5.3 Page Replacement Algorithms FIFO, LRU, Optimal.

5.3 Page Replacement Algorithms

Page replacement algorithms decide which memory pages to replace (swap out) when a page fault occurs and the memory is full. The primary goal is to minimize the number of page faults, thereby optimizing the performance of the system.

1. FIFO (First-In-First-Out) Algorithm

- **Definition:** The **First-In-First-Out (FIFO)** algorithm replaces the oldest page in memory with the new one. It works like a queue, where the first loaded page is the first to be removed.
- **How it Works:**
  1. Pages are loaded into frames in the order they arrive.
  2. When a page fault occurs, the page that has been in memory the longest (the first one) is replaced.
- **Example Scenario:**
  - Page reference string: 2, 3, 1, 4, 3, 2, 5, 1 (Frame size: 3)
  - The page sequence would result in replacing pages in the order they were loaded.
- **Pros:**
  - Simple to implement.
  - Requires minimal bookkeeping.
- **Cons:**
  - Can result in high page faults if the oldest page is frequently used.

2. LRU (Least Recently Used) Algorithm

- **Definition:** The **Least Recently Used (LRU)** algorithm replaces the page that has not been used for the longest period of time.
- **How it Works:**
  1. Every time a page is accessed, it is marked with the current time or placed at the top of a stack.
  2. When a page fault occurs, the page least recently accessed is removed.
- **Example Scenario:**
  - Page reference string: 2, 3, 1, 4, 3, 2, 5, 1 (Frame size: 3)

- The page sequence would result in replacing the least recently accessed pages.

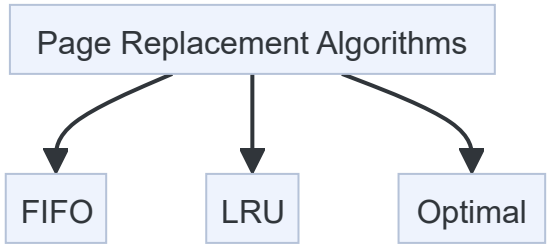
- **Pros:**
  - More efficient than FIFO; lowers the number of page faults in many cases.
- **Cons:**
  - Requires additional memory to track usage, which increases overhead.

3. Optimal Page Replacement Algorithm

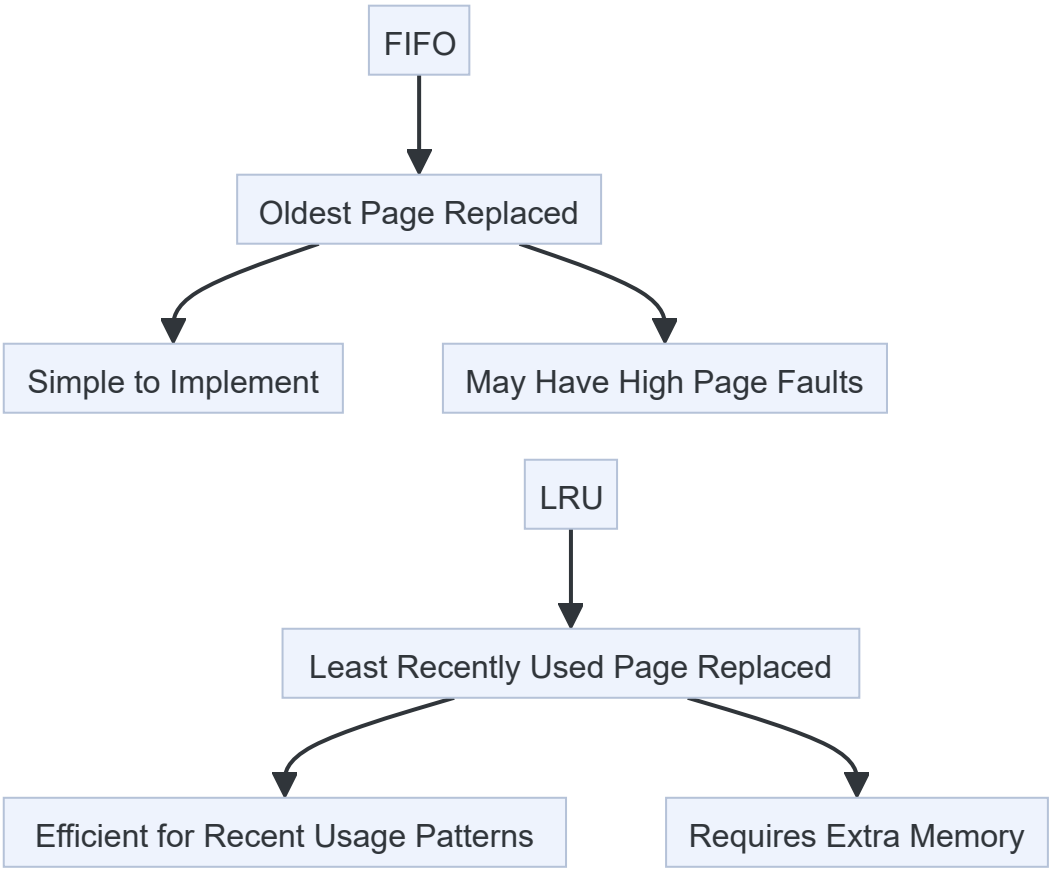
- **Definition:** The **Optimal Page Replacement** algorithm replaces the page that will not be used for the longest period of time in the future.
- **How it Works:**
  1. Looks ahead in the page reference string to determine which page will not be needed for the longest time.
  2. Replaces that page to minimize future page faults.
- **Example Scenario:**
  - Page reference string: 2, 3, 1, 4, 3, 2, 5, 1 (Frame size: 3)
  - The optimal algorithm will foresee future references and select pages for replacement accordingly.
- **Pros:**
  - Guarantees the lowest number of page faults.
- **Cons:**
  - Not practically implementable in real-time systems as it requires future knowledge of the reference string.

