CSCI 3753: Operating Systems Spring 2024

Dylan Sain

Department of Computer Science

University of Colorado Boulder

Week 10: Page Replacement Policies

Reminder: Office Hours

- Office hours:
  - **Wed 10:00am – 11:00am and Thurs 12:30 PM – 2:30 PM**
  - Zoom link: https://cuboulder.zoom.us/j/2163683260
  - CSEL on Thursdays (message me on slack if you cannot find me)
  - Available other times as needed, send me an email or message on slack to make an appointment

Virtual Memory

- Keep only a few pages in memory, rest on disk
- On-demand paging: retrieve a page when needed
- Page fault
  - A referenced page is not loaded in memory
  - OS blocks the process and retrieves the referenced page
  - Significant performance overhead – need to keep page fault frequency low, e.g. less than 1 in 107 for overhead <10%

→ Page replacement algorithms

Page Replacement

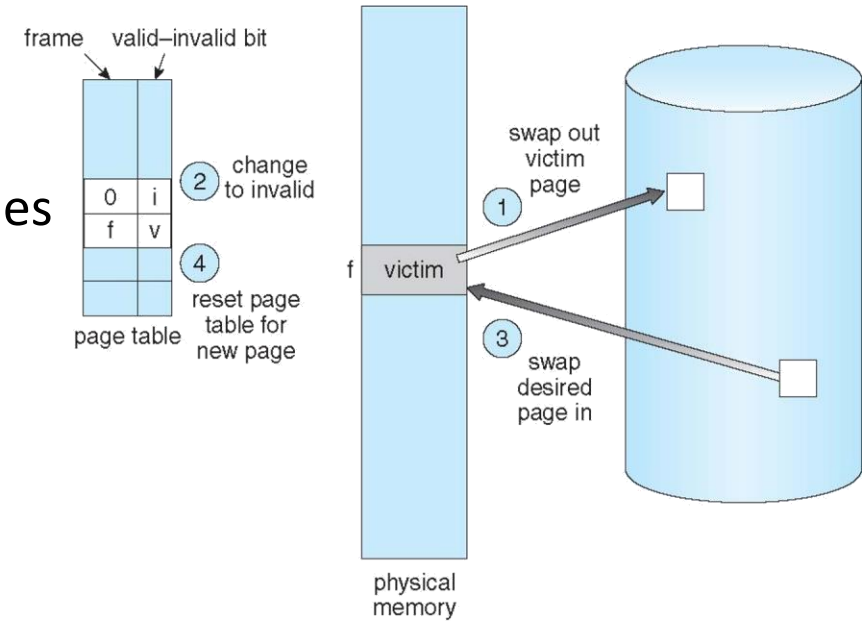
- Page replacement occurs when:
  - A page fault occurs and we need to bring the desired page into memory
  - There are NO free frames.

- Page replacement – find some page in memory, but not really in use, page it out
  - Algorithm – decide which frame to free
  - Performance – want an algorithm which will result in minimum number of page faults

- Same page may be brought into memory several times

Basic Page Replacing Process

1. Find the location of the desired page on disk
2. Find a free frame:
   - If there is a free frame, use it
   - If there is no free frame, use a page replacement algorithm:
     - Select a **victim frame**
     - Write victim frame to disk if dirty
3. Bring the desired page into the (newly) free frame; update the page and frame tables
3. Continue the process by restarting the instruction that caused the trap



Page Replacement Policies

Goal: Want lowest page-fault rate on both first access and re-access

- FIFO
- OPT
- LRU (least recently used)

