

CS 111: Operating System Principles
Lecture 9

Page Replacement

3.0.0

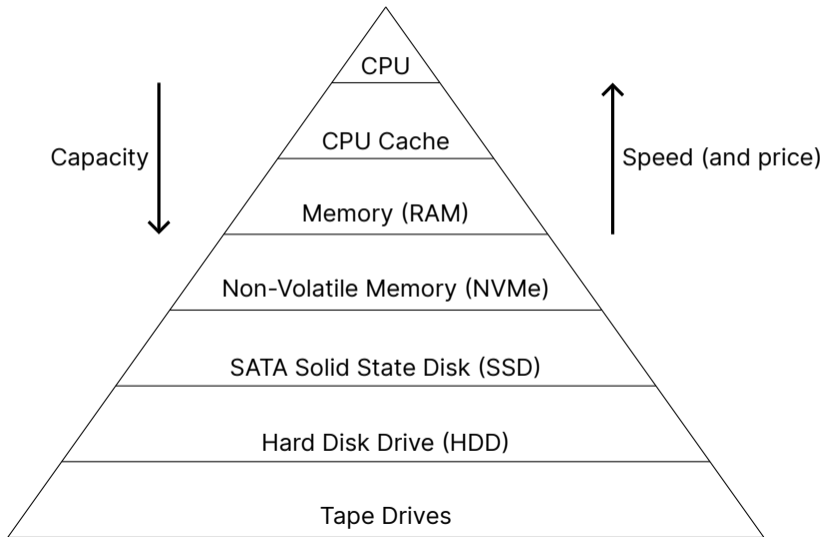
Jon Eyolfson

October 21, 2021



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/)

Computer Memory Hierarchy is a Trade-off of Capacity and Speed



We Want to Hide the Hierarchy from the User

Each level wants to pretend it has the speed of the layer above it
and the capacity of the layer below it

The memory used by all the processes may exceed the amount of physical memory
Not all of them may be in use at the same time

Only keep referenced pages in memory, put others on disk
Swap pages back to memory when they're needed

Page Replacement Algorithms

Optimal

Replace the page that won't be used for the longest

Random

Replace a random page

First-in First-out (FIFO)

Replace the oldest page first

Least Recently Used (LRU)

Replace the page that hasn't been used in the longest time

Page Replacement Evaluation

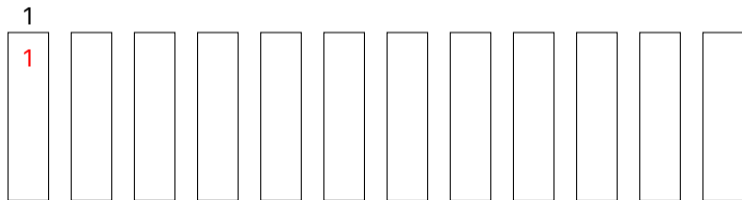
Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)

We'll use this for a bunch of examples during this lecture
We want the fewest number of page faults

For every example we'll find the number of page faults

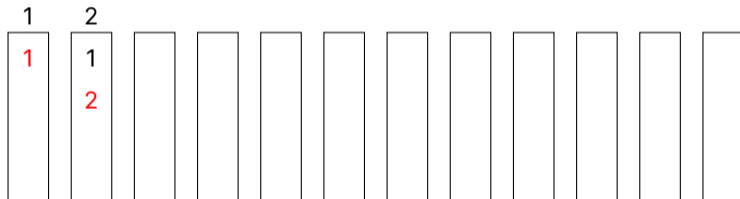
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



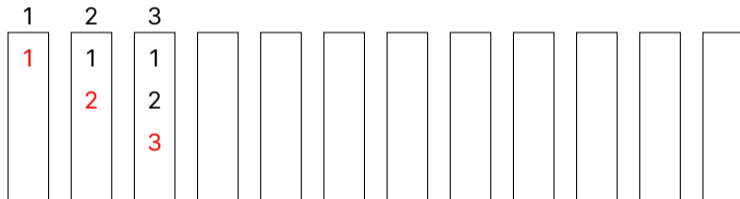
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



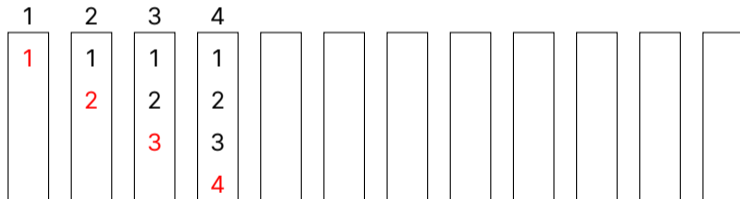
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



