Page Replacement Algorithms

Operating Systems

Virtual Memory Management

- Background
- Demand Paging
- Demand Segmentation
- Paging Considerations
- Page Replacement Algorithms
- Virtual Memory Policies

What is a page frame?

- When using paging, the main memory is partitioned into equal **fixed-size chunks** that are relatively small, and
- Each **process** is also divided into **small fixed-size chunks of the same size**.
- Then, the chunks of a process, known as **pages**,
- are assigned to available chunks of memory, known as **frames** or **page frames**.

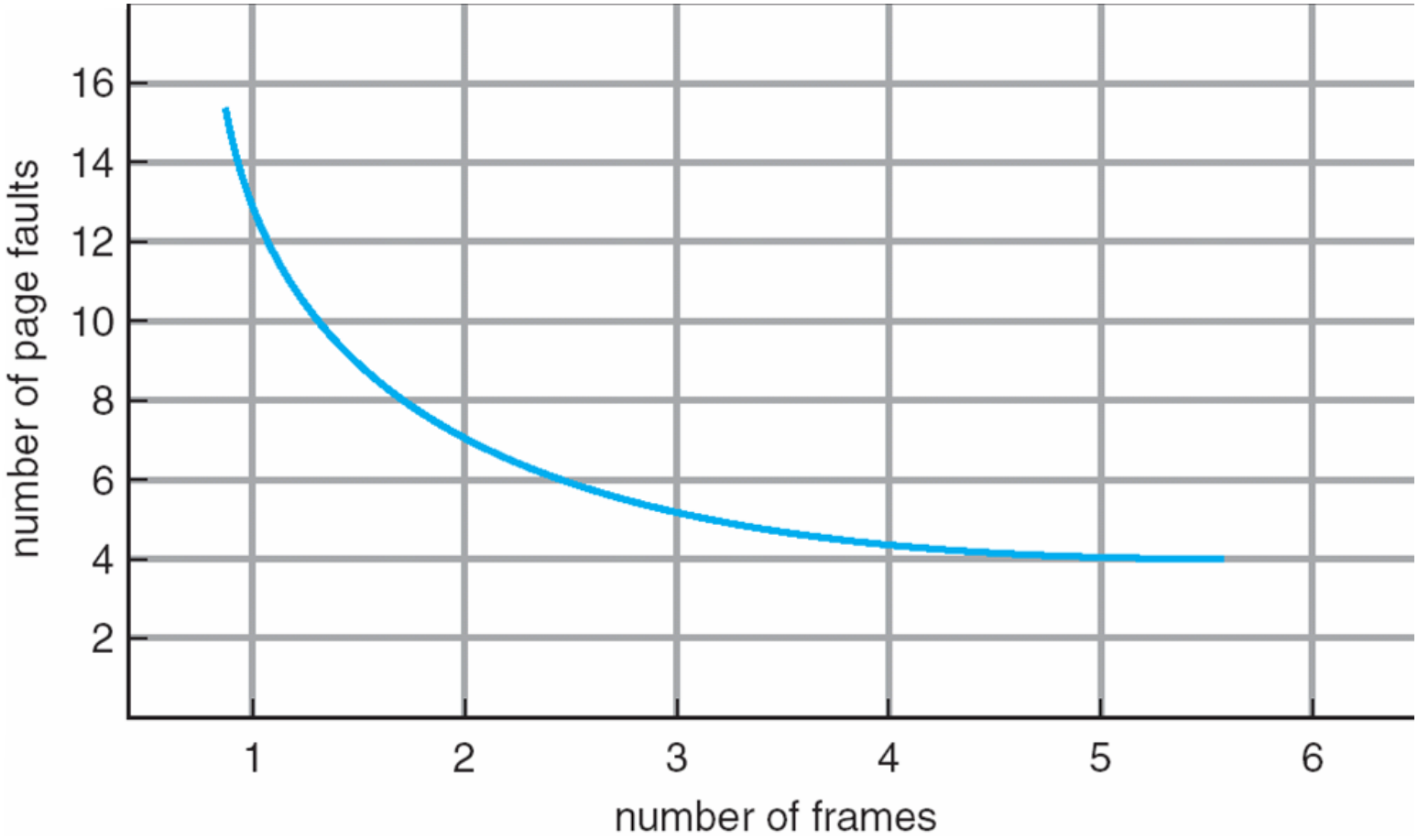
Paging:- In computer operating systems, paging is one of the memory-management schemes by which a computer can store and retrieve data from secondary storage for use in main memory.

- **What is a page table?**
  - A **page table** is the data structure used by a virtual memory system in a computer operating system to store the mapping between virtual addresses and physical addresses.
  - Virtual addresses are used by the accessing process, while physical addresses are used by the hardware or more specifically to the RAM.
- **Page fault:** The main functions of paging are performed when a program tries to access pages that are not currently mapped to physical memory (RAM). This situation is known as a page fault.

Page Replacement Algorithms

- Want lowest page-fault rate.
- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults and page replacements on that string.
- In all our examples, we use a few recurring reference strings.

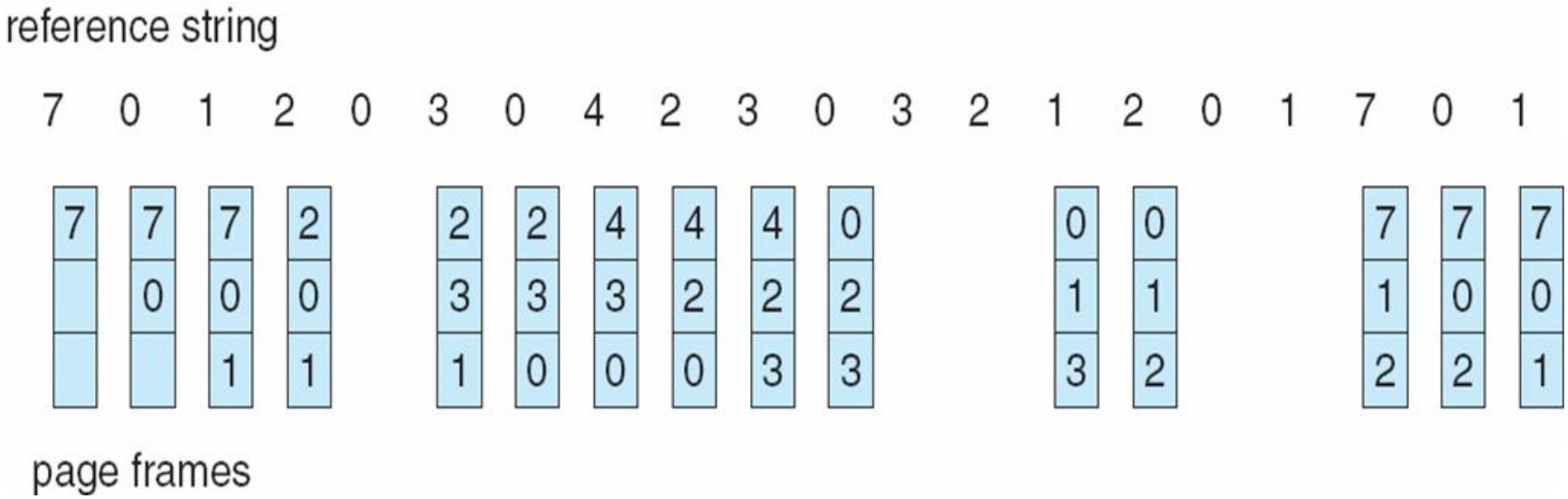
Graph of Page Faults vs. the Number of Frames



