Page Replacement

Minsoo Ryu

Operating Systems and Distributed Computing Lab.
Hanyang University

msryu@hanyang.ac.kr

Topics Covered

- Page Replacement
- LRU Implementations

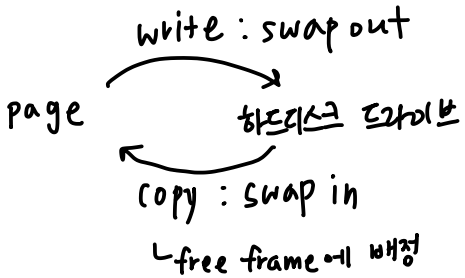
Over-Allocation of Memory

- Suppose that
  - We have 40 frames
  - Processes requires 10 pages
  - Processes actually use 5 pages
  - We could run 4 processes without demand paging
  - We could run 8 processes with demand paging
    - Over-allocation of memory
    - Increased level of multiprogramming
- It is possible that
  - Some process may suddenly try to use all ten of its pages
  - When a page fault occurs, the system would find that there are no free frames
  - What should we do?

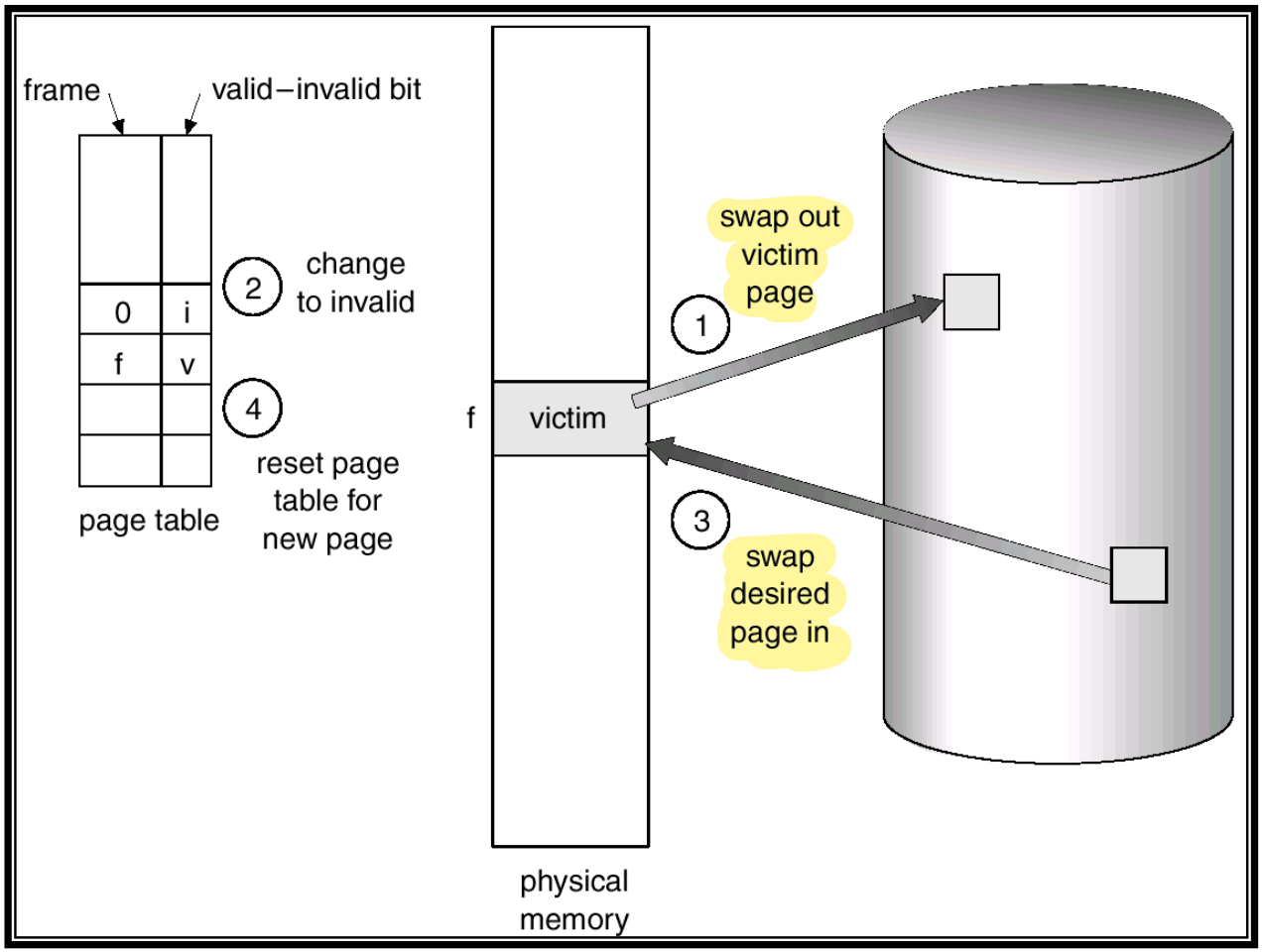
Basic Page Replacement

1. Find the location of the desired page on disk
2. Select a *victim* frame