Diagram: Page Replacement Algorithms

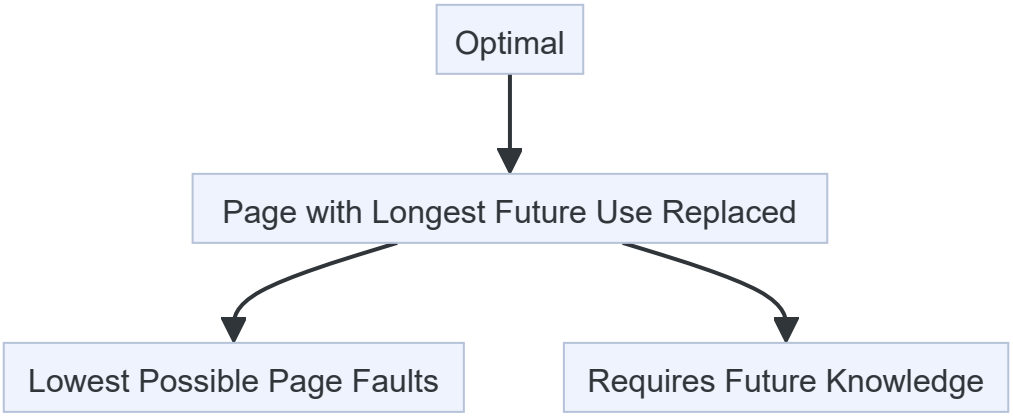
Flowchart 1: Overview of Page Replacement Algorithms



Flowchart 2: FIFO and LRU Algorithms



Flowchart 3: Optimal Algorithm



Summary Table of Page Replacement Algorithms

Algorithm	Definition	How it Works	Pros	Cons	Example Use Case
FIFO	Replaces oldest page in memory	Uses a queue to track page order	Simple and easy to implement	High page faults if old pages are reused	Basic systems with predictable loads

Algorithm	Definition	How it Works	Pros	Cons	Example Use Case
LRU	Replaces least recently used page	Tracks recent usage with time stamps or stack	Reduces page faults effectively	Needs extra memory and management	Systems with varying access patterns
Optimal	Replaces the page not used for the longest future	Foresees future requests and replaces accordingly	Guarantees the least page faults	Impossible to implement practically	Theoretical benchmark for other algorithms

Explanation in Simple Terms:

- **Page Replacement Algorithms** help a computer decide which pages of memory to keep and which ones to throw out when it runs out of space.
- **FIFO (First-In-First-Out)** is like a line at a store; the first person (page) to enter is the first to leave.
- **LRU (Least Recently Used)** is like cleaning up toys you haven't played with in a long time.
- **Optimal** is like magically knowing which toys you won't play with for the longest time and putting those away first.