The FIFO Policy

- Treats page frames allocated to a process as a circular buffer:
  - When the buffer is full, the oldest page is replaced. Hence first-in, first-out:
    - A frequently used page is often the oldest, so it will be repeatedly paged out by FIFO.
  - Simple to implement:
    - requires only a pointer that circles through the page frames of the process.

FIFO Page Replacement



First-In-First-Out (FIFO) Algorithm

- Reference string: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- FIFO Replacement manifests Belady's Anomaly:
  - more frames ⇒ more page faults

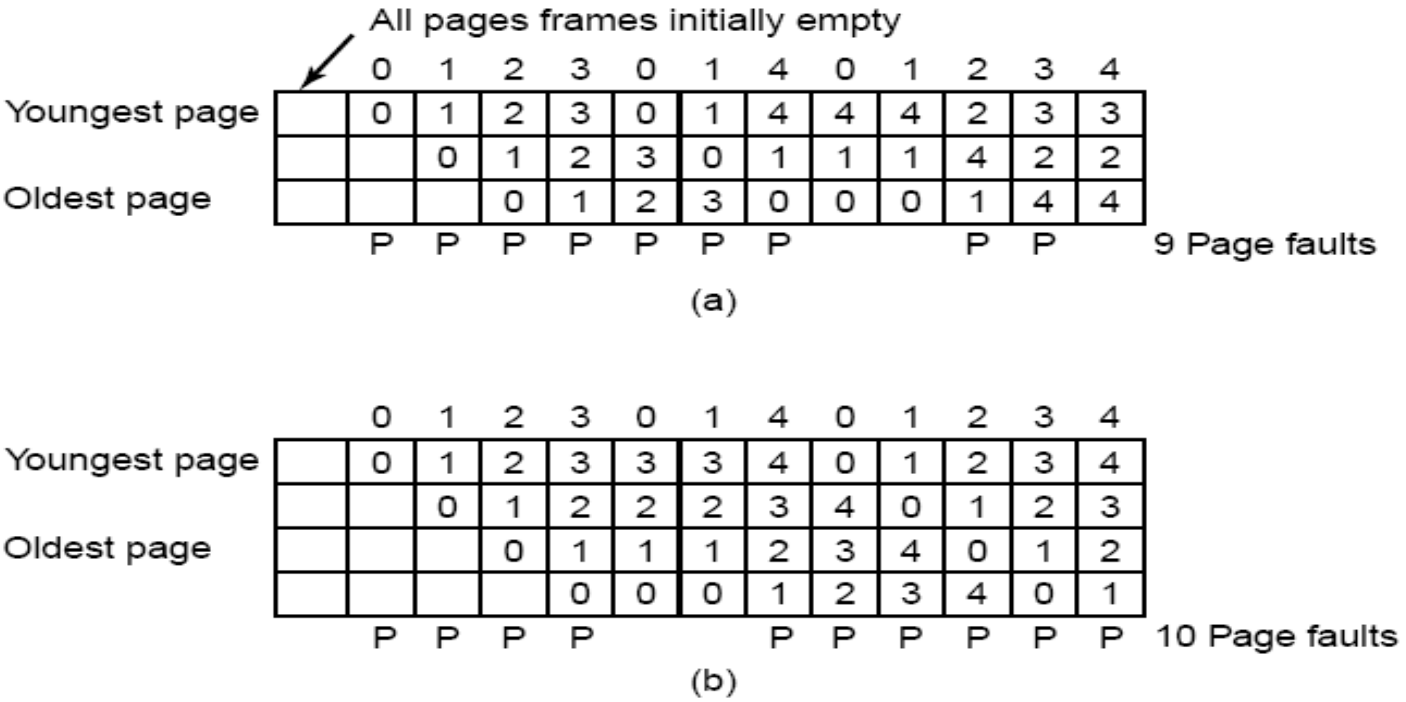
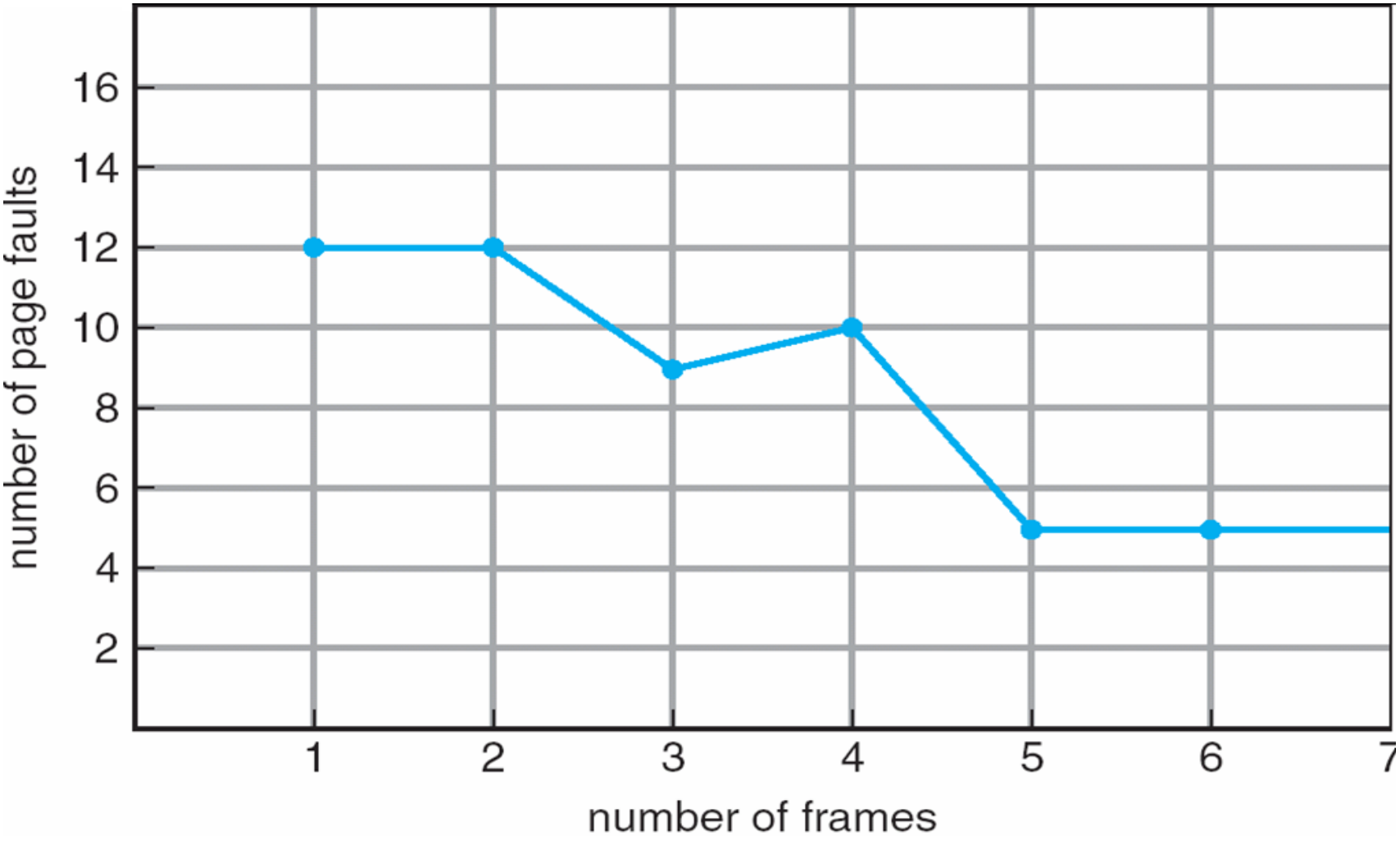


