Operating Systems

Jinghui Zhong （钟竞辉）
Office： B3-515
Email： jinghuizhong@scut.edu.cn

$x = ?$

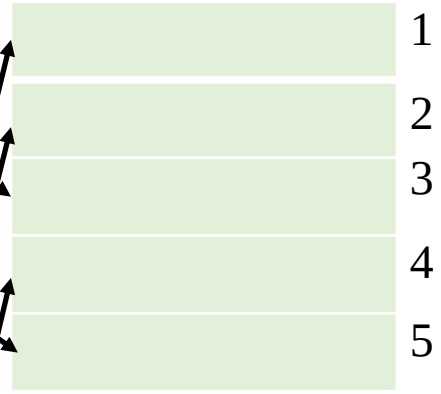
TLB

Page id	Page frame id
x	2
3	*x*
x	1

Page Table

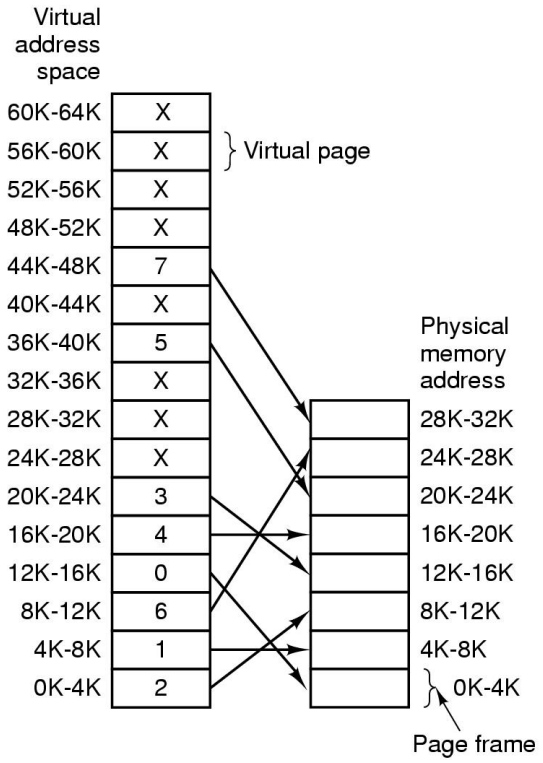
	Page frame id	present
1	*x*	*x*
2		*x*
3	*x*	*x*
4		*x*
5	*x*	*x*
6	*x*	*x*
7		*x*
8	*x*	*x*
9		*x*
10		*x*

Physical Memory



Page Replacement Algorithms

- Page fault forces choice
  - ✓which page must be removed
  - ✓make room for incoming page
- Modified page must first be saved
  - ✓unmodified just overwritten
- Better not to choose an often used page
  - ✓will probably need to be brought back in soon



Optimal Page Replacement Algorithm

Replace the page which will be referenced at the farthest point.

Optimal but impossible to implement.

Not Recently Used Page Replacement Algorithm

- **Each page has Reference bit (R) and Modified bit (M).**
  - ✓ bits are set when page is referenced (read or written recently), modified (written to)
  - ✓ when a process starts, both bits R and M are set to 0 for all pages.
  - ✓ periodically, (on each clock interval (20msec)), the R bit is cleared. (i.e. R=0).
- **Pages are classified**
  - ✓ Class 0: not referenced, not modified
  - ✓ Class 1: not referenced, modified
  - ✓ Class 2: referenced, not modified
  - ✓ Class 3: referenced, modified
- **NRU removes page at random**
  - ✓ from lowest numbered non-empty class

FIFO Page Replacement Algorithm

- Maintain a linked list of all pages
  - ✓Pages came into memory with the oldest page at the front of the list.
- Page at beginning of list replaced
- Advantage?

✓ easy to implement

- Disadvantage?

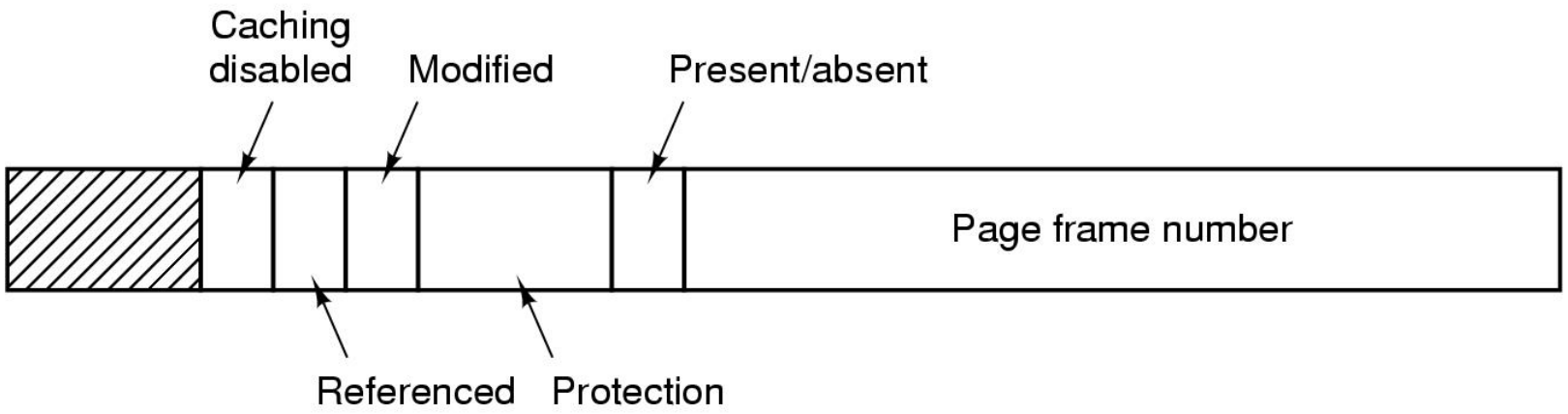
✓page in memory the longest (perhaps often used) may be evicted

