**Memory Management Assessment**

**Question 4**

**Team member: BoNan Mai, HuaYi Lai, RuiMing Zhu**

# What is External Fragmentation

## Defination

When allocating memory, due to the varying duration of each process, it is likely that the previous process has already released memory while the next process has not. As a result, there may be a small remaining memory space between two processes that has been freed but is insufficient to allocate to the next process. This smaller memory space is known as external fragmentation. (Shown in Figure 1)

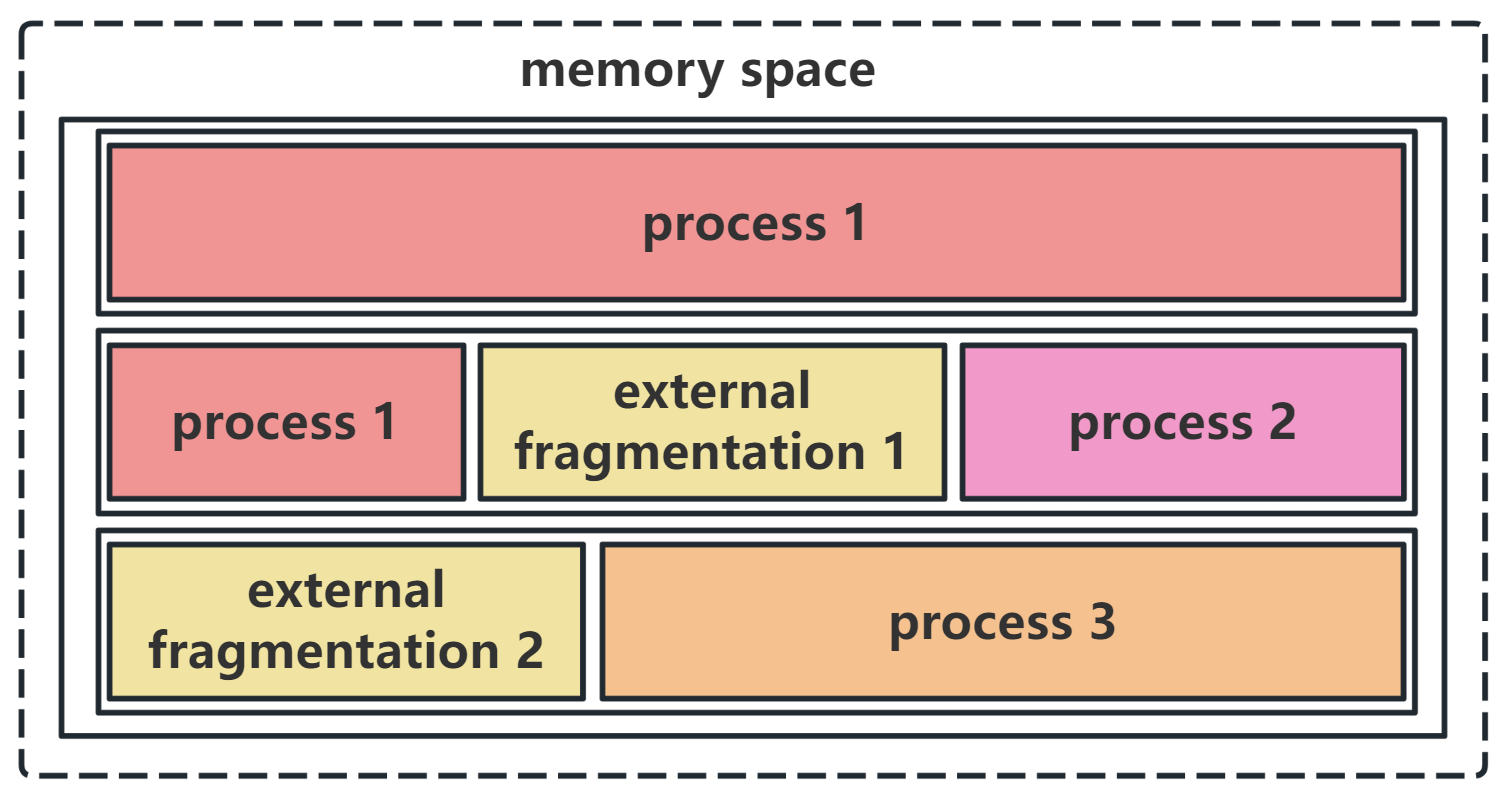


Figure 1: Example of External Fragmentation

In the Figure 1, there is external fragmentation between process 1 and process 2, and this fragmentation may be created because there was a process 4 just now. If the process 4 free the memory before process 1 and process 2 free the memory, then there is a external fragmentation between the process 1 and process 2.

