CSCI-3753: Operating Systems
Fall 2019

Abigail Fernandes

Department of Computer Science

University of Colorado Boulder

Week 10
> Page Replacement

Visual Metaphor

Operating Systems and Toy Shops each have memory (part) management systems

- Uses intelligently sized containers
  - Crates of toy carts
- Not all parts are needed at once
  - Toy orders are completed in stages
- Optimized for performance
  - Reduce wait time for parts
  - ➔ Make more toys! ☺

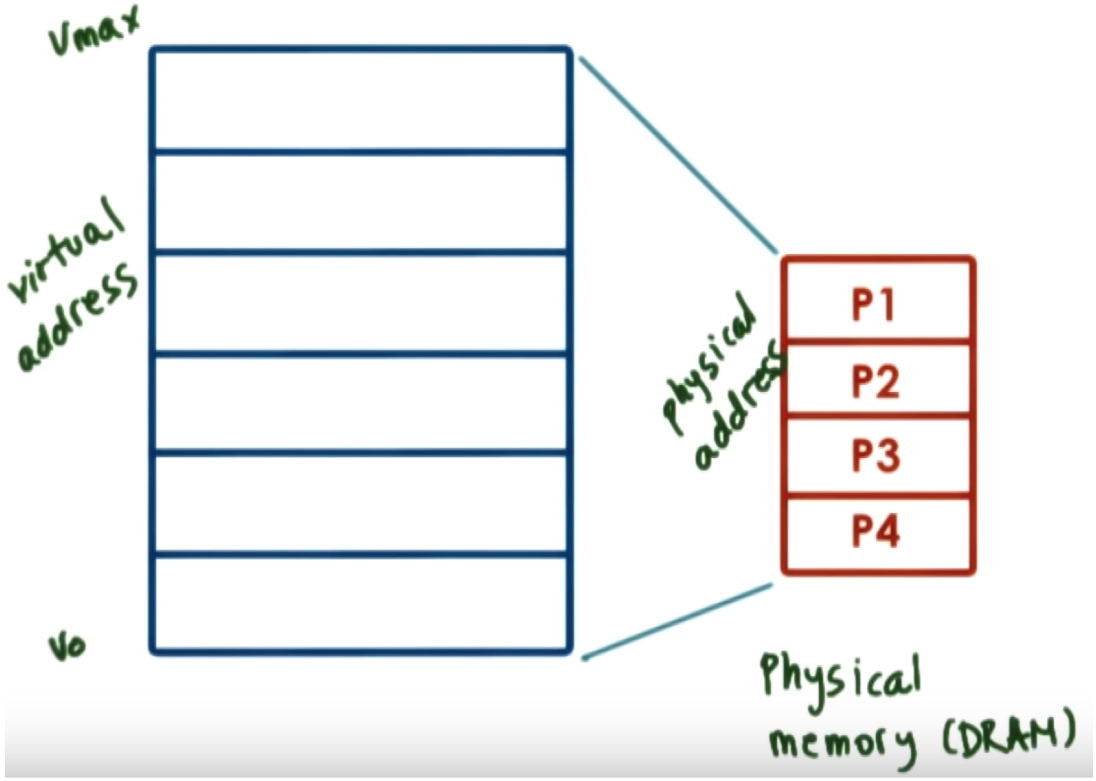
Visual Metaphor

Operating Systems and Toy Shops each have memory (part) management systems

- Uses intelligently sized containers
  - Memory pages or segments
- Not all parts are needed at once
  - Task operate on subset of memory
- Optimized for performance
  - Reduce time to access state in memory
  - ➔ Better performance! ☺