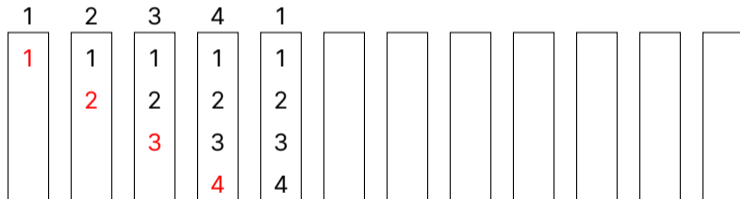
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



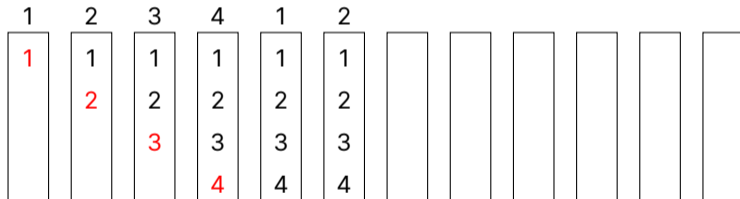
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



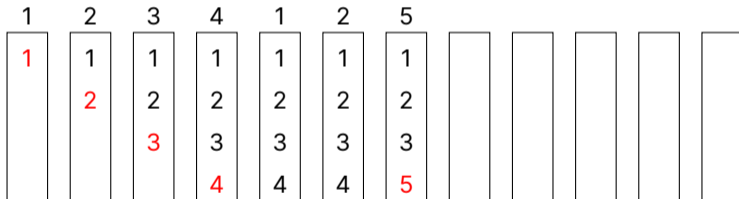
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



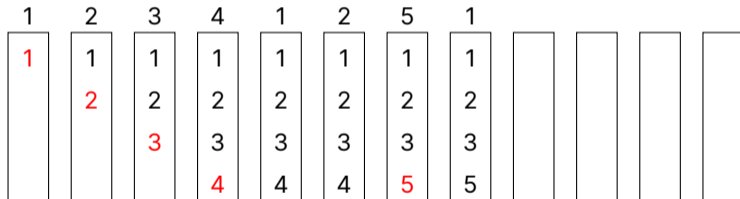
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



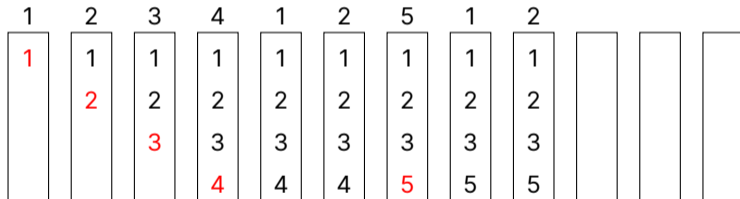
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



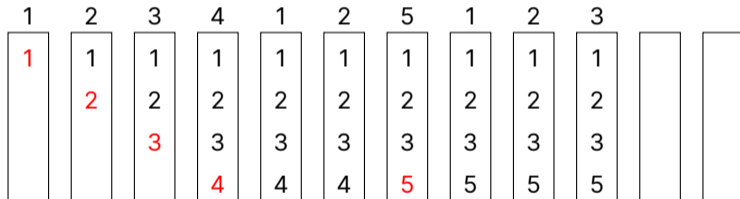
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



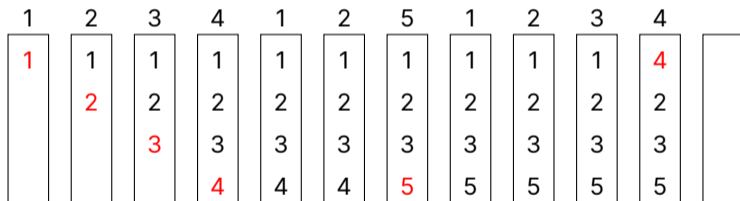
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



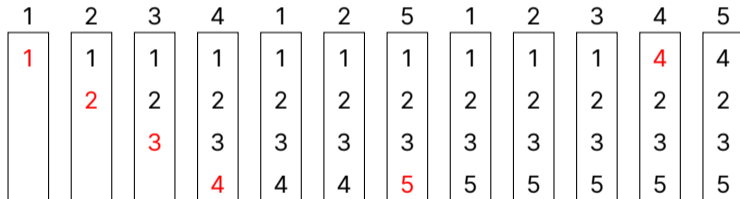
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



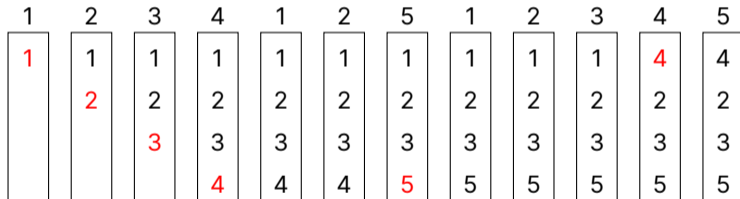
Optimal Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



