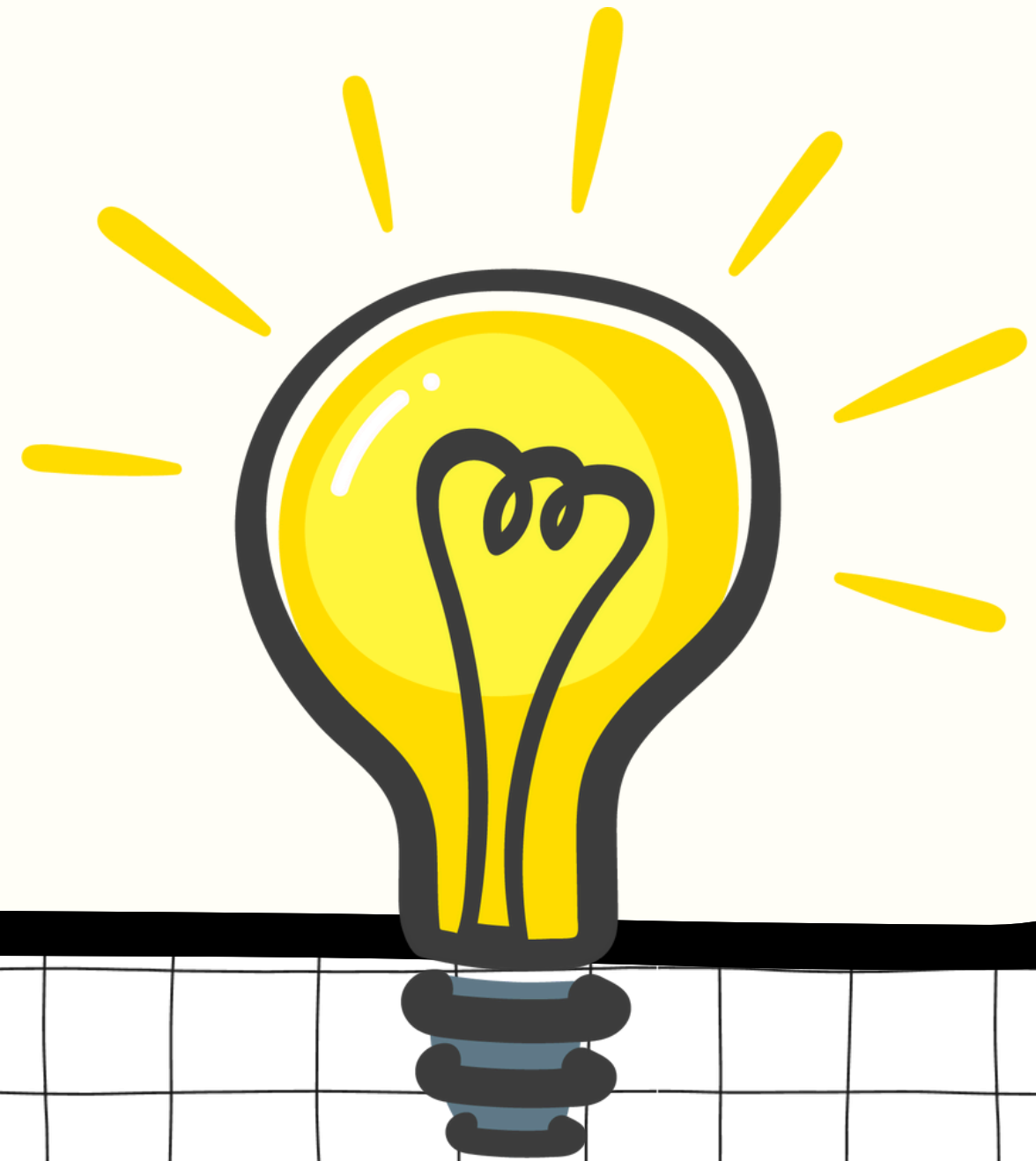




# **Virtual Memory**

Stimulating Virtual Memory



# Contents

**01**

Introduction

**02**

FIFO

**03**

LRU

**04**

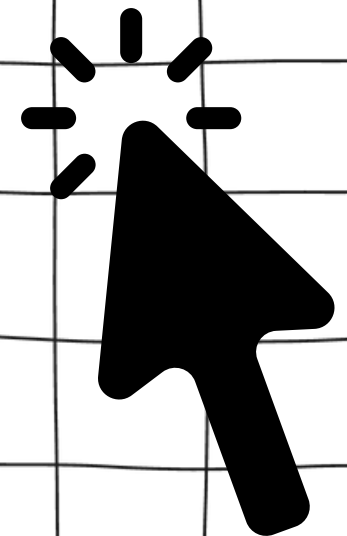
Clock

**05**

Optimal

**06**

Conclusion





# Introduction

- Virtual memory is a technique that uses a portion of a storage drive to simulate additional RAM.
- Extends physical memory by using disk storage.
- Provides a large, contiguous address space to applications.

we use Page Replacement Algorithms

- **FIFO**
  - **LRU**
  - **Clock**
  - **Optimal**
- 
- 

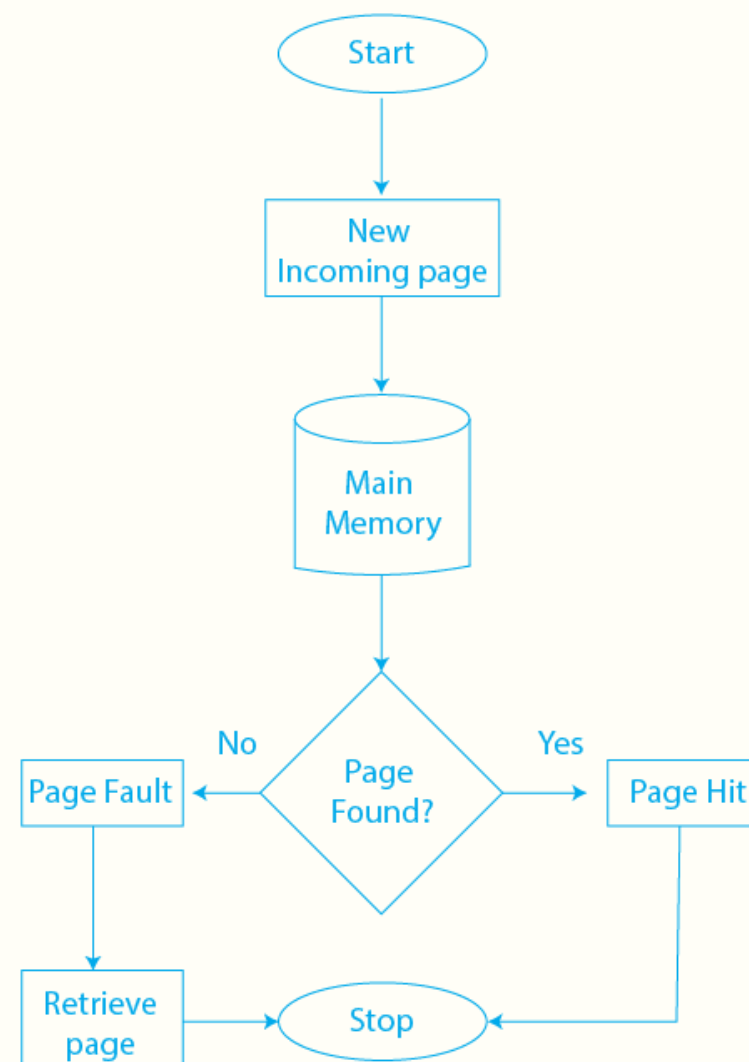
# **Page Replacement**

- **Page replacement refers to the process of swapping pages between physical memory RAM and secondary storage disk) when space in RAM is needed for new pages**
- **It ensures that the most relevant pages are kept in RAM while less frequently used pages are stored on disk**
- **Page Fault Occurs when a program tries to access a page not currently in RAM, triggering the page replacement process.**

**To detect when a page fault happened we use FIFO, LRU, Clock, and Optimal algorithms**

# FIFO

- What is the First In, First Out (FIFO) scheduling algorithm in operating systems?
- How does the FIFO scheduling algorithm determine the order of process execution?
- What are the primary advantages of using the FIFO scheduling algorithm?
- What are the main disadvantages or limitations of the FIFO scheduling algorithm?