Second Chance Page Replacement Algorithm

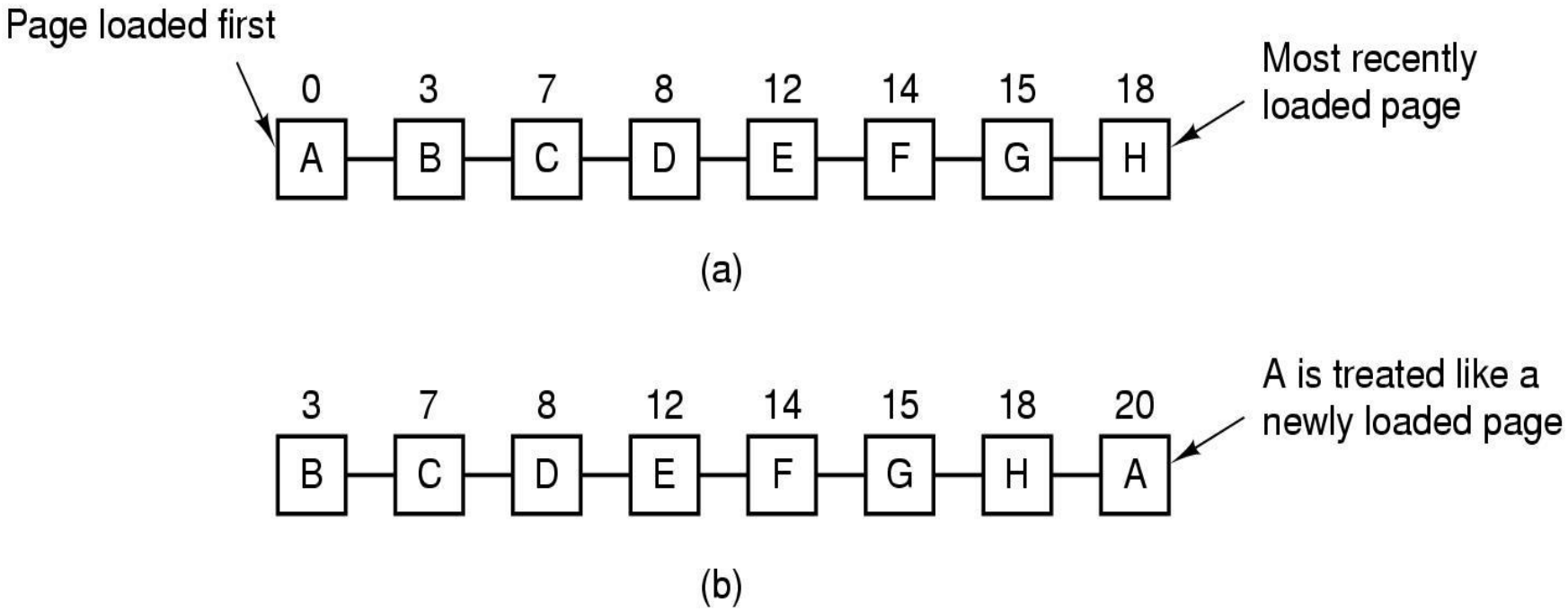
- **Inspect R bit:**

✓ if R = 0 ➔ evict the page

✓ if R = 1 ➔ set R = 0 and put page at end (back) of list. The page is treated like a newly loaded page.

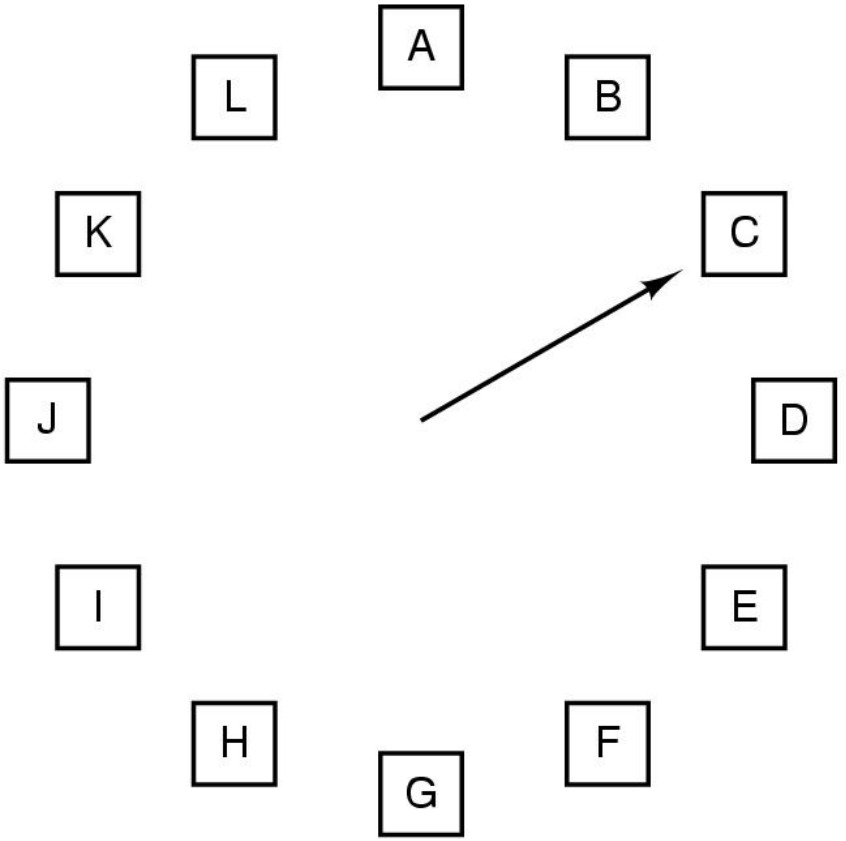


Second Chance Page Replacement Algorithm



- Operation of a second chance
  - Pages sorted in FIFO order
  - Page list if fault occurs at time 20, A has *R* bit set (numbers above pages are loading times)

