几种常见的垃圾回收算法之引用计数算法

在C++中，对象所占的内存在程序结束运行之前一直被占用，需要我们明确释放；而在Java中，当没有对象引用指向原先分配给某个对象的内存时，该内存便成为垃圾。JVM的一个系统级线程会自动释放该内存块。

垃圾收集意味着程序不再需要的对象是"无用信息"，这些信息将被丢弃。当一个对象不再被引用的时候，内存回收它占领的空间，以便空间被后来的新对象使用。

事实上，除了释放没用的对象，垃圾收集也可以清除内存记录碎片。由于创建对象和垃圾收集器释放丢弃对象所占的内存空间，内存会出现碎片。碎片是分配给对象的内存块之间的空闲内存洞。碎片整理将所占用的堆内存移到堆的一端，JVM将整理出的内存分配给新的对象。

垃圾收集能自动释放内存空间，减轻编程的负担。这使Java 虚拟机具有一些优点。首先，它能使编程效率提高。在没有垃圾收集机制的时候，可能要花许多时间来解决一个难懂的存储器问题。在用Java语言编程的时候，靠垃圾收集机制可大大缩短时间。其次是它保护程序的完整性，垃圾收集是Java语言安全性策略的一个重要部份。

垃圾收集的一个潜在的缺点是它的开销影响程序性能。Java虚拟机必须追踪运行程序中有用的对象，而且最终释放没用的对象。这一个过程需要花费处理器的时间。其次垃圾收集算法的不完备性，早先采用的某些垃圾收集算法就不能保证100%收集到所有的废弃内存。当然随着垃圾收集算法的不断改进以及软硬件运行效率的不断提升，这些问题都可以迎刃而解。

在C++中，对象所占的内存在程序结束运行之前一直被占用，在明确释放之前不能分配给其它对象；而在Java中，当没有对象引用指向原先分配给某个对象的内存时，该内存便成为垃圾。JVM的一个系统级线程会自动释放该内存块。垃圾收集意味着程序不再需要的对象是"无用信息"，这些信息将被丢弃。当一个对象不再被引用的时候，内存回收它占领的空间，以便空间被后来的新对象使用。事实上，除了释放没用的对象，垃圾收集也可以清除内存记录碎片。由于创建对象和垃圾收集器释放丢弃对象所占的内存空间，内存会出现碎片。碎片是分配给对象的内存块之间的空闲内存洞。碎片整理将所占用的堆内存移到堆的一端，JVM将整理出的内存分配给新的对象。

垃圾收集能自动释放内存空间，减轻编程的负担。这使Java 虚拟机具有一些优点。首先，它能使编程效率提高。在没有垃圾收集机制的时候，可能要花许多时间来解决一个难懂的存储器问题。在用Java语言编程的时候，靠垃圾收集机制可大大缩短时间。其次是它保护程序的完整性，垃圾收集是Java语言安全性策略的一个重要部份。

垃圾收集的一个潜在的缺点是它的开销影响程序性能。Java虚拟机必须追踪运行程序中有用的对象，而且最终释放没用的对象。这一个过程需要花费处理器的时间。其次垃圾收集算法的不完备性，早先采用的某些垃圾收集算法就不能保证100%收集到所有的废弃内存。当然随着垃圾收集算法的不断改进以及软硬件运行效率的不断提升，这些问题都可以迎刃而解。

今天我们来说说比较简单的垃圾回收算法：引用计数算法。它的含义：给对象中添加一个引用计数器，每当有一个地方引用它时，计数器的值就加1；当引用失效时，计数器值就减1；任何时刻计数器为0的对象就是不可能再被使用的。这也就是需要回收的对象。

客观地说，引用计数算法（Reference Counting）的实现简单，判定效率也非常高。在大部分情况下它都是一个不错的算法。

引用计数算法的优点：（1）、内存管理的开销分布于整个应用程序运行期间，无需挂起应用程序的运行来做垃圾回收；
（2）、空间上的引用局部性比较好，当某个对象的引用计数值变为0时，系统无需访问位于堆中其他页面的单元；
（3）、引用计数算法提供了一种类似于栈分配的方式，废弃即回收。

但是，Java语言中没有选用引用计数算法来管理内存，其中最主要的原因是它很难解决对象之间的互相循环引用的问题。