Fig. 4-24. Belady's anomaly. (a) FIFO with three page frames. (b) FIFO with four page frames. The *P*'s show which page references cause page faults.

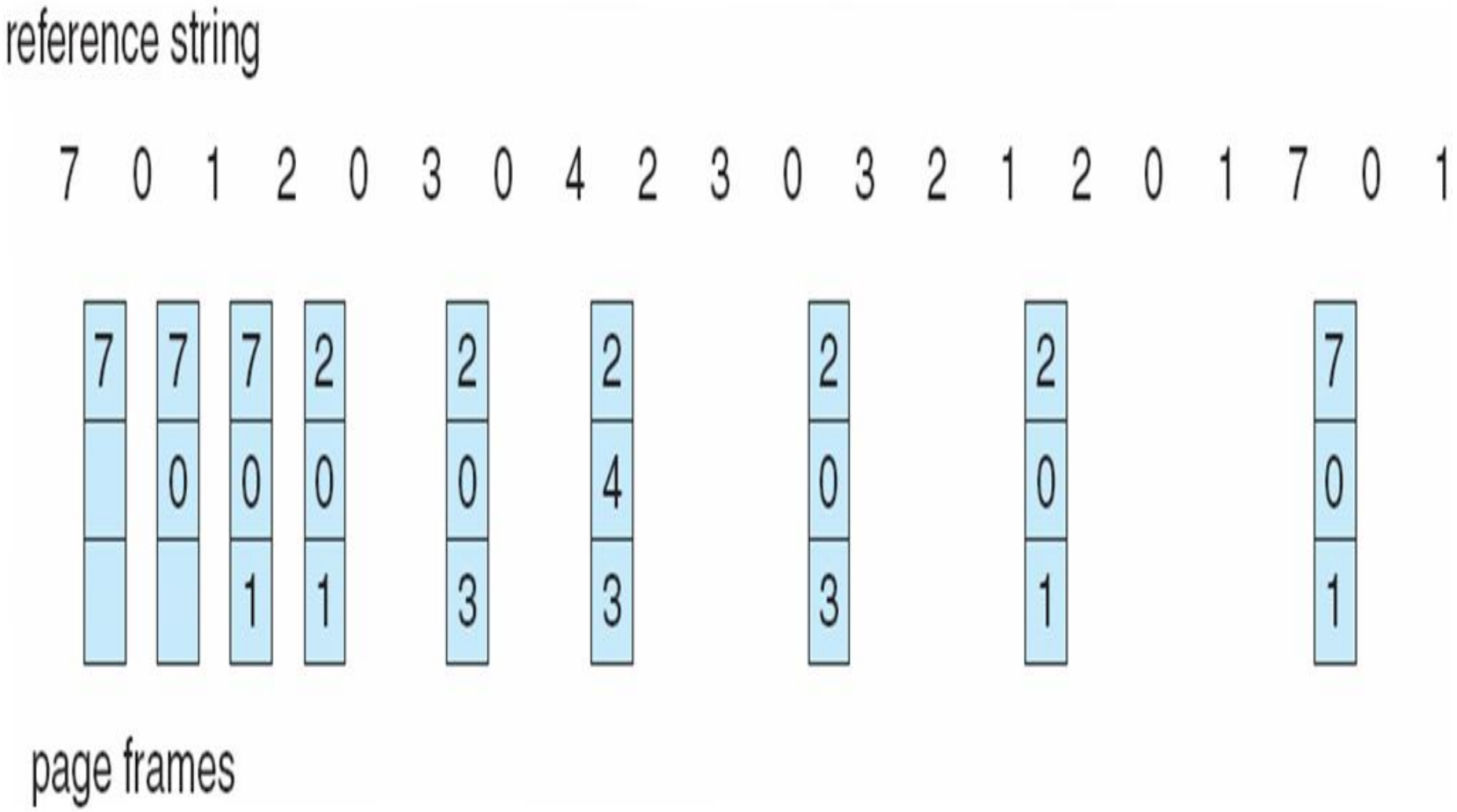
FIFO Illustrating Belady's Anomaly



Optimal Page Replacement

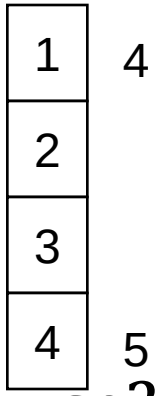
- The Optimal policy selects for replacement the page that will not be used for longest period of time.
- Impossible to implement (need to know the future) but serves as a standard to compare with the other algorithms we shall study.