The Clock Page Replacement Algorithm



When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:

R = 0: Evict the page

R = 1: Clear R and advance hand

Least Recently Used (LRU)

- **Assume pages used recently will used again soon**
  - ✓throw out page that has been unused for longest time
- **Software Solution?**

Must keep a linked list of pages: most recently used at front, least at rear; update this list every memory reference Too expensive!!

- **Hardware solution?**

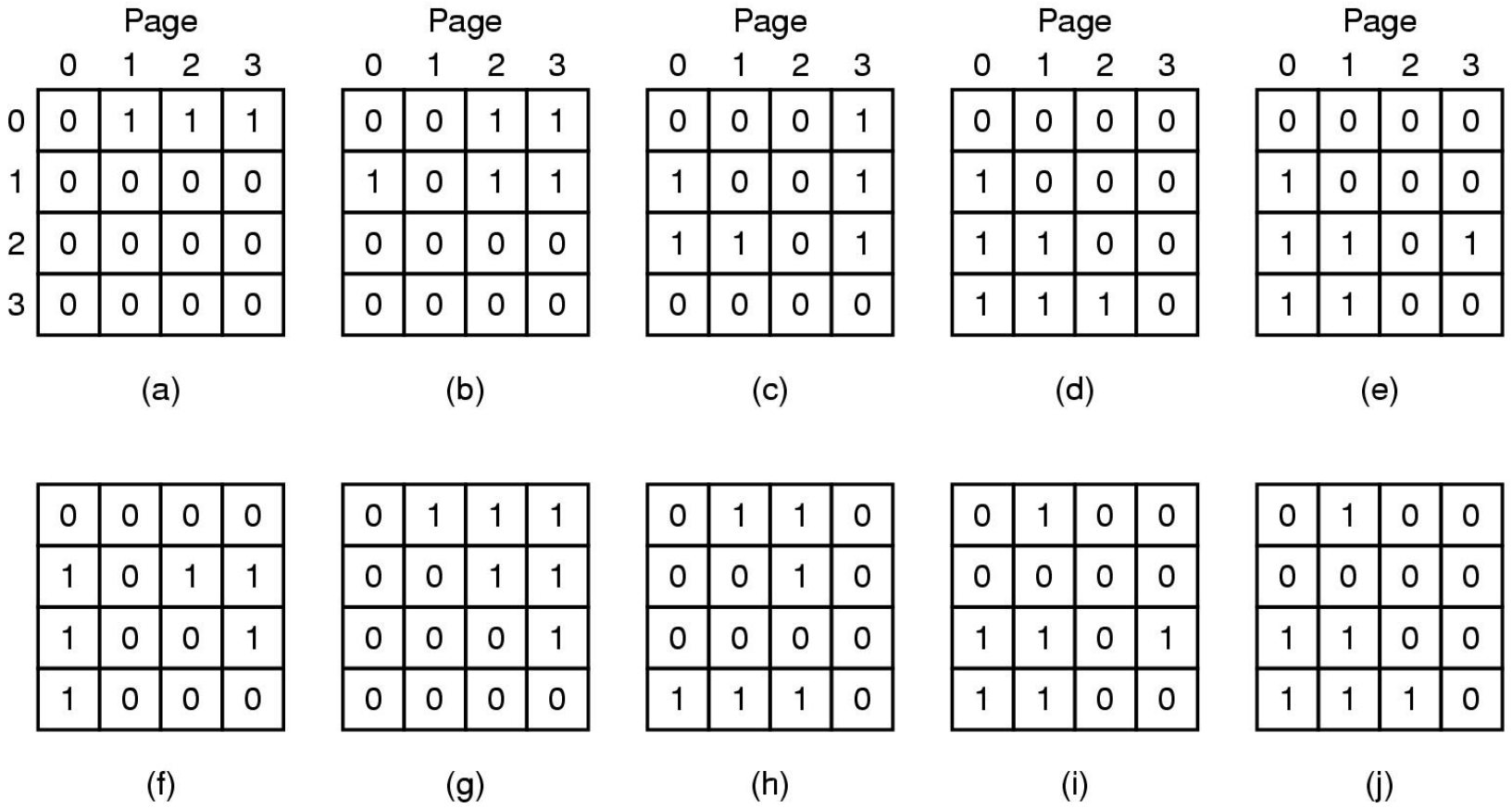
Equip hardware with a 64 bit counter.

Least Recently Used (LRU)

- **Hardware solution:** Equip hardware with a 64 bit counter.
  - That is incrementing after each instruction.
  - The counter value is stored in the page table entry of the page that was just referenced.
  - choose page with lowest value counter
  - periodically zero the counter
  - Problem?

  page table is very large, become even larger.
- **Maintain a matrix of *n* x *n* bits for a machine with *n* page frames.**
  - ✓ When page frame *K* is referenced:
    (i) Set row *K* to all 1s.
    (ii) Set column *K* to all 0s.
  - ✓ The row whose binary value is smallest is the LRU page.