Page  
reference

1, 3, 0, 3, 5, 6, 3

1	3	0	3	5	6	3
		0	0	0	0	3
	3	3	3	3	6	6
1	1	1	1	5	5	5
Miss	Miss	Miss	Hit	Miss	Miss	Miss

Total Page Fault = 6

# FIFO

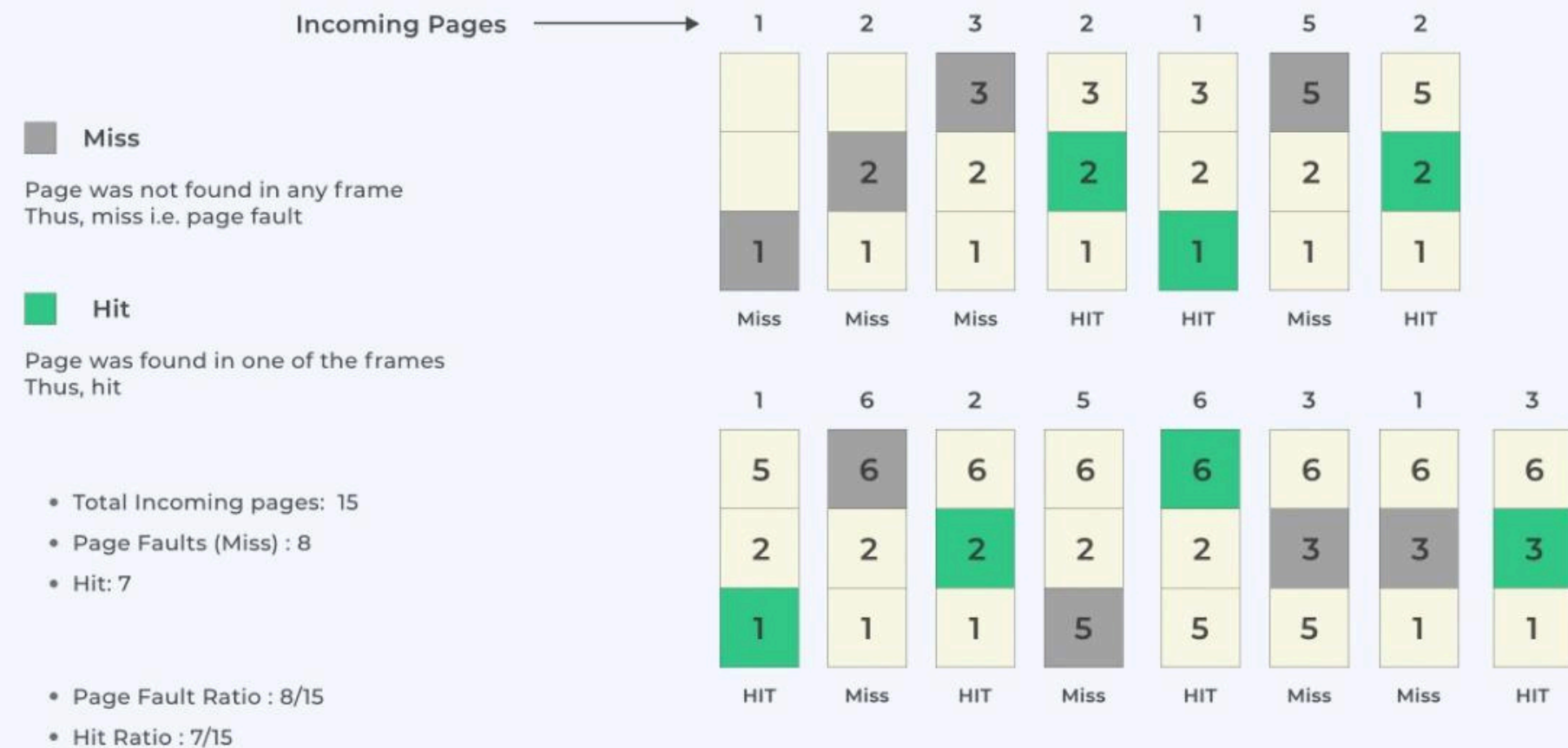
- **BASIC MECHANISM:**
  - **FIFO EXECUTES PROCESSES IN THE ORDER THEY ARRIVE, USING A QUEUE STRUCTURE WHERE THE FIRST PROCESS IN IS THE FIRST TO BE EXECUTED.**
- **ADVANTAGES:**
  - **SIMPLE TO IMPLEMENT AND UNDERSTAND, ENSURING FAIRNESS AS ALL PROCESSES ARE TREATED EQUALLY WITHOUT PRIORITIES.**
- **DISADVANTAGES:**
  - **CAN LEAD TO LONG WAITING TIMES FOR SHORTER PROCESSES DUE TO THE CONVOY EFFECT, REDUCING OVERALL SYSTEM EFFICIENCY.**
- **IDEAL USE CASES:**
  - **SUITABLE FOR SYSTEMS WITH UNIFORM PROCESS LENGTHS AND PREDICTABLE ARRIVAL TIMES, SUCH AS BATCH PROCESSING SYSTEMS.**
- **PERFORMANCE IMPACT:**
  - **MAY RESULT IN HIGHER AVERAGE WAITING TIMES AND LOWER THROUGHPUT, ESPECIALLY IF PROCESS LENGTHS VARY SIGNIFICANTLY.**