Optimal Page Replacement



Optimal Algorithm

- Reference string : 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 4 frames example



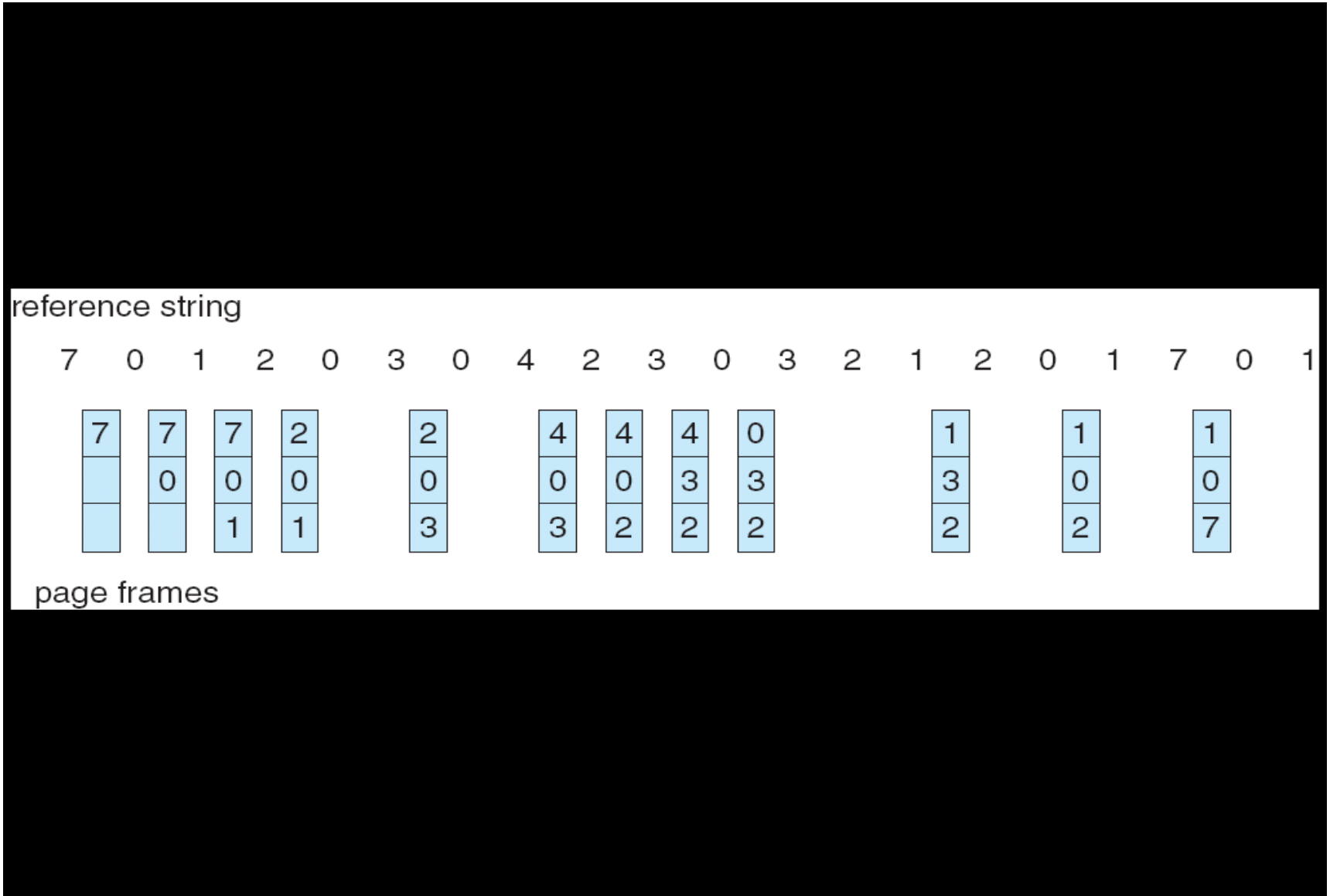
6 page faults

- How do you know future use? You don't!
- Used for measuring how well your algorithm performs.

The LRU Policy

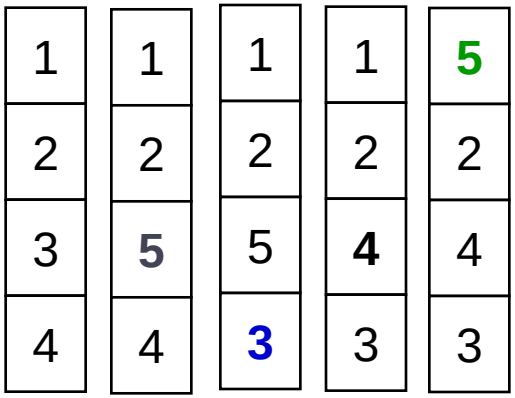
- Replaces the page that has not been referenced for the longest time:
  - By the principle of locality, this should be the page least likely to be referenced in the near future.
  - performs nearly as well as the optimal policy.

