### 实验要求

（1）动态分区管理的主存分配模拟系统的设计—最先适应法、最佳适应法、最坏适应法（选择1~3种），模拟实现内存的分配回收；

（2）能够输入给定的内存大小，进程的个数，每个进程所需内存空间的大小等；

（3）能够选择分配或回收操作；并能显示完成内存分配或回收后内存空间的使用情况；

（4）能够显示进程在内存的存储地址、大小等。

### 二．实验过程

#### （1）类

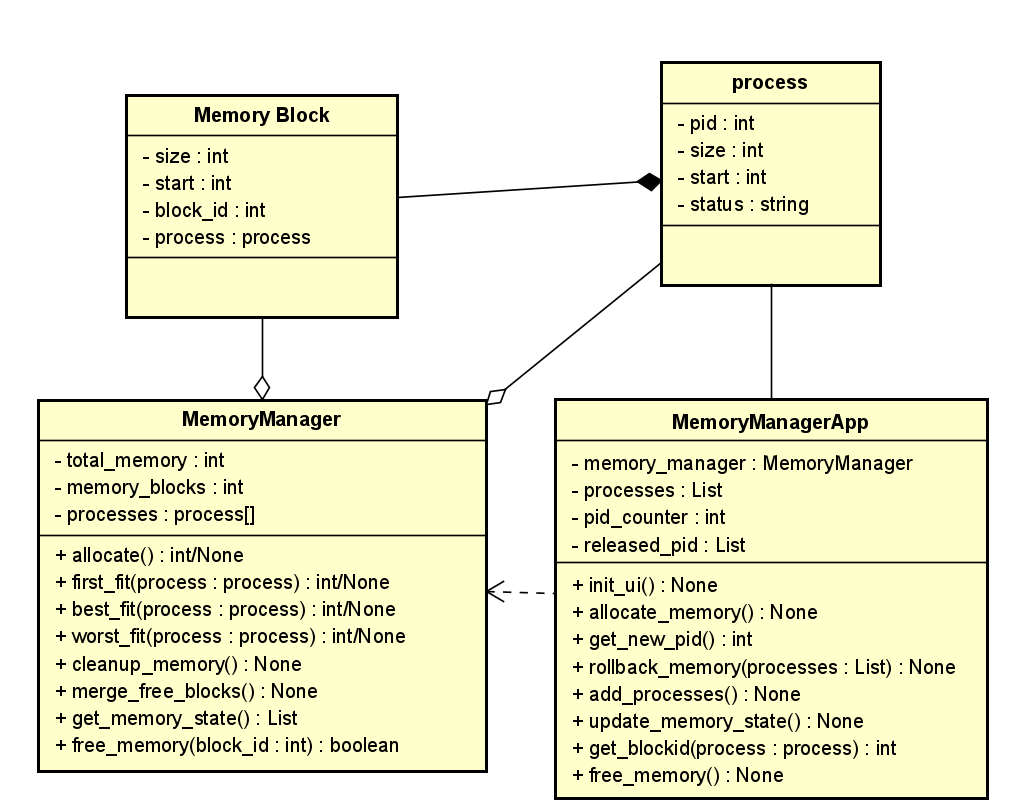


图1 类图

在本次实验实现过程中，我创建了四个主要的类：MemoryManagerApp、MemoryManager、Process 和 MemoryBlock，共同构成了内存管理系统的核心功能模块。

MemoryManagerApp 类是应用程序的主要入口，负责初始化用户界面、处理用户输入以及与内存管理系统的交互。它通过一个 MemoryManager 实例来管理内存的分配和回收，并且负责跟踪当前的进程和内存状态。

MemoryManager 类负责具体的内存管理工作，提供内存分配和回收的功能，包含一个 memory\_blocks 属性，用于存储所有内存块，并通过不同的内存分配策略来执行内存分配操作。每当内存被分配或释放时，MemoryManager 会更新内存状态，并提供方法来查询当前的内存使用情况。

Process 类代表一个正在运行的进程，它有一个唯一的 pid，以及需要分配的内存大小。Process 类还维护着与之关联的内存块，当一个进程结束时，相关的内存块将会被释放回内存池。

MemoryBlock 类表示一个内存块，它具有 block\_id，start，size等属性。每个 MemoryBlock 实例要么处于空闲状态，要么已经被某个 Process 占用，内存块的状态会随进程的内存需求变化而更新。

