OPERATING SYSTEM: CSET209

CONTENT

- Page Replacement
- Page replacement algorithms

PERFORMANCE OF DEMAND PAGING (CONT.)

- Demand paging can significantly affect the performance of a computer system.
  - For most computer systems, the memory-access time, denoted as *ma* ranges from 10 to 200ns.
  - As long as we have no page faults, the effective access time is equal to the memory access time.
  - If a page fault occurs, we must first read the relevant page from the disk and then access the desired word.
  - Let *p* be the probability of a page fault (0 <= *p* <= 1). We expect *p* to be close to 0, that is, we would expect to have only a few page faults
  - The effective access time is then

effective access time = (1 - p)*ma + p*page fault time

DEMAND PAGING EXAMPLE

- Memory access time = 200 nanoseconds
- Average page-fault service time = 8 milliseconds
- EAT = (1 – p) x 200 + p (8 milliseconds)

  = (1 – p x 200 + p x 8,000,000

  = 200 + p x 7,999,800
- If one access out of 1,000 causes a page fault, then

  EAT = 8.2 microseconds.

  This is a slowdown by a factor of 40!!
- If want performance degradation < 10 percent
  - 220 > 200 + 7,999,800 x p
    20 > 7,999,800 x p
  - p < .0000025
  - < one page fault in every 400,000 memory accesses

