Hanoi University of Science and Technology
Data Science and Artificial Intelligence
Class 136315 - IT3070E - Operating system
Responsible: Prof. Do Quoc Huy
Examiners: Prof. Do Quoc Huy

Page Replacement Algorithm

Nguyễn Hùng Anh - 20200028
Nguyễn Hữu Tuấn Duy - 20204907
Phan Thái Việt- 20204895
Hoàng Gia Nguyên - 20204889

Monday 13th February, 2023

-

Contents

1 Abstract

The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in. In this project, we built a program in Java and JavaFX to visualize various page replacement algorithms.
Keyword: Page replacement, OS, Java

2 Paging in Operating System

2.1 Overview

Paging is a memory management scheme used by operating systems to manage and allocate memory in a computer system. In a paging system, the memory is divided into fixed-size blocks called pages, and these pages are assigned to processes as needed. Each process has its own page table, which maps the virtual addresses used by the process to the physical addresses in memory. . The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as paging. The basic purpose of paging is to separate each procedure into pages. Additionally, frames will be used to split the main memory. This scheme permits the physical address space of a process to be non – contiguous. Paging allows the operating system to efficiently allocate memory and manage multiple processes running on a computer system.

2.2 Page fault

In computer systems that utilize virtual memory, a page fault occurs when a program attempts to access a memory page that is currently mapped into the virtual address space but not currently loaded in physical memory. Given that the size of virtual memory is typically much larger than the physical memory, page faults can occur frequently during program execution. In response to a page fault, the operating system is responsible for locating an available physical page frame to load the missing page and update the corresponding page table. If there is no available physical memory, the operating system may need to select a currently loaded page to replace with the newly needed page. Various page replacement algorithms have been developed to determine which page to evict in this scenario, each with their own set of advantages and disadvantages. The primary objective of page replacement algorithms is to minimize the number of page faults that occur, thus improving the performance of the overall system.

3 Page replacement algorithms in operating system

3.1 First in first out (FIFO)

The first-in-first-out (FIFO) page replacement algorithm is a straightforward approach that involves maintaining a queue of all pages in physical memory, where the oldest page is located at the front of the queue. When a page fault occurs and the operating system needs to replace a page, the page at the front of the queue is selected for removal. This algorithm requires minimal overhead, as it only requires tracking the order in which pages were loaded into memory. However, it suffers from several limitations, including a lack of consideration for the frequency of page access or the importance of the page. The FIFO algorithm can lead to poor performance in scenarios where the oldest pages in memory are frequently accessed or hold critical system data. Nonetheless, it is still used in some systems due to its simplicity and low overhead. Below is a simple example of FIFO:

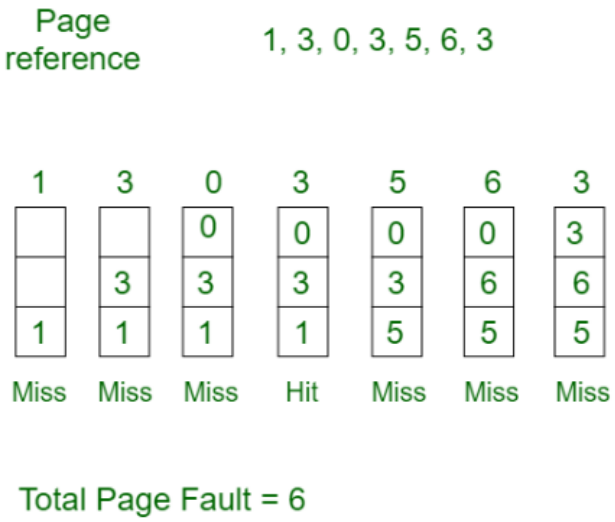


Figure 1: FIFO example

Initially, all slots are empty, so when ’1’, ’3’, ’0’ came they are allocated to the empty slots —> 3 Page faults.
When ’3’ comes, it is already in memory so —> 0 Page faults.
Then ’5’ comes, it is not available in memory so it replaces the oldest page slot i.e 1. —> 1 Page fault.
’6’ comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —> 1 Page fault.
Finally, when ’3’ come it is not available so it replaces ’0’ -> 1 page fault for a total of 6 page faults.