#### （2）三种分配方法

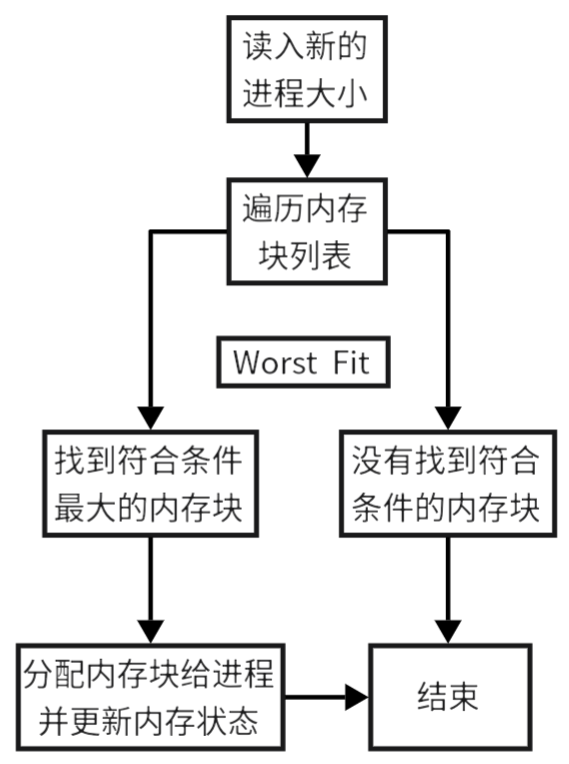
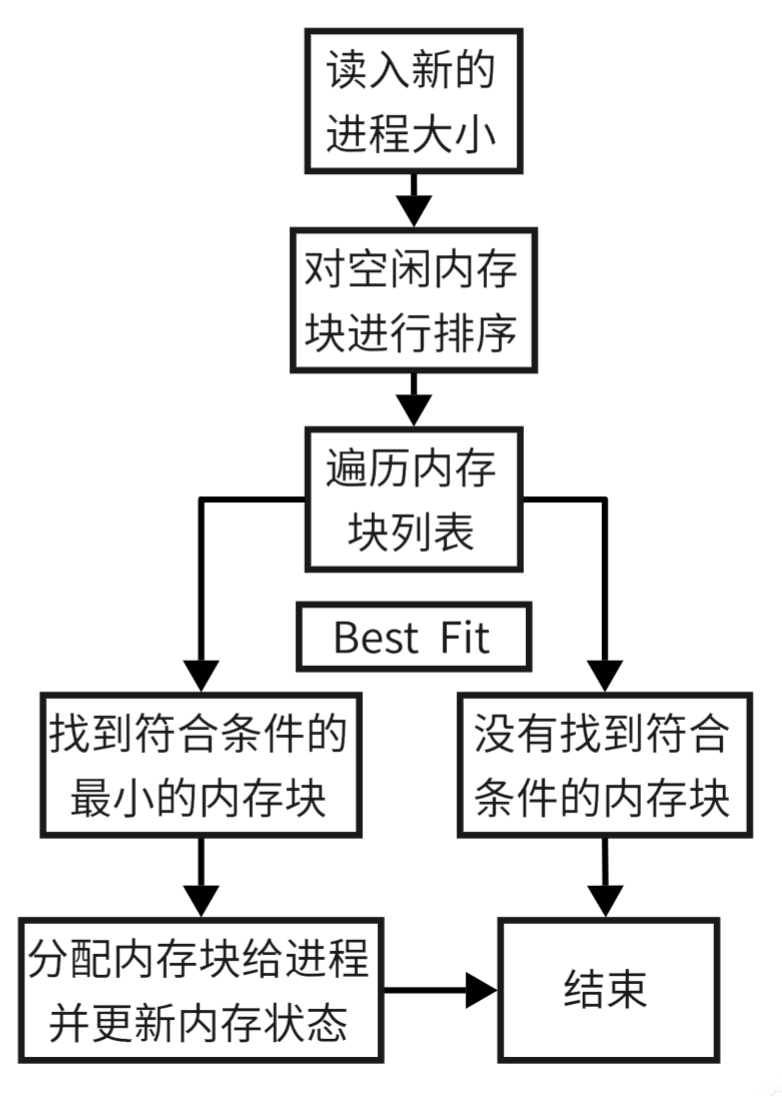
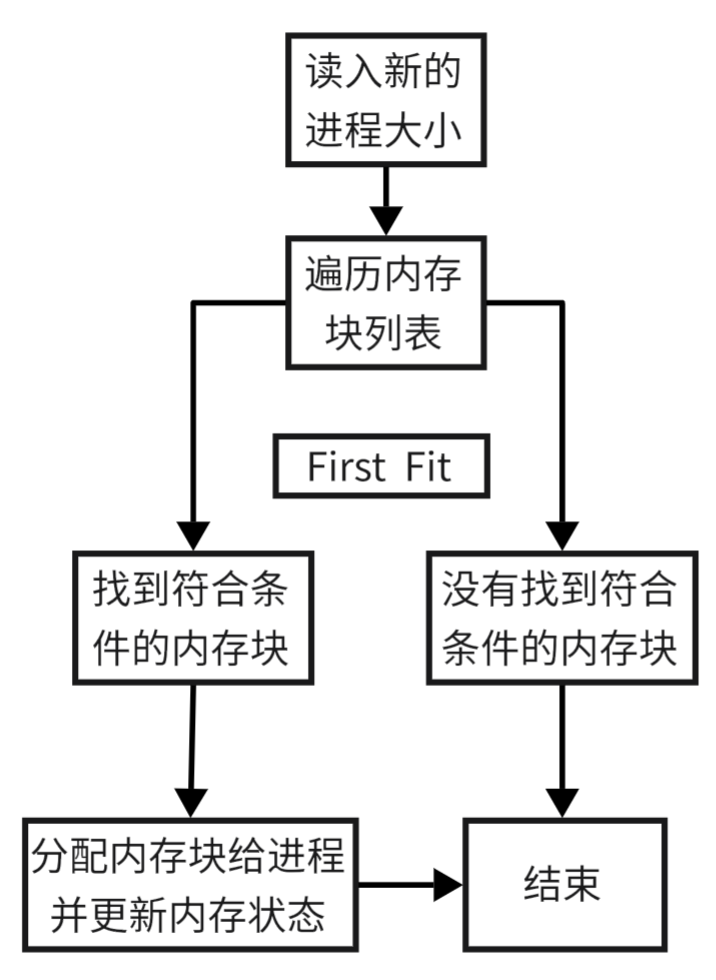


