Page Replacement Algorithms

The Art of Memory Management

Presented by:
Sandesh Shrestha 2K21/CO/417
Sanskar Ojha 2K21/CO/418

Page : Introduction and overview

- A **Page** is a fixed-size block of memory of a process used by the operating system to manage the memory of a running program.

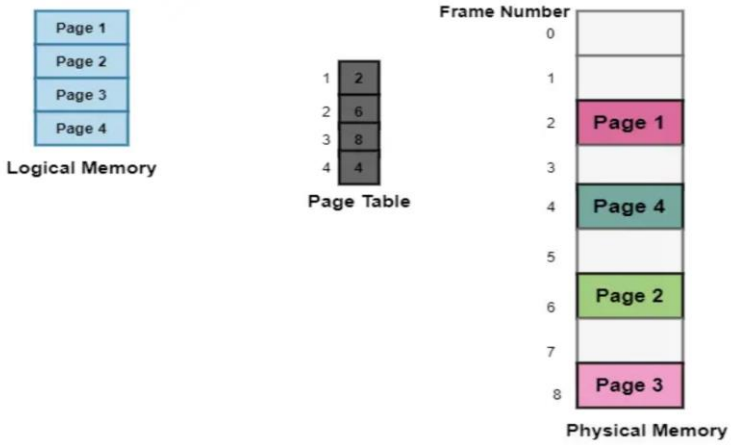
- The memory of a program is divided into pages, and each page is mapped to a corresponding physical page frame in the computer's RAM. When a program needs to access a page of memory, the operating system retrieves the corresponding physical page frame from RAM and maps it to the virtual address of the program.

Paging and Page Faults

This process of mapping virtual pages to physical page frames is called Paging.

Paging is a method of writing and reading data from a secondary storage(Drive) for use in primary Storage(RAM).

The data structure that is used by the virtual memory system in the operating system of a computer in order to store the mapping between physical and logical addresses is commonly known as **Page Table.**



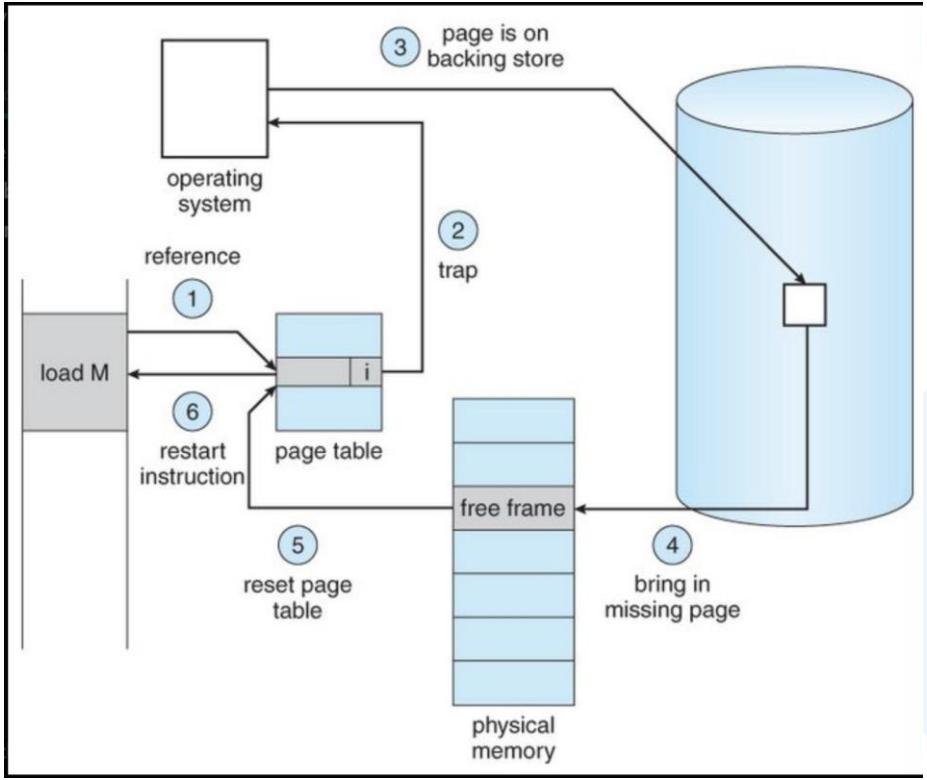
Paging and Page Faults

To understand **why we need page replacement algorithms**, we first need to know about **page faults**. Let's see what is a page fault.