3.2 Least Recently Used (LRU)

The least recently used page replacement algorithm is a popular approach used in modern operating systems to select a page for eviction in response to a page fault. This algorithm works by maintaining a record of the time at which each page in physical memory was last accessed. When a page fault occurs, the operating system examines the time stamps of all pages in memory and selects the page that was least recently accessed for replacement. This approach is based on the principle that pages that have not been accessed recently are less likely to be accessed again in the near future, and thus can be safely removed. The LRU algorithm is generally considered to be an effective approach, as it aims to minimize the number of page faults by evicting the pages that are least likely to be needed in the near future. However, it can be computationally expensive to keep track of the time of every memory access for every page in memory, and special hardware support or efficient software implementations are often required to minimize the overhead.

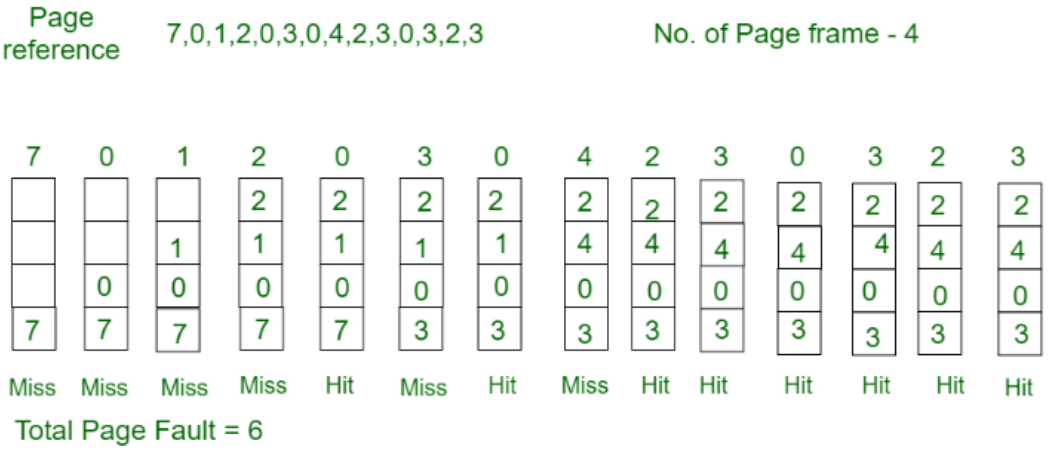


Figure 2: LRU example

Initially, all slots are empty, so when ’7’ ’0’ ’1’ ’2’ are allocated to the empty slots —> 4 Page faults.
’0’ is already in memory —> 0 Page fault.
When ’3’ came it will take the place of 7 because it is least recently used page —> 1 Page fault.
’0’ is already in memory —> 0 Page fault.
’4’ will takes place of 1 —> 1 Page Fault.
The further page reference string —> 0 Page fault because they are already available in the memory. That gives us a total of 6 page fault.

3.3 Optimal Paging Replacement (OPR)

The optimal page replacement algorithm is an idealized approach to page replacement in which the page that will not be used for the longest time in the future is selected for eviction. This

algorithm is considered optimal because it always selects the page that will result in the fewest number of page faults, but it requires perfect knowledge of the future page access pattern, which is generally impossible to obtain in real-world scenarios. Due to its idealized nature, the optimal algorithm is mainly used as a benchmark to compare the performance of other page replacement algorithms.

In practice, no algorithm can achieve the performance of the optimal algorithm, but several practical algorithms have been developed that attempt to approximate the optimal algorithm.

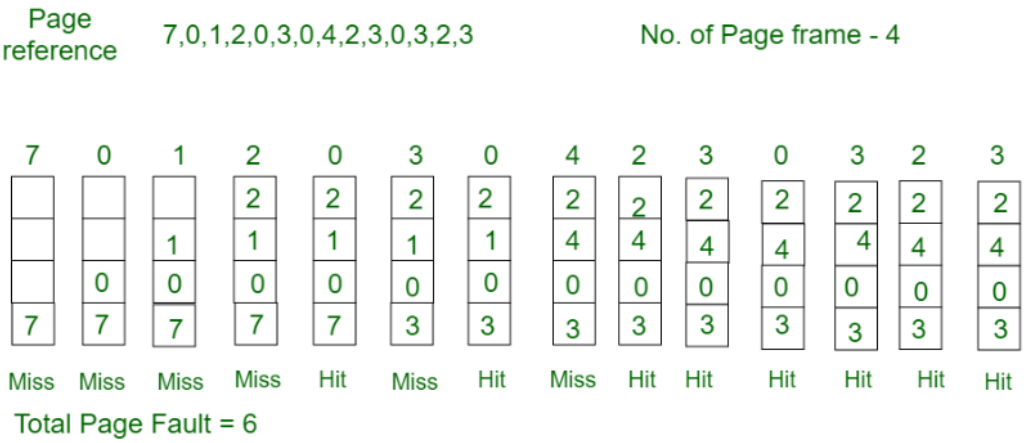


Figure 3: OPR example

Initially, all slots are empty, so when ’7’ ’0’ ’1’ ’2’ are allocated to the empty slots —> 4 Page faults.