Optimal Example

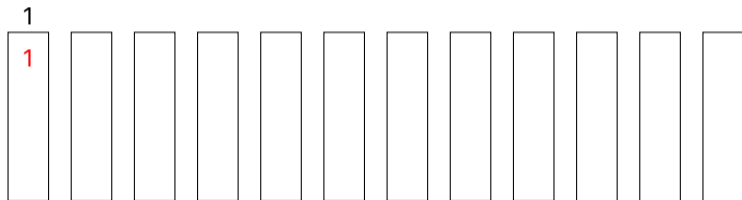
Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



6 page faults

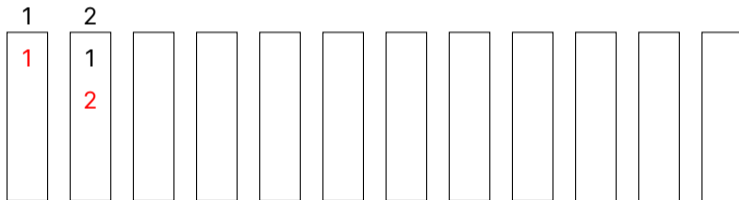
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



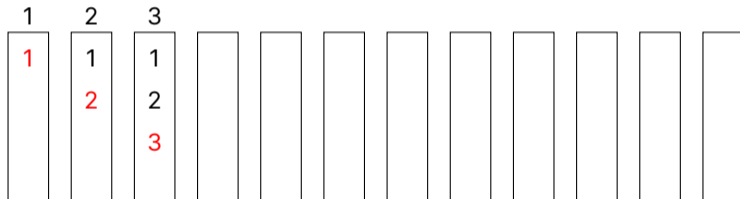
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



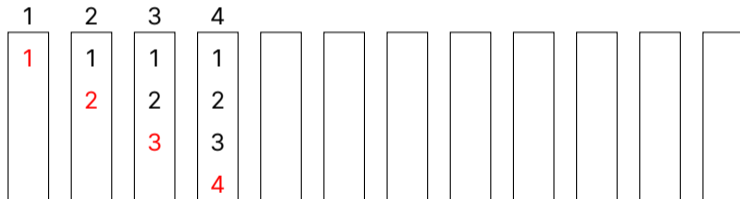
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



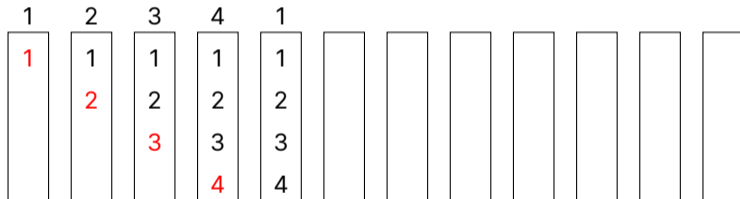
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



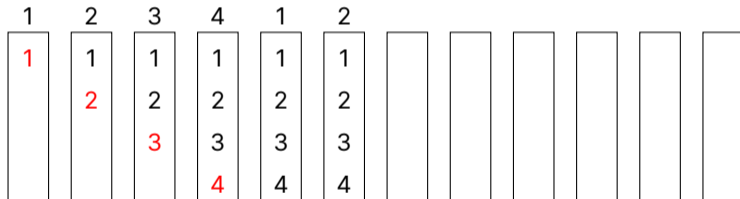
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



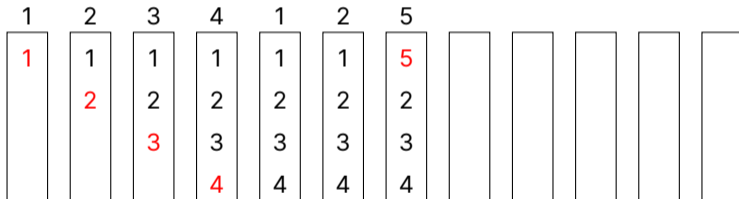
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



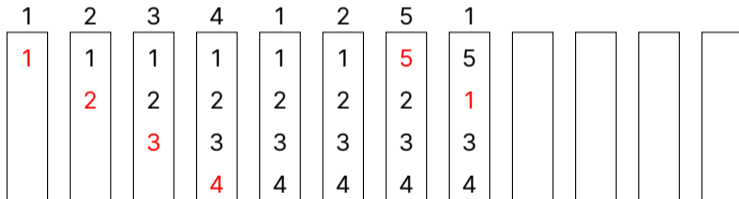
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



