一、Android profile工具查看APP内存泄漏

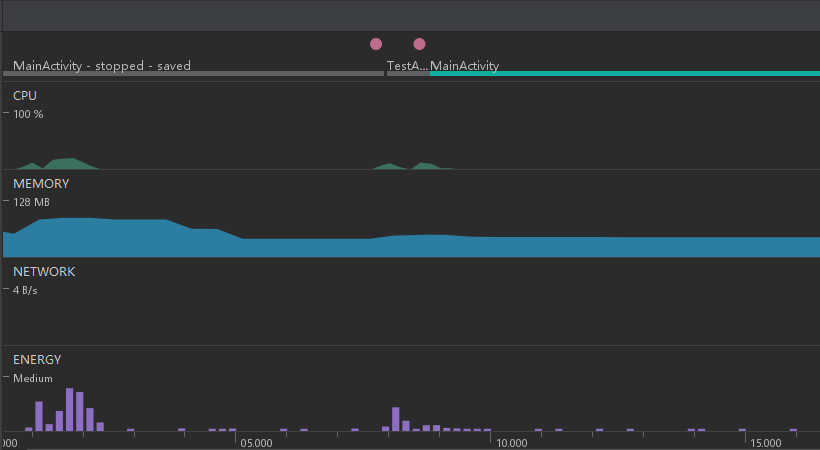
引起ANR示例代码：

*/\*\*  
 \* 测试ANR  
 \*/*public class TestANRActivity extends AppCompatActivity implements View.OnClickListener {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 Handler handler = new Handler();  
 handler.postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 Log.*e*("whh0114", "postDelayed......");  
 }  
 }, 800000L);  
 }  
  