   - Using a page replacement algorithm
3. Read the desired page into the (newly) free frame
4. Update the page and frame tables
5. Restart the process

- 2 page transfers → doubles EAT
  - Swapping out the victim page
  - Swapping in the desired page



Page Replacement



Page Replacement Algorithms

- **To evaluate an algorithm**
  - **Use a particular string of memory references**
    - reference string
  - **Compute the number of page faults on that string**

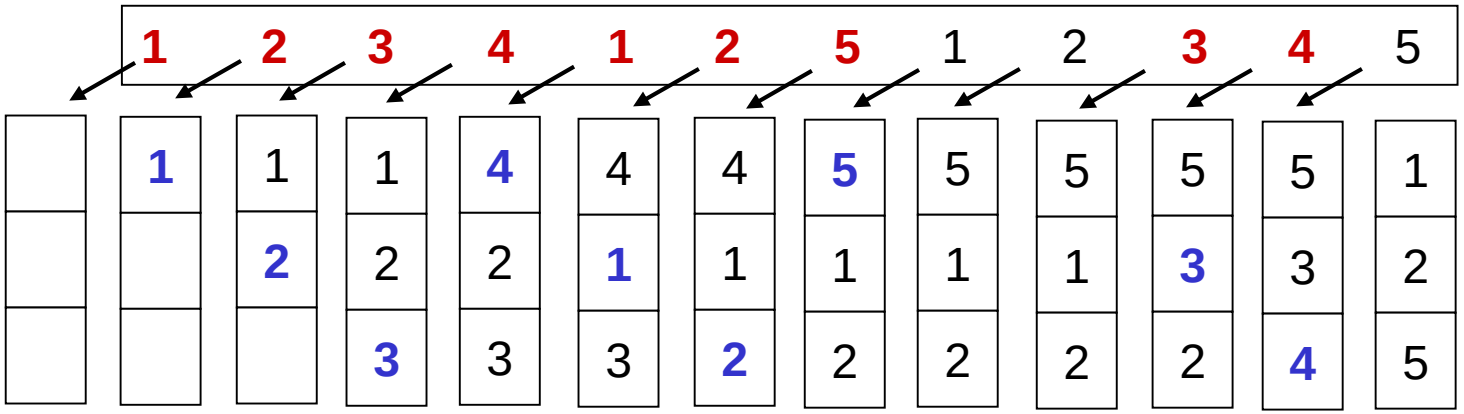
- **Example**
  - **Memory address sequence**
    - 0100, 0232, 0301, 0412, 0102, 0203, 0504, …
  - **Reference string = page number sequence**
    - 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

First-In-First-Out (FIFO) Algorithm

먼저 들어온걸 먼저 쫓아냄 (Swap out)

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frames (3 pages can be in memory at a time)