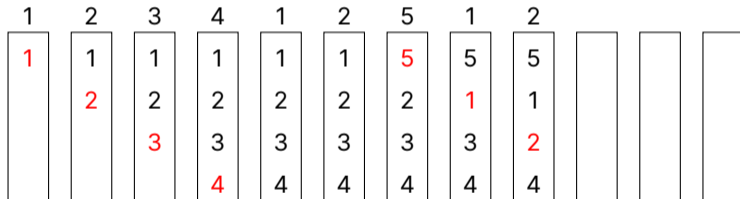
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



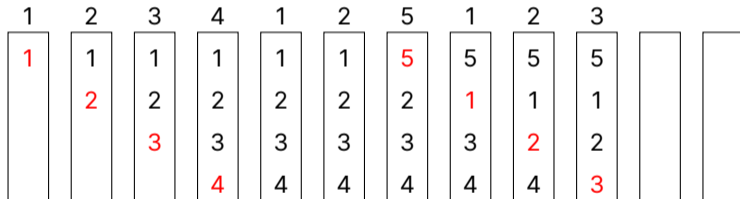
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



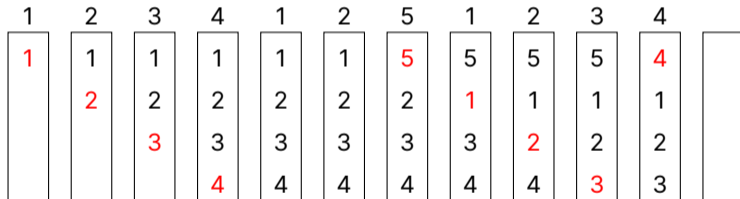
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



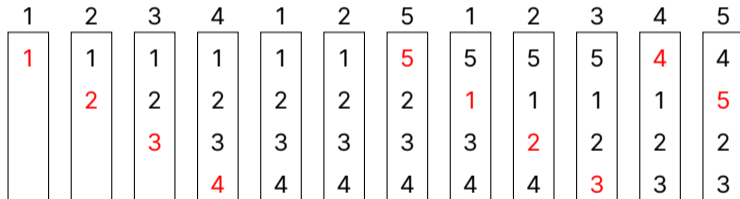
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



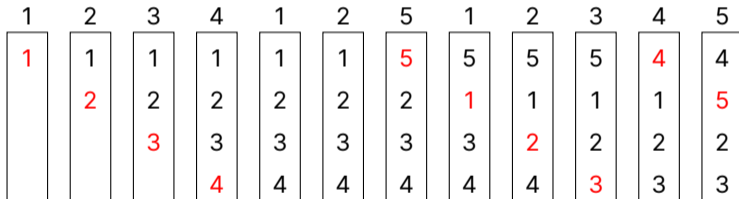
FIFO Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



FIFO Example

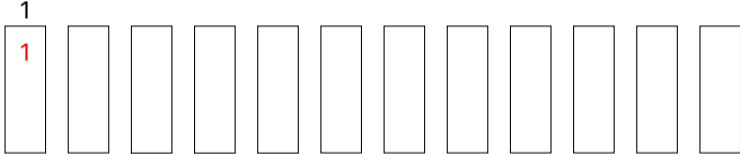
Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



10 page faults

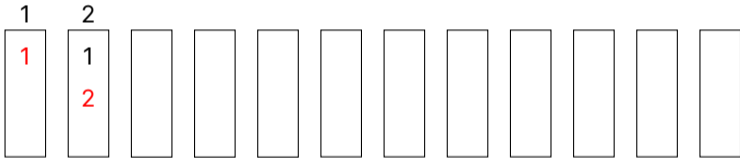
FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



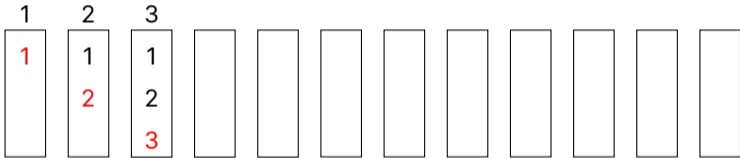
FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



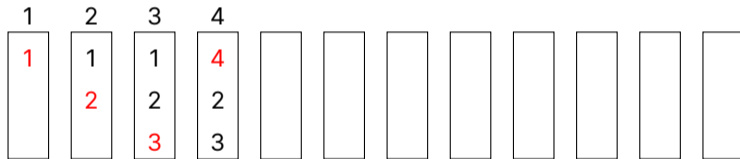
FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



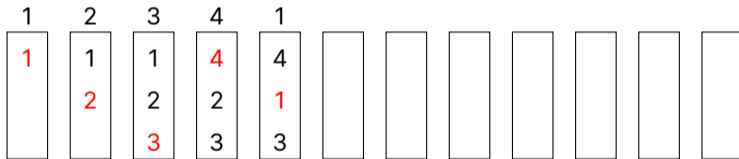
FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



