用户会通过内存分配器在堆上申请内存，而垃圾回收器负责回收堆上的内存空间，共同管理着程序中的堆内存空间。

golang的gc，从最初的标记-清除，到v1.5实现三色标记并发gc。

1.1标记-清除算法，先标记所有需要回收的对象，标记完成后，统一回收掉所有被标记的对象。通过可达性分析，从GC Roots，根据引用关系向下搜索。根节点root主要指的是全局变量、各个G的栈上的变量。缺点：是保证GC期间，标记对象的状态不能变化，需要stop the world。清除会产生大量不连续的内存碎片，当需要分配大对象时，不得不提前触发另一次GC

1.2 三色标记法

白色（尚未被访问，结束仍白，表示不可达），黑色（本身和所有引用都已经被扫描），灰色（引用没有被扫描）。程序和收集器是并发工作的，存活的对象可能被标记死亡，悬挂指针。黑色部分引用了新对象

1.3 写屏障

Go支持并行GC，屏障会记录第一次扫描时每个对象的状态

1.4 辅助GC