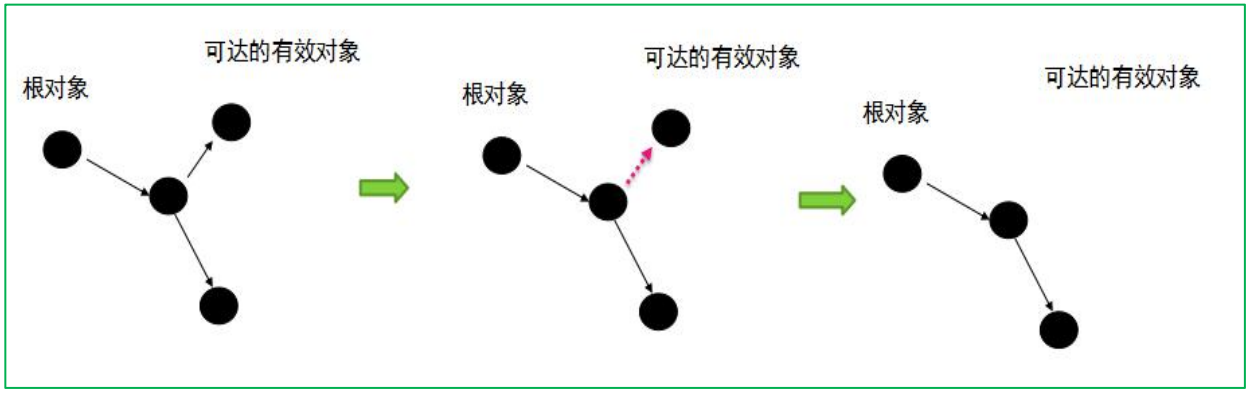
GC 算法与种类

（一）GC 的概念

GC,指 Ganbage Collection 垃圾回收器。**GC 的算法主要分为四类：引用计数法、标记清除、标记压缩、复制算法。**下面将对这几种算法进行逐一说明。

（二）GC 的算法——引用计数法

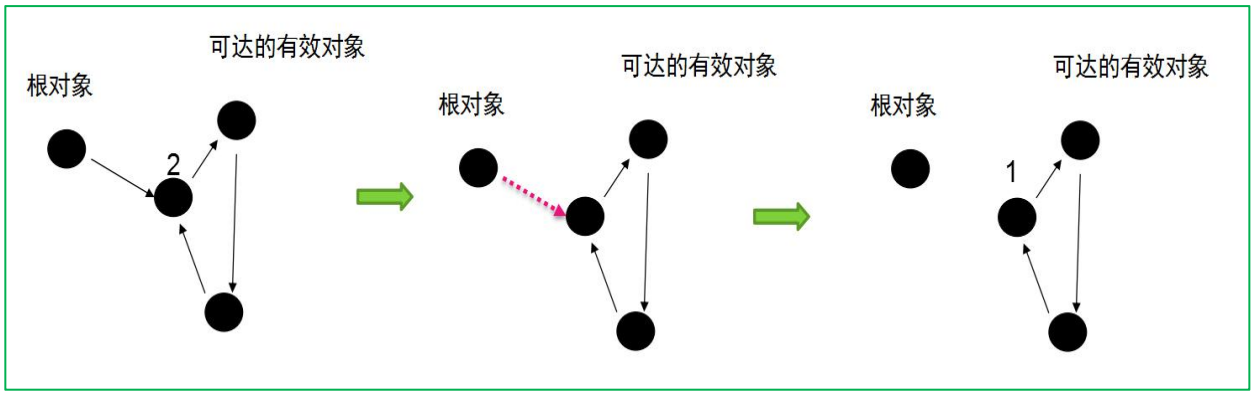
引用计数器的实现很简单，对于一个对象 A，只要有任何一个对象引用了 A，则 A 的引用计数器就加 1，当引用失效时，引用计数器就减 1。只要对象 A 的引用计数器的值为 0，则对象 A 就不可能再被使用。