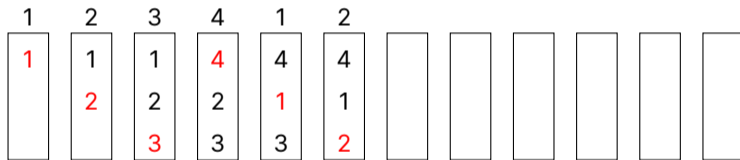
FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



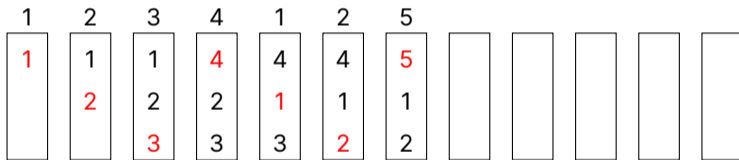
FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



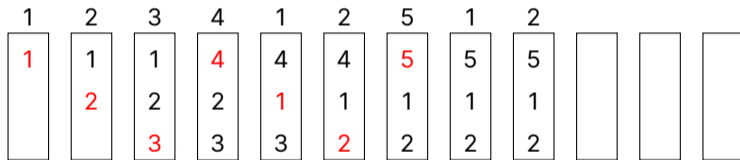
FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



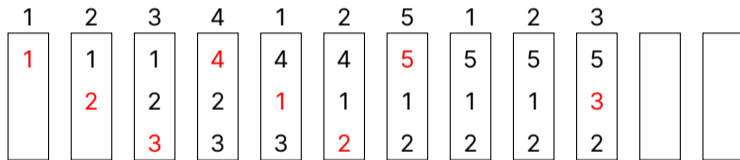
FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



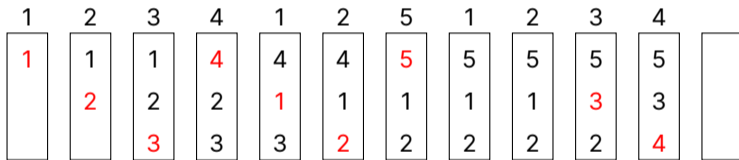
FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



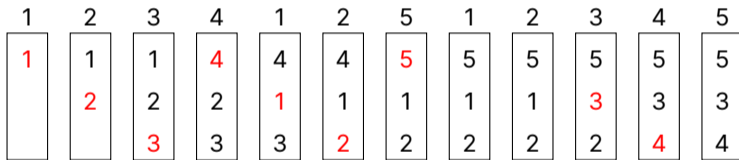
FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



FIFO Example (3 Page Frames)

Assume our physical memory can only hold **3** pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



9 page faults

Bélády's Anomaly Says More Page Frames Causes More Faults

This is a problem with FIFO algorithms

Does not exist with LRU or “stack-based algorithms”

Paper in 2010 found that this FIFO anomaly is unbounded

(<https://arxiv.org/abs/1003.1336>)

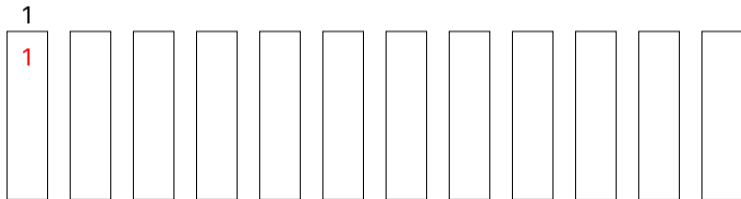
You could construct a sequence to get any arbitrary page fault ratio

For other algorithms:

increasing the number of page frames decreases the number of page faults

LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



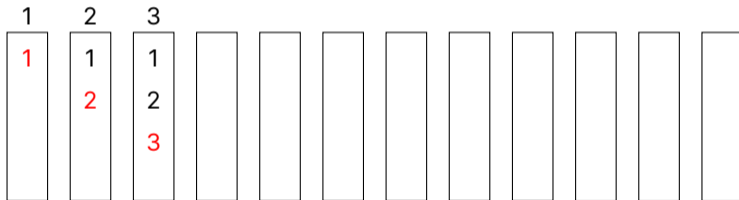
LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



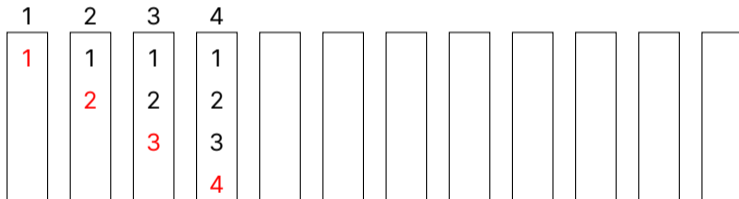
LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