Page Fault: A Page Fault occurs when a program running in CPU tries to access a page that is in the address space of that program, but the **requested page is currently not loaded into the main physical memory**, the RAM of the system.

.

Since the actual RAM is much less than the virtual memory, the page faults occur. So whenever a page fault occurs, the Operating system has to **replace an existing page in RAM with the newly requested page.** In this scenario, page replacement algorithms help the Operating System in deciding which page to replace.



Effective Memory Access time :
EMAT = p * (page fault service time) + (1-p) * (main memory access time)
where,
Main memory access time is in nanosecond
Page fault service time is in milisecond

Need of page replacement policy: Summary

- **The need for a page replacement policy arises because the amount of physical memory (RAM) available in a computer is usually much smaller than the total amount of memory required by all running processes. As** a result, the operating system needs to manage the available memory effectively by swapping pages in and out of memory as needed.

- A page replacement policy is needed to determine which pages to swap out of memory when the available memory becomes insufficient to hold all the required pages. Without a page replacement policy, the operating system would have to swap out pages arbitrarily or rely on a fixed algorithm that may not be optimal for all types of workloads

Page Replacement Policy : Introduction

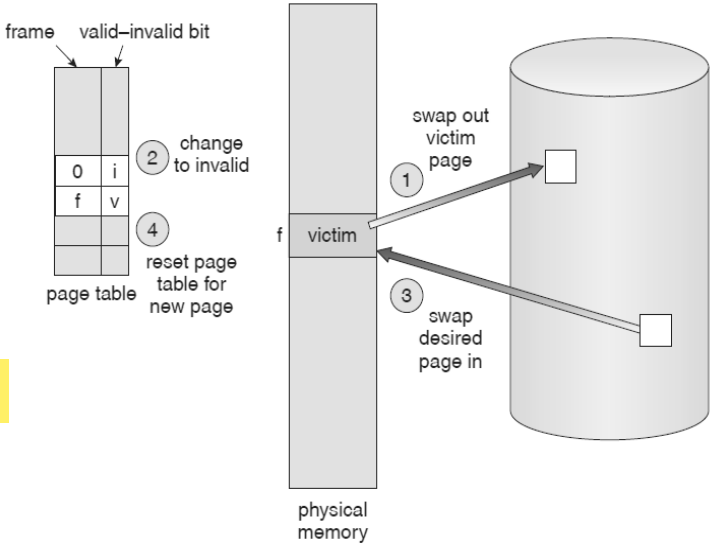
A Page Replacement Policy is an algorithm used by an operating system to select which pages to swap out of main memory (RAM) and into secondary storage (usually disk) when the available memory is not sufficient to hold all the required pages.

The main goal of a page replacement policy is to **maximize the usage of available memory while minimizing the number of page faults** (the number of times a program needs to wait for a page to be loaded into memory).

