**Memory Management Assessment**

**Question 2**

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

# Overall design and structure of the algorithm

**(1)Design and Structure**

Common memory allocation algorithms mainly include: **First Fit**, **Next Fit**, **Best Fit** and **Worst Fit**. Through algorithm design and experiments, we found that the above four algorithms cannot solve all memory allocation problems when dealing with the problem. There will be some processes that cannot be allocated. We will further describe the specific failure cases in Part3. Based on the above situation, we combined the basic principles of the First Fit algorithm and designed a set of algorithms to solve the above problems. We call our algorithm ***“dynamic allocation algorithm”.***

There are Two parts of our algorithm. They are the **First Fit framework function** and the **process reallocation function**. Here is a figure to show the relationships between these two functions(Figure 1)

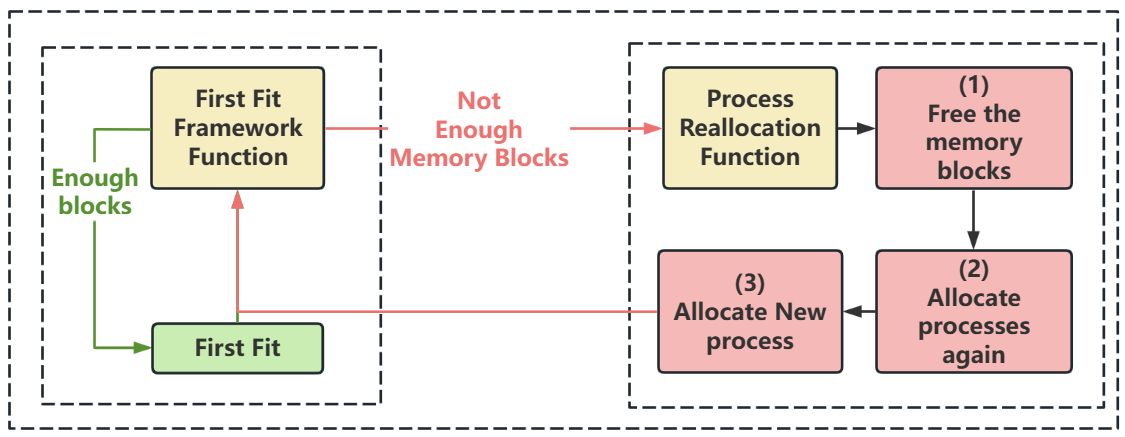


Figure 1:Overall design of the algorithm

First of all, as shown in the left part of Figure 1, we let the memory be allocated according to the First Fit method. Until the memory capacity is full and cannot accommodate more processes, we adopt the idea of compressing memory fragments, as shown in the right half of the above figure, by reallocating memory in the way of First Fit to provide more capacity for the remaining memory. Makes already compressed parts free of external fragmentation. Finally achieve a reasonable allocation of memory.

**(2)Instance Example**

May be it is not clear to introduce the algorithm by words, so we would give an example In the following groughs.

Firstly, if we have **4 processes** and the **memory size is 12**. Just like the table in Figure 2, here are some detailed information about these processes

