Page Replacement Algorithms

Lectuers
by
muntaha

Plan of Action

- What is paging?
- Page frame and hit ratio ?
- What is page replacement?
- Why we need a page replacement algorithm?
- What are the algorithms?

What is Paging?

- In computer operating systems, **paging** is one of the memory-management schemes by which a computer can store and retrieve data from secondary storage for use in main memory.
- In the paging memory-management scheme, the operating system retrieves data from secondary storage in same-size blocks called *pages*.
- ***Advantage:***
- is that it allows the physical address space of a process to be non-contiguous.
- Before paging came into use, systems had to fit whole programs into storage contiguously, which caused
  various storage and fragmentation problems.

Page frame and hit ratio

- Page Frame: - The main memory is divided into fixed sized portions called page frame.
- The size of page frame is same as the size of page because pages have to reside in page frames.
- Hit Ratio: - The ratio of the total number of hit divided by the total CPU access to memory (i.e. hits plus misses) is called hit ratio.
- Hit ratio=$\frac{\text{total number of hits}}{\text{total number f hits + total number of mis}}$

Page fault

A **page fault** (sometimes called **#PF**, **PF** or **hard fault**)
is a type of interrupt, called trap, raised by computer hardware when a running program accesses a memory page that is mapped into the virtual address space, but not actually loaded into main memory. The hardware that detects a page fault is the processor's memory management unit (MMU), and transfers control from the program to the operating system.

The operating system must:

- Obtain an empty page frame in RAM to use as a container for the data.
- Load the requested data into the available page frame.
- Update the page table to refer to the new page frame.
- Return control to the program.
- When all page frames are in use, the operating system must select the page frame to reuse for the page the program now needs., which is its page replacement algorithm.

What is page replacement?

- When memory located in secondary memory is needed, it can be retrieved back to main memory.
- Process of storing data from main memory to secondary memory ->**swapping out**
- Retrieving data back to main memory ->**swapping in**