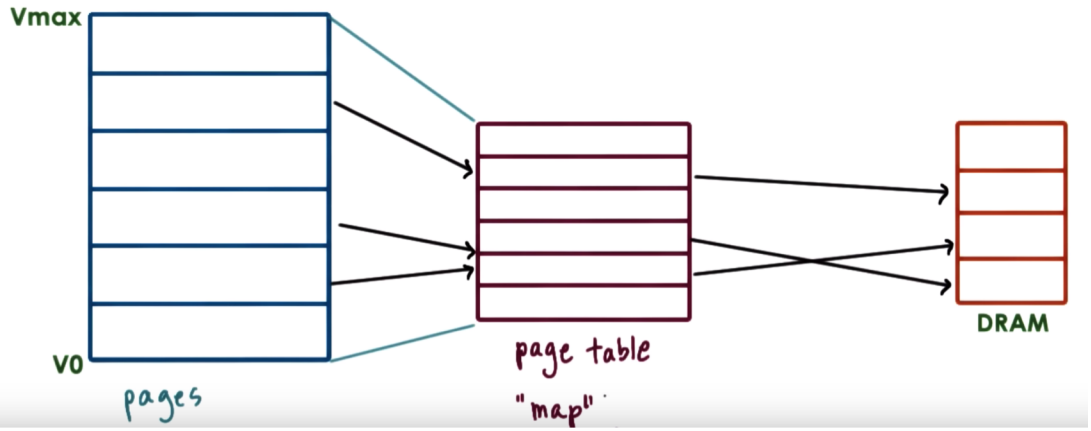
- Uses intelligently sized containers
  - Crates of toy carts
- Not all parts are needed at once
  - Toy orders are completed in stages
- Optimized for performance
  - Reduce wait time for parts
  - ➔ Make more toys! ☺

Memory Management (Page based)



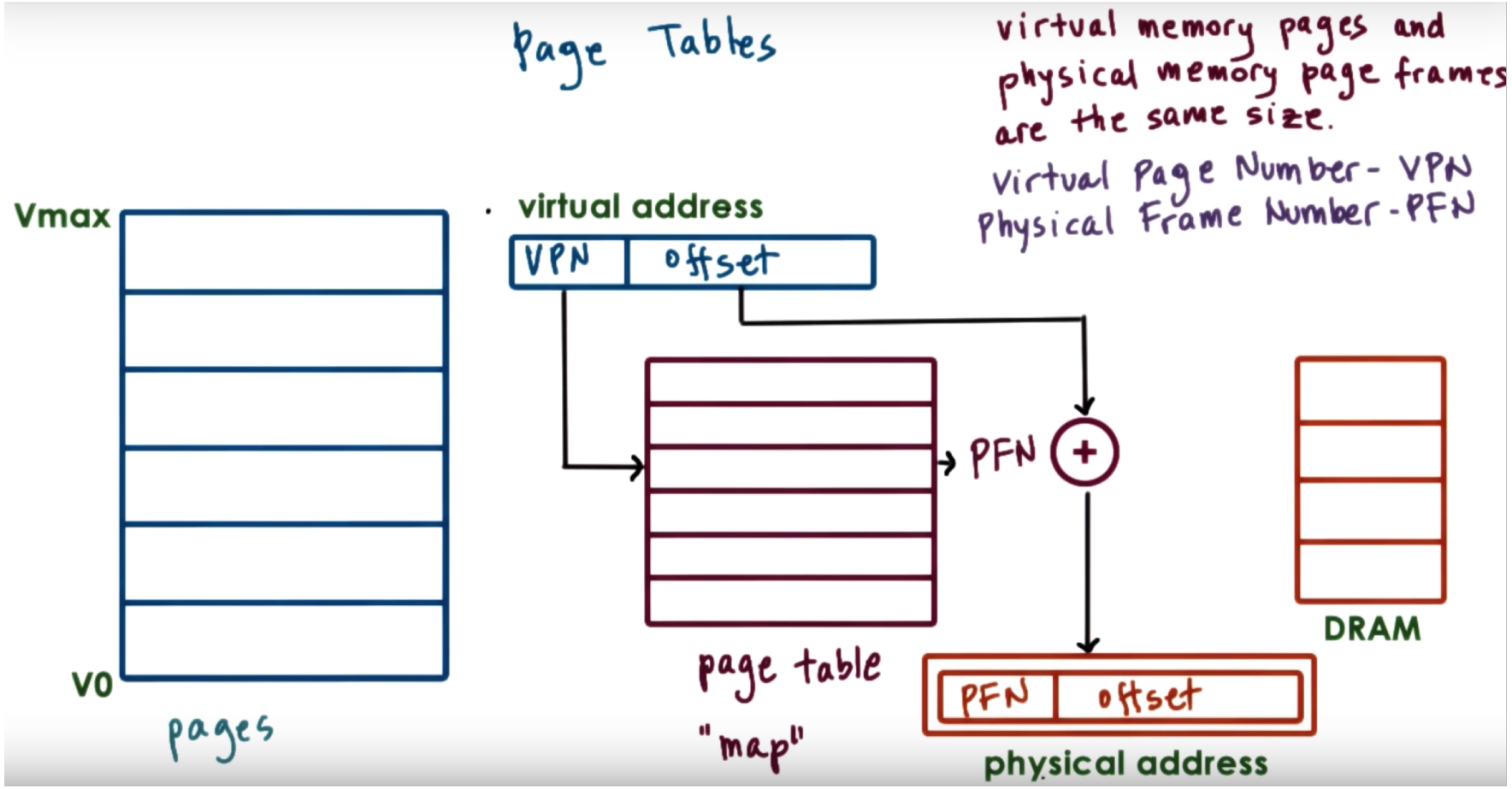
- Virtual vs Physical Memory
- Keep few pages in memory, rest on disk
- Allocate
  - Allocation, replacement
  - Pages -> Page frames
- Arbitrate
  - Address translation and validation
  - Page Tables