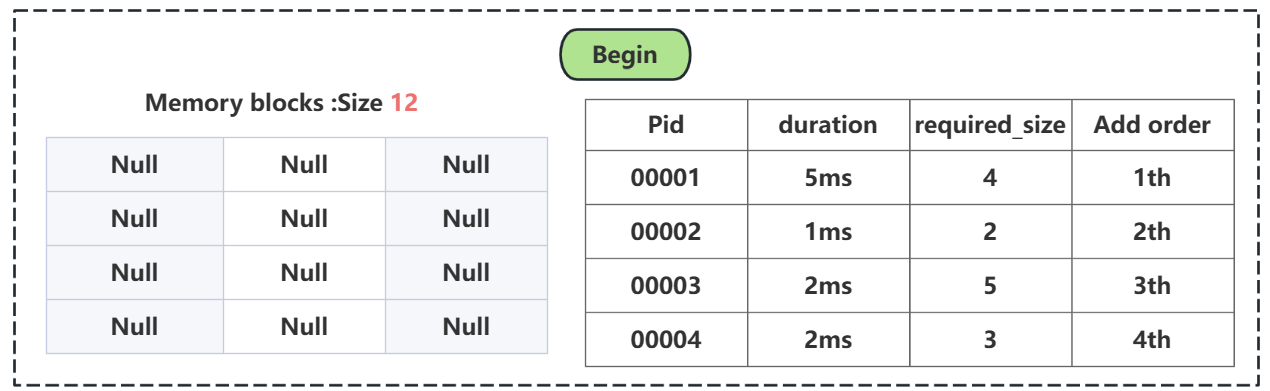


Figure 2:Begin state of the situation

Then, in the first millisecond, The process00001 is added in the memory blocks According to **First Fit**, just like the figure(Figure 3) in below:

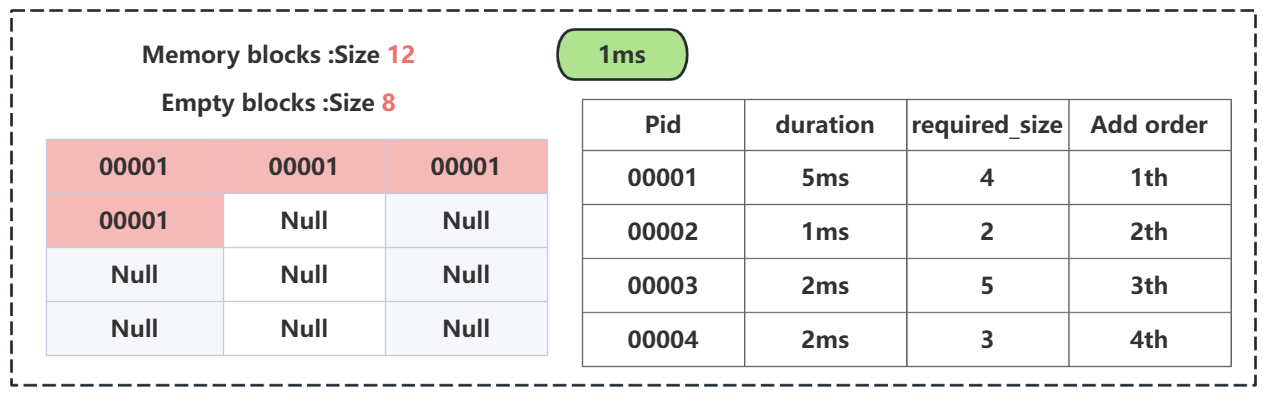


Figure 3: Add the first process

Then in the second millisecond, The process00002 is added in the memory blocks According to the **First Fit**, just like the figure(Figure 4) in below:

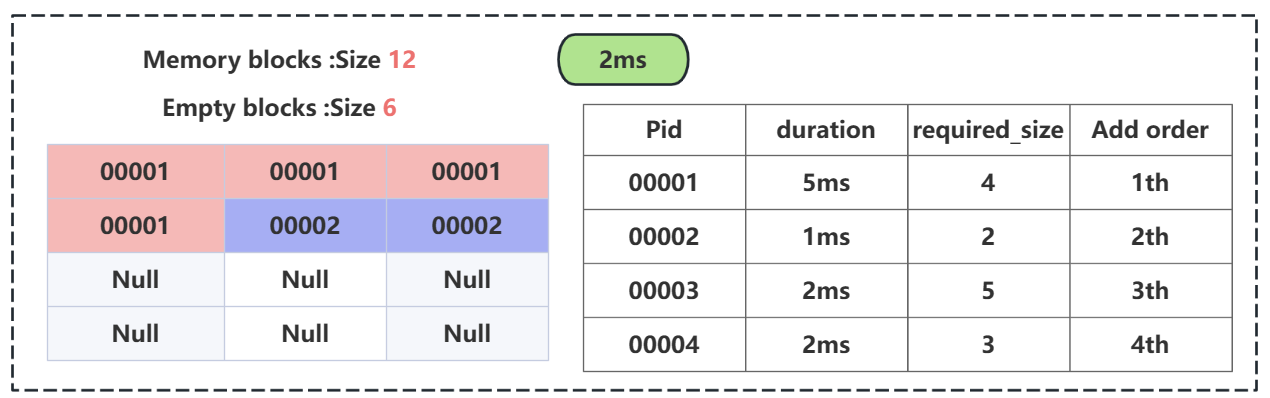


Figure 4: Add the second process

Then in the third millisecond, The process00003 is added in the memory blocks According to the **First Fit,** just like the figure(Figure 5) in below