 @Override  
 public void onClick(View view) {  
 finish();  
 }  
}

在两个Activity来回跳转，其中一个Activity中onCreate()方法中添加上段代码

测试流程：

1. 使用工具运行APP，打开运行内存使用图，如下图：

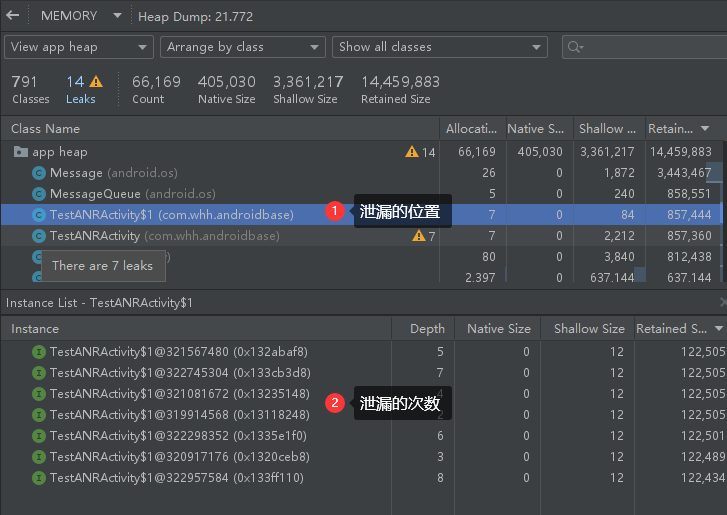


1. 选择MEMORY，来回切换界面，查看内存使用趋势图，如果波动大时，可点击



垃圾回收按钮：手动GC，在内存上升时，可手动回收；

Dump Java Heap：堆转储文件，在程序崩溃前点击，可收集到内存增长、导致内存泄漏的实际代码位置，如下图：



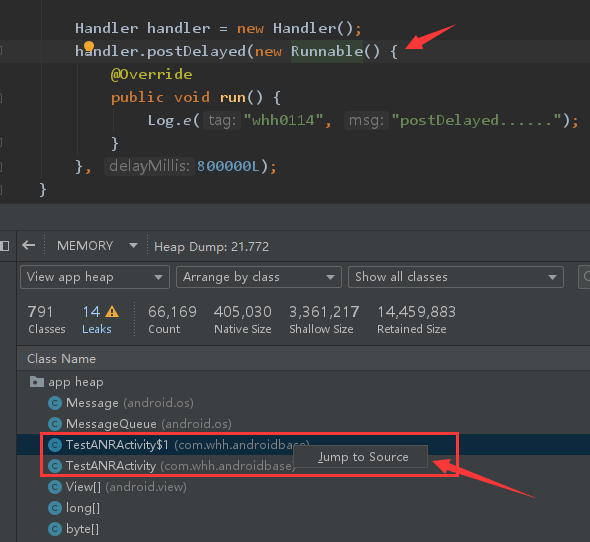
根据上图表现，在TestANRActivity中出现了内存泄漏，右键Jump to Source

名词解释(版本不同，AS UI可能存在差异)：

Reference Tree代表指向该实例的引用，可以从这里面查看内存泄漏的原因；

Shallow Size指的是该对象本身占用内存的大小；

Retained Size代表该对象被释放后，垃圾回收器能回收的内存总和。



可直接跳转泄漏的位置(定位到代码啦！)，哇咔咔！

**<https://www.jianshu.com/p/082766511405>**

· **尽量使用静态内部类。**每个非静态内部类将持有一个外部类的隐式引用，这可能会导致不必要的问题。使用静态内部类代替非静态内部类，并通过弱引用存储一些必要的生命周期引用。（Activity、context不要随意直接传入，如果内部类的生命周期要大于Activity，在Activity释放时，内部类还在，导致Activity一直存在，最后造成内存泄漏，①可改为弱引用存储传入，垃圾回收时随即被回收；②可以将参数context改为context.getApplicationContext()即可，持有的是Application，不是Activity，Application是整个APP的生命周期，要比Activity的寿命长！）

· **考虑后台服务等手段，** Android提供了多种在非主线程工作的方法，如**HandlerThread**，**IntentService**和**AsyncTask**，它们每个都有自己的优缺点。另外，Android提供了一些机制来传递信息给主线程以更新UI。譬如，**广播接收器**就可以很方便实现这一点。

· **不要一味依赖垃圾回收器。**使用具有垃圾回收功能的语言编码很容易有这样的想法：即没必要考虑内存管理。我们的示例清楚地表明，并非如此。因此，请确保你分配的资源都被预期回收。

public class ListenerActivity extends Activity {  
  
 TextView textView;  
 Call call;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.activity\_listener);  
 textView = (TextView) findViewById(R.id.textView);  
  
 GitHubService service = ((LeaksApplication) getApplication()).getService();  
 call = service.listRepos("google");  
 call.enqueue(new RepoCallback(textView));  
 }  
  
 @Override  
 protected void onDestroy() {  
 call.cancel(); //记得释放  
 super.onDestroy();  
 }  
  
 //匿名内部类，非静态内部类将持有一个外部类的隐式引用，改用static，并通过弱引用存储一些必要的生命周期引用  
 private static class RepoCallback implements Callback<List<Repo>> {  
  
 private final WeakReference<TextView> resultTextView; //弱引用  
  
 public RepoCallback(TextView resultTextView) {  
 this.resultTextView = new WeakReference<>(resultTextView);  
 }  
  
 @Override  
 public void onResponse(Call<List<Repo>> call,  
 Response<List<Repo>> response) {  
 TextView view = resultTextView.get();  
 if (view != null) {  
 int numberOfRepos = response.body().size();  
 view.setText(String.*valueOf*(numberOfRepos));  
 }  
 }  
  
 @Override  
 public void onFailure(Call<List<Repo>> call, Throwable t) {  
 // Code omitted.  
 }  
 }  
}

Java对引用的分类（Strong reference, SoftReference, WeakReference, PhatomReference）：

|  |  |  |  |
| --- | --- | --- | --- |
| 级别 | 什么时候被垃圾回收 | 用途 | 生存时间 |
| 强引用 | 从来不会 | 对象的一般状态 | JVM停止运行时终止 |
| 软引用 | 在内存不足时 | 对象简单？缓存 | 内存不足时终止 |
| 弱引用 | 在垃圾回收时 | 对象缓存 | gc运行后终止 |
| 虚引用 | 随时被回收 | 在进入 finalize() 方法后进行特殊的清理过程 | 随时被回收 |