Simulating LRU in Software



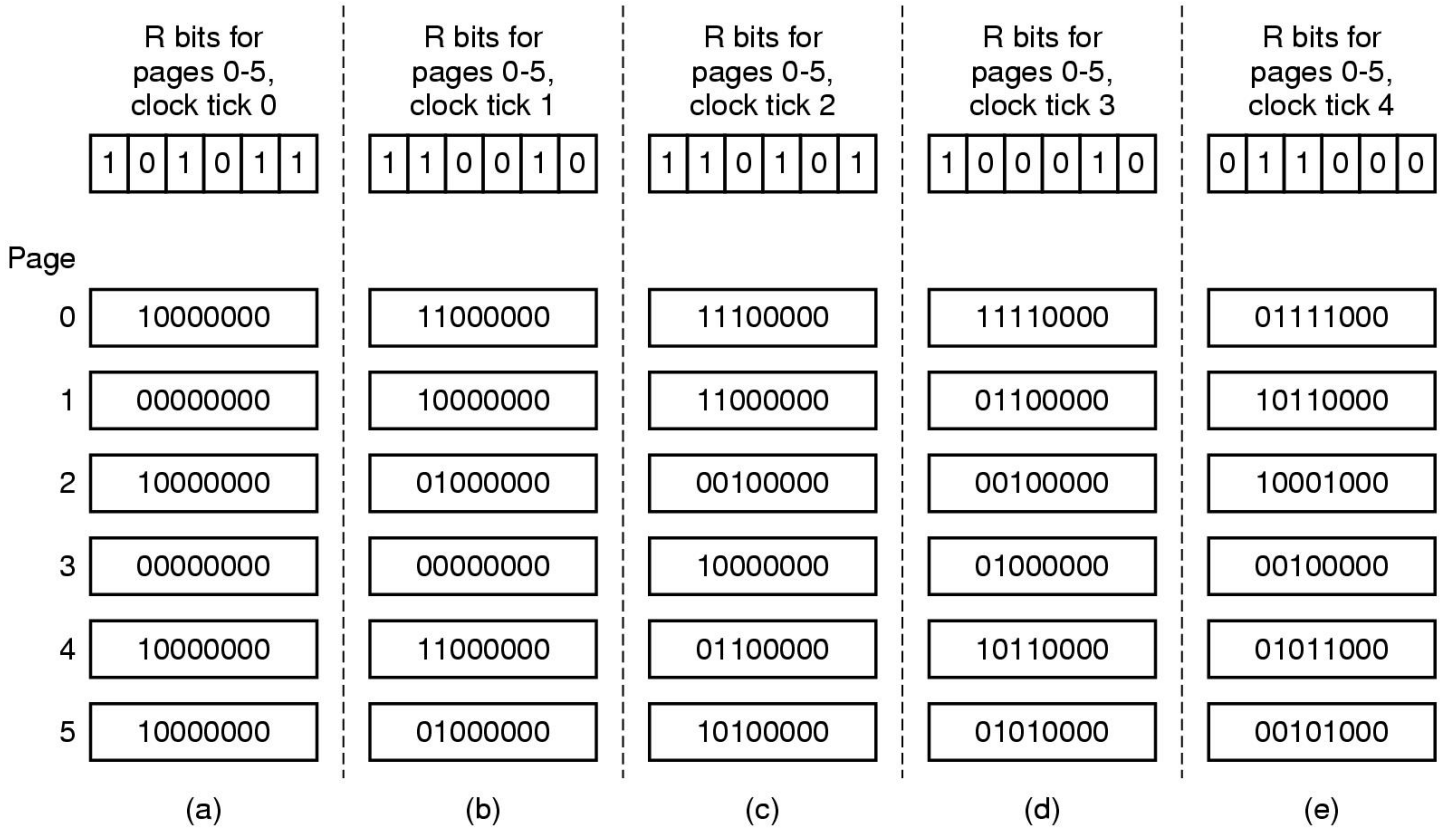
LRU using a matrix – pages referenced in order 0,1,2,3,2,1,0,3,2,3

Simulating LRU in Software

- LRU hardware is not usually available. NFU (Not Frequently Used) is implemented in software.
  - ✓At each clock interrupt, the R bit is added to the counter associated with each page. When a page fault occurs, the page with the lowest counter is replaced.
  - ✓Difference? Problem?

  NFU never forgets, so a page referenced frequency long ago may have the highest counter.
- Modified NFU = NFU with Aging - at each clock interrupt:
  - ✓ The counters are shifted right one bit, and
  - ✓ The R bits are added to the leftmost bit.
  - ✓ In this way, we can give higher priority to recent R values.