’0’ is already in memory —> 0 Page fault.

When 3 came it will take the place of 7 because it is not used for the longest duration of time in the future —> 1 Page fault.

0 is already in memory —> 0 Page fault.

4 will takes place of 1 —> 1 Page Fault.

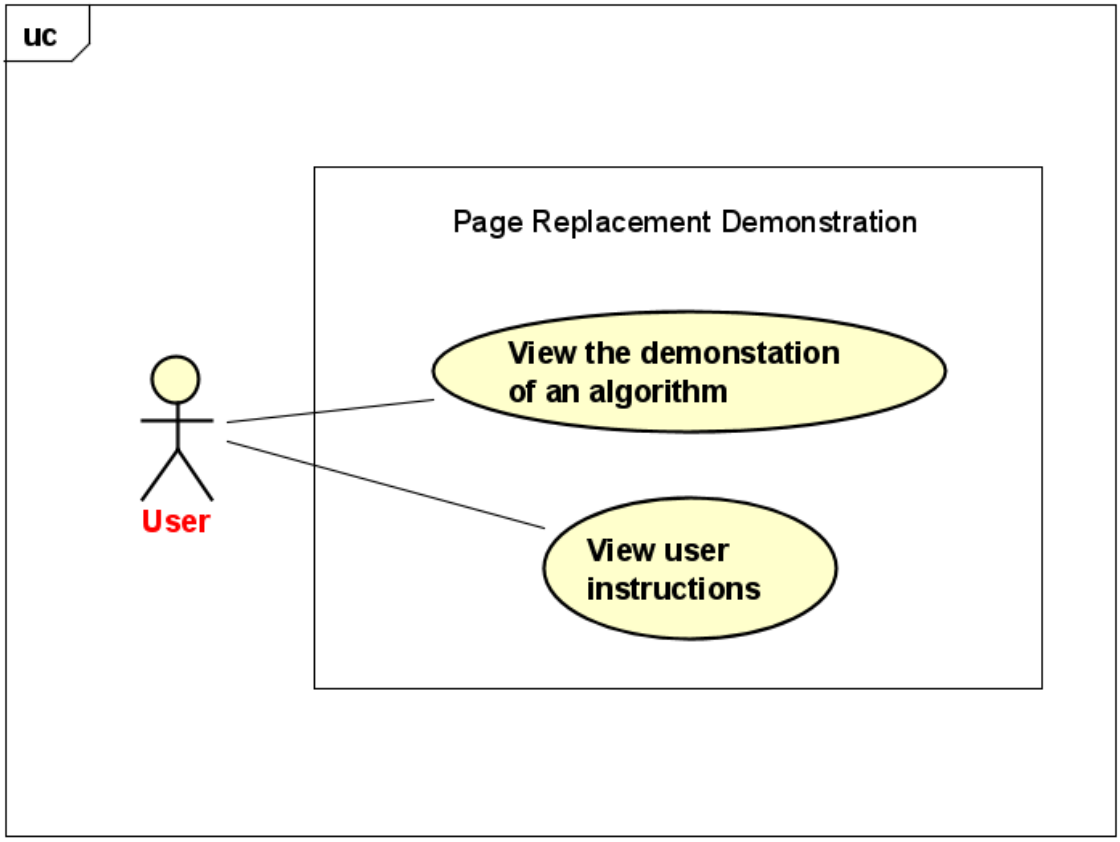
Now for the further page reference string —> 0 Page fault because they are already available in the memory. That gives us a total of 6 page faults.

4 GUI implementation

4.1 Use case diagram

The use case diagram for the GUI implementation of the page replacement algorithms includes the following actors: the user, who interacts with the GUI, and the system, which provides the GUI and runs the page replacement algorithms.