메모리를 배정할 때 최대 3개 page까지 physical memory에 배정할수 있다

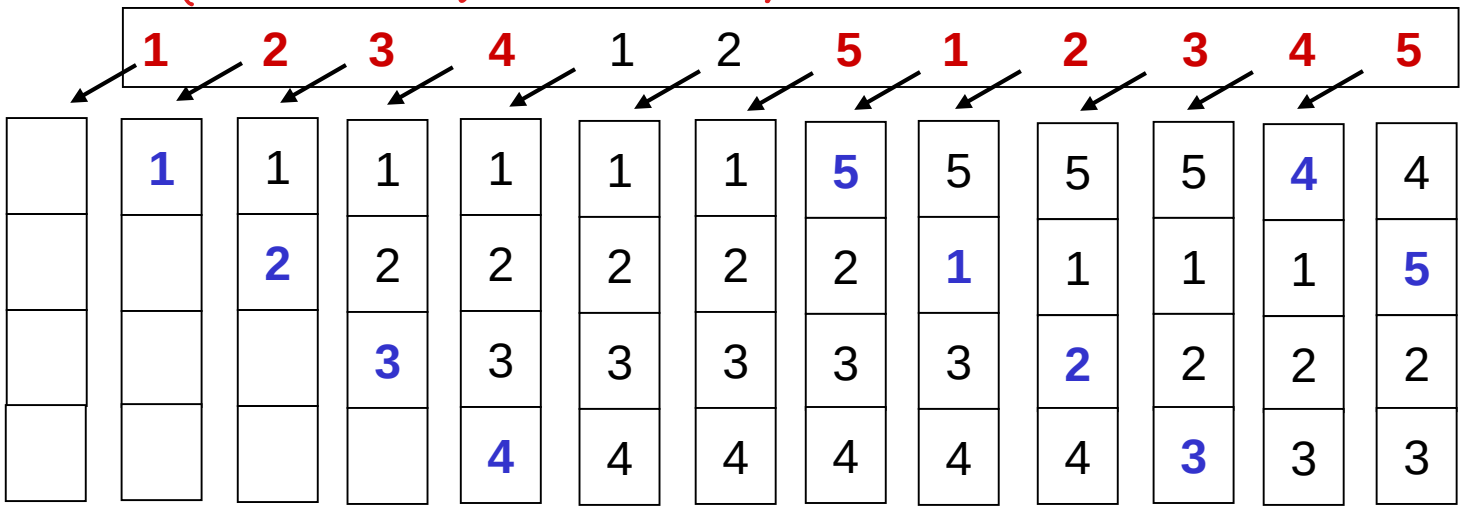


- There are 9 page faults with 3 frames

First-In-First-Out (FIFO) Algorithm

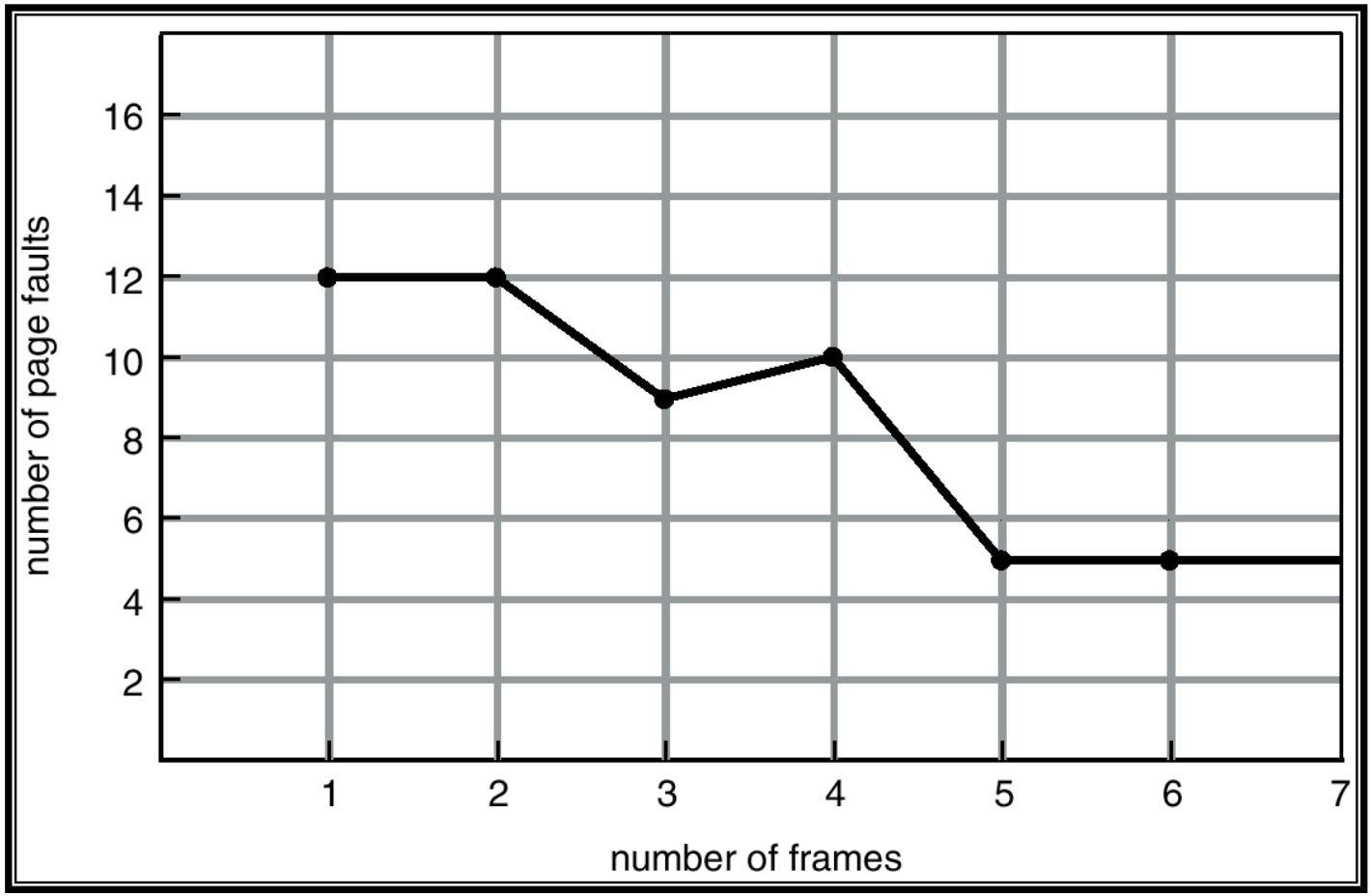
- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 4 frames (4 pages can be in memory at a time)