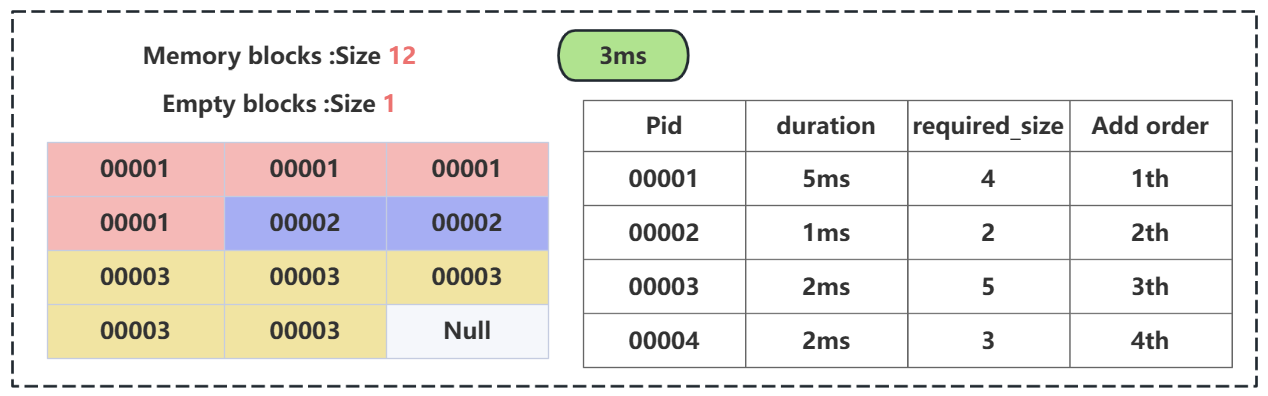


Figure 5: Add the third process

Finally, when we want to add process00004 we found that it is impossible to add it according to **First Fit**, but at the fourth millisecond, the process00002 should be free from the memory block, so although we have 3 Null space, we could not put process00004 in the memory block again. Just like the Figure show in below(Figure 6).

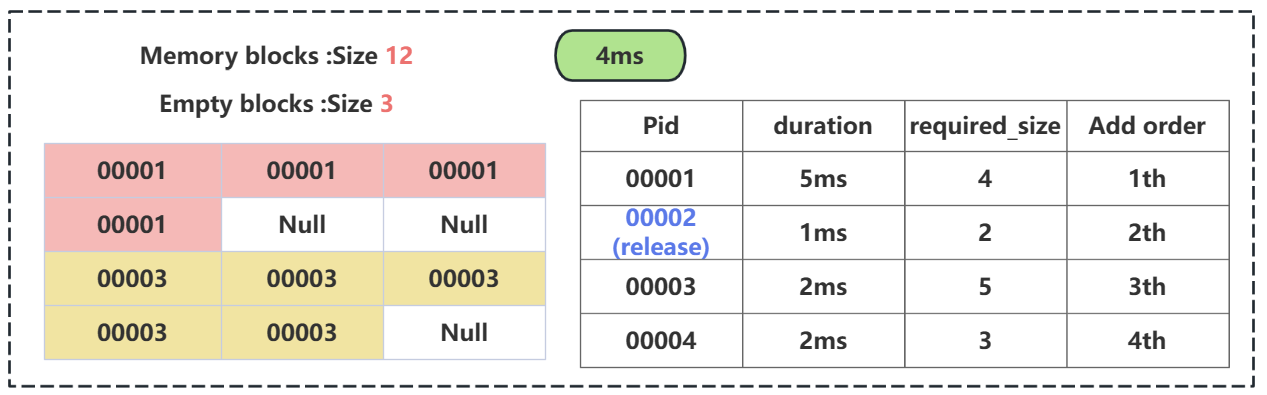


Figure 6: the first step of our algorithm

So according to our **“dynamic allocation algorithm”**, we get process00001 and process00003 out of the memory together, then, we let them **do the First Fit again(shown in Figure 7)**, so there will be 3 continuous blocks in the memory block(shown in Figure 8).