Page Tables



- Virtual memory pages and physical memory page frames are of the same size
- Page table is like a map that tells the OS where to find the virtual memory references
- Virtual address has an offset

Page Tables


Page Tables
virtual memory pages and physical memory page frames are the same size.
Virtual Page Number - VPN
Physical Frame Number - PFN
Vmax
V0
pages
virtual address
VPN
offset
page table
"map"
PFN
+
DRAM
PFN
offset
physical address

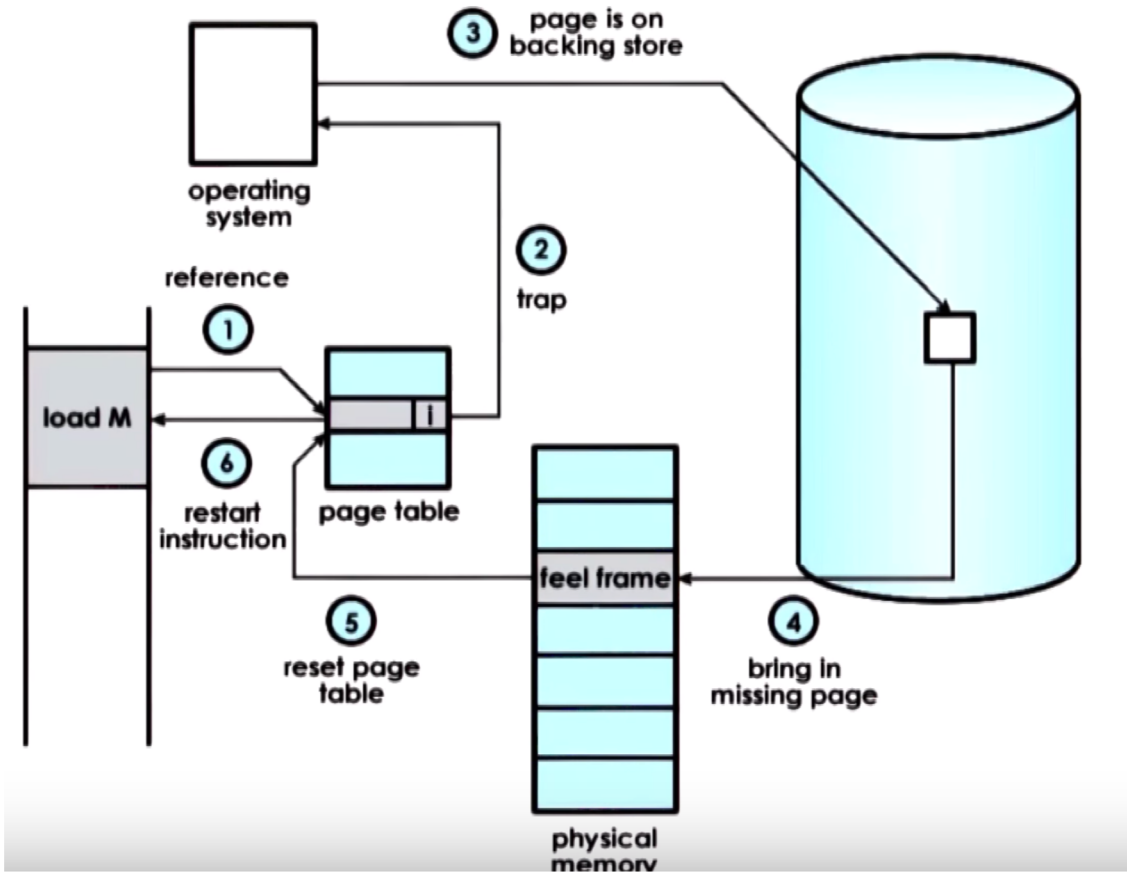
Page Fault

- 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%

Demand Paging

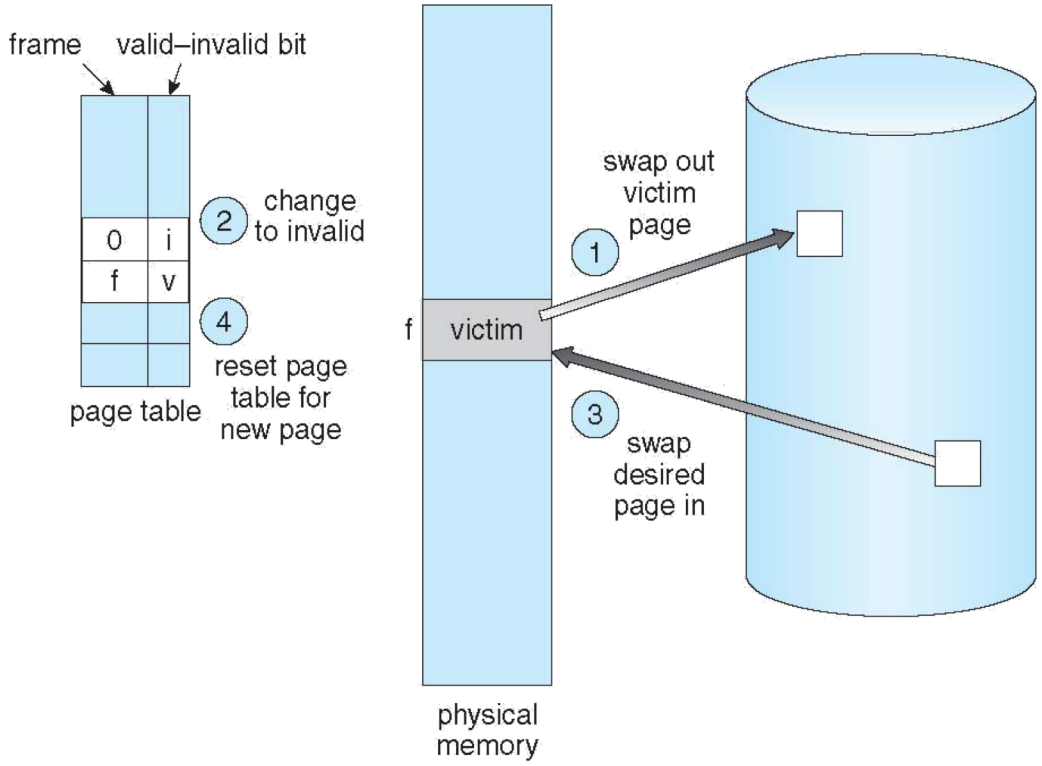
- Virtual memory >> physical memory
  - Virtual memory page not always in physical memory
  - Physical page frame can be saved and restored to/from secondary storage.
- Demand Paging
  - Pages swapped in and out of memory and swap partition (disk)