(빨간색 : page fault가 나는 경우)

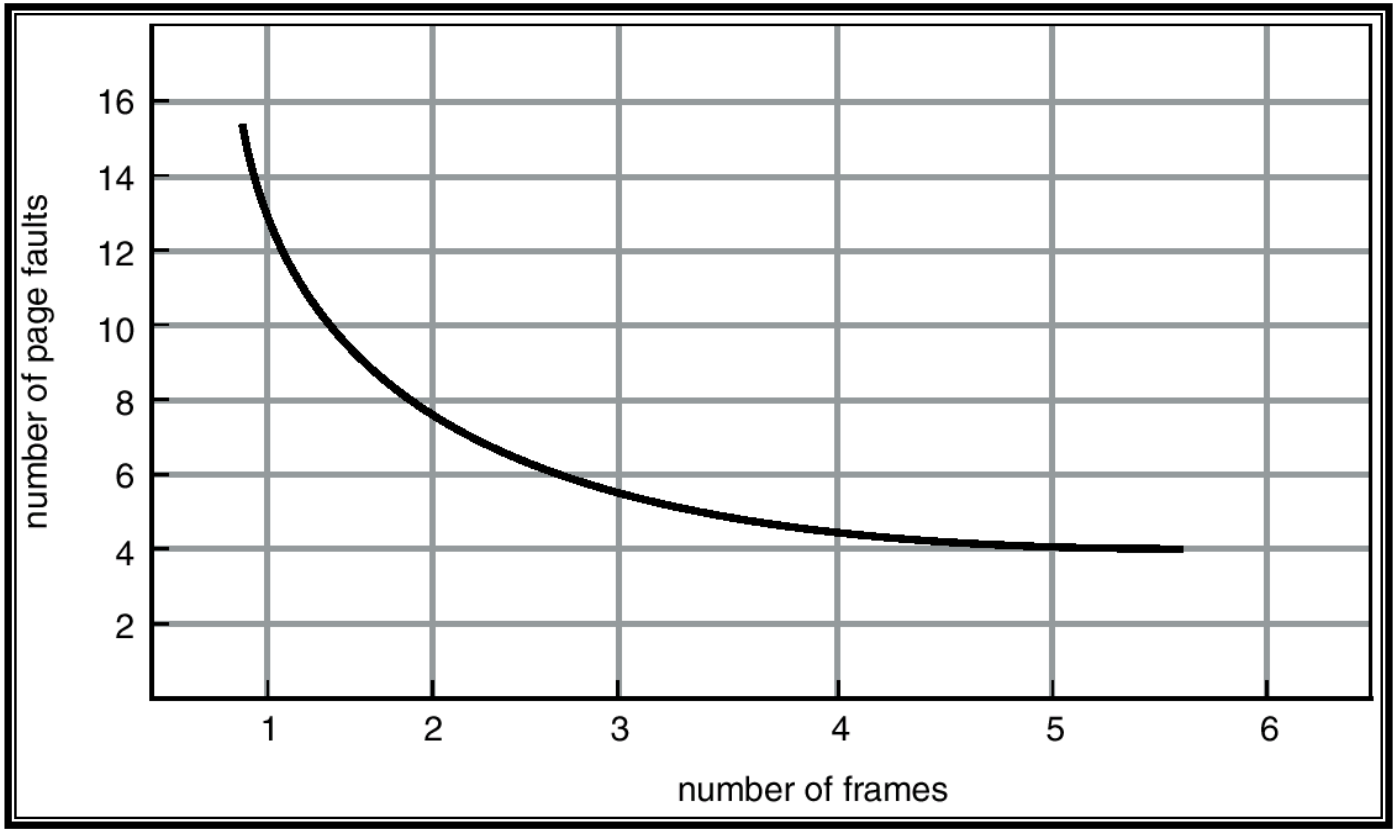


- There are 10 page faults with 4 frames
  - Belady's Anomaly (이상현상) ← physical frame이 하나 늘었지만 page faults가 더 많이 남

