存模型及类加载机制；内存对象的循环引用及避免；

4、ddms 和 traceView；

5、内存回收机制与GC算法(各种算法的优缺点以及应用场景)；GC原理时机以及GC对象；内存泄露场景及解决方法；

6、四大组件及生命周期；ContentProvider的权限管理(读写分离，权限控制-精确到表级，URL控制)；Activity的四种启动模式对比；Activity状态保存于恢复；

7、什么是AIDL 以及如何使用；

8、请解释下在单线程模型中Message、Handler、Message Queue、Looper之间的关系；

9、Fragment生命周期；Fragment状态保存startActivityForResult是哪个类的方法，在什么情况下使用，如果在Adapter中使用应该如何解耦；

10、AsyncTask原理及不足；ntentService原理；

11、Activity 怎么和Service 绑定，怎么在Activity 中启动自己对应的Service；

12、请描述一下Service 的生命周期；

13、AstncTask+HttpClient与AsyncHttpClient有什么区别；

14、如何保证一个后台服务不被杀死；比较省电的方式是什么；

15、如何通过广播拦截和abort一条短信；广播是否可以请求网络；广播引起anr的时间限制；

16、进程间通信，AIDL；

17、事件分发中的onTouch 和onTouchEvent 有什么区别，又该如何使用？

18、说说ContentProvider、ContentResolver、ContentObserver 之间的关系；

19、请介绍下ContentProvider 是如何实现数据共享的；

20、Handler机制及底层实现；

21、Binder机制及底实现；

22、ListView 中图片错位的问题是如何产生的；

23、在manifest 和代码中如何注册和使用BroadcastReceiver；

24、说说Activity、Intent、Service 是什么关系；

25、ApplicationContext和ActivityContext的区别；

26、一张Bitmap所占内存以及内存占用的计算；

27、Serializable 和Parcelable 的区别；

28、请描述一下BroadcastReceiver；

29、请描述一下Android 的事件分发机制；

30、请介绍一下NDK；

31、什么是NDK库，如何在jni中注册native函数，有几种注册方式；

32、AsyncTask 如何使用；

33、对于应用更新这块是如何做的？(灰度，强制更新，分区域更新)；

34、混合开发，RN，weex，H5，小程序(做Android的了解一些前端js等还是很有好处的)；

35、什么情况下会导致内存泄露；

36、如何对Android 应用进行性能分析以及优化；

37、说一款你认为当前比较火的应用并设计(直播APP)；

38、OOM的避免异常及解决方法；

39、屏幕适配的处理技巧都有哪些；

40、两个Activity 之间跳转时必然会执行的是哪几个方法？

40、Okhttp原理

41、Rxjava用法和原理

42，热更新技术有哪些，知道的原理！

43、Activity启动流程

44、Android内存管理

45、Android权限管理

46、将一下7.0的新特性

47、说下你你们项目的架构

48、组件化的有点和具体实施方案

49、内存泄露检测方法

50、Http协议，SSL握手机制。

## Java基础

1、集合类以及集合框架；HashMap与HashTable实现原理，线程安全性，hash冲突及处理算法；ConcurrentHashMap；

2、进程和线程的区别；

3、Java的并发、多线程、线程模型；

4、什么是线程池，如何使用?

答：线程池就是事先将多个线程对象放到一个容器中，当使用的时候就不用new 线程而是直接去池中拿线程即可，节省了开辟子线程的时间，提高的代码执行效率。

5、数据一致性如何保证；Synchronized关键字，类锁，方法锁，重入锁；

6、Java中实现多态的机制是什么；

7、如何将一个Java对象序列化到文件里；

8、说说你对Java反射的理解；

答：Java 中的反射首先是能够获取到Java 中要反射类的字节码， 获取字节码有三种方法， (1).Class.forName(className) (2).类名.class (3).this.getClass()。 然后将字节码中的方法，变量，构造函数等映射成相应的Method、Filed、Constructor 等类，这些类提供了丰富的方法可以被我们所使用。

9、同步的方法；多进程开发以及多进程应用场景；

10、在Java中wait和seelp方法的不同；

答：最大的不同是在等待时wait 会释放锁，而sleep 一直持有锁。wait 通常被用于线程间交互，sleep 通常被用于暂停执行。

11、synchronized 和volatile 关键字的作用；

答：1）保证了不同线程对这个变量进行操作时的可见性，即一个线程修改了某个变量的值，这新值对其他线程来说是立即可见的。 2）禁止进行指令重排序。 12、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。 (1).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 (2).volatile 仅能实现变量的修改可见性，并不能保证原子性；synchronized 则可以保证变量的修改可见性和原子性 (3).volatile 不会造成线程的阻塞；synchronized 可能会造成线程的阻塞。 (4).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