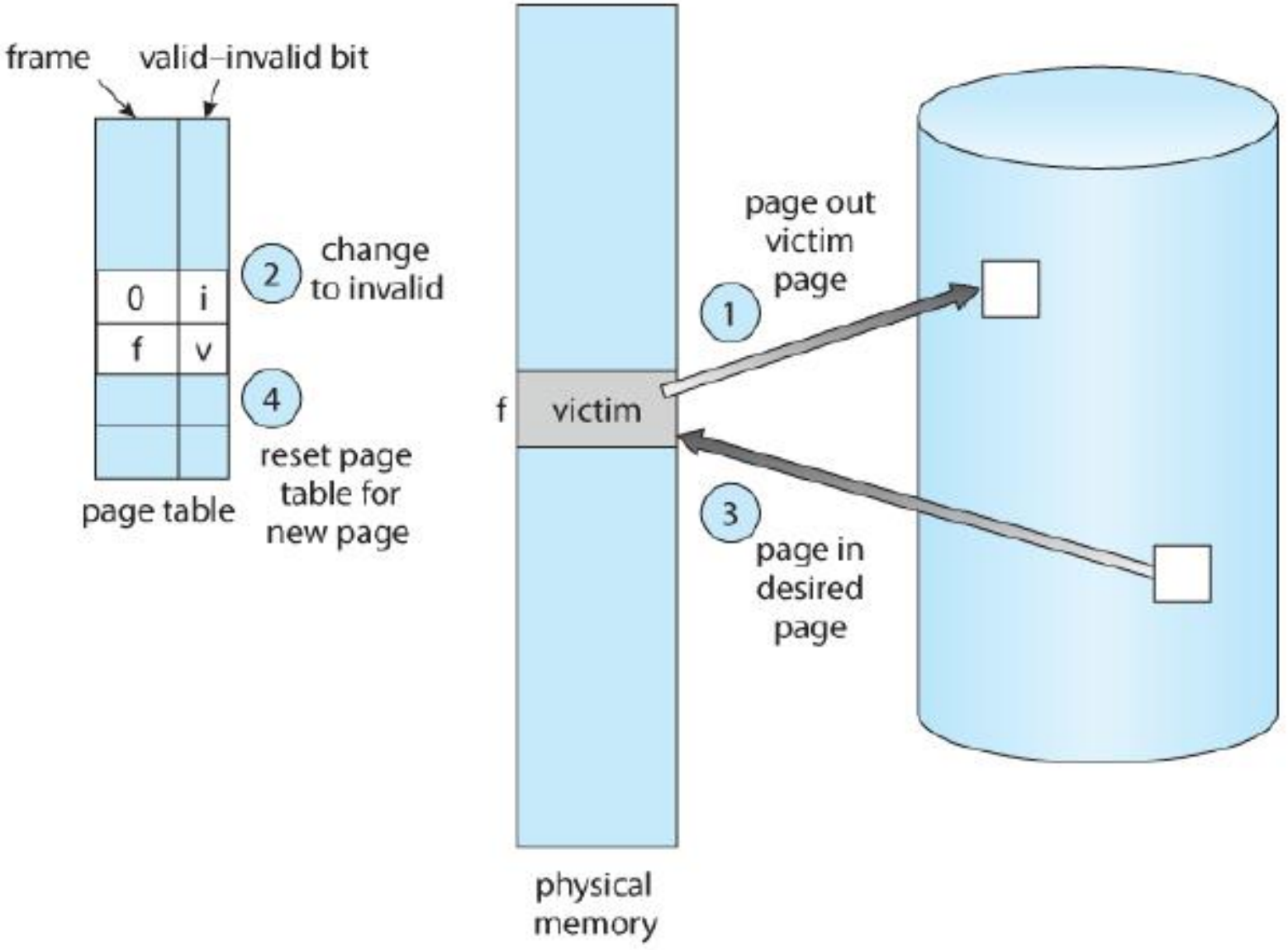
PAGE REPLACEMENT

- Whenever page fault occurs, the page is brought in from the disk and allocated to the free frame in the main memory.
- But what happens if there aren't any free frames available in the main memory?
- There are a few possible solutions:
  - Adjust the memory used for I/O buffering to free up some frames for a user process
  - Put the process requesting more pages into a wait queue until some free frames are available
  - Swap some process out of memory completely freeing up its page frames
  - Find some page in memory that isn't being used right now, and swap that page only out to disk, freeing up a frame that be allocated to the process requesting it. This is called **page replacement** and is the most common solution.

PAGE REPLACEMENT

- Page fault processing assumed that there would be free frames available. So page fault handling must be modified now.
- **Steps for page replacement:**
  1. Find the location of the desired page on the disk (either in swap space or in file system)
  2. Find a free frame.
     1. If there is a free frame, use it.
     2. If no free frame, use page replacement algo to select an existing frame to be replaced known as **victim frame**.
     3. Write the victim frame to the disk. Change all related page tables to indicate that this page is no longer in memory.
  3. Read in the desired page and store it in the frame. Adjust all related page and frame table to indicate the change.
  4. Restart the process that was waiting for this page.