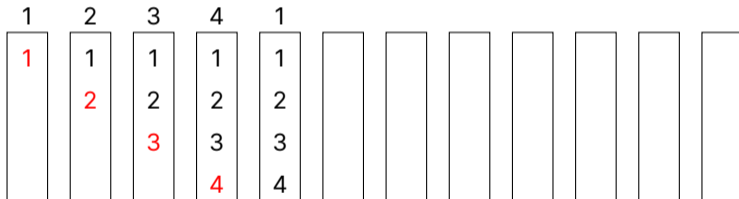
LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



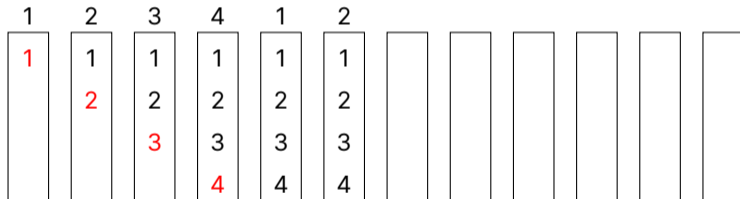
LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



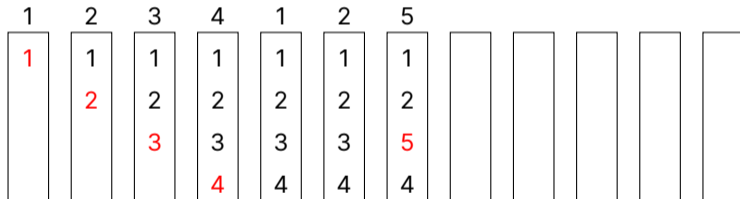
LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



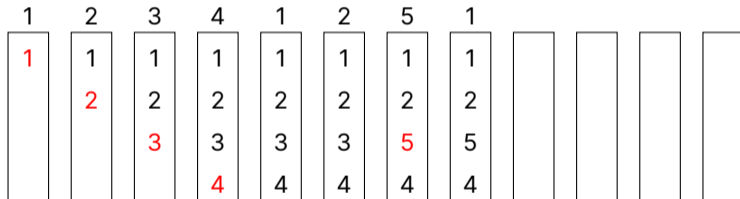
LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



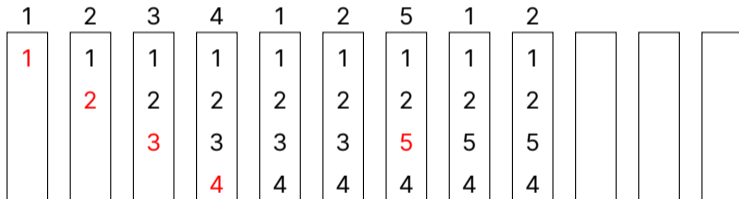
LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



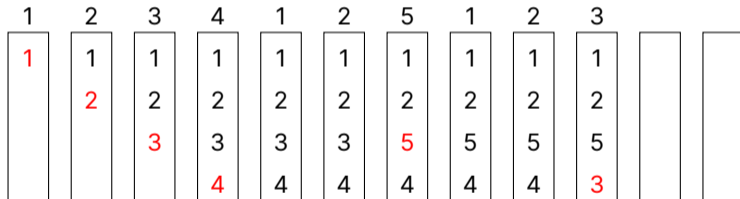
LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



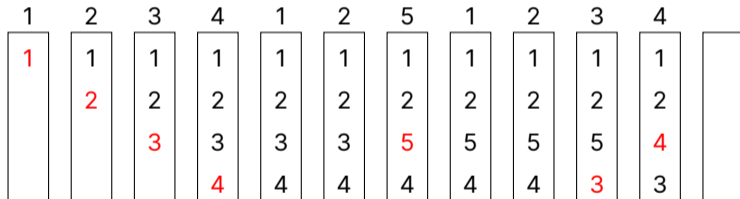
LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



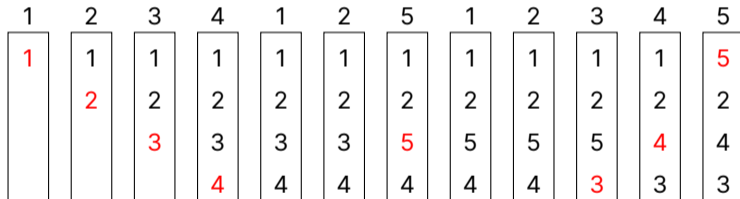
LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)

1	2	3	4	1	2	5	1	2	3	4	5
1	1	1	1	1	1	1	1	1	1	1	5
	2	2	2	2	2	2	2	2	2	2	2
		3	3	3	3	5	5	5	5	4	4
			4	4	4	4	4	4	3	3	3