13、服务器只提供数据接收接口，在多线程或多进程条件下，如何保证数据的有序到达；

14、ThreadLocal原理，实现及如何保证Local属性；

15、String StringBuilder StringBuffer对比；

16、你所知道的设计模式有哪些； 答：Java 中一般认为有23 种设计模式，我们不需要所有的都会，但是其中常用的几种设计模式应该去掌握。下面列出了所有的设计模式。需要掌握的设计模式我单独列出来了，当然能掌握的越多越好。 总体来说设计模式分为三大类： 创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。 结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。 行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式。

17、Java如何调用c、c++语言；

18、接口与回调；回调的原理；写一个回调demo；

19、泛型原理，举例说明；解析与分派；

20、抽象类与接口的区别；应用场景；抽象类是否可以没有方法和属性；

21、静态属性和静态方法是否可以被继承？是否可以被重写？以及原因？

22、修改对象A的equals方法的签名，那么使用HashMap存放这个对象实例的时候，会调用哪个equals方法；

23、说说你对泛型的了解；

24、Java的异常体系；

25、如何控制某个方法允许并发访问线程的个数；

26、动态代理的区别，什么场景使用；

27、Dex加载过程和优化方式；

28、Jvm和Gc机制；

29、常用的设计模式。

## 数据结构与算法

1、堆和栈在内存中的区别是什么(数据结构方面以及实际实现方面)；

2、最快的排序算法是哪个？给阿里2万多名员工按年龄排序应该选择哪个算法？堆和树的区别；写出快排代码；链表逆序代码；

3、求1000以内的水仙花数以及40亿以内的水仙花数；

4、子串包含问题(KMP 算法)写代码实现；

5、万亿级别的两个URL文件A和B，如何求出A和B的差集C,(Bit映射->hash分组->多文件读写效率->磁盘寻址以及应用层面对寻址的优化)

6蚁群算法与蒙特卡洛算法；

7、写出你所知道的排序算法及时空复杂度，稳定性；

8、百度POI中如何试下查找最近的商家功能(坐标镜像+R树)。

9、遍历二叉树

10、自己集合实现一个队列

11、自己实现线程安全类

12、快速排序和冒泡的排序，怎么转换一下。

#### 数组

#### 链表

#### 栈

#### 优先队列

#### 二叉树

#### 递归回溯

#### 动态规划

#### 贪心算法

#### 图论

说明 ： 由于某些数据结构或算法思想（如队列和递归）通常需要配合其他数据结构一起使用， 因此它们可能包含一些其他类别的题目中。

## 题目（类别 - 题目名 - 难度 - LeetCode题号）：

### 数 组

|  |  |
| --- | --- |
| ## 给 比 | ## 数组01 - 0的移动 - 简单 - 283 定一个数组nums，写一个函数，将数组内的0移动到数组末尾，并保持其他非零元素在原数组中的相对位置不变。 如，给定nums = [0, 1, 0, 3, 12]，调用你的函数之后，nums应该变成[1, 3, 12, 0, 0]。 |
| 注 | 意： . 请直接在传入的数组对象上修改，而不是另外创建一份拷贝（术语叫做 in-place，也有中译为“原地”）。 . 尽量减少操作指令代码的行数。 |

#### 数组02 - 删除元素 - 简单 - 27

给定一个数组和一个值，原地移除数组中所有给定的值，并返回新数组的长度。 不允许申请额外空间，确保空间复杂度为O(1)。 数组中的元素可以被改变，不用考虑超出新长度之后的空间遗留。 比如： 给定nums = [3, 2, ,2 3]， val = 3， 你的函数应该返回length = 2， nums = [2, 2]。 - ----------------------------------------------------------------

#### 数组03 - 从有序数组中删除重复元素 - 简单 - 26

给定一个有序数组，原地删除重复元素使得数组中的元素只保留一个，并且返回新长度。 禁止申请额外空间，确保空间复杂度为O(1)。 比如： 给定nums = [1, 1, 2]， 你的函数应该返回length = 2，nums = [1, 2]。 不用考虑超出新长度之后的空间遗留。 - ----------------------------------------------------------------