PAGE REPLACEMENT



PAGE REPLACEMENT

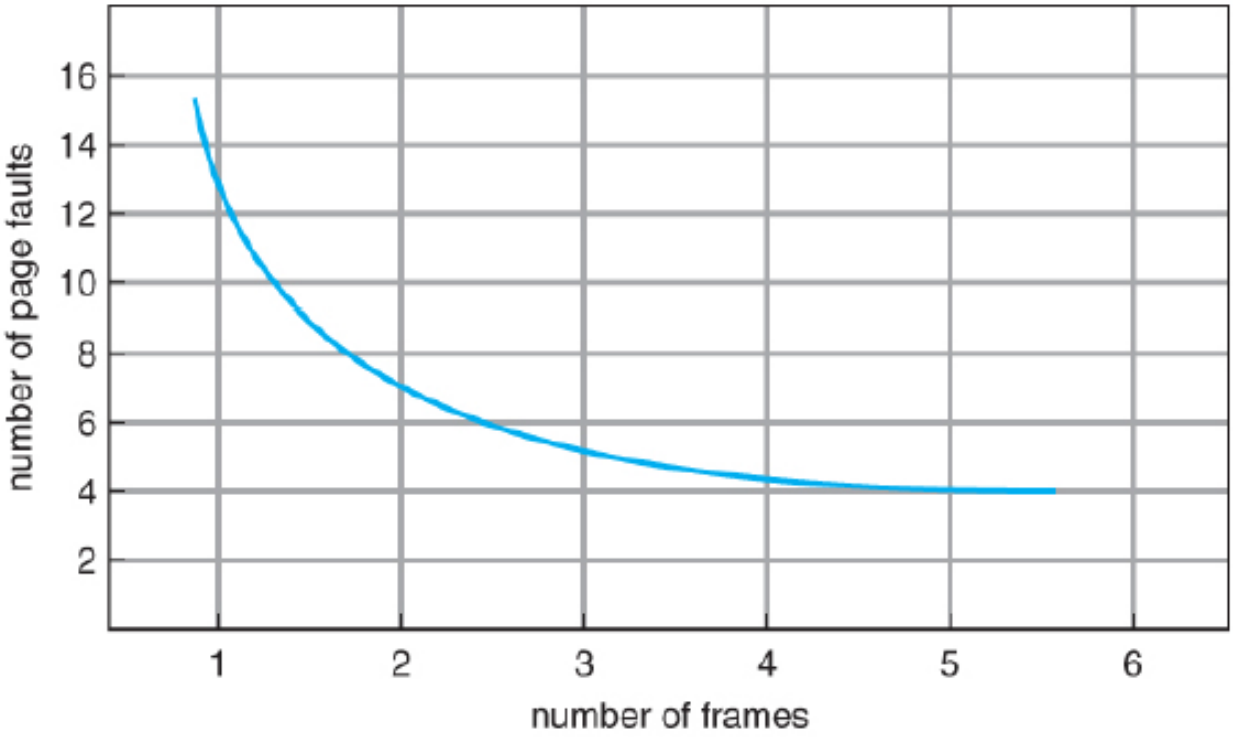
- Note that in page replacement steps, we have added an extra step of disk writing to the page fault handling, effectively doubling the time required to process a page fault.
- This can be alleviated by assigning a **dirty bit** to each page, indicating whether or not it has been changed since it was last loaded from the disk.
- If the dirty bit has not been set (means if it is 0), then the page is unchanged and does not need to be written out to disk. Otherwise page write is required.
- Page replacement techniques look for pages that do not have their dirty bit set, and select clean pages as victim pages.

PAGE REPLACEMENT

- To implement a successful demand paging system, there are two major requirements: **frame allocation algorithm** and **page replacement algorithm**.
  - Frame allocation algorithm decides how many frames are allocated to each process
  - Page replacement algorithm deals with how to select a page for replacement when there are no free frames available.

- The goal of these algorithms is to generate the fewest number of overall page faults. Because disk access is so slow relative to memory access, even slight improvements to these algorithms can yield large improvements in overall system performance.

PAGE REPLACEMENT



- Generally, as the number of available frames increases, the number of page faults should decrease.