Simulating LRU in Software



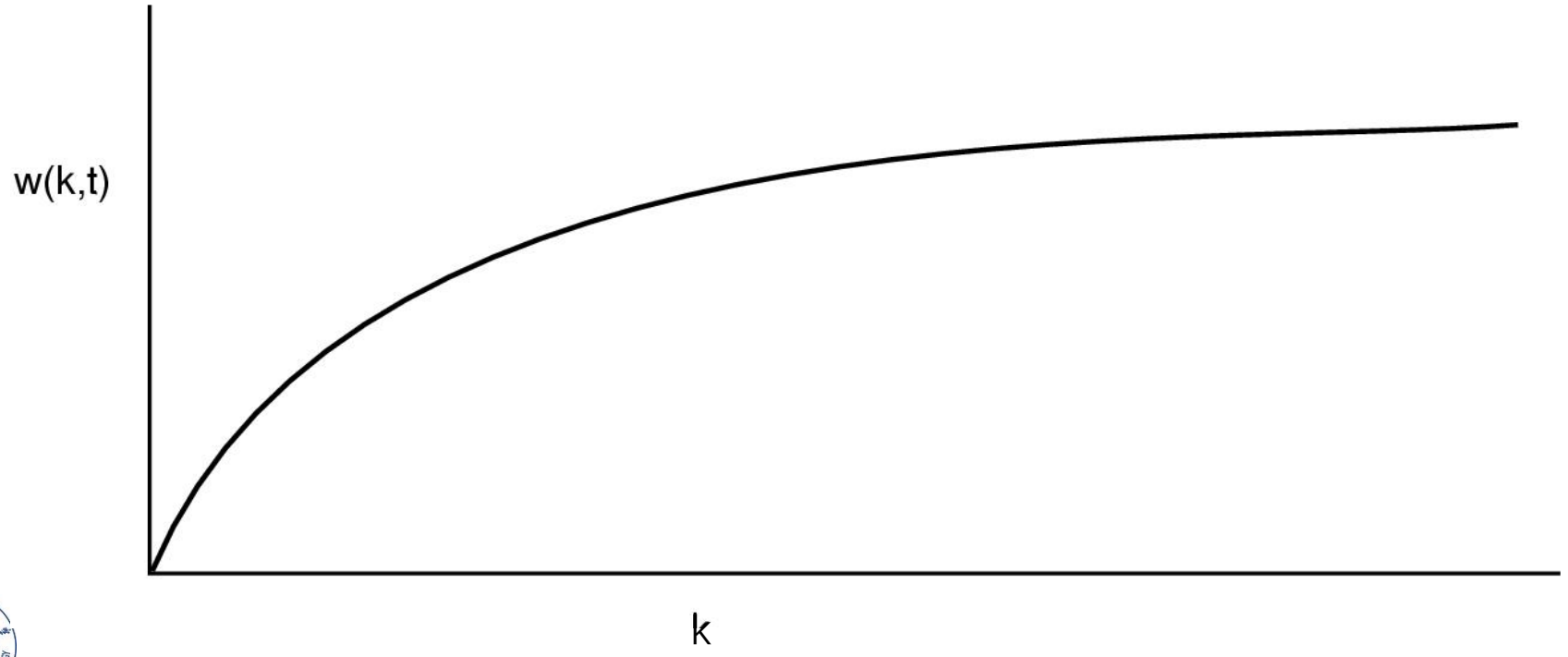
- The aging algorithm simulates LRU in software
- Note 6 pages for 5 clock ticks, (a) – (e)

Working-Set Model

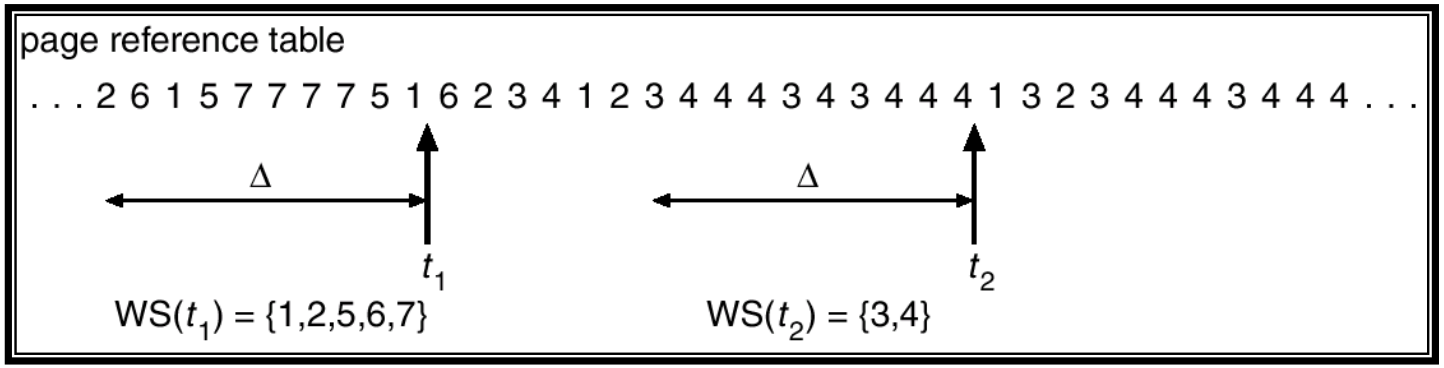
- Pages are loaded only on demand. This strategy is called **demand paging**.
- During the phase of execution the process references relatively small fraction of its pages. This is called a **locality of reference**.
- The set of pages that a process is using currently is called its **working set**.
- A program causing page faults every few instructions is said to be **thrashing**.
- Paging systems keep each process's working set in memory before letting the process run. This approach is called the **working set model**.

Working-Set Model

- Loading the pages before letting processes run is called **prepaging**.
- The working set is the set of pages used by the *k* most recent memory references, *w*(*k*,*t*) is the size of the working set at time *t*.