引用计数法的问题：

（1）引用和去引用伴随着加法和减法，影响性能；

（2）很难处理循环引用；

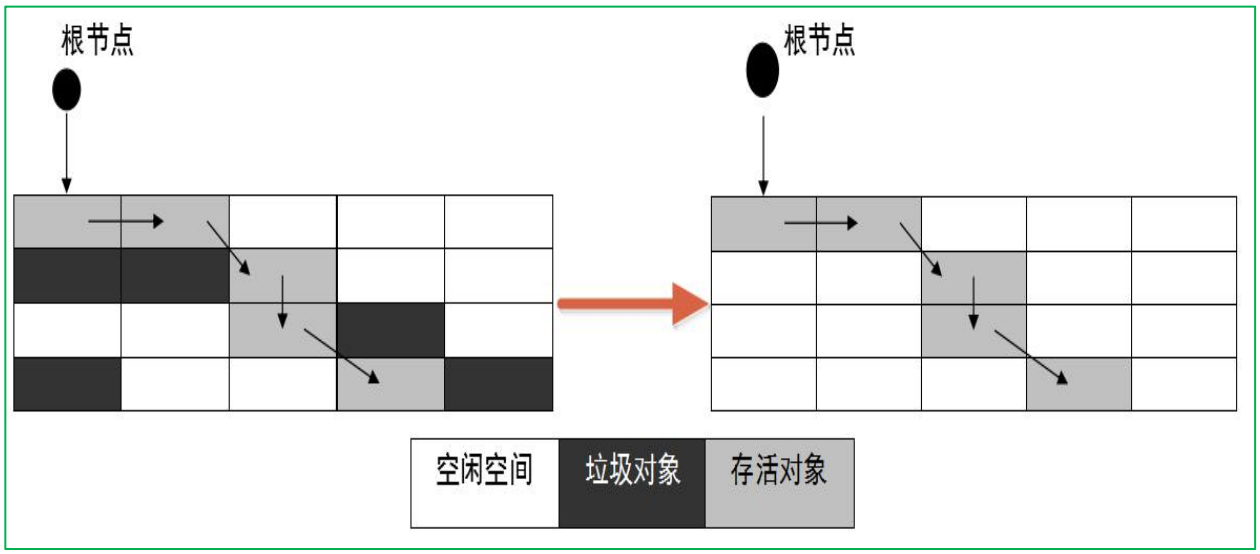


（二）GC 的算法——标记-清除算法

标记-清除算法是现代垃圾回收算法的思想基础。标记-清除算法将垃圾回收分为两个阶段：

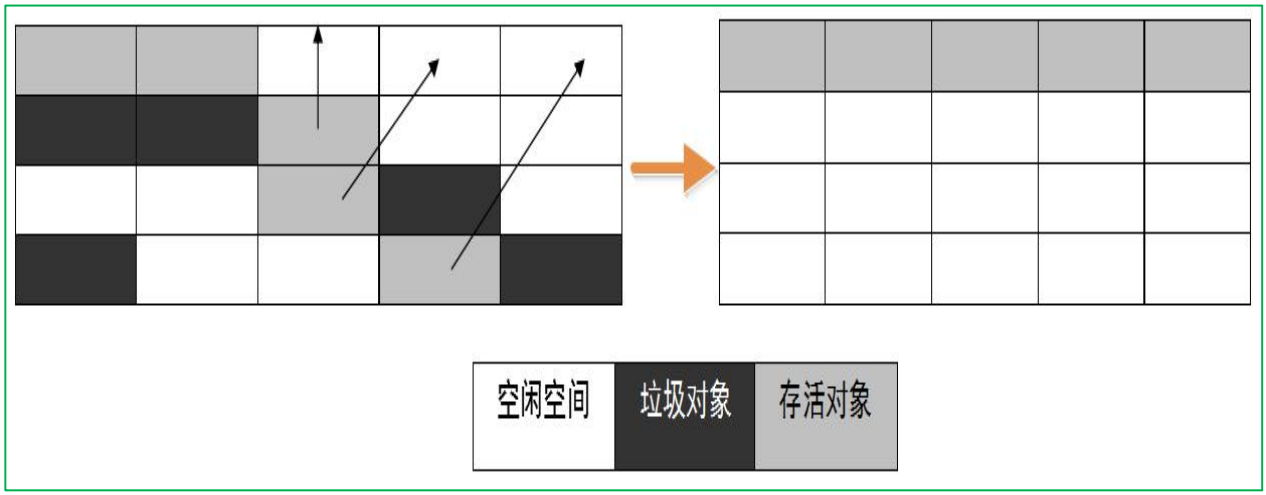
（1）**标记阶段**：通过根节点，标记所有从根节点开始的可达对象。因此，未被标记的对象就是未被引用的垃圾对象。