二、monkey检测方法

adb shell monkey -p xxx(包名) -s 500 -v 10000 压力测试

实际上是执行手机中/system/bin/monkey这个脚本；

https://www.cnblogs.com/yindada/p/9845032.html

1.adb shell pm list package 查看手机上所有的安装包

2.adb shell pm list package -3 查看第三方安装包

3.adb shell pm list packages -s 查看系统的包

4.adb shell pm list packages -e “要搜索的内容” 查看启动包中搜索的包

5.adb shell dumpsys window | findstr mCurrentFocus 获取当前打开的APP包名（前面的为包名，斜杠后面的是当前启动的acivity）

指定一个包执行10次：adb shell monkey -p 包名 10

指定多个包执行10次：adb shell monkey -p 包名 –p 包名 10

指定日志存放目录 adb shell monkey -p 包名 -v -v -v 200>D:AAlog.txt

指定日志存放在手机上

adb shell

monkey -p com.kugou.android --pct-touch 25 -v -v 200 1>/mnt/sdcard/monkey.txt 2>/mnt/sdcard/error.txt &

日志分析：

1、查找出差步骤：

a、找到monkey里哪个地方出错，查看Monkey执行的是哪一个Activity，在switch后面找，两个swtich之间如果出现了崩溃或其他异常，可以在该Activity中查找问题的所在。

b、查看Monkey里面出错前的一些事件动作，手动执行该动作

Sleeping for XX milliseconds这是执行Monkey测试时，throttle设定的间隔时间，每出现一次，就代表一个事件，Sending XX 就是代表一个操作，如下图的两个操作 应该就是一个点击事件。

c、若以上步骤还不能找出，则可以使用之前一样的seek再执行monkey命令一遍，便于复现

2、测试结果分析：

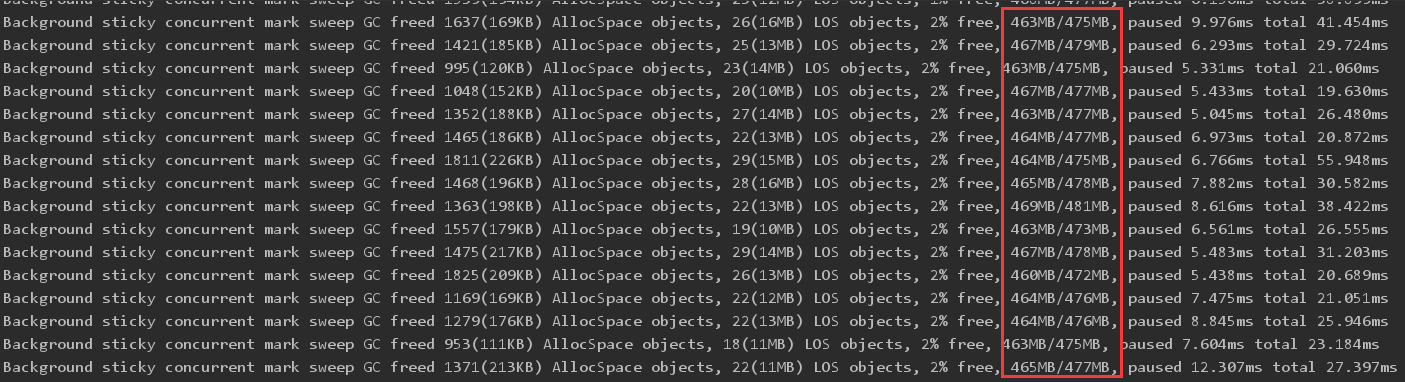
a、程序无响应，ANR问题：在日志中搜索“ANR”

b、崩溃问题：在日志中搜索“CRASH”

c、其他问题:在日志中搜索”Exception”

3、检测内存泄漏问题

a、JNI泄漏：



如果这个数值一直增长不降，说明程序存在内存泄漏！