#### 数组04 - 从有序数组中删除重复元素2 - 中等 - 80

与数组03题条件相同，但是变更一个要求：可以允许元素最多重复2次。 比如，给定nums = [1, 1, 1, 2, 2, 3]， 返回length = 5， nums = [1, 1, 2, 2, 3]。同样不用考虑超出新长度之后的空间遗留。 - ----------------------------------------------------------------

#### 数组05 - 两数的和（输入数组已排序） - 简单 - 167

【题中包含的数组的进阶技术：对撞指针技术】 给定一个整形数组，并且数组内元素已经按升序排列，找出两个元素，使得它们之和与给定的数相等。 函数应该返回找到的这两个元素的索引，并且第一个元素的索引小于等于第二个元素的索引，并且元素索引起始位置是基于1而不是基于0。 你可以假设给定的目标数在数组中必定找得到对应的两个元素。 比如： 输入： numbers = [2, 7, 11, 15], target = 9 输出： index = 1, index = 2 - ----------------------------------------------------------------

#### 数组06 - 装最多的水 - 中等 - 11

【题中包含的数组的进阶技术：对撞指针技术】 给出一个非负整数 a1, a2, ..., an,它们分别代表x轴上的一个点(i, ai)，在每个点上画高度为ai的“墙”， 用来代表容器。选择两堵墙，使得它们和x轴围起来的容器装水容量最大。 注意：给出的n>=2。 - ----------------------------------------------------------------

#### 数组07 - 排序颜色 - 中等 - 75

【题中包含的数组的进阶技术：对撞指针、三路快速排序】 给定一个数组，其中有n个元素，分别为红色、白色和蓝色，请将数组中的元素进行排序，使得颜色相同的元素排在一起，并且颜色顺序为红、白、蓝。 我们使用整数0、1、2分别代表红、白、蓝3种颜色。 注意：禁止使用标准库提供的排序算法。 提示：尝试使用三路快速排序的思路以O(n)的时间复杂度解决问题。 - ----------------------------------------------------------------

#### 数组08 - 找到数组中第k大的元素 - 中等 - 215

【题中包含的数组的进阶技术：对撞指针、快速排序】 在一个无序数组中找到第k大的元素。注意这里的第k大是指在排序顺序中第k大的元素，而不是第k个不同的元素 比如： 给定[3, 2, 1, 5, 6, 4]，k = 2，则应该返回5。 注意：你可以假设k的值是有效的。 提示：使用快速排序的思想可以以O(n)的时间复杂度解决该问题。 - ----------------------------------------------------------------

#### 数组09 - 最小尺寸子数组之和 - 中等 - 209

【题中包含的数组的进阶技术：滑动窗口技术】 给定一个整形数组和一个数组s，找出数组中最短的一个连续子数组，使得连续子数组中的元素之和sum>=s。 返回这个最短连续子数组。 比如：nums = [2, 3, 1, 2, 4, 3], s = 7 答案为[4, 3] - ----------------------------------------------------------------

#### 数组10 - 没有重复字符的最长子串 - 中等 - 3

【题中包含的数组的进阶技术：滑动窗口技术】 在一个字符串中寻找没有重复字母的最长子串 比如： “abcabcbb”,结果为“abc” “bbbbbb”，结果为“b” “pwwkew”，结果为“wke” - ----------------------------------------------------------------

#### 数组11 - 最小窗口子串 - 困难 - 76

给定一个字符串S和字符串T，在S中寻找最短的子串，包含T中所有的字符。 比如： S=“ADOBECODEBAXC”，T=“ABC” 结果为“BAXC”。

### 链 表

辅助数据结构：

struct ListNode {  
 int val;  
 ListNode \*next;  
 ListNode(int x): val(x), next(NULL) {}  
};

public class ListNode {  
 int val;  
 ListNode next;  
 ListNode(int x) { val = x; }  
}

#### 链表01 - 进阶反转链表 - 中等 - 92

反转一个链表中从m到n的元素。 比如：对于一个链表1->2->3->4->5->NULL，m = 2， n = 4 则返回链表 1->4->3->2->5->NULL 注意：可以假设1<=m<=n<=链表长度。 - ----------------------------------------------------------------

#### 链表02 - 成对交换链表节点 - 中等 - 24

给定一个链表，为每两个相邻节点做一次交换给定一个链表，为每两个相邻节点做一次交换。 要求不能修改节点的值，只能修改链表结构。要求O(1)的空间复杂度。 比如：1->2->3->4，应该返回2->1->4->3。 - ----------------------------------------------------------------