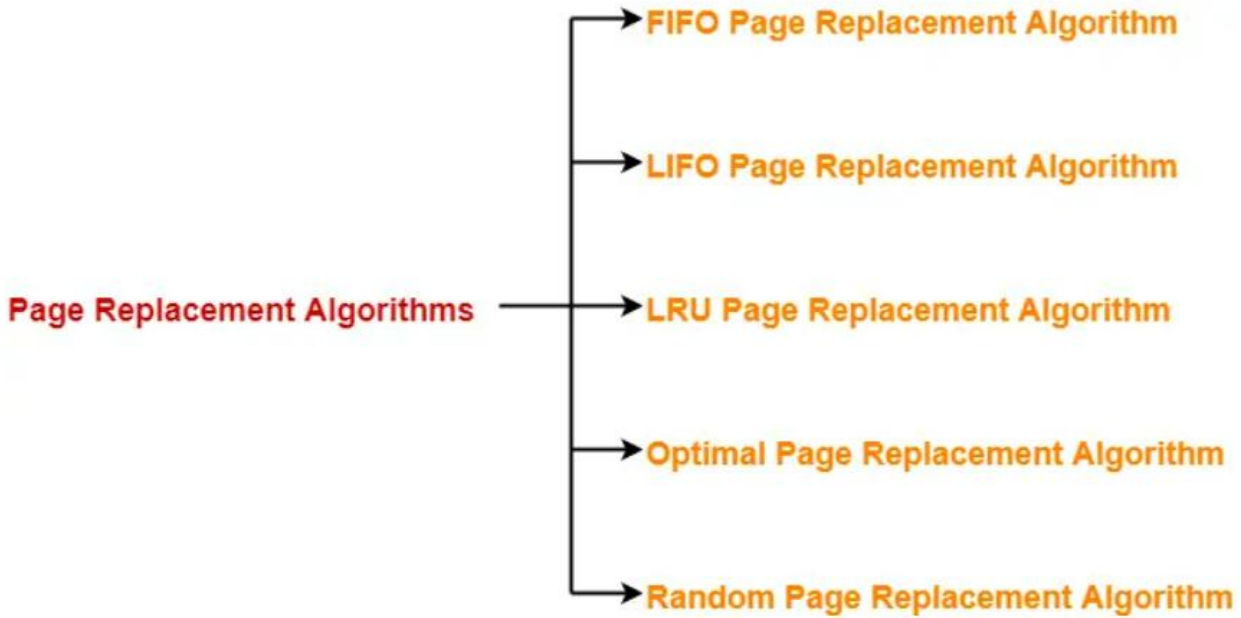
Page Replacement Algorithms

- Primary Objective: Lower Page-fault Rate

- determines which pages should be removed from memory when there is no more space available in main memory for new pages.