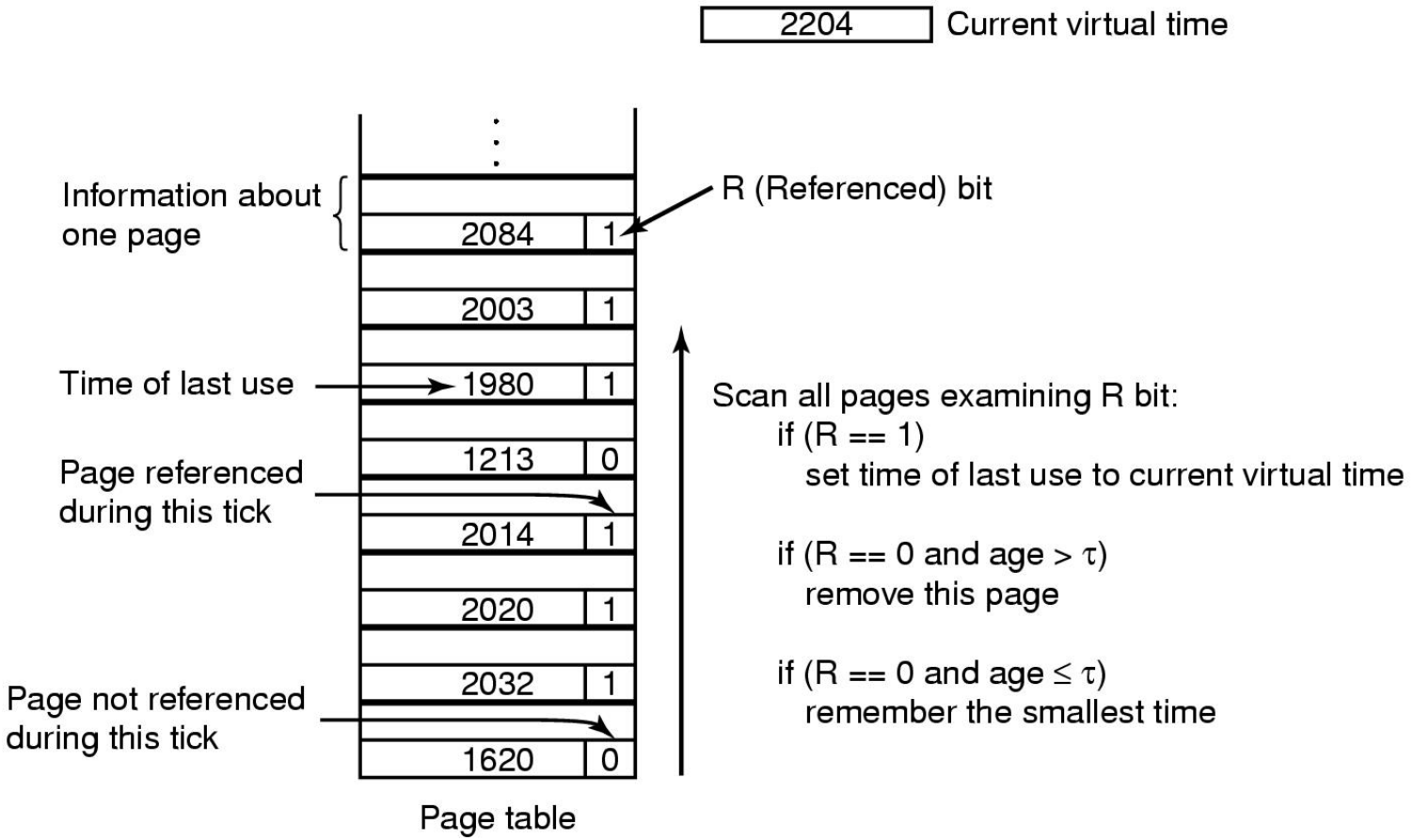
Working-set model



Working-Set Model

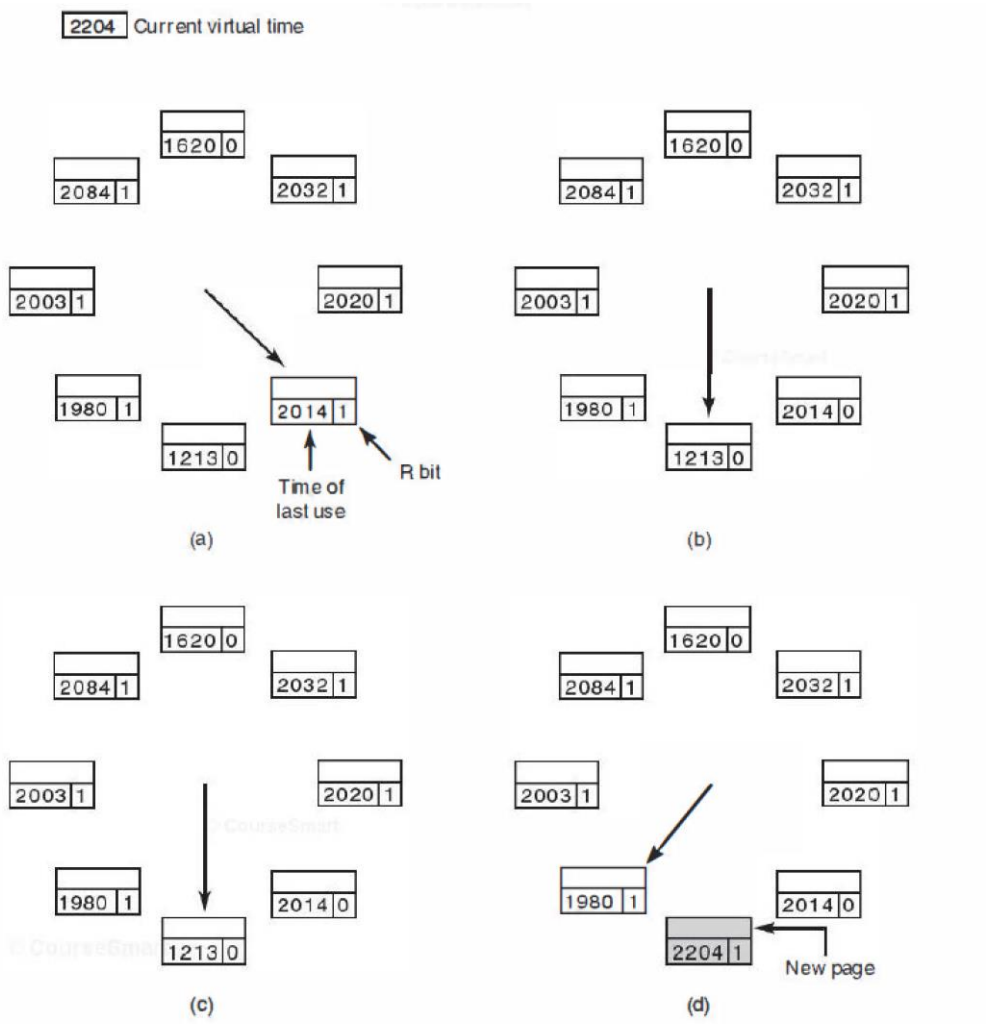
- The idea is to examine the most recent page references. Evict a page that is not in the working set.
- The working set of a process is the set of pages it has referenced during the past τ seconds of **virtual time** (the amount of CPU time a process has actually used).
- Scan the entire page table and evict the page:
  - ✓R= 0, its age is greater than τ.
  - ✓R = 0, its age is not greater than τ and its age is largest.
  - ✓R = 1, randomly choose a page.
- The basic working set algorithm is expensive. Instead, WSCLock is used in practice.