LRU Page Replacement



Least Recently Used (LRU) Algorithm

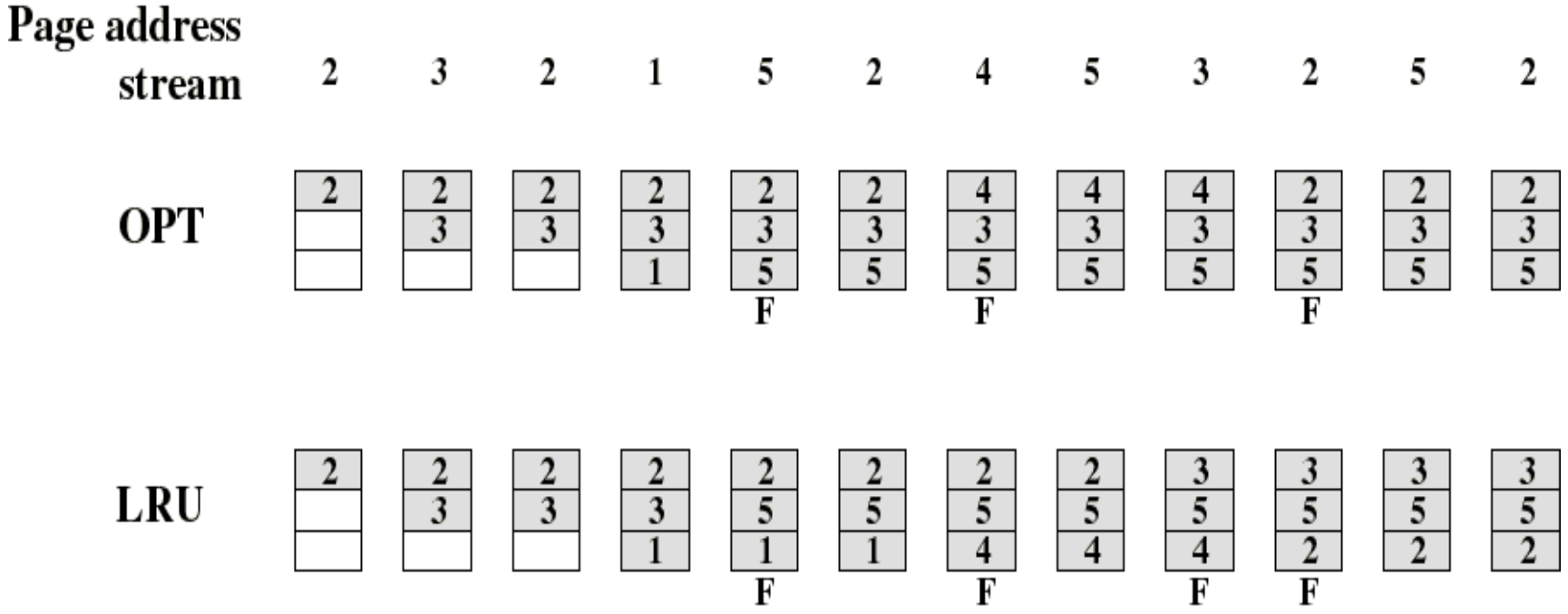
- Reference string: 1, 2, 3, 4, 1, 2, **5**, 1, 2, **3**, **4**, **5**



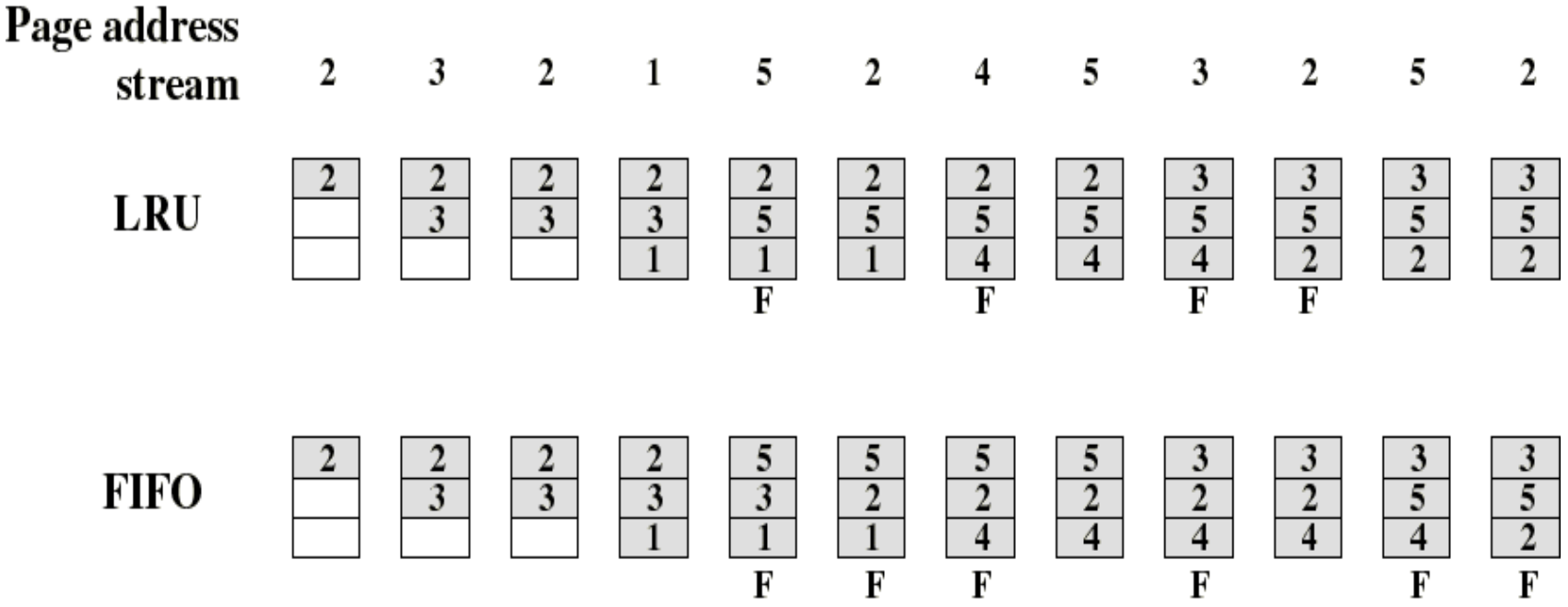
8 page faults

Comparison of OPT with LRU

- Example: A process of 5 pages with an OS that fixes the resident set size to 3.



Comparison of FIFO with LRU



- LRU recognizes that pages 2 and 5 are referenced more frequently than others but FIFO does not.

