参考文档：<https://www.cnblogs.com/c-xiaohai/p/12489336.html>

8、总结：

一般常见的OOM，要么是短时间内涌入大量的对象，导致你的系统根本支持不住，此时你可以考虑优化代码，或者是加机器；要么是长时间来看，你的很多对象不用了但是还被引用，就是内存泄露了，你也是优化代码就好了；这就会导致大量的对象不断进入老年代，然后频繁full gc之后始终没法回收，就撑爆了

要么是加载的类过多，导致class在永久代理保存的过多，始终无法释放，就会撑爆

我这里可以给大家最后提一点，人家肯定会问你有没有处理过线上的问题，你就说有，最简单的，你说有个小伙子用了本地缓存，就放map里，结果没控制map大小，可以无限扩容，最终导致内存爆了，后来解决方案就是用了一个ehcache框架，自动LRU清理掉旧数据，控制内存占用就好了。

另外，务必提到，线上jvm必须配置-XX:+HeapDumpOnOutOfMemoryError，-XX:HeapDumpPath=/path/heap/dump。因为这样就是说OOM的时候自动导出一份内存快照，你就可以分析发生OOM时的内存快照了，到底是哪里出现的问题。

Jstat

Jmap