FIFO Illustrating Belady's Anomaly



General Behavior

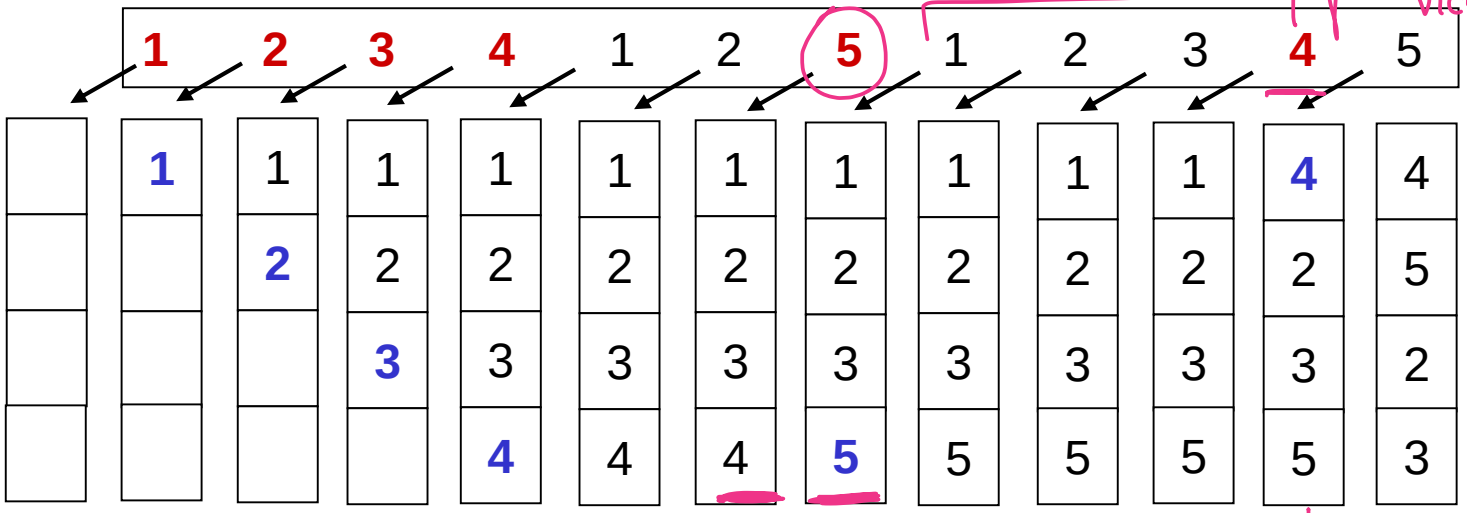


page fault의 횟수 최소화

Optimal Algorithm

- Replace page that will not be used for longest period of time
- 4 frames example ⇒ victim으로 선택

제일 마지막에 참조되는 4번을 victim으로 선택



5번을 제외한 아무 프레임 victim으로 선택

- Now there are 6 page faults
- But this algorithm requires future knowledge about the reference string ⇒ 실제 사용은 어렵고, 다른 page 교체 알고리즘과 비교하는 용도로 사용
- It is used mainly for comparison studies