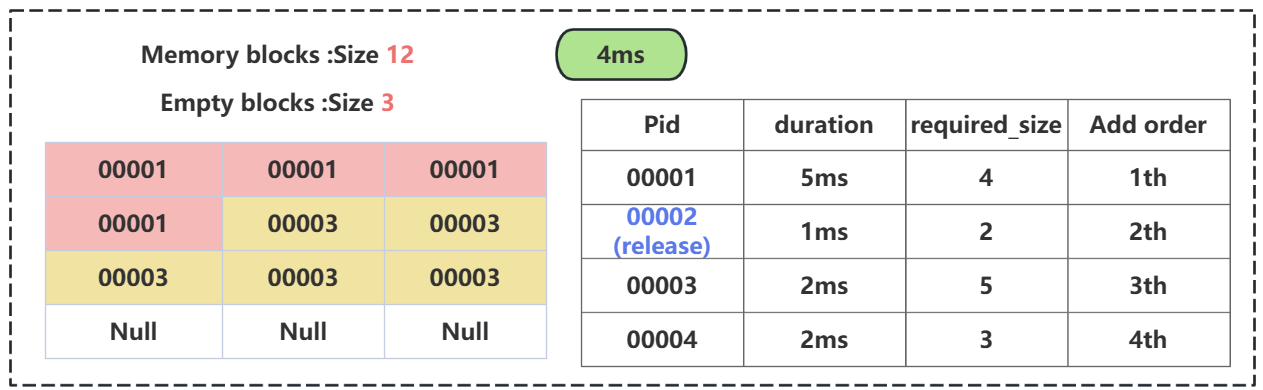


Figure 7: the second step of our algorithm

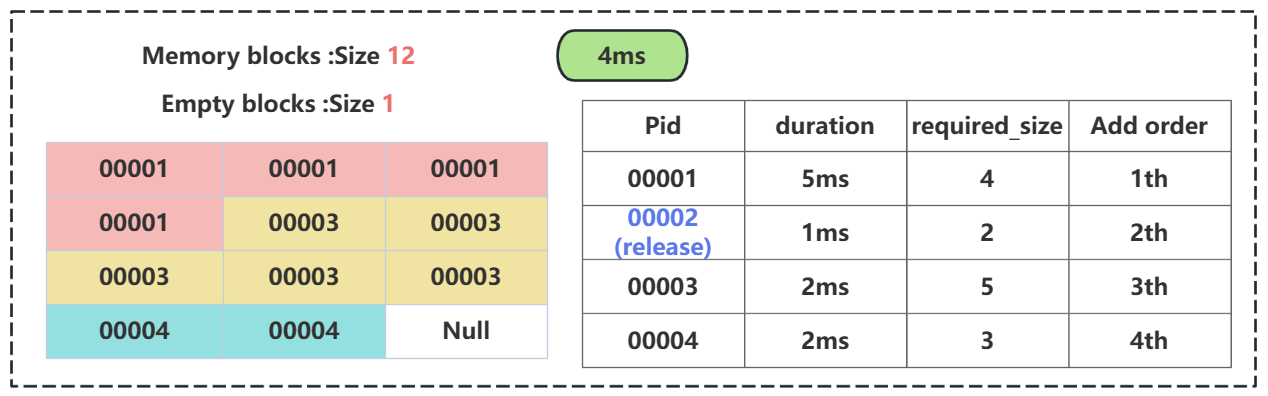


Figure 8: the final step of our algorithm

# Implementation of the design in Python

Here is the code that we use to implement our algorithm(Figure 9):