→ Evaluation

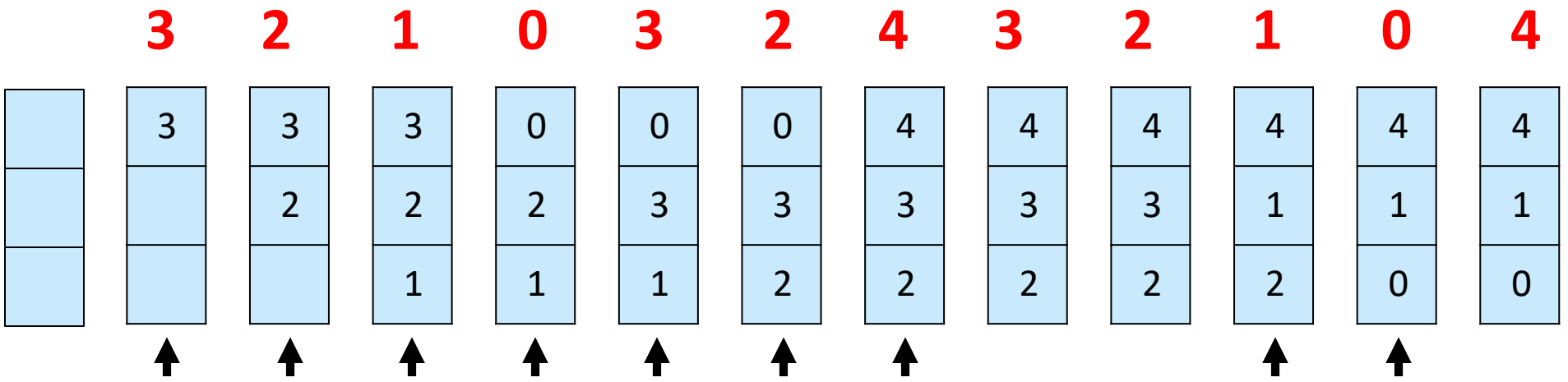
- Parameters: algorithm, page reference string, # of memory frames
- Algorithm with lowest # of page faults is most desirable

First-In-First-Out (FIFO) Algorithm

- Reference string:

3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4

- 3 frames (3 pages can be in memory at a time per process)



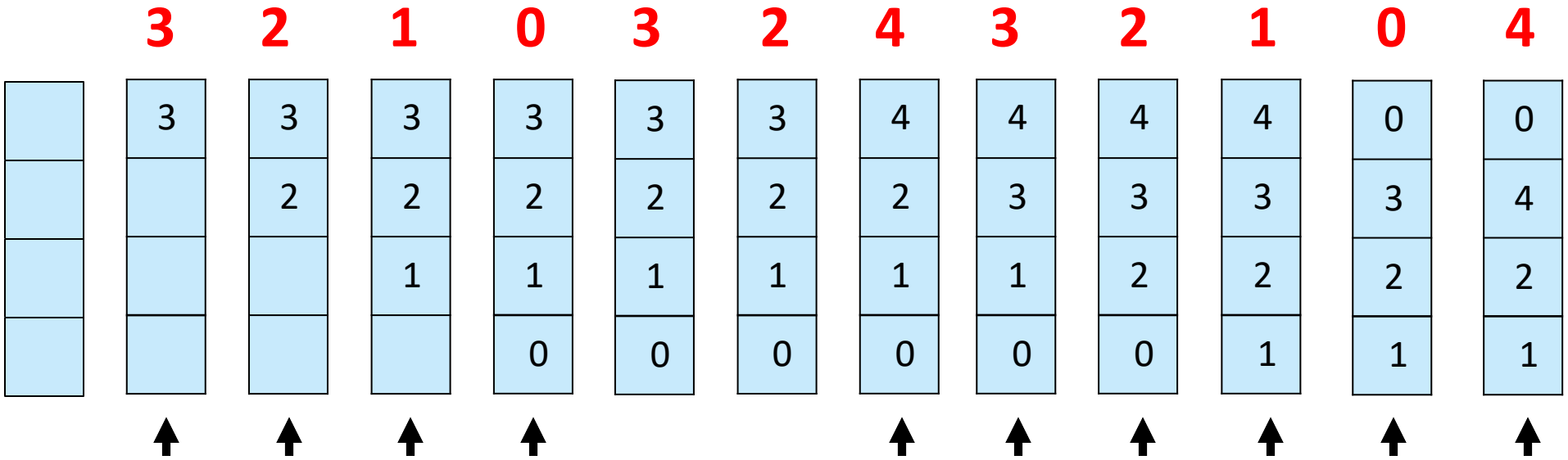
9 page faults

First-In-First-Out (FIFO) Algorithm

- Reference string:

3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4

- 4 frames (4 pages can be in memory at a time per process)



10 page faults

First-In-First-Out (FIFO) Algorithm

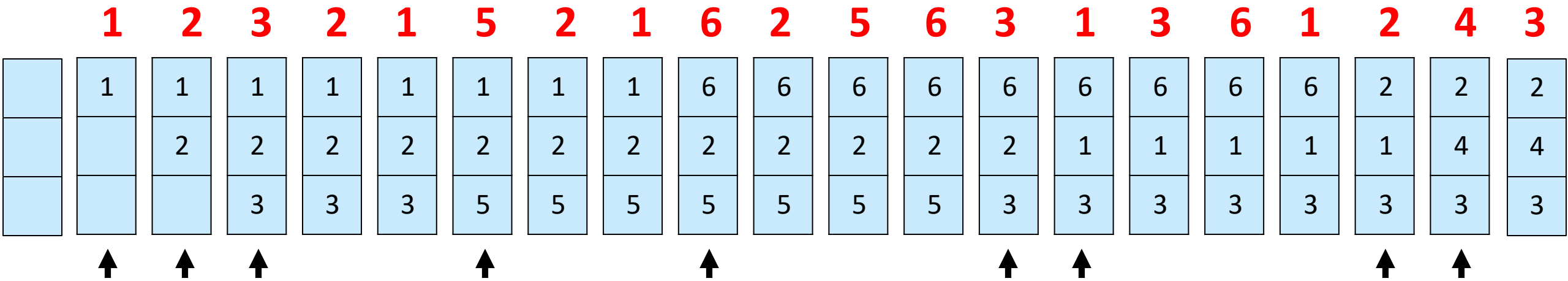
- FIFO is easy to understand and implement
- Performance can be poor
  - In the worst case, each page that is paged out could be the one that is referenced next, leading to a high page fault rate
  - Ideally, keep around the pages that are about to be used next – this is the basis of the OPT algorithm in the next slide

Optimal (OPT) Algorithm

- Reference string:

1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3

- 3 frames (3 pages can be in memory at a time per process)



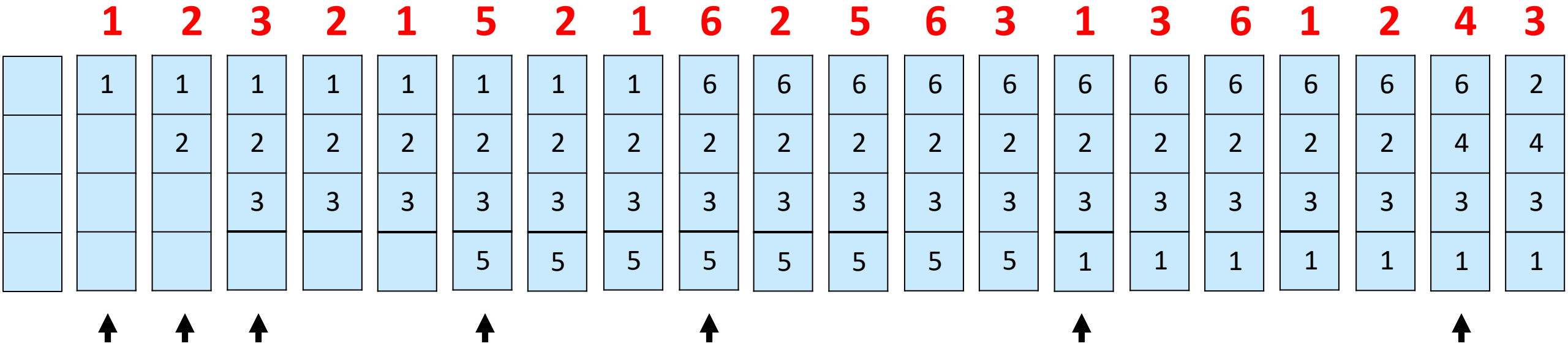
9 page faults

Optimal (OPT) Algorithm

- Reference string:

1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3

- 4 frames (4 pages can be in memory at a time per process)