#### 链表03 - 链表相加低阶 - 中等 - 2

给定两个非空链表，分别代表两个非负整数。链表中的数字以逆序排列并且每个节点只包含一个个位数。 将两个链表所代表的数字相加并且以链表的形式返回这个和。 比如：输入2->4->3和5->6->4 应该返回 7->0->8 - ----------------------------------------------------------------

#### 链表04 - 链表相加高阶 - 中等 - 2

【题中包含的进阶技术：设计数据结构】 给定两个非空链表，分别代表两个非负整数。这次链表中的数字以顺序排列，同样每个链表包含一个个位数字。 计算两个链表所代表的数字的和，并以链表的形式返回这个和。 你可以假设给定的输入不会包含为0的开头，除非这个链表代表的数字本身就是0。 比如： 给定7->2->4->3和5->6->4， 输出7->8->0->7 提示： 如果我们要求不可以修改给出的链表呢？也就是说，先反转链表再利用链表02的解，这个做法是不被允许的。 提示也许可以创建一种数据结构来解决这个问题。 - ----------------------------------------------------------------

#### 链表05 - 删除链表元素 - 简单 - 203

【题中包含的进阶技术：虚拟头结点】 删除链表中值为val的元素。 比如： 给出: 1->2->6->3->4->5->6，val = 6 返回: 1->2->3->4->5 - ----------------------------------------------------------------

#### 链表06 - 从有序链表中删除重复元素 - 简单 - 83

【题中包含的进阶技术：虚拟头结点】 给定一个有序链表，删除其中所有重复的元素，只留下不存在重复的元素。 比如： 给出1->2->3->3->4->4->5，返回1->2->5， 给定1->1->2->3->3 返回 1->2->3 - ----------------------------------------------------------------

#### 链表07 - 从链表中删除倒数第N个元素 - 中等 - 19

【题中包含的进阶技术：双指针技术】 给定一个链表，删除其倒数第N个元素并返回头结点。 比如： 给定： 1->2->3->4->5， n = 2， 则应该返回 1->2->3->5。 注意：n为有效值。 提示：使用双指针技术来实现只用一次遍历求解。 - ----------------------------------------------------------------

#### 链表08 - 旋转链表 - 中等 - 61

【题中包含的进阶技术：双指针技术】 给定一个链表，让链表向右旋转k位，其中k为非负数。 比如: 1->2->3->4->5->NULL，k = 2 返回: 4->5->1->2->3->NULL。 - ----------------------------------------------------------------

#### 链表09 - 重排链表 - 难 - 143

【题中包含的进阶技术：双指针技术】 给定一个链表L: L0→L1→…→Ln-1→Ln,将其重排序为 L0→Ln→L1→Ln-1→L2→Ln-2→… 注意：原地排序，并且请思考如何只用一次遍历求解。 比如：1->2->3->4，返回1->4->2->3。 - ----------------------------------------------------------------

### 栈

#### 栈01 - 括号配对 - 简单 - 20

给定一个字符串，其中只包含()，[]，{},判定字符串中的括号匹配是否合法。 比如 “()”，“()”是合法的，“(}”，“([)]”是非法的。 - ----------------------------------------------------------------

#### 栈02 - 逆波兰表达式求值 - 中等 - 150

给定一个数组，表示一个逆波兰表达式，求其值。 运算类型只有+、-、*、/。 比如： ["2", "1", "+", "3", "\*"] -> ((2 + 1)*  3) -> 9 ["4", "13", "5", "/", "+"] -> (4 + (13 / 5)) -> 6 - ----------------------------------------------------------------

#### 栈03 - 简化路径 - 中等 - 71

给定一个Unix风格的绝对路径，进行简化。 比如： path = "/home/" => "/home" path = "/a/./b/../../c/" => "/c" 注意：考虑一些边界条件： . 对于path = "/../" 可以返回"/"； . 路径中可能存在的连续'/'，比如"/home//foo"，这种情况下应该忽略重复的斜杠并返回"/home/foo"。 - ----------------------------------------------------------------

#### 栈04 - 嵌套列表平滑迭代器 - 中等 - 341

给定一个整形的嵌套列表，实现一个迭代器来使对它的遍历平滑化。 其中列表中的元素可能是一个整形元素，也可能是一个列表 —— 这个列表同样也可能同时包含整形元素或另一个子列表。 比如：给定一个列表[[1,1],2,[1,1]]，连续调用迭代器的next方法，直到hasNext返回false，则遍历的元素依次为[1,1,2,1,1]。 给定一个列表[1,[4,[6]]]，同样，遍历结果应该为[1,4,6]。 （题目中已经给出了嵌套列表这个数据结构的接口的定义，请自行查看） - ----------------------------------------------------------------