# LRU

- What is the Least Recently Used (LRU) page replacement algorithm in operating systems?
- How does the LRU page replacement algorithm determine which page to replace?
- What are the primary advantages of using the LRU page replacement algorithm?
- What are the main disadvantages or limitations of the LRU page replacement algorithm?

## LRU Page Replacement Algorithm In OS



# LRU

## **BASIC MECHANISM:**

- **LRU REPLACES THE PAGE THAT HAS NOT BEEN USED FOR THE LONGEST PERIOD OF TIME, USING DATA STRUCTURES TO KEEP TRACK OF PAGE ACCESS HISTORY.**

## **ADVANTAGES:**

- **MORE EFFICIENT THAN SIMPLER ALGORITHMS LIKE FIFO IN MANAGING PAGES, AS IT BETTER APPROXIMATES OPTIMAL PAGE REPLACEMENT BY LEVERAGING TEMPORAL LOCALITY.**

## **DISADVANTAGES:**

- **CAN BE COMPLEX TO IMPLEMENT DUE TO THE NEED FOR MAINTAINING AND UPDATING PAGE ACCESS ORDER, POTENTIALLY LEADING TO HIGHER OVERHEAD.**

## **IDEAL USE CASES:**

- **SUITABLE FOR SYSTEMS WHERE RECENT PAGE ACCESS PATTERNS ARE A GOOD PREDICTOR OF FUTURE ACCESSES, SUCH AS GENERAL-PURPOSE OPERATING SYSTEMS AND APPLICATIONS WITH STRONG TEMPORAL LOCALITY.**