Page Replacement Algorithms

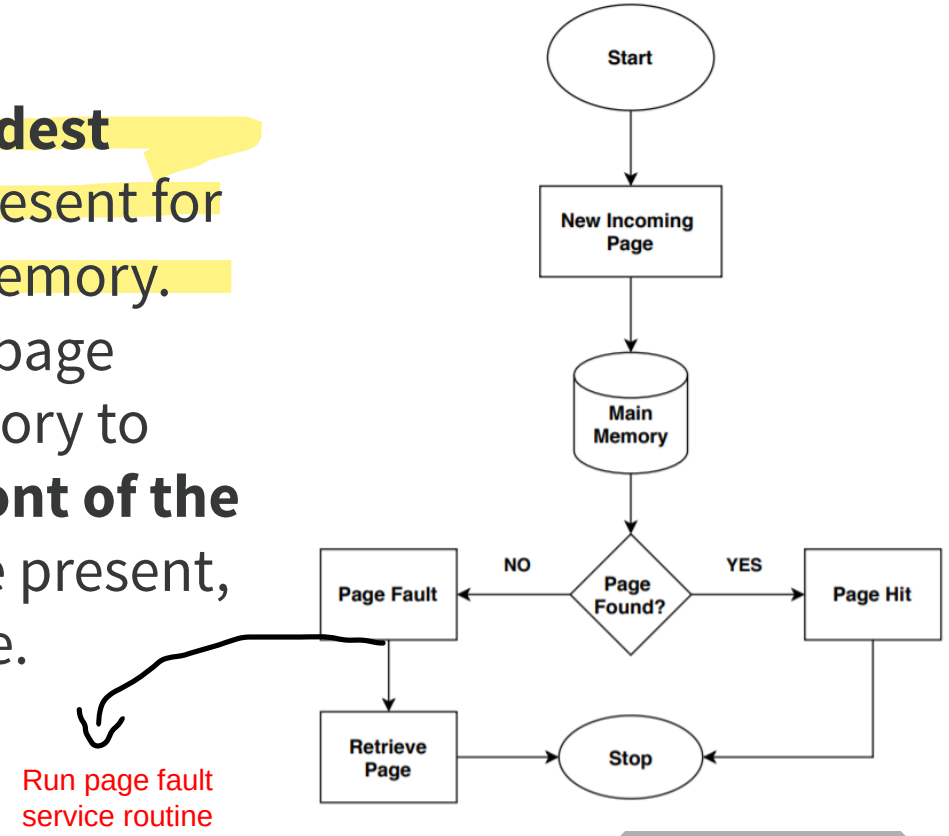


Page Replacement Algorithms

- **FIFO (First In First Out):** the page that has been in memory **the longest is replaced first.**
- **LRU (Least Recently Used):** the page that has **not been accessed for the longest time is replaced first.**
- **Optimal Page Replacement:** replaces the page that **will not be used for the longest time in the future.**

FIFO Page Replacement Algorithm

- FIFO algorithm replaces the **oldest (First) page** which has been present for the longest time in the main memory.
- **In simple words**, When a new page comes in from secondary memory to main memory, It selects the **front of the queue** which is the oldest page present, and replaces it with a new page.



A system uses 3 page frames for storing process pages in main memory. It uses **the First in First out (FIFO) page replacement policy. Assume that all the page frames are initially empty**. What is the total number of page faults that will occur while processing the page **reference string** given **- 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1.** Also calculate the hit ratio and miss ratio.

Pages >>	6	7	8	9	6	7	1	6	7	8	9	1
Frame 3			8	8	8	7	7	7	7	7	9	9
Frame 2		7	7	7	6	6	6	6	6	8	8	8
Frame 1	6	6	6	9	9	9	1	1	1	1	1	1
	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Hit

Hit ratio = $\frac{Total\ number\ of\ page\ hits}{Total\ number\ of\ references} = \frac{3}{12}$

Page fault/miss ratio = $\frac{Total\ number\ of\ page\ fault\ or\ miss}{Total\ number\ of\ references} = \frac{9}{12}$

FIFO Page Replacement Algorithm

```
# Declare 'S' as a set of current pages in the system.
# Declare a Queue name QPage which will be used to store the incoming pages.
# Queue will store in a FIFO manner
# page fault PF is Initially set to zero.
S= set();
QPage = Queue();
PF=0;
for k = 0 to length (N) do
    if length (S) < C then
        if P[k] not in S then
            S.add(P[k]);
            PF = PF + 1;
            QPage.put(P[k]);
        end
    else
        if P[k] not in S then
            val = QPage.queue[0];
            QPage.get();
            S.remove(val);
            S.add(P[k]);
            QPage.put(P[k]);
            PF = PF + 1;
        end
    end
end
return PF;
```

Advantages of FIFO Page Replacement Algorithm

- commonly known for its simplicity.
- much easy to implement as well as understand.
- Small systems can use the FIFO algorithm efficiently.

Disadvantages of FIFO Page Replacement Algorithm

- Uses an additional **Queue** data structure.
- It suffers from **Belady's anomaly** problem i.e when the number of page frames increases, more memory is given to processes, but instead of decreasing, the number of page faults increases.

Belady’s anomaly

- For LRU and optimal page replacement algorithms, **increasing** the number of **frames** generally **reduces** the number of **page faults**.

- However, Belady found that , for the FIFO page replacement algorithm, **increasing** the number of **frames** can actually **increases** the number of **page faults**.

Belady's anomaly

Reference String : 0 1 5 3 0 1 4 0 1 5 3 4.