Implementation of the LRU Policy

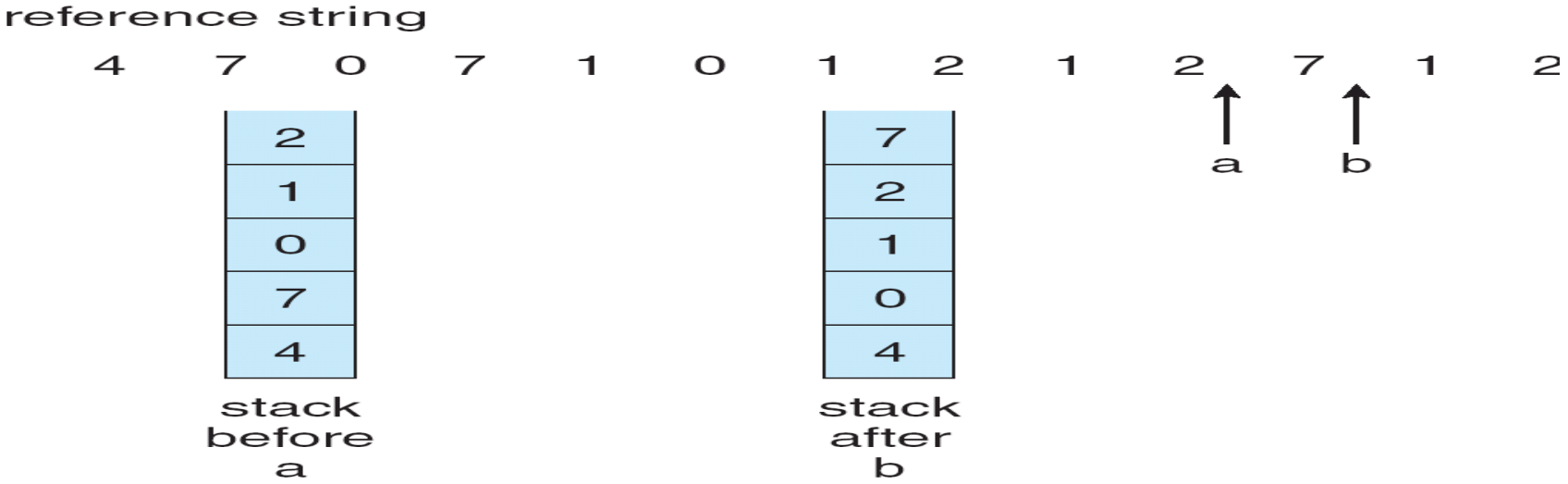
- Each page could be tagged (in the page table entry) with the time at each memory reference.
-
- The LRU page is the one with the smallest time value (needs to be searched at each page fault).
- This would require expensive hardware and a great deal of overhead.
- Consequently very few computer systems provide sufficient hardware support for true LRU replacement policy.
- Other algorithms are used instead.

LRU Implementations

- Counter implementation:
  - Every page entry has a counter; every time a page is referenced through this entry, copy the clock into the counter.
  - When a page needs to be changed, look at the counters to determine which are to change.
- Stack implementation – keep a stack of page numbers in a double link form:
  - Page referenced:
    - move it to the top
    - requires 6 pointers to be changed
  - No search for replacement.

Use of a stack to implement LRU

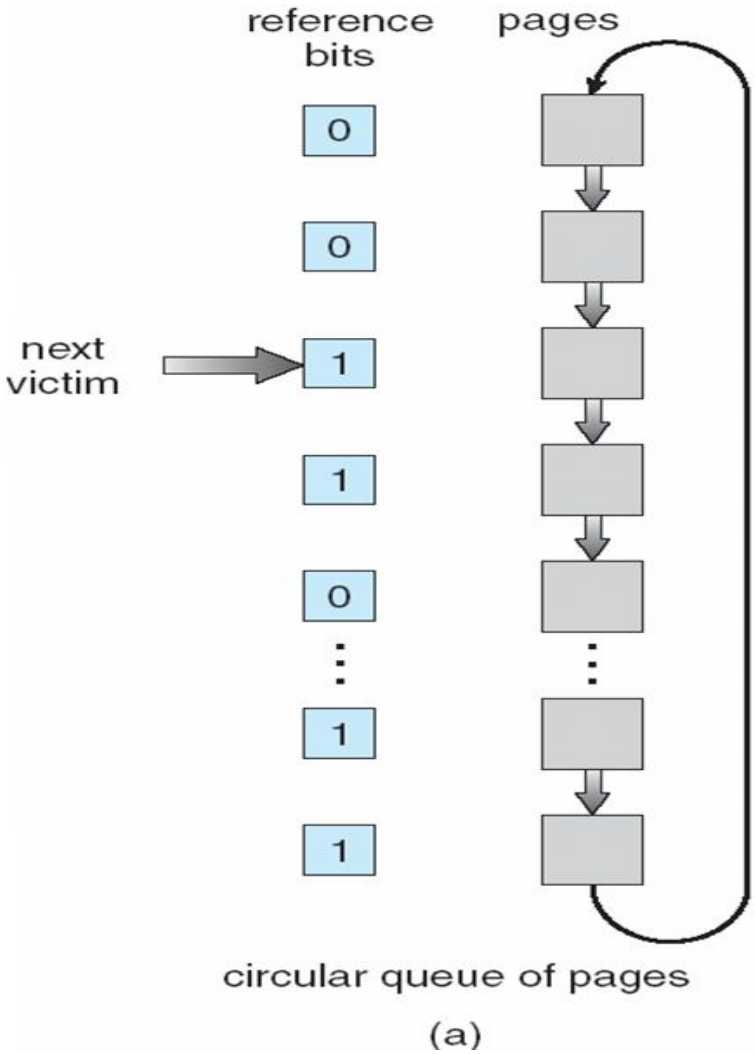
- Stack implementation – keep a stack of page numbers in a double link form:
  - Page referenced:
    - move it to the top
    - requires 6 pointers to be changed
  - No search for replacement – always take the bottom one.