图2 三种分配方法

最先适应法从内存的开始位置依次遍历内存块，找到第一个满足进程内存需求的空闲内存块，并将其分配给进程。如果找到合适的内存块，则立即分配；如果没有合适的内存块，分配失败。

最佳适应法首先对所有的空闲内存块按照大小进行排序，选择最小的一个且能够满足进程需求的内存块进行分配。

最坏适应法选择最大的空闲内存块进行分配，将进程放置在剩余空间最大的内存块中。

#### 主要函数实现

**allocate\_memory()**

用于处理内存分配的逻辑。用户输入总内存大小、进程数量及各个进程的内存需求后，该函数会检查是否有足够的内存可供分配。如果内存足够，它会创建进程对象并使用选定的分配策略进行内存分配。如果分配成功，内存状态将更新；如果分配失败，所有操作将回滚。

**add\_processes()**

用于添加新的进程。当用户输入新的进程内存需求时，系统会首先检查当前剩余的空闲内存是否足够。若足够，则创建新进程并使用选定的分配策略进行内存分配。如果内存不足或分配失败，将回滚新增进程的分配，并打印错误信息。

**rollback\_memory()**

用于回滚分配失败的进程。当内存分配操作失败时，rollback\_memory()会释放已经分配的内存，并回收被分配给进程的内存块。同时，它还会将已回收的进程编号放入一个回收列表，以便下次分配时可以重用这些编号。

**update\_memory\_state()**

用于更新内存的状态，并将当前的内存状态显示在用户界面中。它会遍历内存管理器的内存块，检查每个内存块是否被分配给某个进程。如果是，它会更新表格中显示的进程编号、内存起始地址和大小。如果是空闲内存块，则显示空闲状态。

**get\_block\_id\_by\_process()**

通过进程的起始地址来查找对应的内存块ID。它遍历内存管理器中的所有内存块，寻找与进程对应的内存块，并返回该内存块的ID，以便进行后续的内存释放操作。

**free\_memory()**

用于回收用户指定进程的内存。用户输入进程编号后，系统会查找对应的进程，并释放其所占用的内存块。如果该进程存在并成功释放内存，内存状态将更新，并删除进程记录；如果输入无效，则提示错误信息。

#### （4）UI界面

#### 

图3 界面原型与成果展示

本系统的UI界面是采用PyQt5完成的，包含三个主要部分：内存与进程配置区、操作区和内存状态表格。在配置区，用户可以输入内存总大小、进程数量以及每个进程的内存需求，并通过“分配内存”按钮启动内存分配操作。操作区提供了回收内存、新增进程和选择内存分配策略（包括最先适应法、最佳适应法、最坏适应法）的功能，用户可通过输入框和按钮进行操作。内存状态表格实时展示当前内存分配情况，显示每个内存块的编号、起始地址、大小和分配状态，帮助用户清晰了解内存的使用情况。

### 三．实验小结

本次实验是基于课程中的内存分配内容展开的。我实现了一个动态内存分配模拟系统，采用了最先适应法、最佳适应法和最坏适应法来模拟内存的分配与回收。在这个过程中，我更好地理解了内存管理算法的工作原理，并把它们应用到实际的内存分配操作中。

首先，我用PyQt5设计了一个简单易用的界面，用户可以输入总内存、进程数和每个进程所需的内存空间等参数来配置系统。每种分配方法都有对应的按钮，用户可以根据自己的需求选择不同的分配策略。系统也允许用户回收内存，进行内存的动态管理。每次分配或回收内存后，内存的状态会实时显示在表格中，用户可以看到每个内存块的状态、进程在内存中的起始地址和大小等信息。

通过这次实验，我深刻体会到内存管理的重要性和复杂性，让我更好地理解了操作系统中内存分配算法的实现，也加深了对内存管理基本原理的理解。