Case 1: Number of frames = 3

Request	0	1	5	3	0	1	4	0	1	5	3	4
Frame 3			5	5	5	1	1	1	1	1	3	3
Frame 2		1	1	1	0	0	0	0	0	5	5	5
Frame 1	0	0	0	3	3	3	4	4	4	4	4	4
Miss/Hit	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Hit

Number of Page Faults = 9
Page miss ratio = 9/12

Belady's anomaly

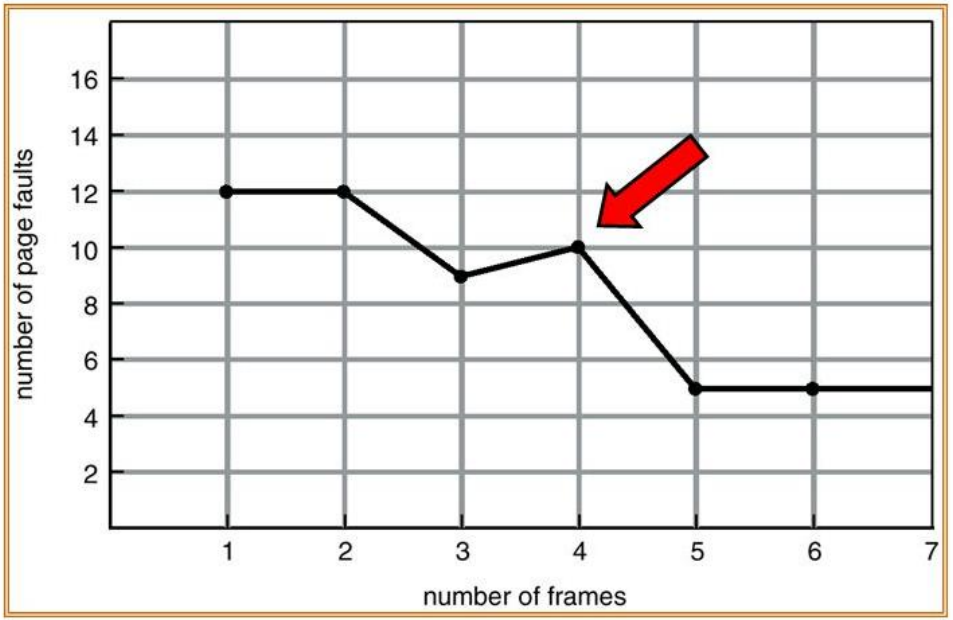
Reference String : 0 1 5 3 0 1 4 0 1 5 3 4.

Case 2: Number of frames = 4

Request	0	1	5	3	0	1	4	0	1	5	3	4
Frame 4				3	3	3	3	3	3	5	5	5
Frame 3			5	5	5	5	5	5	1	1	1	1
Frame 2		1	1	1	1	1	1	0	0	0	0	4
Frame 1	0	0	0	0	0	0	4	4	4	4	3	3
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Miss	Miss	Miss	Miss

Number of Page Faults = 10
Page miss ratio = 10/12

Belady’s anomaly



increasing the number of page frames in a memory can sometimes lead to an increase in the number of page faults in a FIFO page replacement algorithm.

LRU Page Replacement Algorithm

- LRU stands for Least Recently Used.
- based on the strategy that whenever a page fault occurs, the **least recently used page will be replaced with a new page**. So, the page not utilized for the longest time in the memory (compared to all other pages) gets replaced.
- This algorithm works on **previous data.** The page which is used the **earliest** is replaced or which appears the earliest in the sequence is replaced.

A system uses 3 page frames for storing process pages in main memory. It uses **the LRU page replacement policy. Assume that all the page frames are initially empty**. What is the total number of page faults that will occur while processing the page **reference string** given **- 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1, 7, 9, 6.** Also calculate the hit ratio and miss ratio.