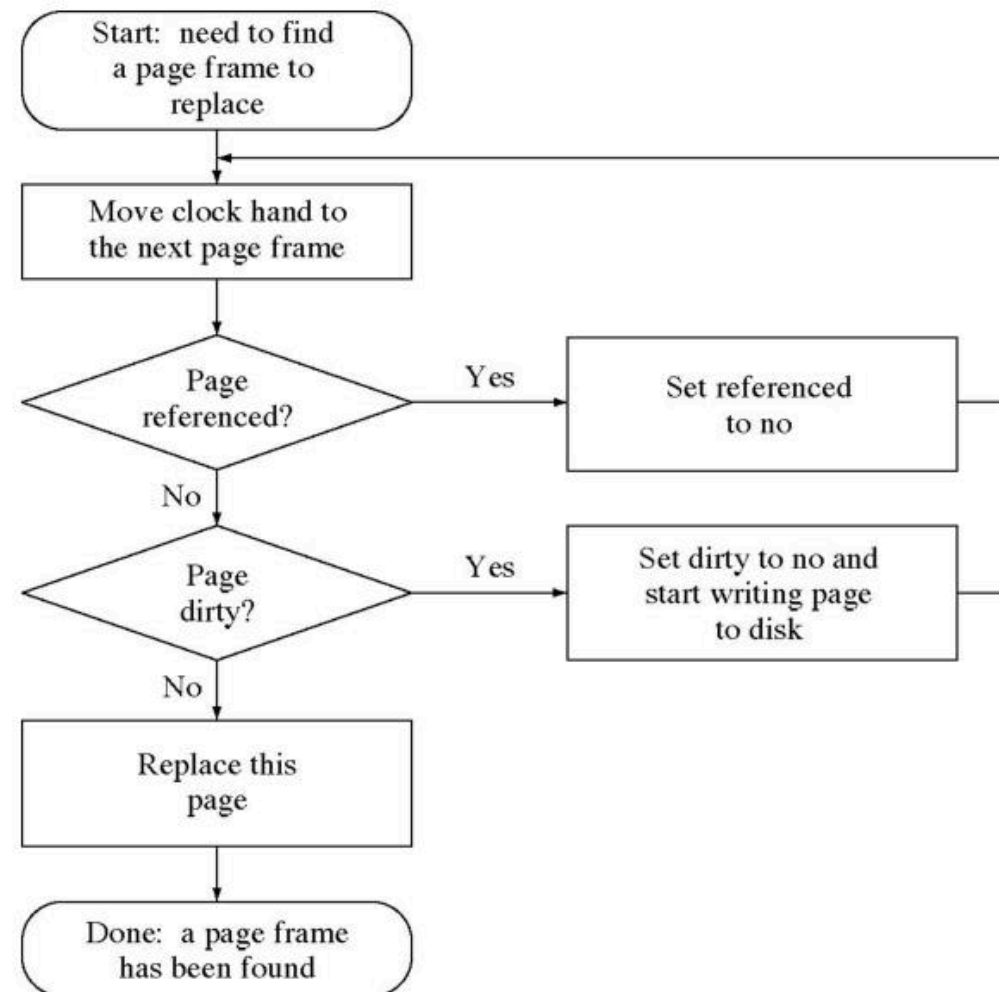
## **PERFORMANCE IMPACT:**

- **GENERALLY RESULTS IN LOWER PAGE FAULT RATES COMPARED TO FIFO, BUT THE OVERHEAD OF MAINTAINING ACCESS ORDER CAN IMPACT PERFORMANCE, ESPECIALLY WITH A LARGE NUMBER OF PAGES.**



# Clock

## Clock algorithm flow chart



Crowley OS Chap. 12

### Replacement Process:

1. **Move Clock Hand:** To the next page.
2. **Check Reference Bit:**
  - If 0: Replace the page.
  - If 1: Clear bit, move clock hand to next page.
3. **Repeat:** Until a page with bit 0 is found.

### Comparison:

- **LRU:** Accurate but high overhead.
- **Clock Algorithm:** Efficient LRU approximation using a circular list.

### Advantages:

- **Efficiency:** Low overhead in searching for replacement.
- **Simplicity:** Easy implementation with a circular list and single pointer.
- **Effectiveness:** Balances performance with complexity.