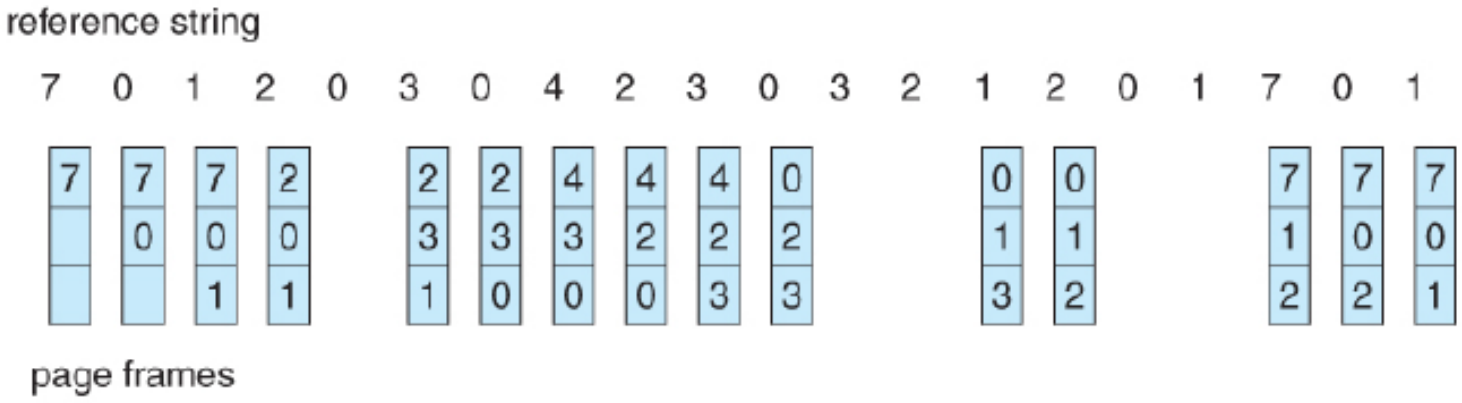
PAGE REPLACEMENT ALGORITHMS

1. FIFO Page replacement:

FIFO: First In First Out

As the new pages are brought in, they are added to the tail of a queue, and the page at the head of the queue is the next victim.

Example:



Here, 20 page requests result in 15 page faults

PAGE REPLACEMENT ALGORITHMS

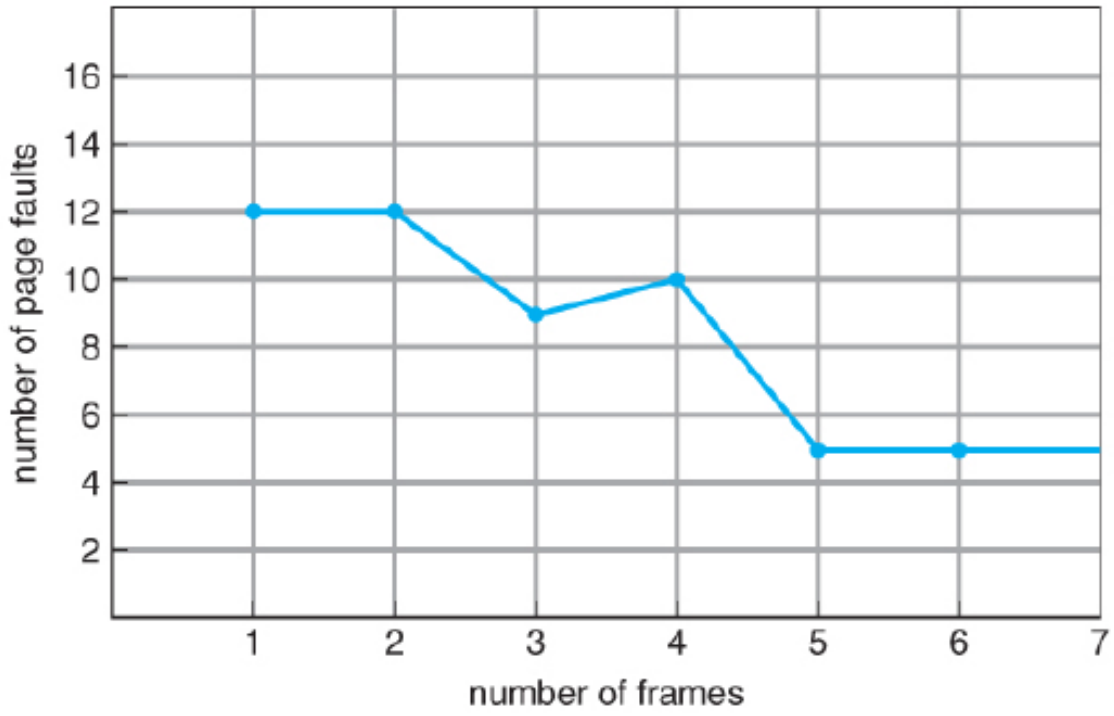
1. **FIFO Page replacement:**

Although FIFO is simple and easy, it is not always optimal

An interesting effect that can occur with FIFO is **Belady's anomaly,** in which increasing the number of frames actually increase the number of page faults that occur.