LRU Example (Use FIFO to Break Ties)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 1 2 5 1 2 3 4 5 (all of the pages are initially on disk)



8 page faults

Implementing LRU in Hardware Has to Search All Pages

You could implement it by keeping a counter for each page

For each page reference, save the system clock into the counter

For replacement, scan through the pages and find the one with the oldest clock

Implementing LRU in Software is Too Expensive

Create a doubly linked list of pages

For each page reference, move it to the front of the list

For replacement, remove from the back of the list

It requires 6 pointer updates for each page reference, and
also creates a high contention bottleneck for multiple processors

Implementing LRU in Practice Isn't Going to Work

We settle for approximate LRU

LRU is an approximation of the optimal case anyways

There's lots of different tweaks you can do to implement it more efficiently

We'll be looking at the clock algorithm, but there's also:

Least Frequently Used (LFU), 2Q, Adaptive Replacement Cache (ARC)

Clock Algorithm

Data structures:

- Keeps a circular list of pages in memory
- Uses a reference bit for each page in memory (light grey in next slides)
- Has a “hand” (iterator) pointing to the last element examined

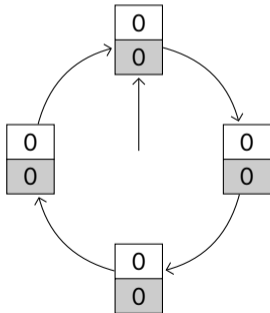
Algorithm, to insert a new page:

- Check the hand’s reference bit, if it’s 0 then place the page and advance hand
- If the reference bit is 1, set it to 0, advance the hand, and repeat

For page accesses, set the reference bit to 1

Clock Example (with Diagram)

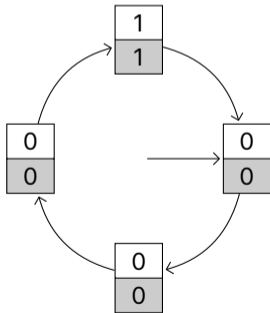
Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