### • Purpose:

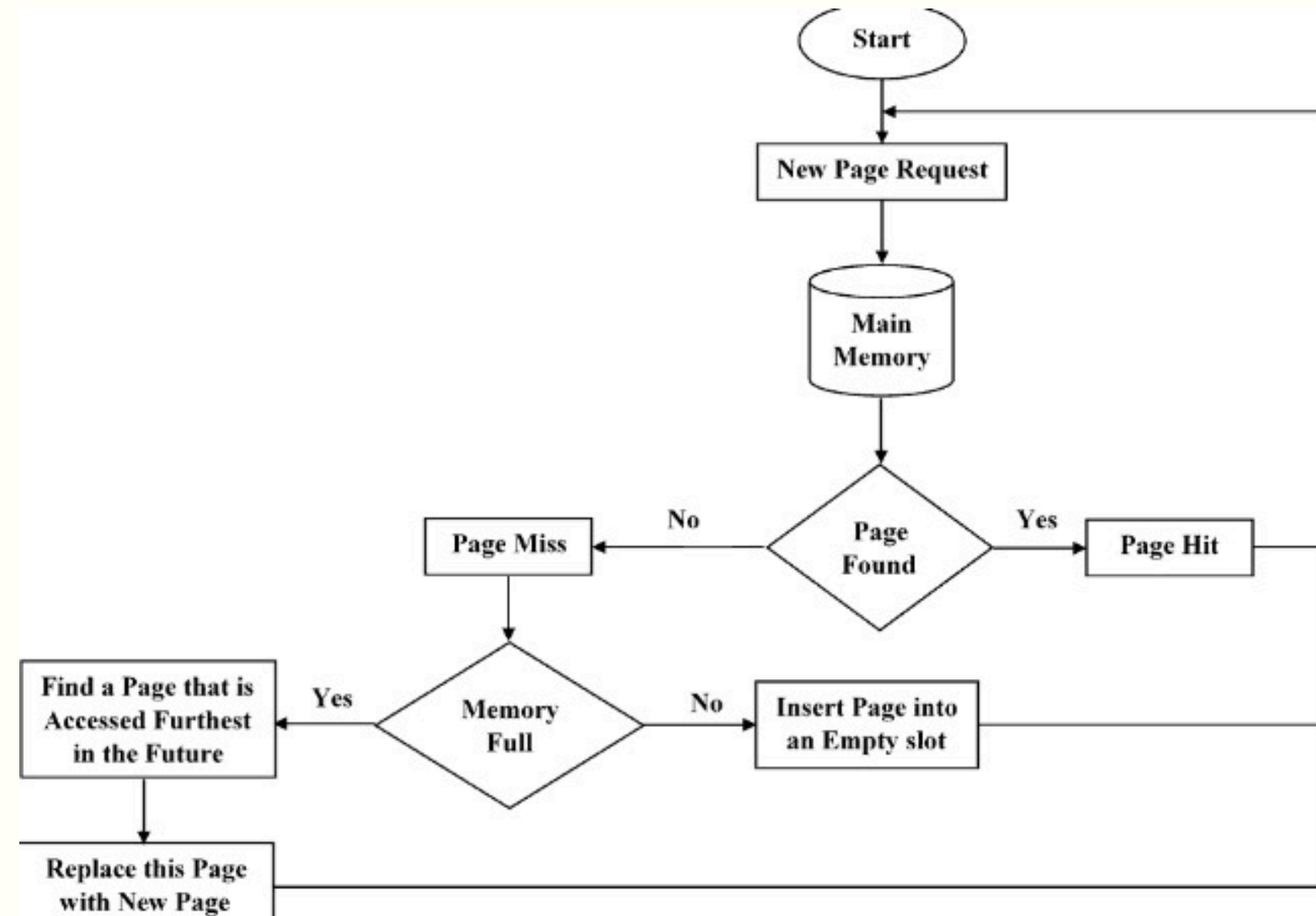
- Efficient page replacement strategy approximating LRU.
- LRU is accurate but costly in time and space complexity.

### • How It Works:

- **Circular List:** Pages organized in a circular list.
- **Clock Hand:** Pointer moves around the list for potential page replacement.
- **Reference Bit:** Each page has a reference bit (0 or 1).
  - Set to 1 on page access.

