**垃圾收集器与内存分配策略**

**垃圾收集器**

**确定回收对象**

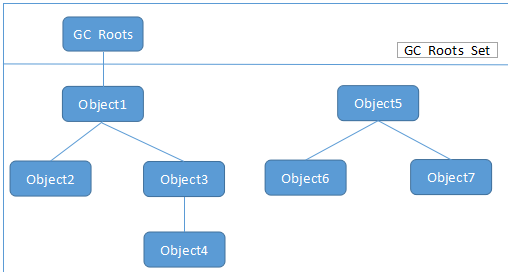
**引用计数法**

定义：给对象添加一个计数器，每一次被引用，计数器加1。引用失效时，减1，但计数为0时，该对象不可用。

缺点：A对象引用B对象，B对象引用A对象，那么计数永远也不为0

可达性分析算法

定义：当一个对象没有路径到达GC Roots对象时，该对象可以回收



概念：

GC Roots对象

虚拟机栈引用的对象

方法区静态属性引用的对象

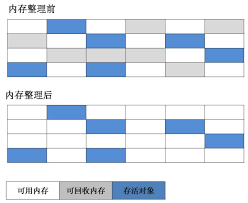
方法区常量属性引用的对象

**垃圾回收算法**

**标记-清除算法**

定义：

标记回收对象，把该对象的内存回收



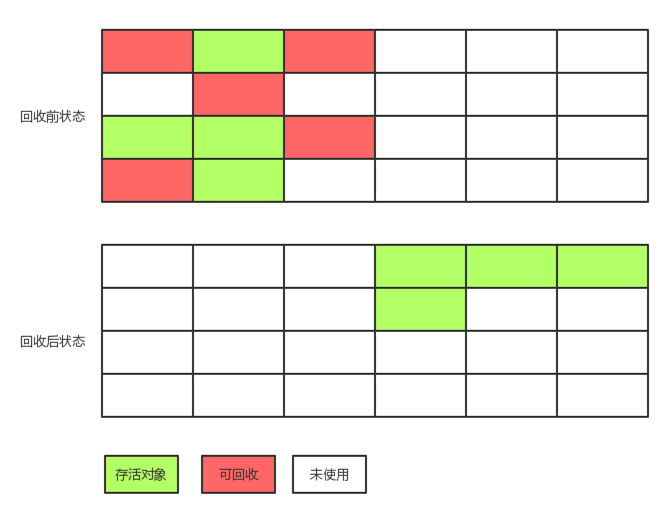
缺点：

1. 效率低
2. 容易产生内存碎片

**复制算法**：

定义：

把内存分成两半，一半用来为对象分配内存，另一半不使用。但回收内存时把活的对象拷贝到没有使用的内存，把另一半的内存全部回收



缺点：

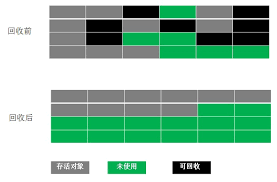
1. 浪费资源，每一次只使用一半的内存
2. 当存活对象比较多时，复制操作比较多，效率低

使用：

由于新生代的对象98%是很快不可达，Eden:Survivor:Survivor=8:1;1,每一次回收都把活的对象拷贝到空闲的Survivor块，把Eden和另一块Survivor回收。

**标记-整理算法：**

所有存活对象往内存的一边靠拢，把其他部分全部回收



分代收集算法：

新生代使用复制-清除算法，老年代使用标记-清除或标记-整理算法。

**确定回收的时间**：

安全点：

长时间执行的语句，比如for循环

安全区域：

对象的应用关系不会发生变化

**内存分配：**

1. 对象优先在Eden分配
2. 大对象直接进入老年代
3. 长期存活的对象进入老年代

逃过多次的新生代回收（15次）