The use cases for the user include selecting the page replacement algorithm to be used, entering the size of the memory, and entering the sequence of page references. The use cases for the system include processing the input data and displaying the results of the page replacement algorithm in the GUI

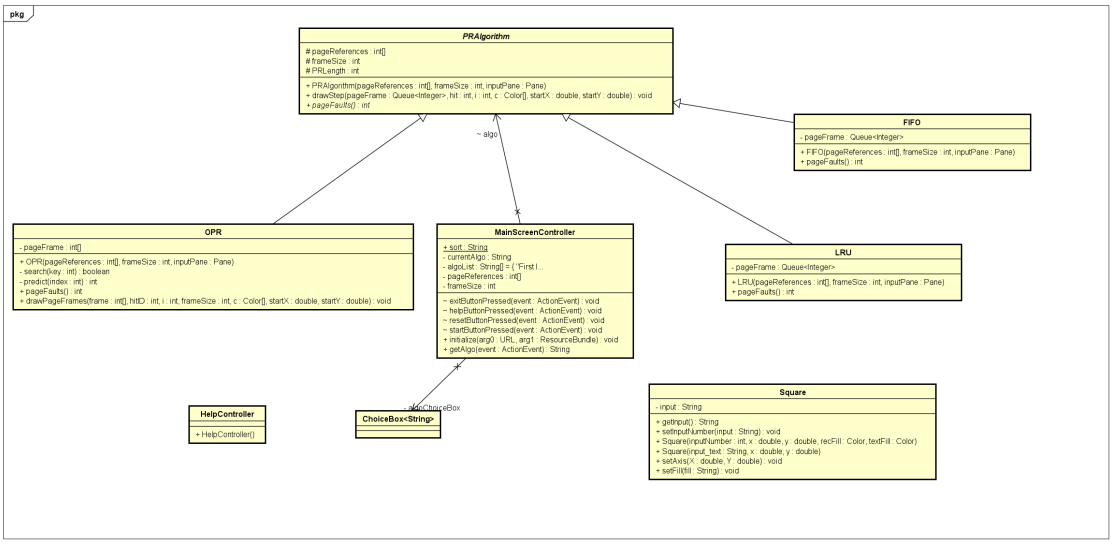


4.2 Class diagram

The class diagram for the GUI implementation of the page replacement algorithms includes the following classes: Main, GUI, Algorithm, and InputData.

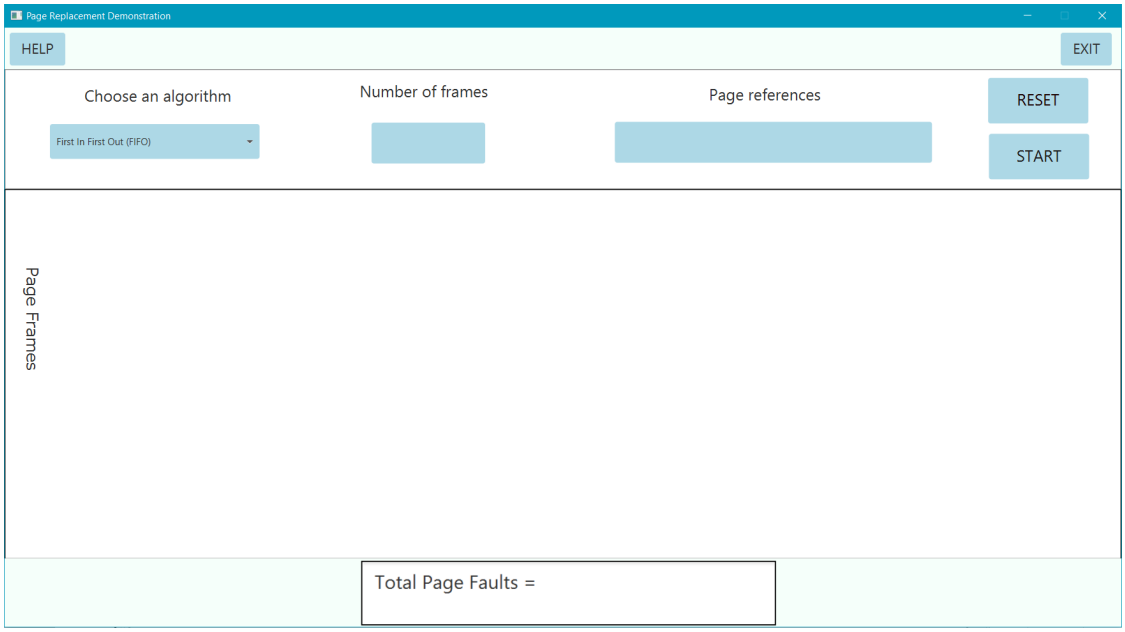
The Main class is the starting point of the application and contains the main method. The GUI class handles the graphical user interface and user input. The Algorithm class implements the page replacement algorithms, and the InputData class handles the input data for the algorithms. The Algorithm class includes the following methods: LRU, FIFO, Optimal, and Clock. These methods implement the corresponding page replacement algorithms.