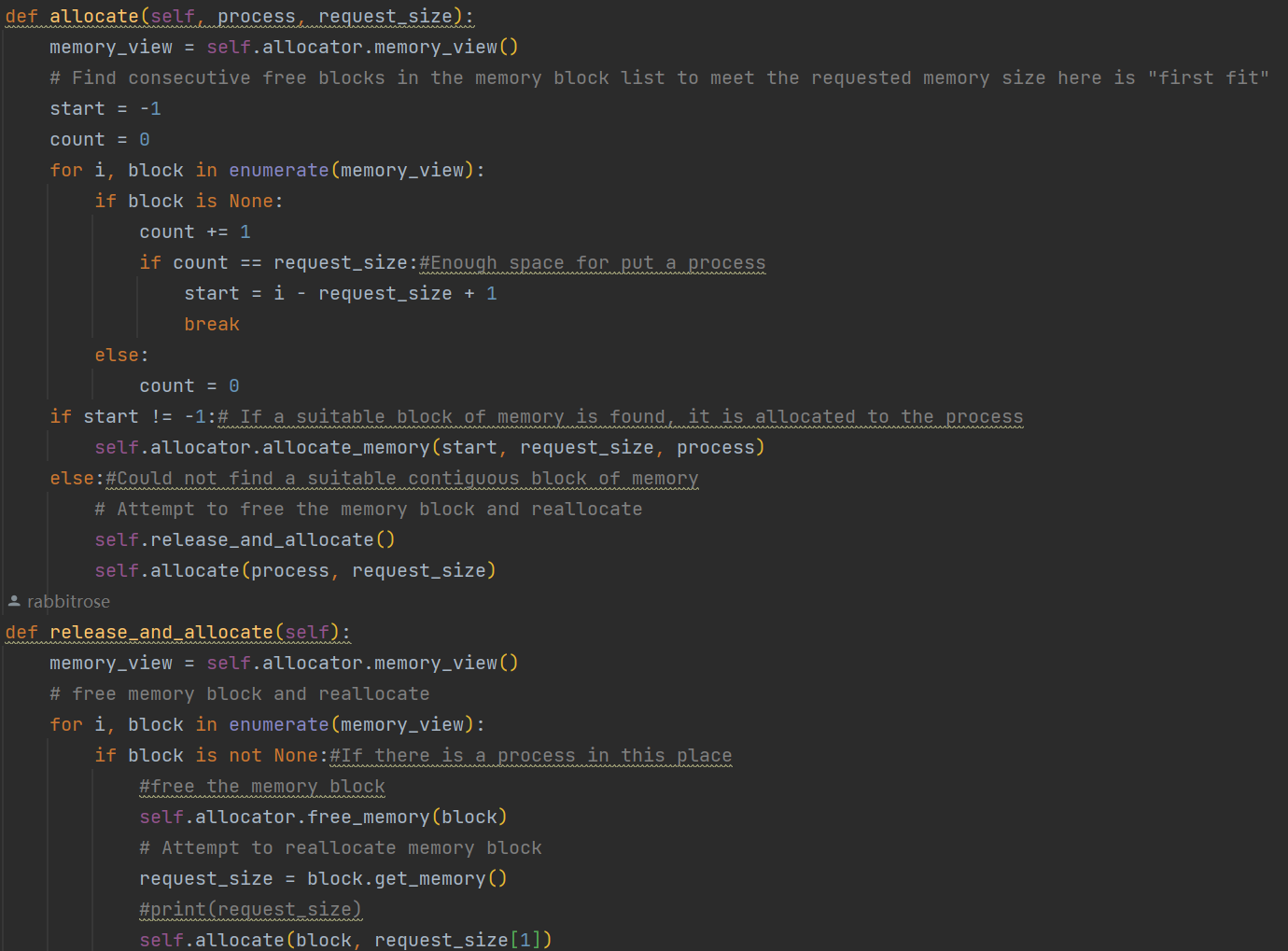


Figure 9: the code that we use to achieve the algorithm

In order to realize the functions I mentioned above, we have improved on the basis of the First Fit algorithm. Every time there is a new process waiting to be added to the memory block, we perform a fragment compression (as shown in the function **release\_and\_allocate**) to ensure that Each process joins with no external fragments ahead

# Challenges or obstacles that we encounter and solutions

## Difficult to find a perfect algorithm to solve this problem

We have tested the ‘First Fit’, ‘Best Fit’, ‘Worst Fit’ and ‘Next Fit’. Those only get **26 to 35 points** in this project. We found that those algorithms could not **solve the external fragmentation problem** perfectly, so we decided to design an algorithm to solve the problem, finally, we make design our algorithm successfully

## Lacking of API

To begin with, we want to operate the external fragmentations and concentrate them together, but after long times trying, we are failed because we lack the relevant functions to do it, finally we change our minds to move the process to achieve our goals