### 队列

由于队列的一个重要的作用，就是实现广度优先算法，因此队列经常用来解决树和图中的相关问题。所以，我们将会在树和图的相关问题中，尝试用队列这种数据结构解决问题。因此我们可以看到队列一般是与其他数据结构结合使用。大家可以回忆一下，这一点的另一个体现， 也就是优先队列。 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#### 优先队列01 - 出现频率第k的元素 - 中等 - 347

给定一个非空整形数组，返回出现频率第k的元素。 比如：给定[1,1,1,2,2,3]，k=2，返回[1,2]。 注意： \* 你可以假设k为有效的值，1 <= k <= 独一元素数量 \* 算法时间复杂度必须至少为O(nlogn)。 提示：这里提供三个思路： . 扫描一遍统计频率；排序找到前k个出现频率最高的元素 O(nlogn)； . 维护优先队列，O(nlogk) . 维护优先队列，时间复杂度为(Onlog(n-k)) - ----------------------------------------------------------------

#### 优先队列02（链表） - 合并k个有序链表 - 困难 - 23

合并k个有序的链表并返回合成的有序列表。 提示：当k为2时，其实就是经典的归并排序中的归并过程。当这个问题解决后， 我们就自然可以设计出k分归并排序算法了。 - ----------------------------------------------------------------

### 树和递归

大家知道，树这种数据结构中最经典的应用自当是二叉树。二叉树具备天然的递归结构，因此， 二叉树相关的大部分题目当中，都可以运用递归这种思想解决问题。 这里简单地谈一谈如何设计一个递归算法： 你需要深刻地认识“递”和“归”这两个字，“递”意味着传递，因此在设计时，你要明白你的代码如何传递到所有子问题；“归”意味着边界条件，递归程序必须在适当的时候返回，如何考虑返回条件，来达成最终的结果，掌握这两个字，就能更加深入地理解递归这个思想的精髓。 - -----------------------------------------------------------------

#### 二叉树01（队列） - 二叉树的层序遍历 - 中等 - 102

给定一个二叉树，返回其层序遍历结构（从左往右，一层一层地遍历）。 比如： 给定二叉树 [3, 9 , 20 , null, null, 15, 7]，

3  
 / \  
 9 20  
 / \  
 15 7

返回的结果应该是：

[  
 [3],  
 [9, 20],  
 [15, 7]  
]

#### 二叉树02（队列） - 从右侧观察二叉树的结果 - 中等 - 199 给定一棵二叉树，相信你站在树的右边观察它，返回你能看到的结点，顺序为自上而下：

比如：

1<---  
 / \  
2 3<---  
 \ \  
 5 4 <---

应该返回[1, 3, 4] - ----------------------------------------------------------------

#### 二叉树03 - 二叉树最低深度 - 简单 - 111

求一棵二叉树的最低深度，也就是从根节点到叶子结点的最短路径的长度。

#### 二叉树04 - 反转二叉树 - 简单 - 226

这个题目大有来头~当年homebrew的作者去面试Google，就是因为这道基础题做不出来被pass掉了，这在业界曾经引起了广泛的反响。 Google: 90% of our engineers use the software you wrote (Homebrew), but you can’t invert a binary tree on a whiteboard so fuck off. 反转二叉树。 原树：

4  
 / \  
 2 7  
 / \ / \  
 1 3 6 9

反转后：

4  
 / \  
 7 2  
 / \ / \  
 9 6 3 1

#### 二叉树05 - 判断二叉树是否对称 - 简单 - 101

给定一棵二叉树，检查它是否为对称的。 比如：

1  
 / \  
 2 2  
 / \ / \  
3 4 4 3

是一棵对称的二叉树，而

1  
 / \  
 2 2  
 \ \  
 3 3

则为非对称。 注意：尝试用递归和迭代两种方式解决。 - ----------------------------------------------------------------

#### 二叉树06 - 计算完全二叉树的节点个数 - 中等 - 222

给定一个完全二叉树，求它的节点个数。 概念：完全二叉树： 除了最后一层，其他所有层的节点数达到最大，同时最后一层的所有节点都在最左侧。 满二叉树： 所有节点数达到最大。 - ----------------------------------------------------------------