# The solution to eliminate external fragmentation

## Move the process in the memory block(What we have done)

The algorithm we design to pass the test is to move the process in the memory block. When the memory allocator is unable to allocate sufficient memory to a process, it initiates memory compaction. The compaction process involves traversing the entire memory space. For each non-empty memory block encountered, it is released and reallocated. This effectively "moves" the process to the front, eliminating external fragmentation. Once the compaction process is completed, all fragments are eliminated, creating larger contiguous space to allocate to subsequent processes. The code is as follow(figure 2):

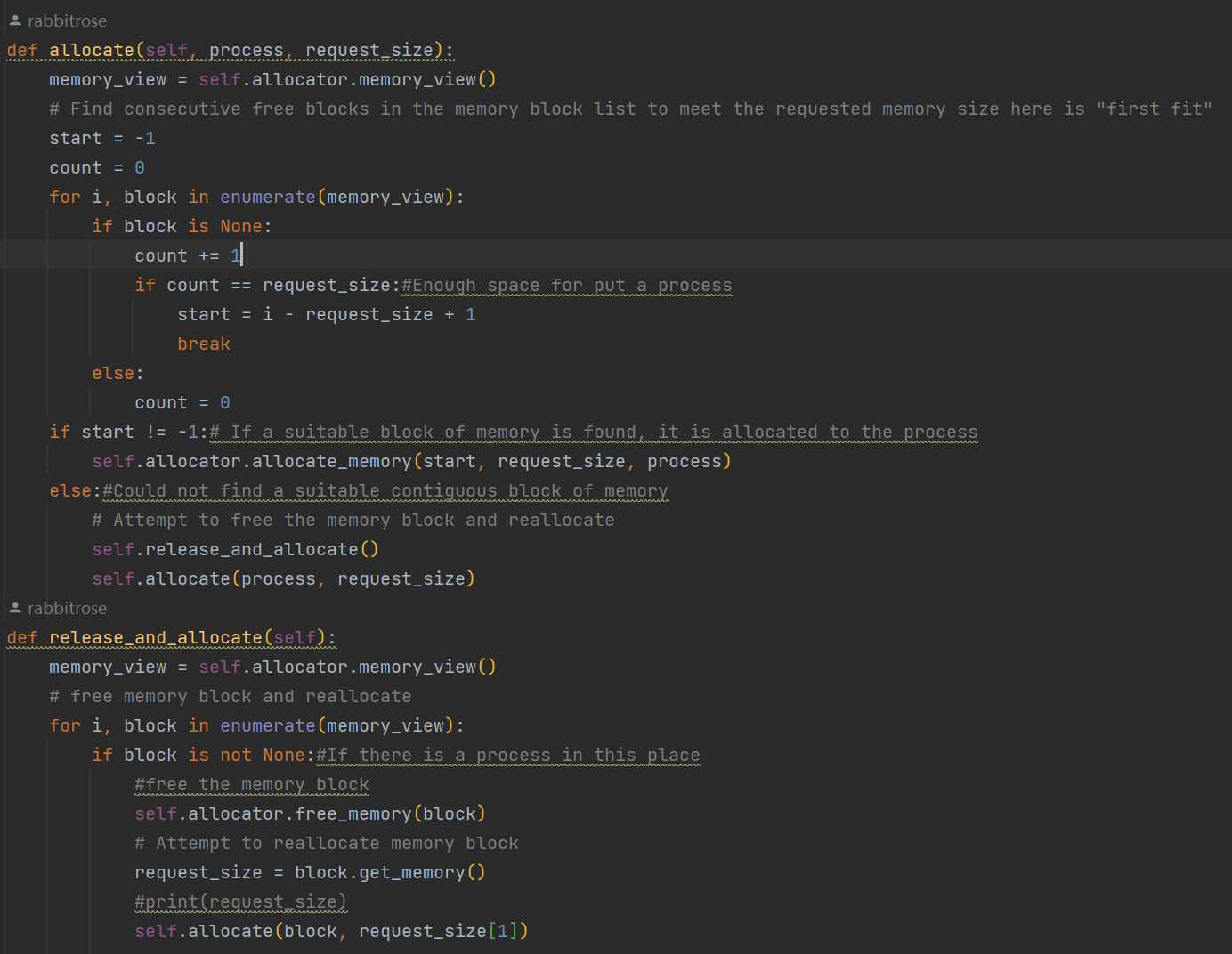


Figure 2: the algorithm that we use

## Move and combine fragmentation to a big memory block

This solution my conjecture which is different from the **solution 2.1,** the first conjecture is that if we can not move the process in the memory block, then we can move the free block in the memory and combine the fragmentation to a big memory block. The steps are as follow:

The first step is to find and collect all the fragmentation. In **Figure 3**

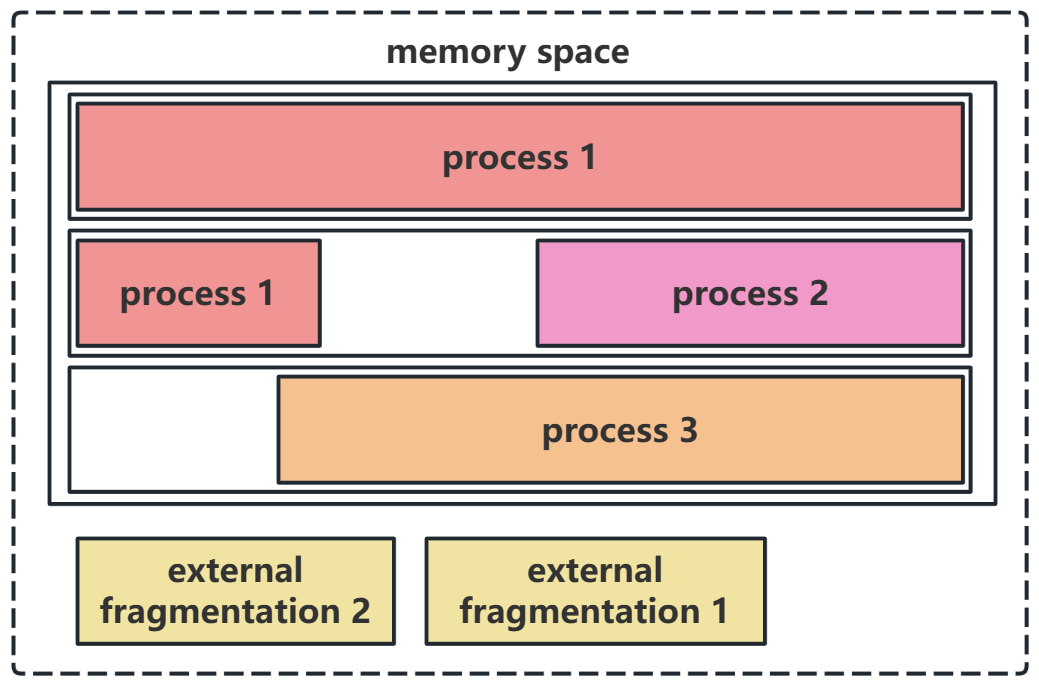


Figure 3: Find all the fragmentation

Then we combine the fragmentation we find, and we get a huge free memory. Just like **Figure 4.**

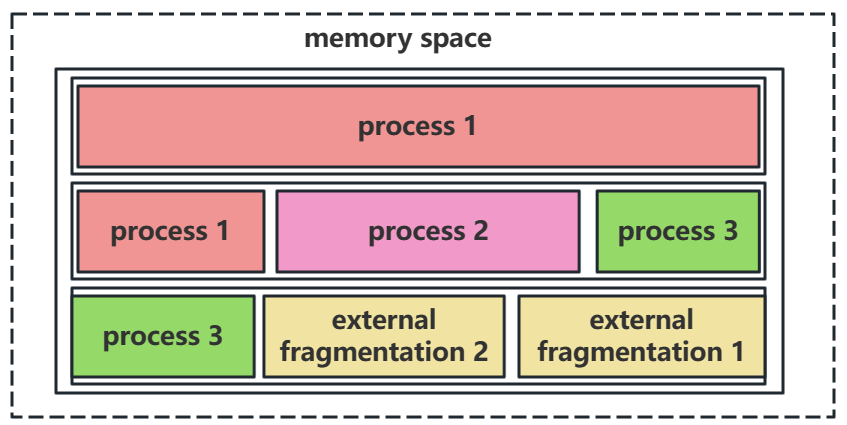


Figure 4: combine the fragmentation

This solution can not be achieve because I find that the allocator API do not have the function to move the fragmentation in the memory. And the API to free the memory is based on the first address, so even the allocator have the API to find and combine the fragmentation in the memory, this solution still can not be achieved. And this is only my conjecture, maybe it can not be achieve in fact.