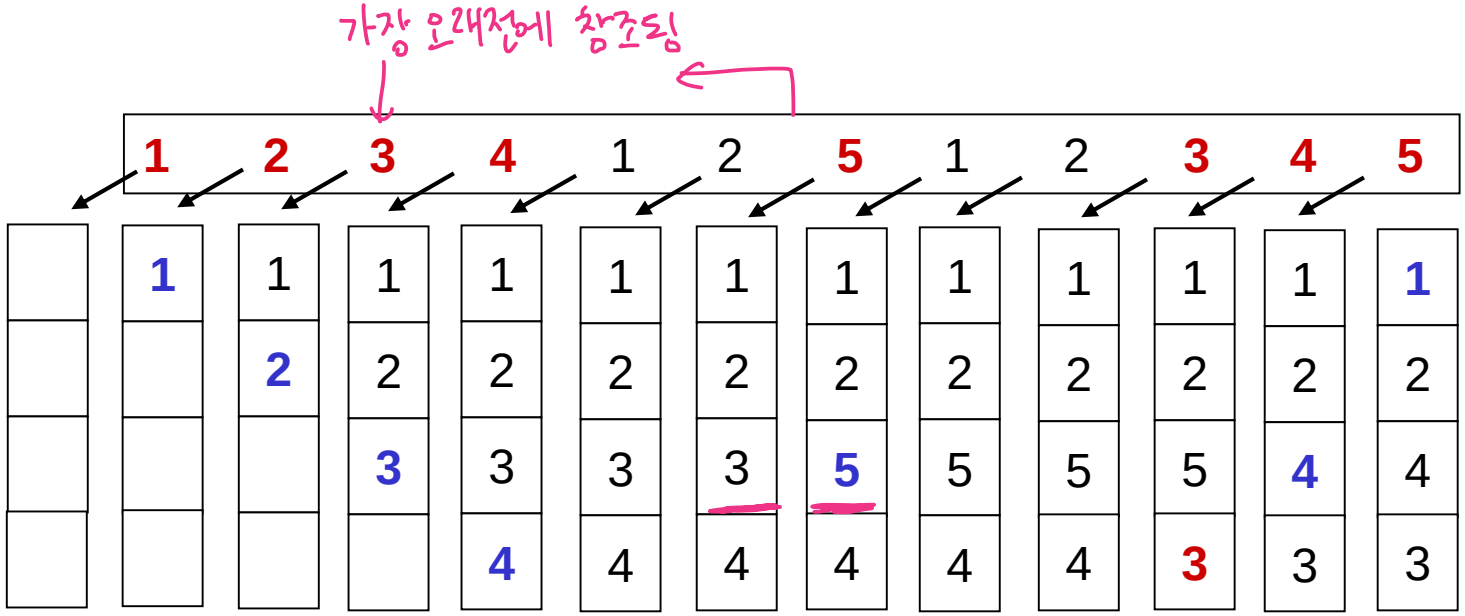
Least Recently Used (LRU) Algorithm

가장 오래전에 참조됐던 page frame을 swap out

- FIFO uses
  - The time when a page was brought into memory
- OPT uses
  - The time when a page is to be used
- LRU (Least Recently Used) uses
  - The recent past as an approximation of the near future
  - Chooses the page that has not been used for the longest period of time
  - Programs tend to have locality of reference
  - Optimal backward-looking algorithm

Least Recently Used (LRU) Algorithm

- 4 frames example



FIFO optimal

- Now there are 8 page faults (compare with 10 and 6)

Locality and LRU Performance

컴퓨터가 메모리 location을 참조할 때 반복적으로 참조하는 경향

- Locality of reference is the tendency of a processor to access the same set of memory locations repetitively over a short period of time
  - There are two basic types of reference locality – temporal and spatial locality
  - **Temporal locality** refers to the reuse of specific data, and/or resources, within a relatively small time duration

    어떤 메모리 주소를 참조했을 때 그 주소가 조만간 다시 참조될 확률이 높은 경향
  - **Spatial locality** refers to the use of data elements within relatively close storage locations