#### 二叉树07 - 判断一棵二叉树是否为平衡二叉树 - 简单 - 110

判断一棵二叉树是否为平衡二叉树。 平衡二叉树： 每一个节点的左右子树的高度差不超过1。 - ----------------------------------------------------------------

#### 二叉树08 - 路径和 - 简单 - 112

给出一棵二叉树以及一个数字sum，判断在这棵二叉树上是否存在一条从根节点到叶子的路径， 其路径上的所有节点和为sum。 技巧：如何在递归过程中保存数据。 比如：

5  
 / \  
 4 8  
 / / \  
 11 13 4  
 / \ \  
 7 2 1

如果sum = 22，则可以找到这条路径满足条件： 5->4->11->2。 - ----------------------------------------------------------------

#### 二叉树09 - 左叶子的和 - 简单 - 404

找到树中所有左叶子的和。

比如：  
 3  
 / \  
 9 20  
 / \  
 15 7  
这棵树中有两个左叶子，9和15，因此返回结果应该为24。

#### 二叉树10 - 二叉树路径 - 简单 - 257

给定一棵二叉树，返回所有表示从根节点到叶子结点路径的字符串。 技巧：如何利用递归函数的返回值。

如：  
 1  
 / \  
 2 3  
 \   
 5   
返回结果为：  
["1->2->5", "1->3"]

#### 二叉树11 - 根节点到叶子结点的和 - 简单 - 129

给定一棵二叉树，每个节点都是一个0-9的数字。从根节点到叶子结点的每条路径可以表示成一个数，求这些数的和。

比如：  
 1  
 / \  
2 3  
1->2，可以表示成12；  
1->3，可以表示成13；  
所以结果为12+13=25。

#### 二叉树12 - 路径和3 - 简单 - 437

技巧：更加复杂的递归逻辑。 给出一棵二叉树以及一个数字sum，判断二叉树上存在多少条路径，使其路径上的所有节点的和为sum。 注意： \* 其中路径不一定要起始于根节点、终止于叶子结点。 \* 路径虽然可以从任意节点开始，但只能往下走。 - ----------------------------------------------------------------

#### 二叉树13（二分搜索树） - 二叉搜索树中的最近公共祖先 - 简单 - 235

给定一棵二叉搜索树和两个节点，寻找这两个节点的最近公共祖先。

比如：  
 6  
 / \  
 2 8  
 /\ /\  
0 4 7 9  
 / \  
3 5  
给定2和8，则结果为6。给定2和4则结果为2。

#### 二叉树14（二分搜索树） - 在二分搜索树中删除一个节点 - 中等 - 450

给定一个二分搜索树，删除其中一个节点。 一般来说，删除操作可以分为两个不走： . 查找到要删除的那个节点 . 如果找到，则删除它。 注意： 时间复杂度至少得小于等于O(树的高度)

比如：  
root = [5,3,6,2,4,null,7]  
key = 3  
 5  
 / \  
 3 6  
 / \ \  
 2 4 7  
给定要删除的节点为3。  
其中一个可行的答案为[5,4,6,2,null,null,7]，如下：  
 5  
 / \  
 4 6  
 / \  
 2 7  
另一种有效的答案为 [5,2,6,null,4,null,7].  
 5  
 / \  
 2 6  
 \ \  
 4 7

#### 二叉树15（二分搜索树） - 将有序数组转换为一颗平衡的二叉搜索树 - 简单 - 108

给定一个有序数组，生成一棵平衡的二叉搜索树。 - ----------------------------------------------------------------

|  |  |
| --- | --- |
| ## | # 递归和回溯 |
| 之 | 前的递归算法问题都是建立在二叉树上的，那么在更广义的范围内运用递归呢？ |
| 在 | 递归问题中一个经典的思想就是回溯法，而它们适用的问题一般都是树形问题。 |
| 实 | 际上回溯法是一个很经典的思想，其核心在于搜索，它也是古典人工智能的基础。 |
| 我 | 们经常听说的8皇后问题、数独，都可以采用回溯法来解决。 |

#### 递归回溯01 - 9宫格键盘中的字母组合 - 中等 - 17

给定一个数字字符串，返回这些数字能在手机的9宫格键盘中组合成的所有字母组合。

比如：  
给出“23”，则可以组合成：  
["ab","ae","af","bd","be","bf","cd","ce","cf"]。

#### 递归回溯02（排列问题