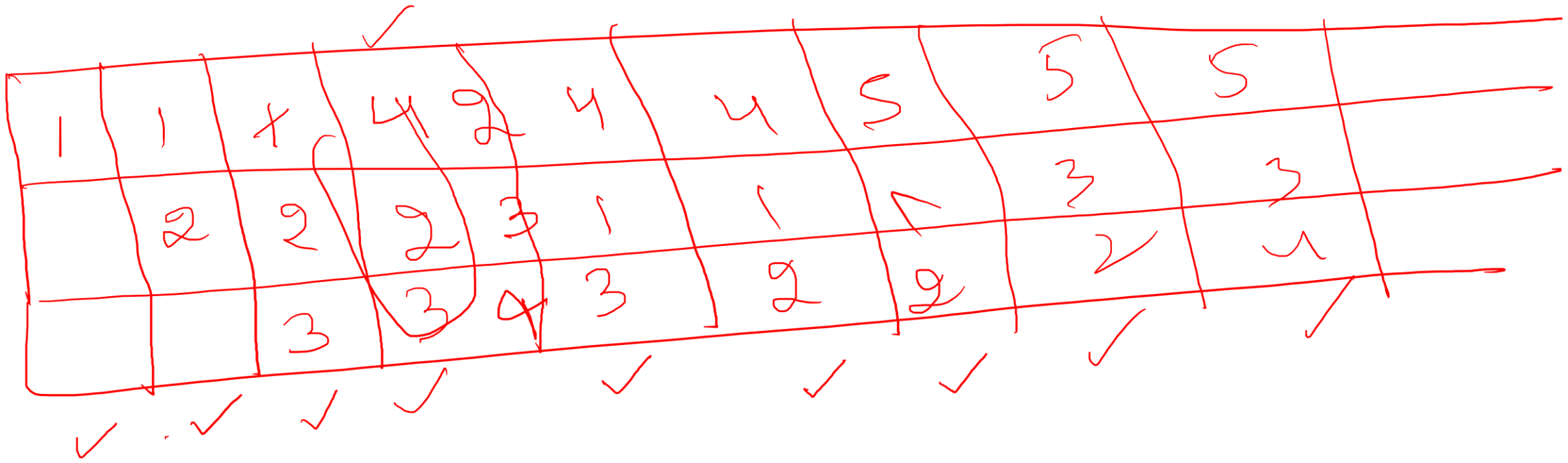
FIFO PAGE REPLACEMENT

Example: A page sequence (1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5) and a varying number of available frames. The maximum number of faults is 12 (every request generates a fault), and the minimum number is 5 (each page loaded only once), but in between, there are some interesting results:



BELADY'S ANOMALY IN FIFO –

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



BELADY'S ANOMALY IN FIFO – 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	1	1	2	3	4	1	1	1	2	5	5
	2	2	3	4	1	2	2	2	5	3	3
		3	4	1	2	5	5	5	3	4	4
PF	PF	PF	PF	PF	PF	PF	X	X	PF	PF	X

BELADY'S ANOMALY IN FIFO – 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

1	1	1	1	1	1	2	3	4	5	1	2
	2	2	2	2	2	3	4	5	1	2	3
		3	3	3	3	4	5	1	2	3	4
			4	4	4	5	1	2	3	4	5
PF	PF	PF	PF	X	X	PF	PF	PF	PF	PF	PF