Demand Paging



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
4. Continue the process by restarting the instruction that caused the trap



Page Replacement

- Typically $\sum_i VAS_i \gg Physical\ Memory$
- With demand paging, physical memory fills up quickly
- When a process faults and memory is full, some page must be swapped out.
  - Handling a page fault now requires 2 disk accesses, not 1
- Which page should we replace?
  - Local replacement – Replace a page of the faulting process
  - Global replacement – Possibly replace the page of another process

Page Replacement Evaluation

- Record a trace of the pages accessed by a process
  - Generates page trace. Example, 3, 1, 4, 2, 5, 2, 1, 2, 3, 4
- Simulate the behavior of a page replacement algorithm on the trace and record the number of page faults generated
- Parameters: algorithm, page reference string, # of memory frames
- Fewer page faults ➔ Better performance

Page Replacement Algorithms

A page replacement algorithm picks a page to be paged out and frees up a frame

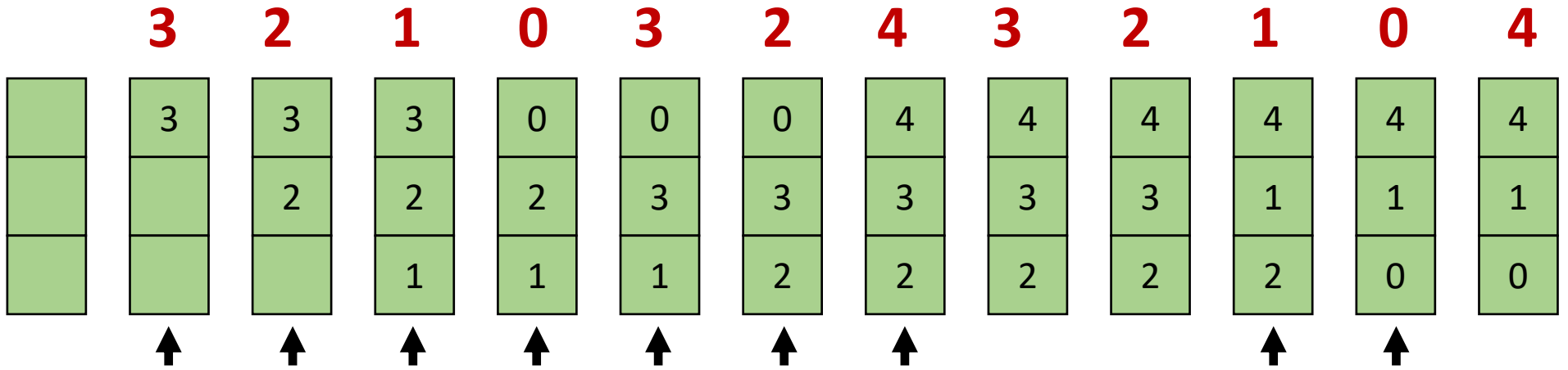
- **Optimal** – The one that leads to the least faults
- **FIFO** - First In First Out
- **LRU** - Least Recently Used Replacement

First In First Out (FIFO) Algorithm

- Basic Idea: The oldest page in physical memory is the one selected for replacement
- Extremely simple to implement
  - Keep a list
    - Victims are chosen from the tail
    - New pages are placed at the head

FIFO Example

- Reference String: 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4
- 3 Frames (At a time, only 3 pages can be in memory for a process)