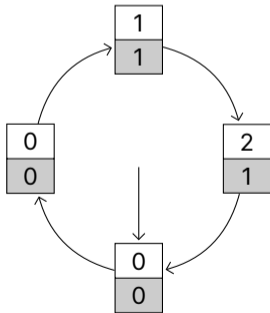
1



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

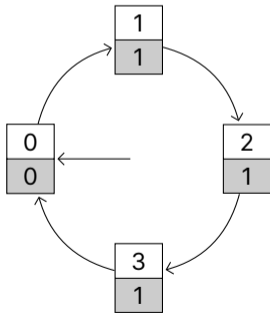
1 2



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

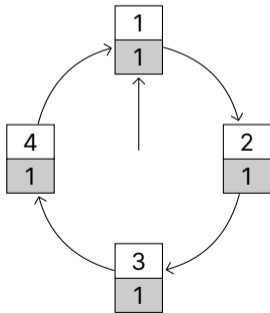
1 2 3



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

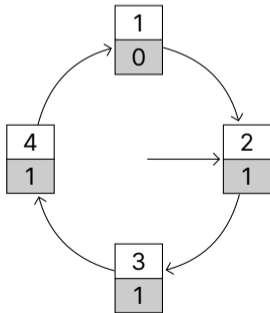
1 2 3 4



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

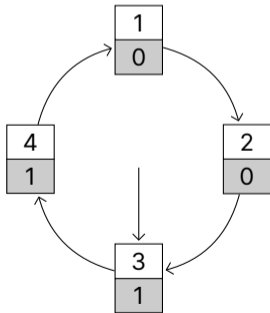
1 2 3 4 5



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

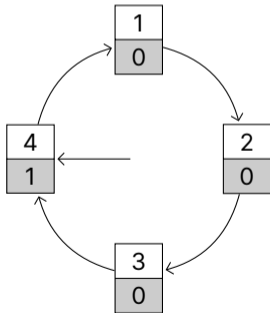
1 2 3 4 5



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

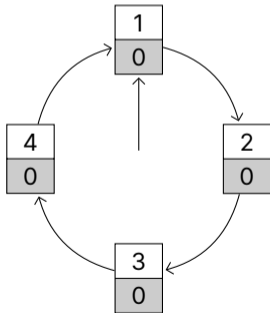
1 2 3 4 5



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

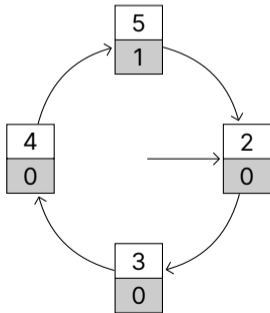
1 2 3 4 5



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

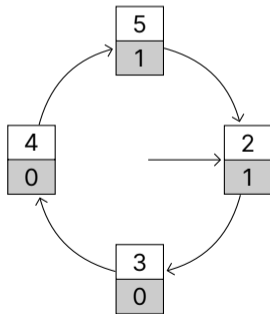
1 2 3 4 5



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

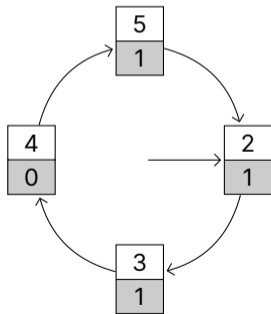
1 2 3 4 5 2



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

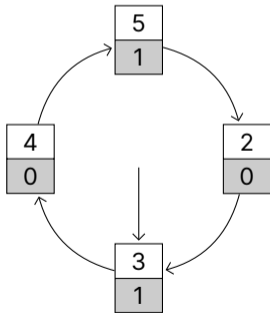
1 2 3 4 5 2 3



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

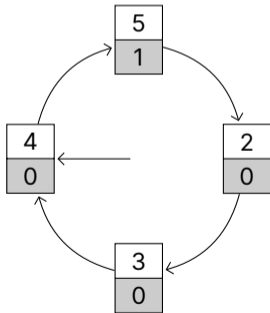
1 2 3 4 5 2 3 1



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

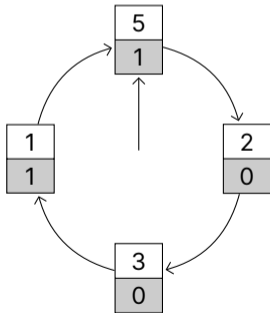
1 2 3 4 5 2 3 1



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

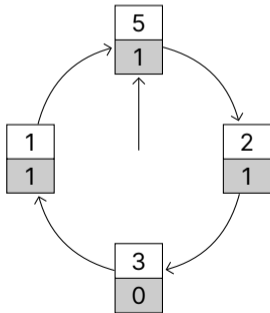
1 2 3 4 5 2 3 1



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

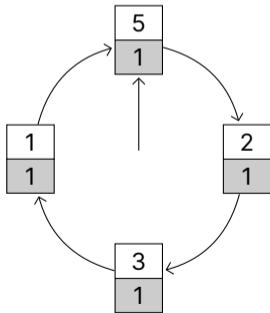
1 2 3 4 5 2 3 1 2



Clock Example (with Diagram)

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)

1 2 3 4 5 2 3 1 2 3



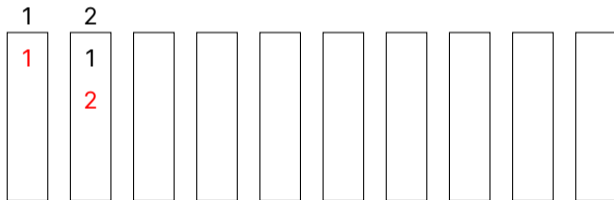
Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



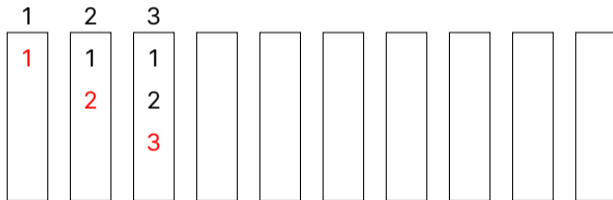
Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



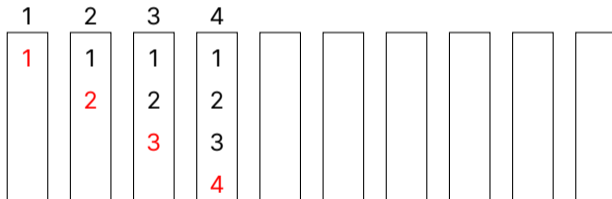
Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



Clock Example

Assume our physical memory can only hold 4 pages, and we access the following:
1 2 3 4 5 2 3 1 2 3 (all of the pages are initially on disk)



6 page faults

For Performance You May Choose to Disable Swapping

Memory is cheap, and has quite high capacity

You'd rather know you need more memory than run slowly

Linux runs an OOM (out of memory) killer, that SIGKILLS the memory hog

Larger page sizes allow for speedup

Trade more fragmentation for more TLB coverage

Page Replacement Algorithms Aim to Reduce Page Faults

We saw the following:

- Optimal (good for comparison but not realistic)
- Random (actually works surprisingly well, avoids the worst case)
- FIFO (easy to implement but Bélády's anomaly)
- LRU (gets close to optimal but expensive to implement)
- Clock (a decent approximation of LRU)