（2）**清除阶段**：清除所有未被标记的对象。



（三）GC 的算法——标记-压缩算法

标记-压缩算法适合用于存活对象较多的场合，如老年代。它在标记-清除算法的基础上做了一些优化。和标记-清除算法一样，标记-压缩算法也首先需要从根节点开始，对所有可达对象做一次标记。但之后，它并不简单的清理未标记的对象，而是将所有的存活对象压缩到内存的一端。之后，清理边界外所有的空间。



【引申】标记压缩对标记清除而言，有什么优势呢？

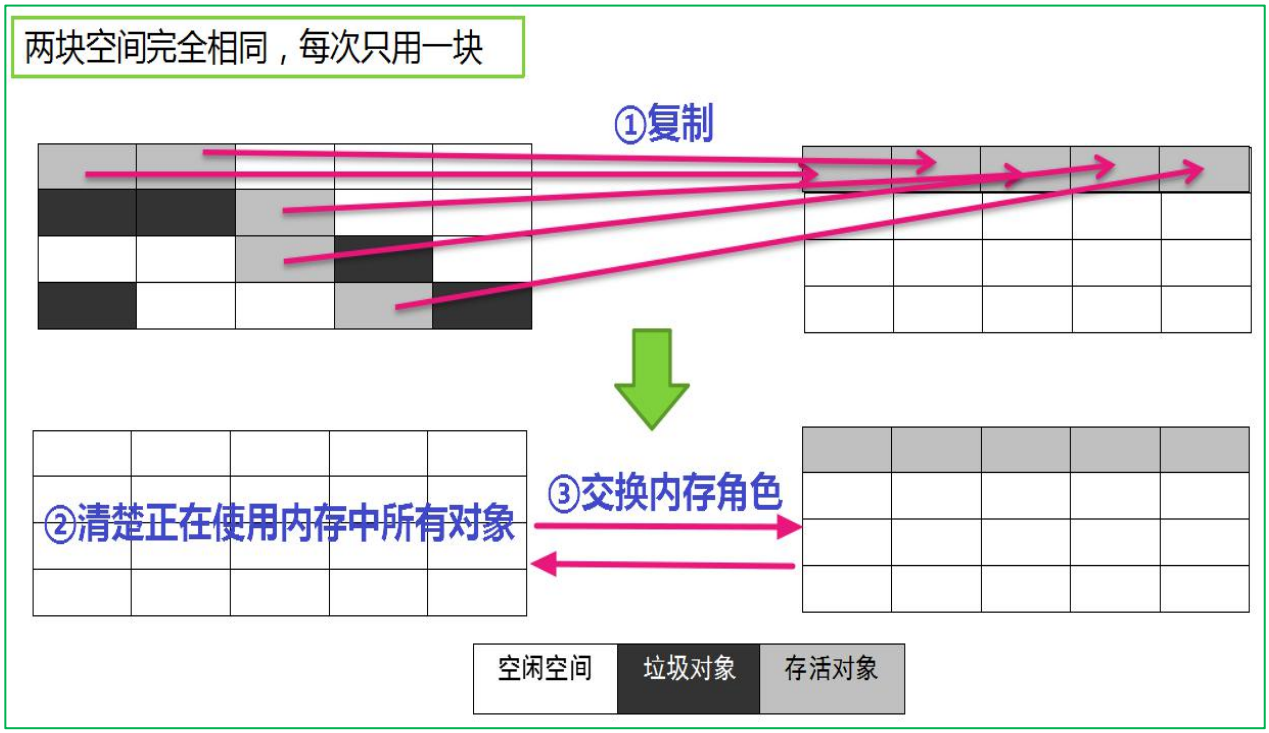
优势就是能够整理内存碎片，避免分配大对象时，空间不足导致 FullGC。

（四）GC 的算法——复制算法

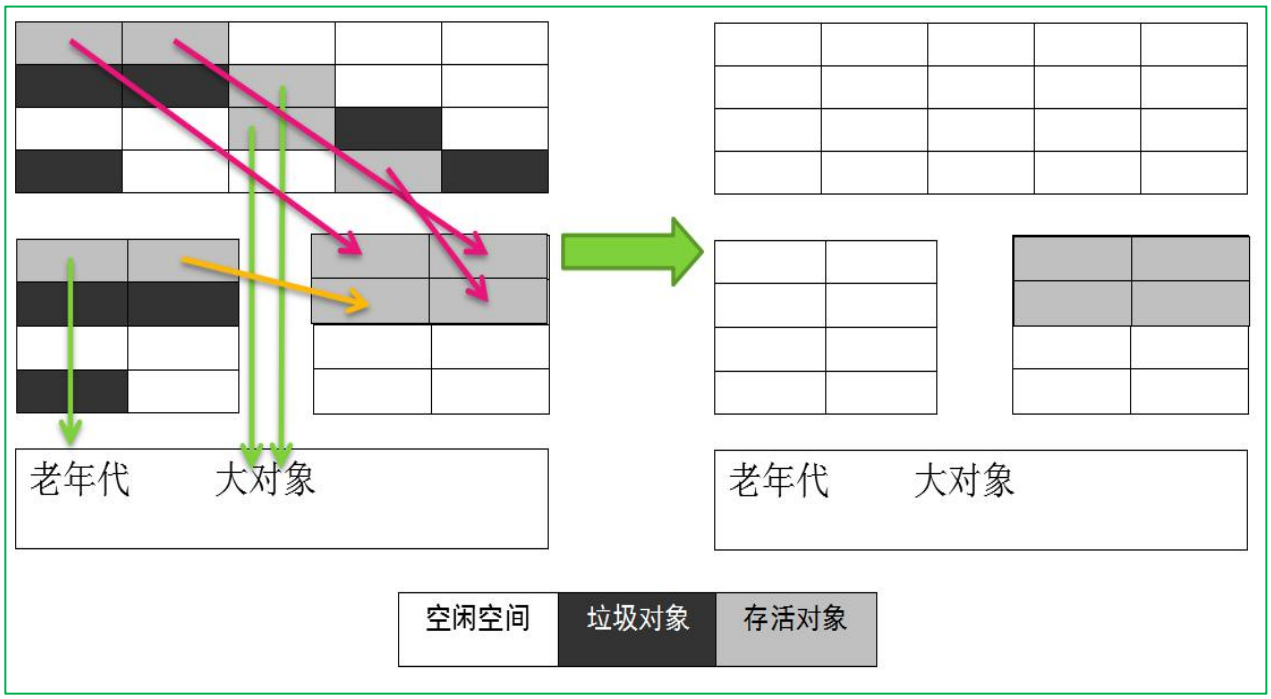
与标记-清除算法相比，复制算法是一种相对高效的回收方法；

不适用于存活对象较多的场合 如老年代；

将原有的内存空间分为两块，每次只使用其中一块，在垃圾回收时，将正在使用的内存中的存活对象复制到未使用的内存块中，之后，清除正在使用的内存块中的所有对象，交换两个内存的角色，完成垃圾回收。



复制算法的最大问题是：空间浪费！



（五）分代思想

依据对象的存活周期进行分类，短命对象归为新生代，长命对象归为老年代。

根据不同代的特点，选取合适的收集算法：

①少量对象存活，比如新年代，适合复制算法。

②大量对象存活，比如老生代，适合标记清理或者标记压缩

【注意】所有的算法，需要能够识别一个垃圾对象，因此需要给出一个可触及性的定义。

（六）可触及性

（1）**可触及的**：从根节点可以触及到这个对象。

（2）**可复活的**：一旦所有引用被释放，就是可复活状态，因为在 finalize()中可能复活该对象。

（3）**不可触及的**：在 finalize()后，可能会进入不可触及状态，不可触及的对象不可能复活，可以回收。

看如下例子：

```
package com.liyan.gcTest;
public class CanReliveObj {
    public static CanReliveObj obj;
    @Override
    protected void finalize() throws Throwable {
        super.finalize();
        System.out.println("CanReliveObj finalize called");
        obj = this;
    }
    @Override
    public String toString() {
        return "I am CanReliveObj";
    }

    public static void main(String[] args) throws InterruptedException {
        obj = new CanReliveObj();
        obj = null; // 可复活
        System.gc();
        Thread.sleep(1000);
        if (obj == null) {
            System.out.println("obj 是 null");
        } else {
            System.out.println("obj 可用");
        }
        System.out.println("第二次 gc");
        obj = null; // 不可复活
        System.gc();
        Thread.sleep(1000);
        if (obj == null) {
            System.out.println("obj 是 null");
        } else {
            System.out.println("obj 可用");
        }
    }
}
```

输出结果

```
CanReliveObj finalize called
obj 可用
第二次 gc
obj 是 null
```