Pages>>	6	7	8	9	6	7	1	6	7	8	9	1	7	9	6
Frame 3			8	8	8	7	7	7	7	7	7	1	1	1	6
Frame 2		7	7	7	6	6	6	6	6	6	9	9	9	9	9
Frame 1	6	6	6	9	9	9	1	1	1	8	8	8	7	7	7
	Miss	Miss	Miss	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Miss	Miss	Miss	Hit	Miss

Hit ratio = $\frac{Total\ number\ of\ page\ hits}{Total\ number\ of\ references} = \frac{3}{15}$

Page fault/miss ratio = $\frac{Total\ number\ of\ page\ fault\ or\ miss}{Total\ number\ of\ references} = \frac{12}{15}$

Advantages of LRU Page Replacement Algorithm

- Performs well in many scenarios, particularly when there is a high degree of locality in the memory access pattern.
- Can work well with large numbers of pages in memory.

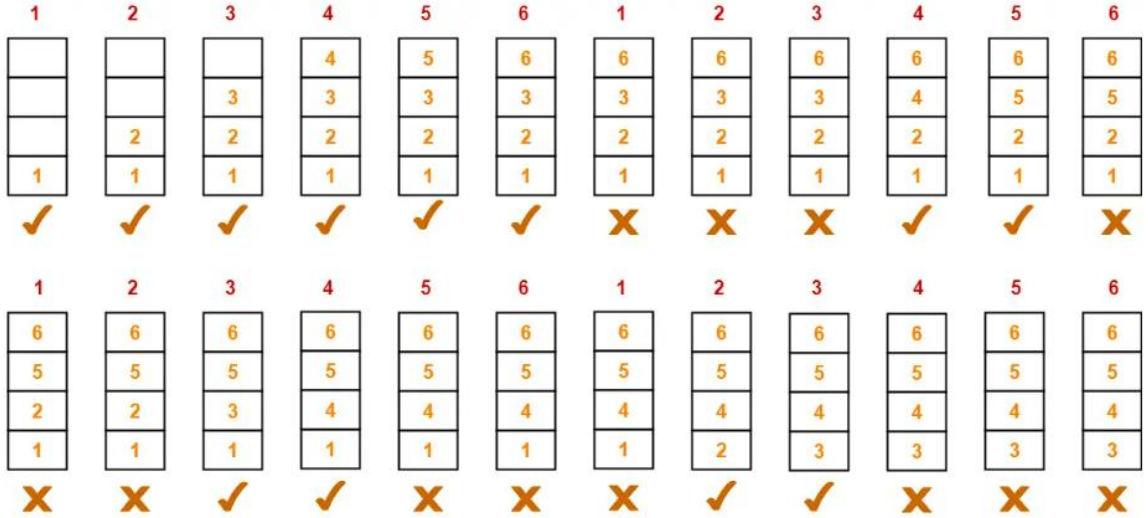
Disadvantages of LRU Page Replacement Algorithm

- Requires tracking every memory access to determine which page is least recently used, which can be computationally expensive.
- May require specialized hardware support to implement efficiently.

MRU Page Replacement Algorithm

- MRU stands for Most Recently Used.
- selects the page that was most recently accessed for replacement when a page fault occurs, rather than the least recently used page as in the LRU algorithm..
- To implement the MRU algorithm, the page table keeps track of the timestamp of the last access for each page in memory.

A system uses 3 page frames for storing process pages in main memory. It uses **the LRU page replacement policy. Assume that all the page frames are initially empty**. What is the total number of page faults that will occur while processing the page **reference string** given **- 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6.** Also calculate the hit ratio and miss ratio.



Hit ratio = $\frac{Total\ number\ of\ page\ hits}{Total\ number\ of\ references} = \frac{12}{24}$

Page fault/miss ratio = $\frac{Total\ number\ of\ page\ fault\ or\ miss}{Total\ number\ of\ references} = \frac{12}{24}$

Optimal Page Replacement Algorithm

- The **most desirable** page replacement algorithm for replacing pages.
- the page which would be used after the longest interval is replaced. In other words, the page which is farthest to come in the upcoming sequence is replaced.

Advantages of MRU Page Replacement Algorithm

- Can perform well in situations where a small number of pages are accessed frequently and need to be kept in memory.