The InputData class includes the following attributes: algorithm type, memory size, and page reference sequence. These attributes are used by the algorithms to determine the page replacements.

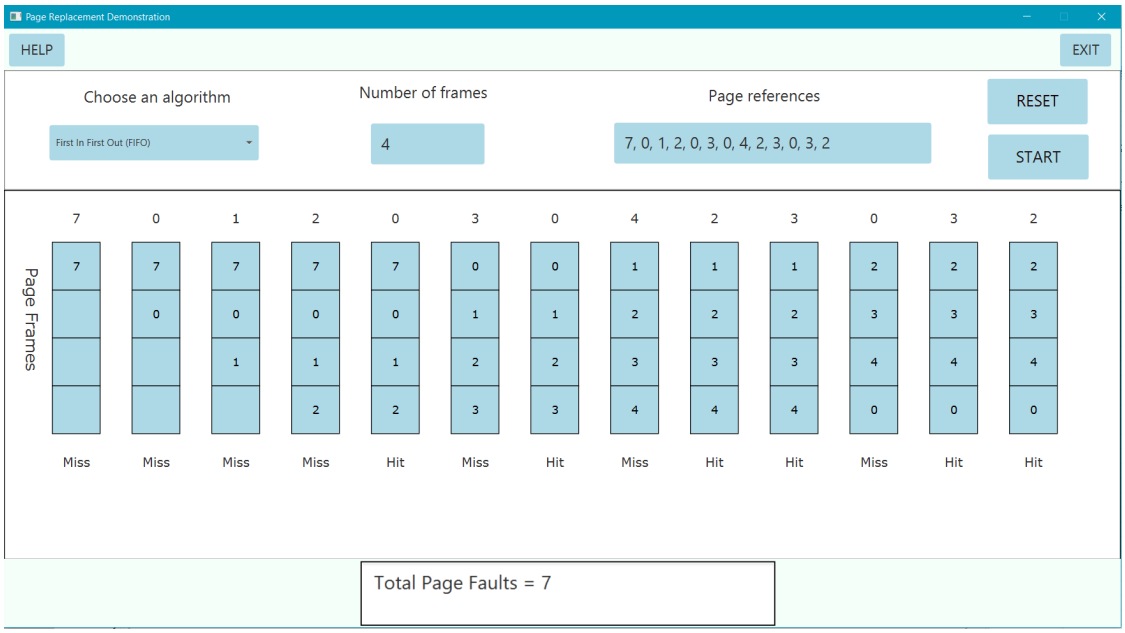


4.3 Java implementation

The GUI implementation of the page replacement algorithms is demonstrated in Java using the Swing library. The demonstration includes a window that allows the user to select the page replacement algorithm to be used, enter the size of the memory, and enter the sequence of page references.



Once the input data is entered, the page replacement algorithm is run and the results are displayed in a table in the GUI. The results include the page faults and the sequence of page replacements. The demonstration shows how the page replacement algorithms can be implemented in a user-friendly way, allowing the user to select the desired algorithm and input the necessary data.


Page Replacement Demonstration
HELP
EXIT
Choose an algorithm
First In First Out (FIFO)
Number of frames
4
Page references
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2
RESET
START
Page Frames
7 0 1 2 0 3 0 4 2 3 0 3 2
Miss Miss Miss Miss Hit Miss Hit Miss Hit Hit Miss Hit Hit
Total Page Faults = 7

5 Conclusion

In conclusion, the GUI implementation of the page replacement algorithms provides a user-friendly way to demonstrate and compare the different algorithms. The use case and class diagrams provide a clear understanding of the application's functionality and structure. The Java demonstration shows how the page replacement algorithms can be implemented in a practical and useful way.

Overall, the GUI implementation of the page replacement algorithms is a valuable tool for understanding and comparing the different page replacement algorithms and their performance in different scenarios.