9 page faults

- 4 page frames → **10** page faults 😲

FIFO Issues

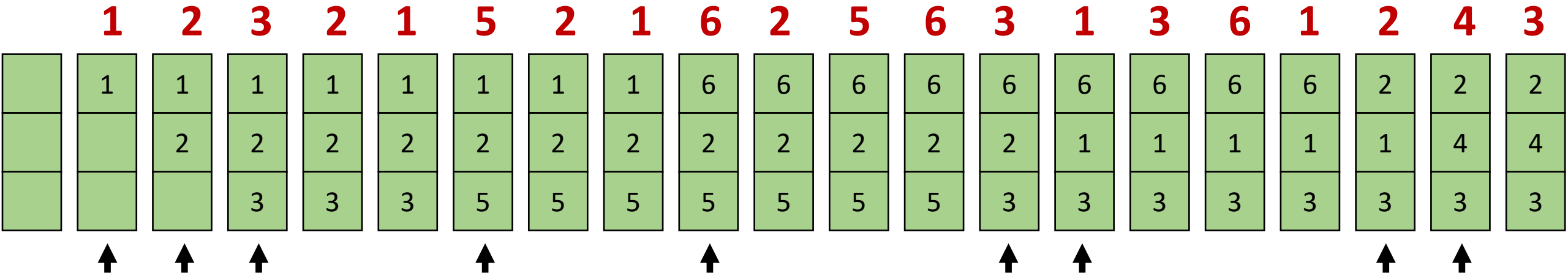
- Performance can be poor
- FIFO does not consider Page Usage
  - In the worst case, each page that is paged out could be the one that is referenced next, leading to a high page fault rate
  - Ideally, keep around the pages that are about to be used next –
  - This is the basis of the OPT algorithm in the next slide

Optimal(OPT) Algorithm

- Basic Idea: Replace the page that will not be referenced for the longest time
- This gives the lowest possible fault rate
- Issues: Impossible to implement 😕
- Does provide a good measure for other techniques

OPT Example

- Reference String: 1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3
- 3 Frames (At a time, only 3 pages can be in memory for a process)



9 page faults

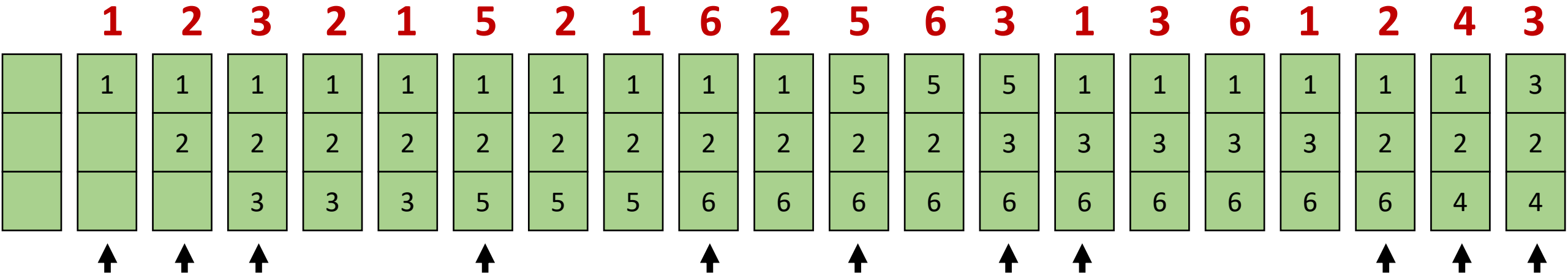
- 4 page frames → **7** page faults

Least Recently Used (LRU)

- Basic Idea: Replace the page in memory that has not been accessed for the longest time
  - 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
- Optimal policy looking back in time
  - As opposed to forward in time
  - Fortunately programs tend to follow similar behaviour

LRU Example

- Reference String: 1, 2, 3, 2, 1, 5, 2, 1, 6, 2, 5, 6, 3, 1, 3, 6, 1, 2, 4, 3
- 3 Frames (At a time, only 3 pages can be in memory for a process)



11 page faults

- 4 page frames → **10** page faults

Your turn now!!

LRU Quiz

Suppose you have an array with 11 page sized entries that are accessed one by one in a loop.

Also, suppose you have a system with 10 pages of physical memory.

What is the percentage of pages that will need to be demand paged using the LRU policy? (round to the nearest %)

100 %