Disadvantages of MRU Page Replacement Algorithm

- Can suffer from the same computational overhead as LRU, as it also requires tracking every memory access.
- May perform poorly in scenarios where there is a low degree of locality in the memory access pattern, as it may **replace pages that are likely to be needed again soon.**

A system uses 3 page frames for storing process pages in main memory. It uses **the Optimal page replacement algorithm. Assume that all the page frames are initially empty**. What is the total number of page faults that will occur while processing the page **reference string** given **- 6, 7, 8, 9, 6, 7, 1, 6, 7, 8, 9, 1, 7, 9, 6.** Also calculate the hit ratio and miss ratio.

Pages >>	6	7	8	9	6	7	1	6	7	8	9	1	7	9	6
Frame 3			8	9	9	9	1	1	1	1	1	1	1	1	1
Frame 2		7	7	7	7	7	7	7	7	7	7	7	7	7	7
Frame 1	6	6	6	6	6	6	6	6	6	8	9	9	9	9	6
	Miss	Miss	Miss	Miss	Hit	Hit	Miss	Hit	Hit	Miss	Miss	Hit	Hit	Hit	Miss

Hit ratio = $\frac{Total\ number\ of\ page\ hits}{Total\ number\ of\ references} = \frac{7}{15}$

Page fault/miss ratio = $\frac{Total\ number\ of\ page\ fault\ or\ miss}{Total\ number\ of\ references} = \frac{8}{15}$

Advantages of Optimal Page Replacement Algorithm:

- **Least page fault** occurs as this algorithm replaces the page that is not going to be used for the longest time in the future.
- Belady's Anomaly does not occur in this algorithm.
- Optimal page replacement algorithm always gives **the best performance in every scenario.**

Optimal PR Algorithm: Setbacks

- **not practical** because it **cannot predict whether the page it is going to remove will not be used in the future.**
- Maybe it removes a page and then immediately after an iteration we will again need that removed page in a frame.

Important Results :

Optimal page replacement algorithm always gives **the best performance in every scenario.**

Result 1: For a Reference String Consisting Of **Repeated Sequence** Of Page Numbers, **MRU** page replacement algorithm will behave as Optimal page replacement algorithm.

Result 2: For a Reference String Where First Half String Is a **Mirror Image** Of Other Half - **LRU and FIFO** page replacement algorithm will behave as Optimal page replacement algorithm.

Importance of Page replacement policy

The importance of an efficient page replacement policy lies in *its ability to optimize the use of available memory and reduce the overhead associated with frequent page faults*. If the page replacement policy is not effective, the computer may spend a lot of time swapping pages in and out of memory, which can significantly slow down the performance of the system.

Therefore, an efficient page replacement policy can help ensure that the computer runs smoothly and can handle larger and more complex programs with ease, while minimizing the impact on the system's performance.

Impacts of Page replacement policy

The page replacement policy is an important aspect of memory management in an operating system (OS) and has a significant impact on the smooth functioning of the OS. The page replacement policy determines which pages in memory should be replaced with new pages when the memory is full and a new page needs to be loaded.

The impact of page replacement policy on the smooth functioning of an OS can be seen in several ways:

1.Performance: The page replacement policy can have a significant impact on the performance of the OS. A good policy can minimize the number of page faults and optimize memory usage, leading to better overall system performance. On the other hand, a poor policy can result in excessive page faults, which can slow down the system.

Impacts of Page replacement policy

2.**Fairness**: The page replacement policy can also impact the fairness of the system, as different processes may have different memory requirements. A good policy should ensure that all processes are given a fair share of memory and that no process is unfairly penalized.

3.**Complexity**: The complexity of the page replacement policy can also impact the smooth functioning of the OS. A more complex policy may require more resources and processing time, leading to slower performance and increased overhead.

In summary, the page replacement policy is a critical component of memory management in an OS, and its impact on the smooth functioning of the system cannot be overstated. A good policy can optimize performance, ensure fairness, and minimize complexity, leading to a more stable and reliable system.