The Working Set Page Replacement Algorithm



The working set algorithm

The WSClock Page Replacement Algorithm



Operation of the WSClock algorithm

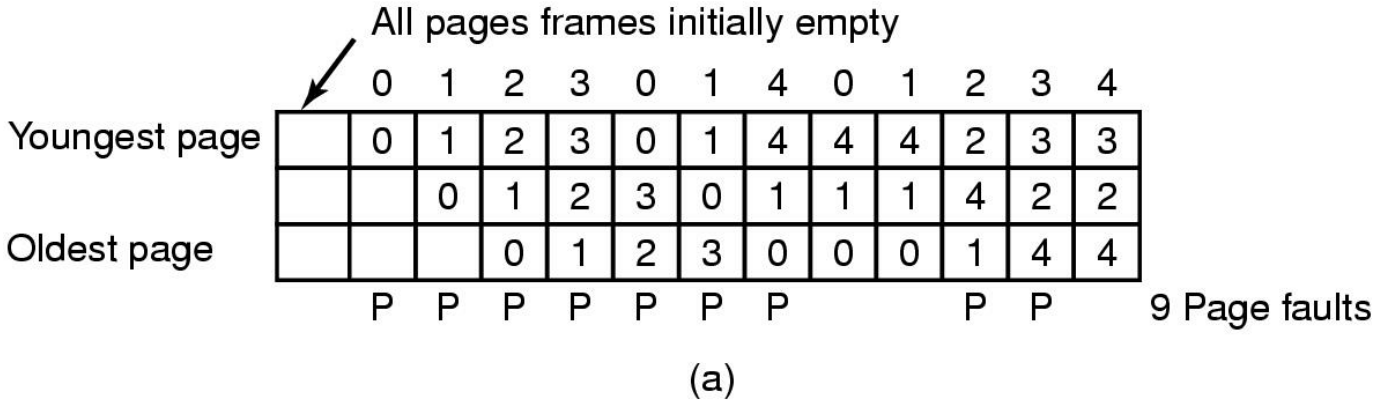
Review of Page Replacement Algorithms

Algorithm	Comment
Optimal	Not implementable, but useful as a benchmark
NRU (Not Recently Used)	Very crude
FIFO (First-In, First-Out)	Might throw out important pages
Second chance	Big improvement over FIFO
Clock	Realistic
LRU (Least Recently Used)	Excellent, but difficult to implement exactly
NFU (Not Frequently Used)	Fairly crude approximation to LRU
Aging	Efficient algorithm that approximates LRU well
Working set	Somewhat expensive to implement
WSClock	Good efficient algorithm

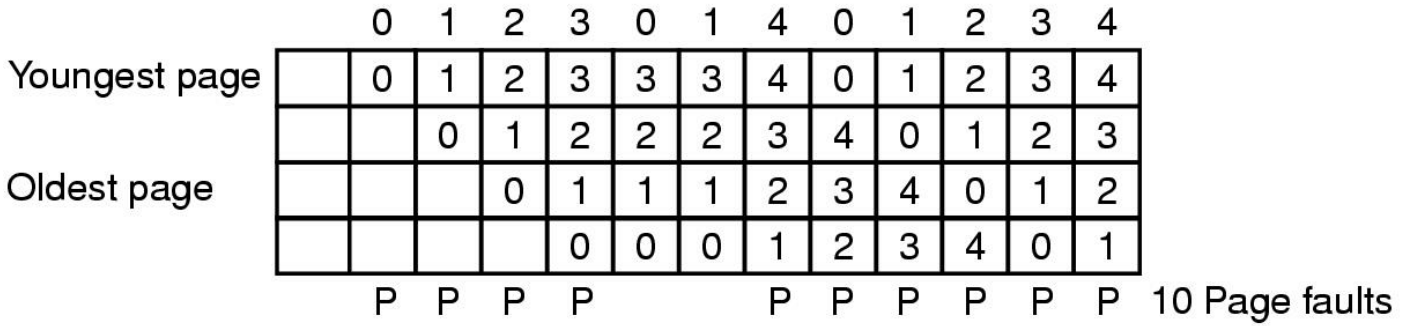
Modeling Page Replacement Algorithms

- **Belady's anomaly:**

More page frames might not always have fewer page faults.