    어떤 주소가 참조됐을 때, 주변의 주소가 참조될 가능성이 높음
- **LRU is based on temporal locality**

LRU Implementation

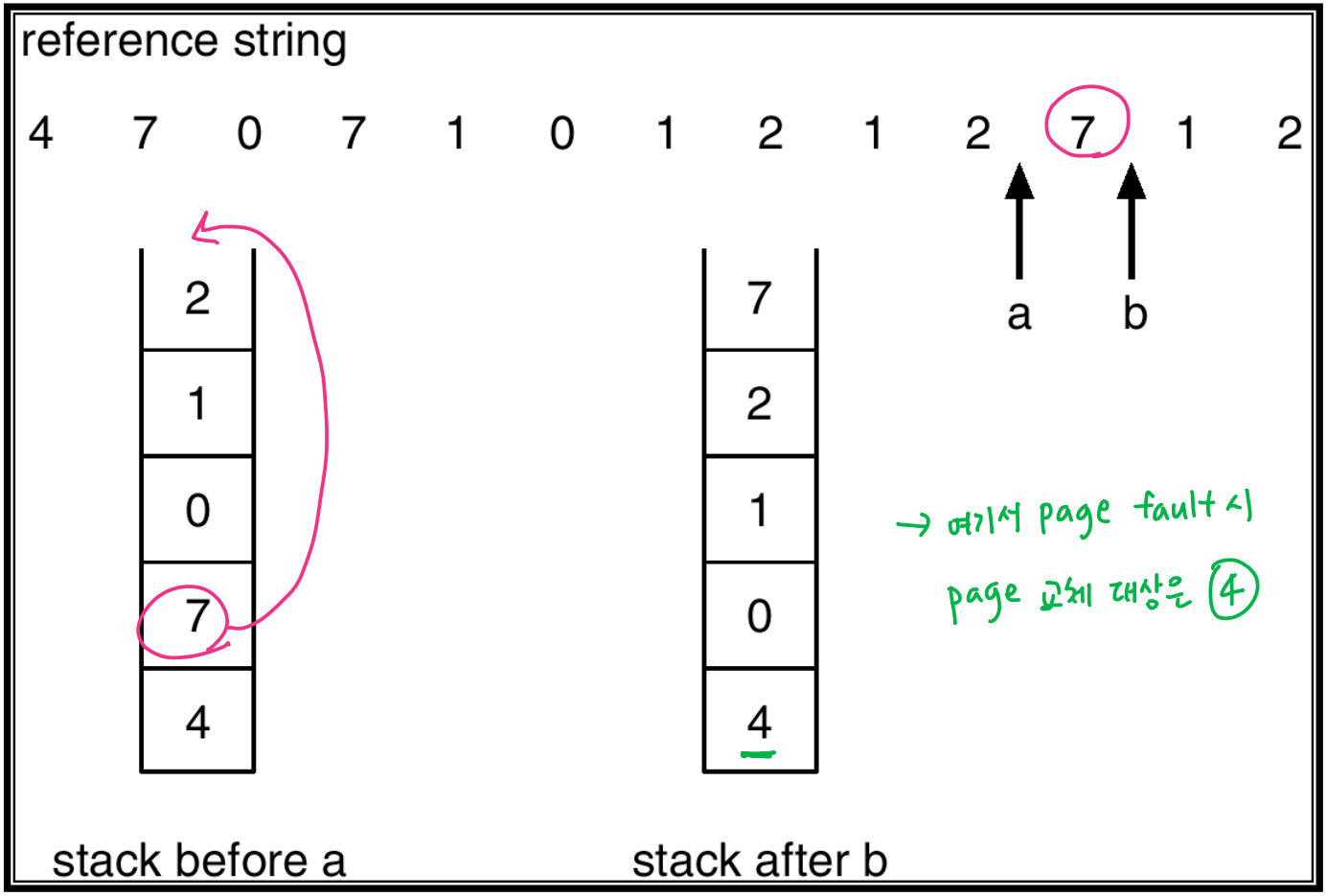
❒ Counters 페이지가 참조된 시각

- **Every page entry has a counter (time-of-use)**
- **Every time a page is referenced through this entry, copy the clock value into the counter**
- **Replace the page with the smallest time value**

❒ Stack 페이지 참조 순서

- **Keep a stack of page numbers (order-of-reference)**
- **Whenever a page referenced, move it to the top**
  - Top → most recently used
  - Bottom → least recently used
- **Best implemented by a doubly linked list**

Stack Implementation



LRU Implementation

- **LRU implementation is not conceivable without hardware assistance**
  - **The updating of the clock fields or stack must be done for every memory reference**
  - **It we were to use an interrupt for every reference, we would see unacceptable performance**

- **Few computer systems provide sufficient hardware support**

CPU에서 명령어를 실행할때마다 page 참조 발생

(시간값 기록/스택 위치 조작) 또한 CPU가 instruction을 처리하는 속도로 진행돼야함
Counters stack

→ 너무 빈번하게 발생하므로 순수하게 소프트웨어 적으로 구현해서는 성능 향상이 저하

→ special한 하드웨어의 도움 필요 → 아직은 X.