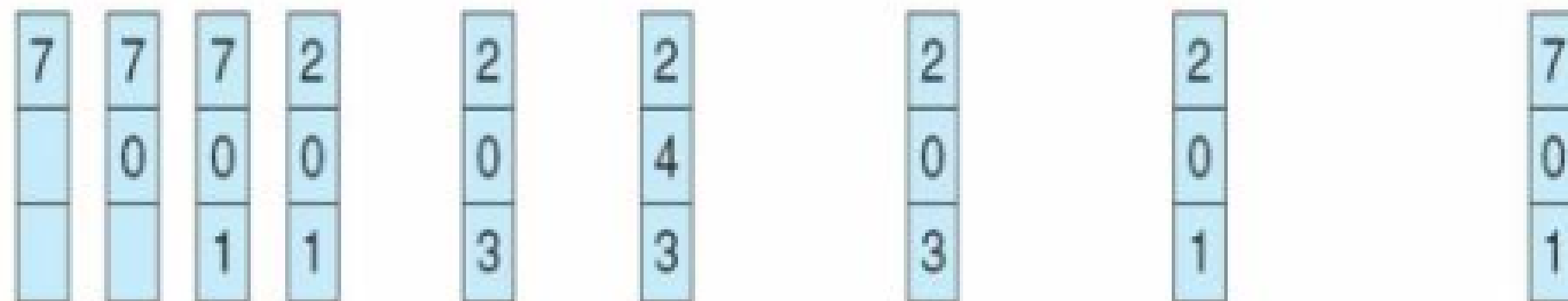
# Optimal

- Replace page that will not be used for longest period of time.
- Used for measuring how well our algorithm performance.
- how it works:
  - a.If referred page is already present, increment hit count.
  - b.If not present, find if a page that is never referenced in future. If such a page exists, replace this page with new page. If no such page exists, find a page that is referenced farthest in future. Replace this page with new page.



reference string

7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

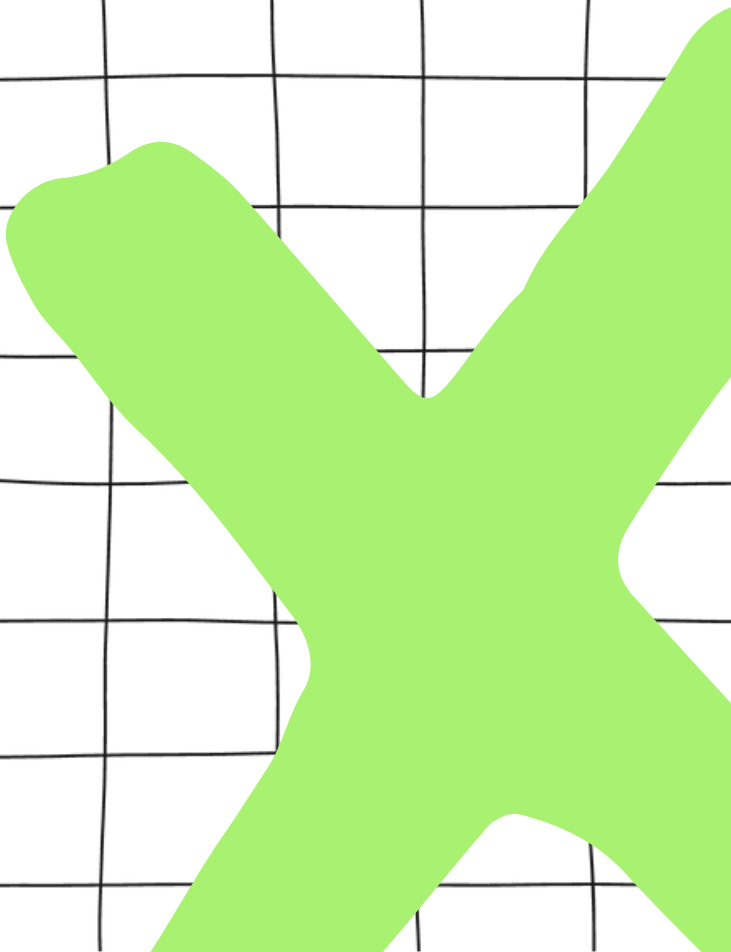


page frames



## conclusion

**In conclusion, this project offers a robust simulation environment for virtual memory management systems. By implementing various page replacement algorithms, including FIFO, LRU, and Clock, alongside support for different page sizes and paging strategies like demand and pre-paging, it facilitates in-depth analysis and comparison of system performance. Through experimentation and evaluation of page swap metrics, the project provides valuable insights into the efficiency and effectiveness of different memory management techniques, aiding in the optimization of virtual memory systems in real-world applications.**





**Thank you**