(a)



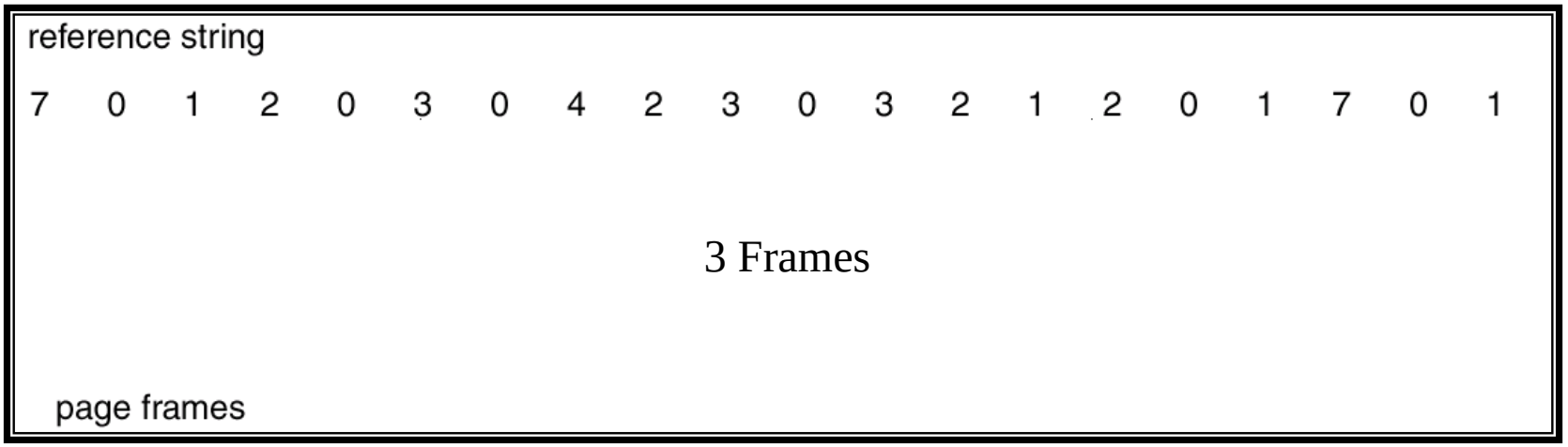
(b)

Modeling Page Replacement Algorithms

- Modeling LRU Algorithms:
  - ✓When a page is referenced, it is always moved to the top entry in pages in memory.
  - ✓If the page referenced was already in memory, all pages above it move down one position. Pages that below the referenced page are not moved.

Reference string	0	2	1	3	5	4	6	3	7	4	7	3	3	5	5	3	1	1	1	7	1	3	4	1
	0	2	1	3	5	4	6	3	7	4	7	3	3	5	5	3	1	1	1	7	1	3	4	1
		0	2	1	3	5	4	6	3	7	4	7	7	3	3	5	3	3	3	1	7	1	3	4
			0	2	1	3	5	4	6	3	3	4	4	7	7	7	5	5	5	3	3	7	1	3
				0	2	1	3	5	4	6	6	6	6	4	4	4	7	7	7	5	5	5	7	7
					0	2	1	1	5	5	5	5	5	6	6	6	4	4	4	4	4	4	5	5
						0	2	2	1	1	1	1	1	1	1	1	6	6	6	6	6	6	6	6
							0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Page faults	P	P	P	P	P	P	P		P					P			P						P	
Distance string	∞	∞	∞	∞	∞	∞	∞	4	∞	4	2	3	1	5	1	2	6	1	1	4	2	3	5	3

Page Replacement



- FIFO： 15 page faults
- LRU： 12 page faults

Stack Replacement Algorithms

- A page replacement algorithm is called a **stack replacement algorithm** if the set of pages in a k-frame memory is always a subset of the pages in a (k + 1) frame memory.

Reference string	0	2	1	3	5	4	6	3	7	4	7	3	3	5	5	3	1	1	1	7	1	3	4	1
	0	2	1	3	5	4	6	3	7	4	7	3	3	5	5	3	1	1	1	7	1	3	4	1
		0	2	1	3	5	4	6	3	7	4	7	7	3	3	5	3	3	3	1	7	1	3	4
			0	2	1	3	5	4	6	3	3	4	4	7	7	7	5	5	5	3	3	7	1	3
				0	2	1	3	5	4	6	6	6	6	4	4	4	7	7	7	5	5	5	7	7
					0	2	1	1	5	5	5	5	5	6	6	6	4	4	4	4	4	4	5	5
						0	2	2	1	1	1	1	1	1	1	1	6	6	6	6	6	6	6	6
							0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Page faults	P	P	P	P	P	P	P		P					P			P						P	
Distance string	∞	∞	∞	∞	∞	∞	∞	4	∞	4	2	3	1	5	1	2	6	1	1	4	2	3	5	3

M is the set of memory array, after each item in reference string is processed， m is the number of page frames, then M(m)⊆ M(m+1).

Check Points

1. What is the drawback of FIFO?
2. What is working set?
3. What is thrashing?
4. What is Belady's anomaly ?
5. What is demand paging?

Session 1

【Objective and Requirement】

Objective： Be familiar with the creation of process and thread.

Requirement：

Task 1: Create a console application, “child”, which keeps printing out “The child is talking at [system time]” (in a loop, one per 1s).

Task 2: Create another console application, “parent”. It create a child **process** to execute “child”. At the same time, the “parent” process keeps printing out “The parent is talking at [system time]”. (one per 1s). Execute “parent” and explain the output you see.

Task 3: Create a child thread in the “mainThread” program. Both the main thread and the child thread keep printing out “[ThreadID] + [System time]”.

Task 4: Create a console application, which contains a shared integer shared_var. The initial value of shared_var is 0. The application will create a child **thread** after it starts. The main thread keeps increasing the value of shared_var by 1, while the child thread keeps decreasing the value of shared_var by 1. Explain the observed results.

【Environment】

Operating System：Linux;