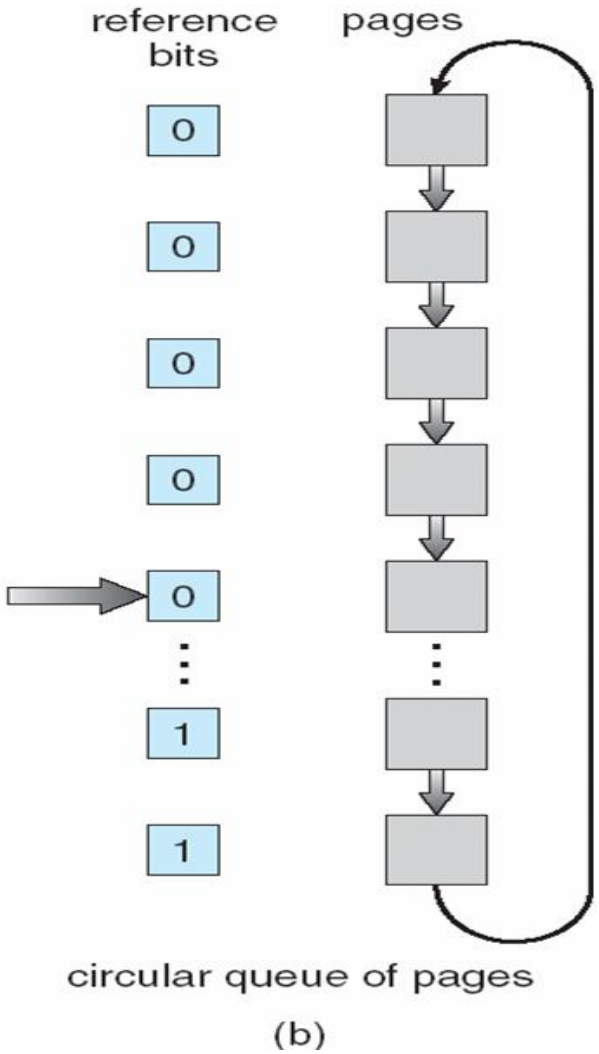


The Clock (Second Chance) Policy

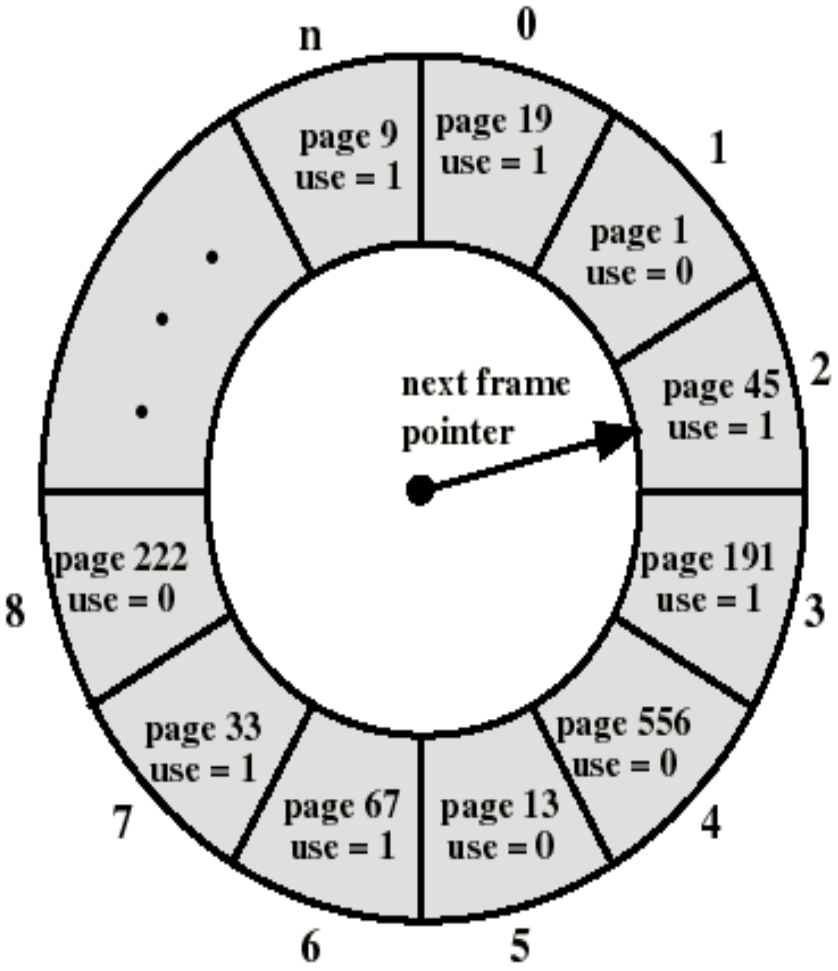
- The set of frames candidate for replacement is considered as a circular buffer.
- When a page is replaced, a pointer is set to point to the next frame in buffer.
- A reference bit for each frame is set to 1 whenever:
  - a page is first loaded into the frame.
  - the corresponding page is referenced.
- When it is time to replace a page, the first frame encountered with the reference bit set to 0 is replaced:
  - During the search for replacement, each reference bit set to 1 is changed to 0.

