内存溢出 & 内存泄漏 & 内存抖动

定义 原因 解决

内存溢出

应用程序所需内存超出系统分配的内存限额，从而导致内存溢出

1. 内存中加载的数据量过于庞大，如一次从数据库取出过多数据
2. 内存泄露
3. 代码中存在死循环或循环产生过多重复的对象实体（内存抖动）
4. 使用的第三方软件中的BUG
5. 启动参数内存值设定的过小

内存泄露

当一个对象已经不需要再使用本该被回收时，另外一个正在使用的对象持有它的引用从而导致它不能被回收，这导致本该被回收的对象不能被回收而停留在堆内存中，这就产生了内存泄漏

持有引用者的生命周期>被引用者的生命周期：

1. 集合类：回收集合元素
2. Static关键字修饰的成员变量：避免Static引用过多实例
3. 非静态内部类/匿名类：使用静态内部类
4. 资源使用后未关闭：关闭资源对象

内存抖动

内存大小不断浮动的现象

由于大量、临时的小对象频繁创建，导致程序频繁地分配内存 & 垃圾回收器（GC）频繁回收内存

垃圾收集器（GC）频繁地回收内存会导致卡顿，甚至内存溢出（OOM）——大量、临时的小对象频繁创建会导致内存碎片，使得当需分配内存时，虽总体上有剩余内存可分配，但由于这些内存不连续，导致无法模块分配。系统则视为内存不够，故导致内存溢出OOM

尽量避免频繁创建大量、临时的小对象

内存泄露

定义

当一个对象已经不需要再使用本该被回收时，另外一个正在使用的对象持有它的引用从而导致它不能被回收，这导致本该被回收的对象不能被回收而停留在堆内存中，这就产生了内存泄漏。

本质原因

持有引用的对象的生命周期>被引用的对象的生命周期



# Bitmap优化

原因

图片资源（Bitmap）非常消耗内存，占用App内存大部分。Android系统分配给每个应用程序内存有限，因此可能引发内存溢出（OOM），导致应用崩溃（Crash）

方案

1. 使用完毕后释放图片资源
2. 根据分辨率适配 & 缩放图片
3. 按需选择合适的解码方式
4. 设置图片缓存——三级缓存机制

# 谈谈你项目中内存优化的一些经验？



# 启动优化

黑白屏优化

系统在启动Activity的setContentView之前绘制窗体，此时布局资源还未加载，于是使用了默认的背景色。

解决：把启动图bg\_splash设置为窗体背景，避免刚刚启动App的时候出现，黑/白屏

onCreate优化

onCreate()耗时长会影响应用程序布局绘制的时间。因此应该减少onCreate工作量。

一般重写Application，在onCreate()方法中做一些初始化操作（如第三方SDK配置），可以将这些较大的第三方库通过开启一个异步线程中进行初始化。

# 布局优化

布局优化思想

减少Overdraw（过度绘制）（一般通过减少UI层级、简化布局实现）

布局优化方法

1. 善用相对布局RelativeLayout

可以通过扁平的RelativeLayout降低LinearLayout嵌套所产生布局树的层级

1. 使用抽象布局标签include、merge、ViewStub
2. 使用Android最新的布局方式ConstaintLayout