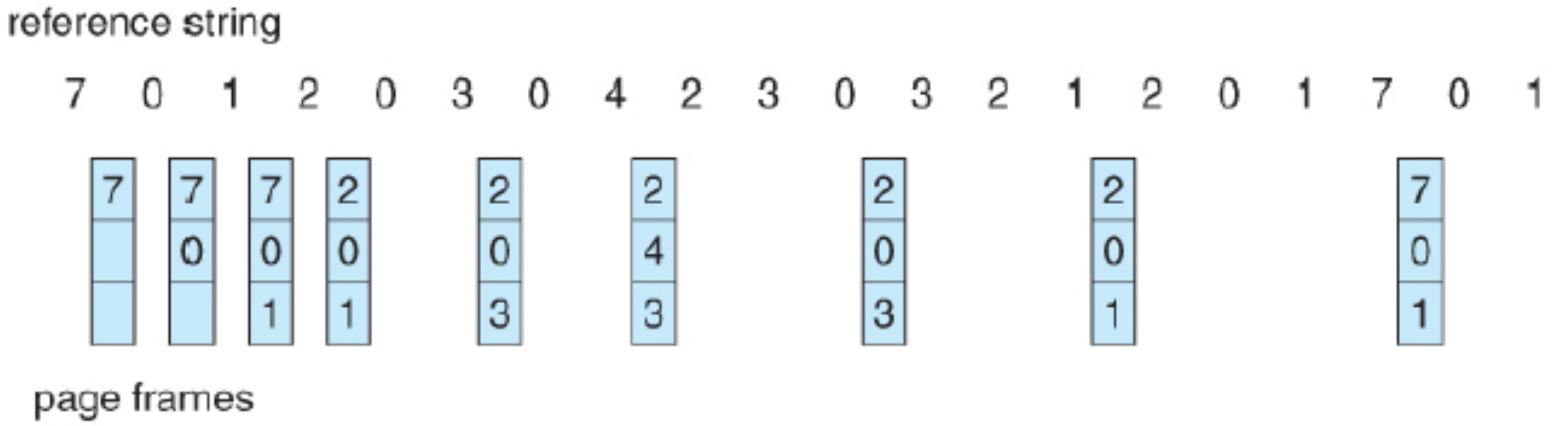
OPTIMAL PAGE REPLACEMENT

2. Optimal page replacement

The discovery of Belady's anomaly lead to the search for an optimal page-replacement algorithm which yields the lowest of all possible page-faults, and which does not suffer from Belady's anomaly.

This algorithm is simply "Replace the page that will not be used for the longest time in the future."

EXAMPLE:



It generates 9 page faults (of which 6 are unavoidable, the first reference to each new page)

For the same string, FIFO required 3 times as many extra page faults as required by Optimal strategy.

OPTIMAL PAGE REPLACEMENT

2. Optimal page replacement

Cannot be implemented practically, because it requires foretelling the future but it makes a good benchmark for other replacement strategies.

CONSIDER A REFERENCE STRING: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. THE NUMBER OF FRAMES IN THE MEMORY IS 3. FIND OUT THE NUMBER OF PAGE FAULTS RESPECTIVE TO:

OPTIMAL PAGE REPLACEMENT ALGORITHM
FIFO PAGE REPLACEMENT ALGORITHM

CONSIDER A REFERENCE STRING: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. THE NUMBER OF FRAMES IN THE MEMORY IS 3. FIND OUT THE NUMBER OF PAGE FAULTS RESPECTIVE TO:

FIFO PAGE REPLACEMENT ALGORITHM

Request	**4**	**7**	**6**	**1**	**7**	**6**	**1**	**2**	**7**	**2**
Frame 3			6	6	6	6	6	6	7	7
Frame 2		7	7	7	7	7	7	2	2	2
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Miss	Hit

CONSIDER A REFERENCE STRING: 4, 7, 6, 1, 7, 6, 1, 2, 7, 2. THE NUMBER OF FRAMES IN THE MEMORY IS 3. FIND OUT THE NUMBER OF PAGE FAULTS RESPECTIVE TO:

OPTIMAL PAGE REPLACEMENT ALGORITHM

Request	4	7	6	1	7	6	1	2	7	2
Frame 3			6	6	6	6	6	2	2	2
Frame 2		7	7	7	7	7	7	7	7	7
Frame 1	4	4	4	1	1	1	1	1	1	1
Miss/Hit	Miss	Miss	Miss	Miss	Hit	Hit	Hit	Miss	Hit	Hit

THANK YOU