LRU Approximation Algorithms

- **Reference bit**
  - Many systems provide some help in the form of reference bit
  - With each page associate a bit, initially = 0
  - When page is referenced the bit is set to 1 by HW
  - Replace the one which is 0 (if one exists)
    - We do not know the order, however

- **Additional-Reference-Bits**
  - Keep an 8-bit byte for each page
  - At regular intervals (e.g., 100 milisec), the OS shifts the reference bit
  - 10000000 → 01000000 → 10100000 → 11010100
  - 11000000 has been used more recently than 01110111

참조발생

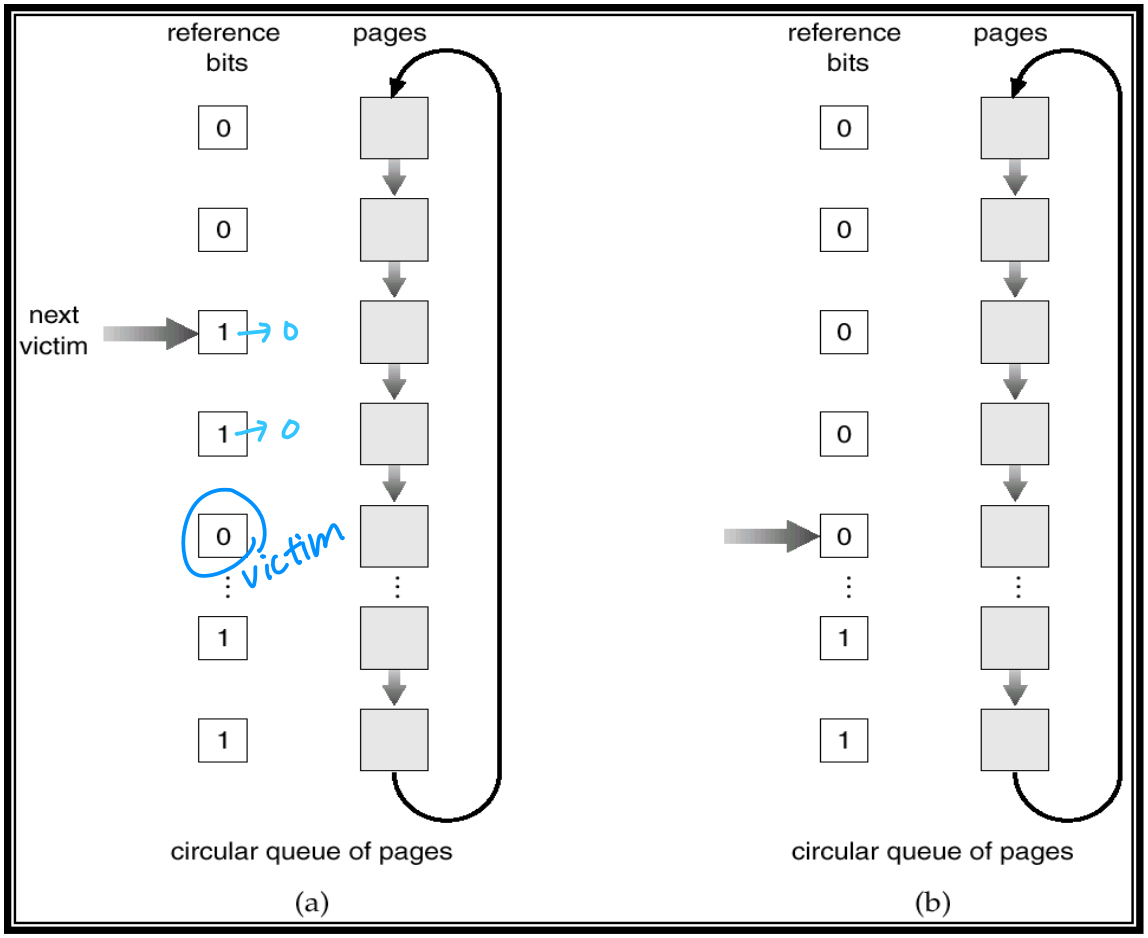
참조발생

victim

LRU Approximation Algorithms

- **Second chance**
  - **FIFO algorithm + reference bit**
  - **Need reference bit**
  - **If reference bit value is 0, replace the page**
  - **If reference bit value is 1**
    - Set reference bit 0 (← second chance 의 뜻)
    - Leave page in memory
    - Move onto the next FIFO page
  - **In the worst case, when all values are 1,**
    - The algorithm degenerates to FIFO

Second-Chance (Clock) Algorithm



Counting Algorithms

- Keep a counter of the number of references that have been made to each page

- LFU (Least Frequently Used) Algorithm
  - Replaces page with smallest count
  - Give preference to actively used pages

- MFU (Most Frequently Used) Algorithm
  - Replaces page with largest count
  - Based on the argument that the page with the smallest count was probably just brought in and has yet to be used

- Neither MFU nor LFU is common

참조된 시간의 간격을 봄
LRU와 달리
참조의 빈도를 가지고
page 교체를 수행.
LRU보단 성능 떨어짐

thank you!