android 因其系统的特殊性,安装的软件默认都安装到内存中,所以随着用户安装的软件越来越多,可供运行的程序使用的内存越来越小,这就要求我们在开发 android 程序时,尽可能的少占用内存。根据我个人的开发经验总结了如下几点优化内存的方法:

1 创建或其他方式获得的对象如不再使用,则主动将其置为 null。

2 尽量在程序中少使用对图片的放大或缩小或翻转.在对图片进行操作时占用的内存可能比图片本身要大一些。

3 调用图片操作的后，及时的清空，调用 recycle()提醒经行垃圾回收。

4 尽可能的将一些静态的对象(尤其是集合对象),放于 SQLite 数据库中。并且对这些数据的搜索匹配尽可能使用 sql 语句进行。

5 一些连接资源在不使用使应该释放,如数据库连接文件输入输出流等。应该避免在特殊的情况下不释放(如异常或其他情况)

6 一些长周期的对像引用了短周期的对象,但是这些短周期的对象可能只在很小的范围内使用。所以在查内存中也应该清除这一隐患。

如果你想写一个 Java 程序，观察某对象什么时候会被垃圾收集的执行绪清除，你必须要用一个 reference 记住此对象，以便随时观察，但是却因此造成此对象的 reference 数目一直无法为零，使得对象无法被清除。

java.lang.ref.WeakReference

不过，现在有了 Weak Reference 之后，这就可以迎刃而解了。如果你希望能随时取得某对象的信息，但又不想影响此对象的垃圾收集，那

么你应该用 Weak Reference 来记住此对象，而不是用一般的 reference。

```
A obj = new A();
WeakReference wr = new WeakReference(obj);
obj = null;
//等待一段时间，obj 对象就会被垃圾回收
…
if (wr.get()==null) {
System.out.println(“obj 已经被清除了 “);
} else {
System.out.println(“obj 尚未被清除，其信息是
“+obj.toString());
}
…
```

在此例中，透过 get() 可以取得此 Reference 的所指到的对象，如果传出值为 null 的话，代表此对象已经被清除。

这类的技巧，在设计 Optimizer 或 Debugger 这类的程序时常会用到，因为这类程序需要取得某对象的信息，但是不可以影响此对象的垃圾收集。

java.lang.ref.SoftReference

Soft Reference 虽然和 Weak Reference 很类似，但是用途却不同。被 Soft Reference 指到的对象，即使没有任何 Direct Reference，也不会被清除。一直要到 JVM 内存不足时且 没有 Direct Reference

时才会清除，SoftReference 是用来设计 object-cache 之用的。如此一来 SoftReference 不但可以把对象 cache 起来，也不会造成内存不足的错误 （OutOfMemoryError）。我觉得 Soft Reference 也适合拿来实作 pooling 的技巧。

```
A obj = new A();
SoftRefenrence sr = new SoftReference(obj);
//引用时
if(sr!=null){
obj = sr.get();
}else{
obj = new A();
sr = new SoftReference(obj);
}
```