7 page faults

Optimal (OPT) Algorithm

- Replace the page that will not be referenced for the longest time
- Guarantees the lowest page-fault rate
- Problem: requires future knowledge

Least Recently Used (LRU) Algorithm

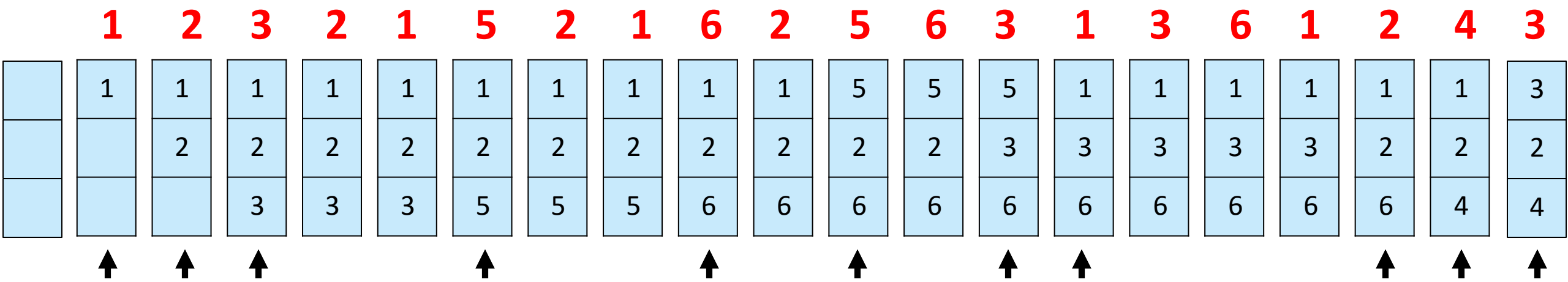
- Use the past to predict the future
  - If a page wasn't used recently, then it is unlikely to be used again in the near future
  - If a page was used recently, then it is likely to be used again in the near future
  - So select a victim that was least recently used

Least Recently Used (LRU) Algorithm

- Reference string:

1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3

- 3 frames (3 pages can be in memory at a time per process)



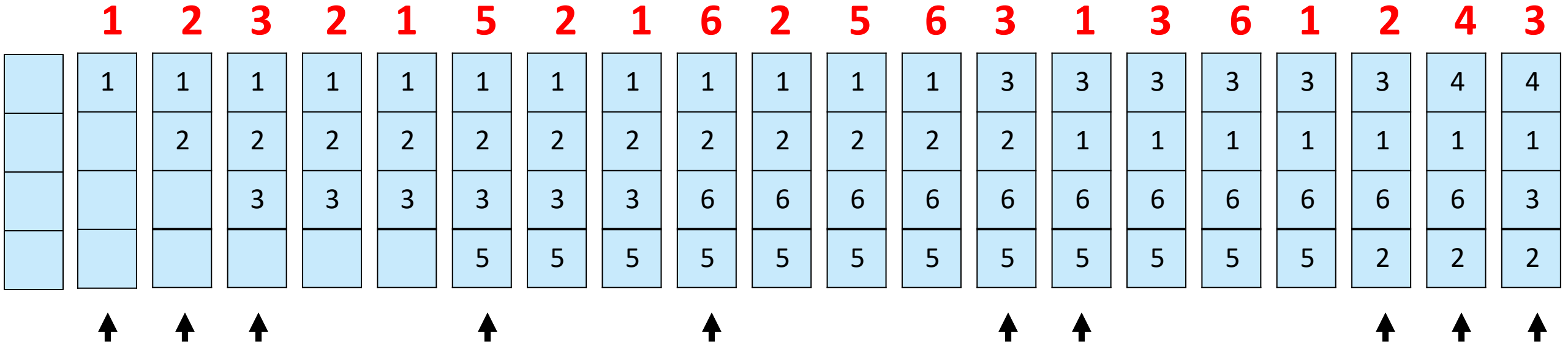
11 page faults

Least Recently Used (LRU) Algorithm

- Reference string:

1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3

- 4 frames (4 pages can be in memory at a time per process)



10 page faults