原文地址：<https://www.cnblogs.com/nele/p/5673215.html>

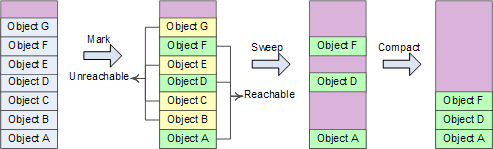
**什么是GC**

Garbage Collector（垃圾收集器）以应用程序的root为基础，遍历应用程序在Heap上动态分配的所有对象[2]，通过识别它们是否被引用来确定哪些对象是已经死亡的、哪些仍需要被使用。已经不再被应用程序的root或者别的对象所引用的对象就是已经死亡的对象，即所谓的垃圾，需要被回收。

**GC回收算法**

**Mark-Compact 标记压缩算法**

1. 标记清除阶段，先假设heap中所有对象都可以回收，然后找出不能回收的对象，给这些对象打上标记，最后heap中没有打标记的对象都是可以被回收的；
2. 压缩阶段，对象回收之后heap内存空间变得不连续，在heap中移动这些对象，使他们重新从heap基地址开始连续排列



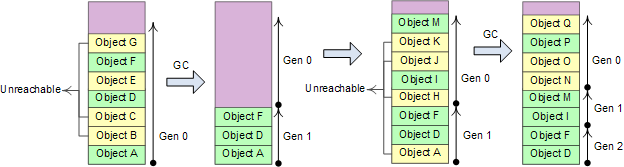
**Generational 分代算法**

程序可能使用几百M、几G的内存，对这样的内存区域进行一次GC操作成本很高

Generational 分代算法如下：

刚分配的对象大多数生命周期都很短，这些对象称为0代对象，进行一次GC之后，存活下来的对象生命周期都较长，这些对象称为1代对象，这些对象一般不会在下一个GC时间内死亡，所以这些对象的下一次GC操作是在10个GC时间后，存活下来的对象生命周期更长，这些对象称为2代对象，这些对象的下一次GC操作是在100个GC时间后

GC使用三个指针将heap分为0，1，2代



**Finalization Queue和Freachable Queue**

当某个类具有析构函数时，其对象被生成时，Finalization Queue队列中会添加一个指向对象的指针，当对象要被回收时，如果Finalization Queue存在对象的指针，那么对象不会被回收，Freachable Queue会添加对象的指针，在某个时刻，Freachable Queue队列的对象的析构函数会被调用，这时对象会被回收

GC方法：

ReRegisterForFinalize：将对象注册到Finalization Queue中

SuppressFinalize：禁止对象的Finalize方法（告诉GC回收时不必调用其析构方法了）

**值类型与引用类型**

.NET中的所有类型都是（直接或间接）从System.Object类型派生的。

CTS中的类型被分成两大类

引用类型：分配在内存堆上，如类，数组，String等

值类型：分配在堆栈上，如int，char，指针等

**托管资源与非托管资源**

**托管资源**

.NET中超过80%的资源都是托管资源，我们自定义的类都是托管资源，当某个托管资源不被任何其它资源引用时，GC会自动回收

**非托管资源**

GC无法回收系统资源，需要程序自己释放，如：

ApplicationContext, Context, FileStream, OdbcDataReader, OleDBDataReader, Regex, Socket, 文件句柄, GDI资源, 数据库连接等等资源

**非托管资源释放**

非托管资源释放的两种方式：

1. 手动释放：手动释放需要释放自身引用的非托管资源和所引用类引用的非托管资源，手动释放一般继承IDisposable（目的是我们可以使用using）
2. GC回收释放：在GC回收时释放，GC释放只需释放自身引用的非托管资源（因为所引用的类也会被GC回收，不需要帮其他类释放），需要实现析构函数

我们一般将手动释放和GC回收释放结合，当手动释放时会释放资源，当我们没有手动释放时，GC回收时也会释放

示例代码：

public class ResourceHolder : IDisposable

{

private bool isDisposabled = false;

~ResourceHolder()

{

// 释放自身资源

Dispose(false);

}

public void Dispose()

{

// 释放所有资源

Dispose(true);

}

/// <summary>

/// 释放资源

/// </summary>

/// <param name="disposing"></param>

protected virtual void Dispose(bool disposing)

{

if (isDisposabled == true)

return;

// 释放自身的资源

isDisposabled = true;

if (disposing == false)

return;

// 释放所引用的对象的资源（调用所有引用对象的 Dispose()）

}

}

**GC.Collect() 方法**

作用：强制进行垃圾回收。

|  |  |
| --- | --- |
| 名称 | 说明 |
| Collect() | 强制对所有代进行即时垃圾回收。 |
| Collect(Int32) | 强制对零代到指定代进行即时垃圾回收。 |
| Collect(Int32, GCCollectionMode) | 强制在 GCCollectionMode 值所指定的时间对零代到指定代进行垃圾回收 |