Clock Page-Replacement Algorithm

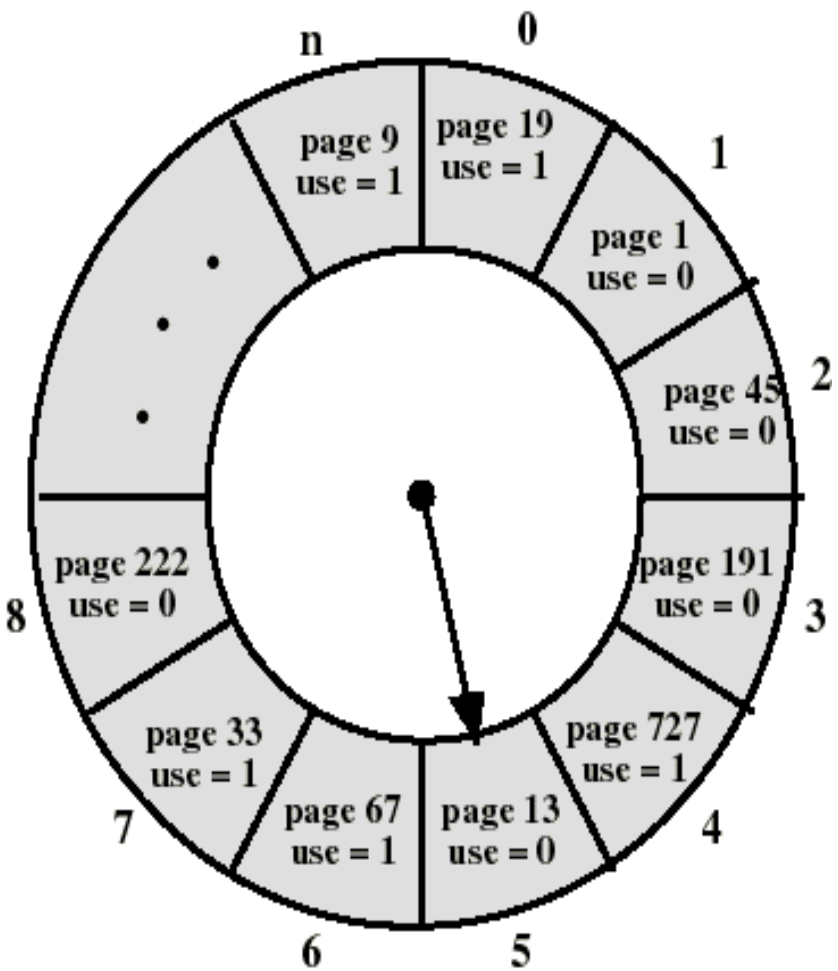




The Clock Policy: Another Example

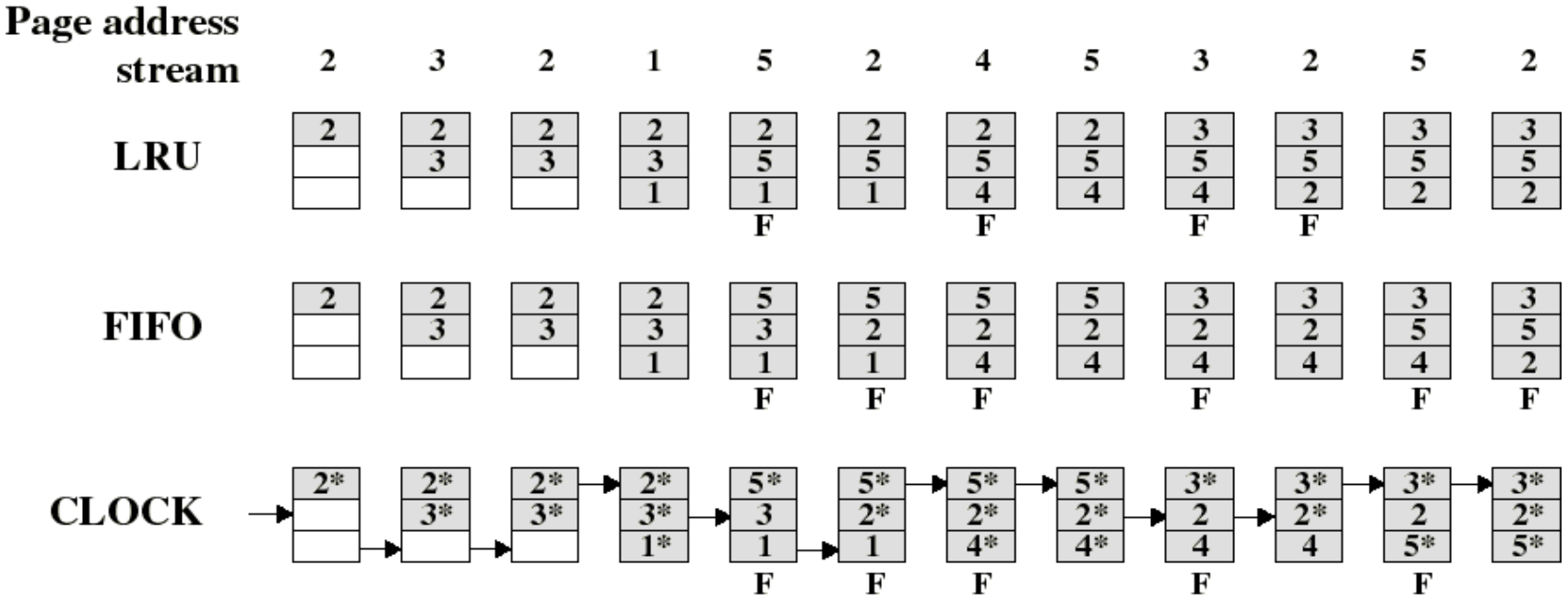


(a) State of buffer just prior to a page replacement



(b) State of buffer just after the next page replacement

Comparison of Clock with FIFO and LRU (1)

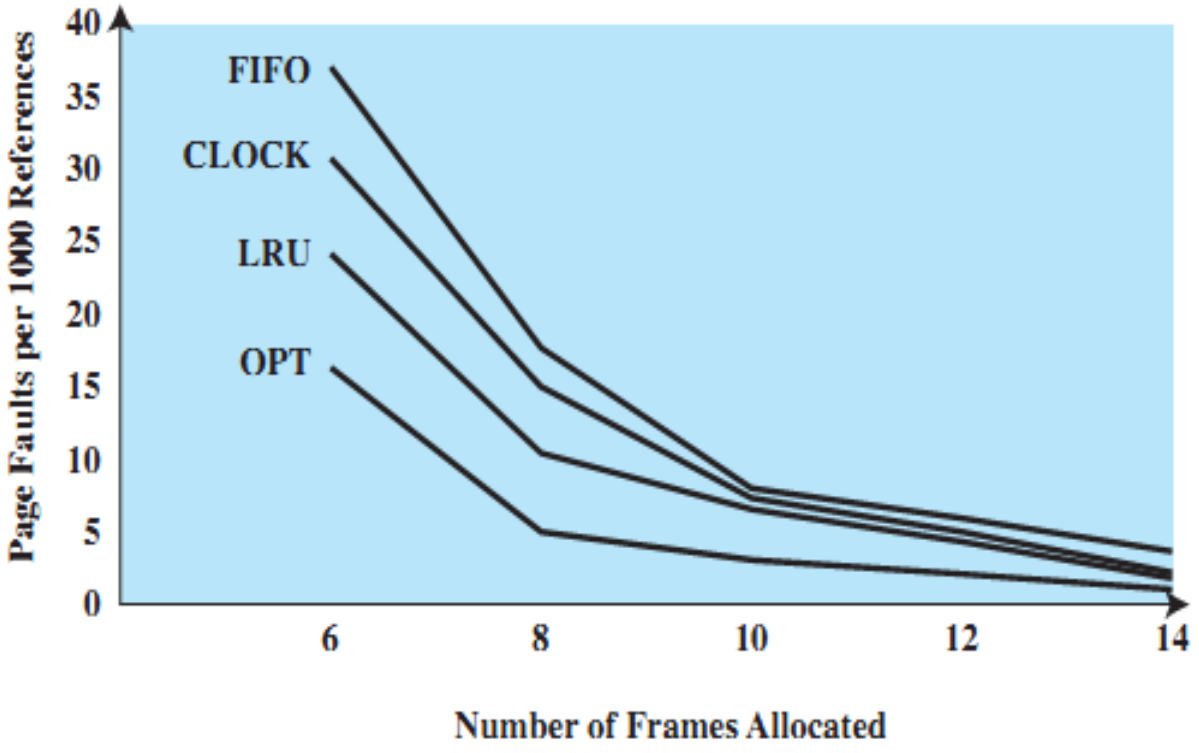


- Asterisk indicates that the corresponding use bit is set to **1.**
- The arrow indicates the current position of the pointer.
- Note that the clock policy is adept at protecting frames 2 and 5 from replacement.

Comparison of Clock with FIFO and LRU (2)

- Numerical experiments tend to show that performance of Clock is close to that of LRU.
- Experiments have been performed when the number of frames allocated to each process is fixed and when pages local to the page-fault process are considered for replacement:
  - When few (6 to 8) frames are allocated per process, there is almost a factor of 2 of page faults between LRU and FIFO.
  - This factor reduces close to 1 when several (more than 12) frames are allocated. (But then more main memory is needed to support the same level of multiprogramming).

Fixed-Allocation, Local Page Replacement



Counting-based Algorithms

- Keep a counter of the number of references that have been made to each page.
- Two possibilities: Least/Most Frequently Used (LFU/MFU).
- LFU Algorithm: replaces page with smallest count; others were and will be used more.
- MFU Algorithm: based on the argument that the page with the smallest count was probably just brought in and has yet to be used.