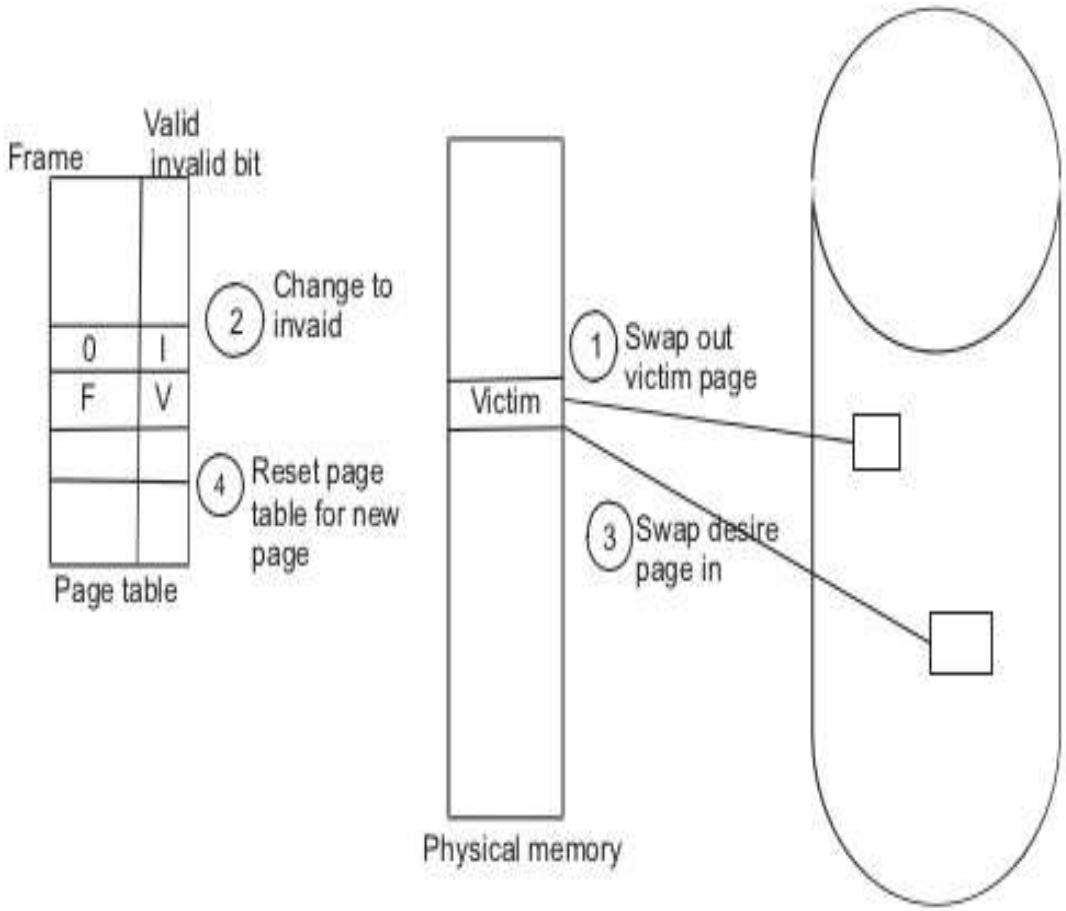
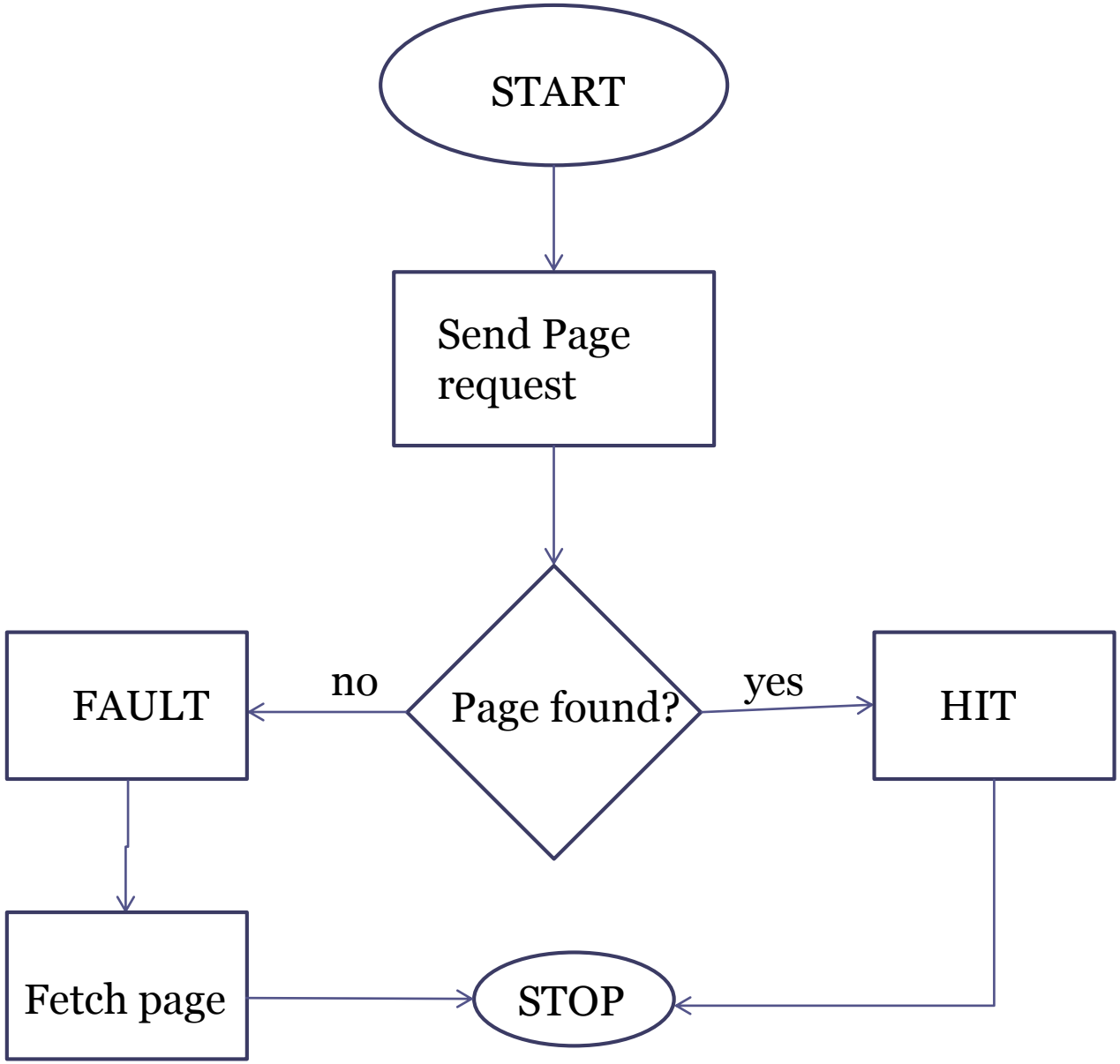


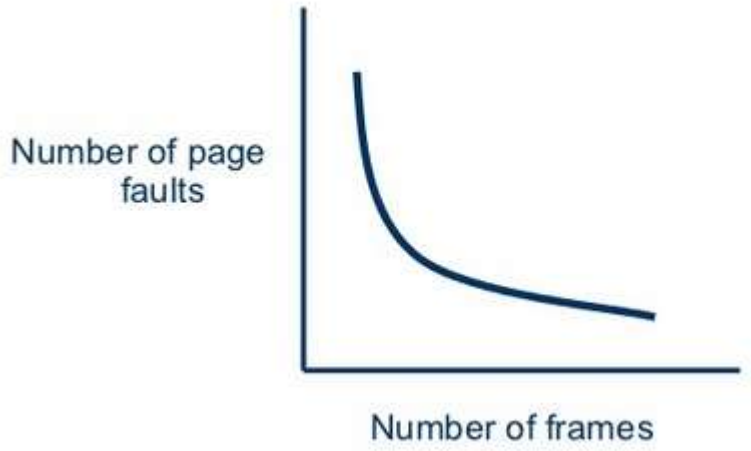
Fig: Page Replacement

Why we need a page replacement algorithm?

- The main goal of page replacement algorithms is to provide lowest page fault rate.


START
Send Page request
Page found?
no
yes
FAULT
HIT
Fetch page
STOP

No. of Page Faults Vs No. of Frames



Algorithms

- First In First Out.
- Optimal Replacement.
- Second Chance.
- The Clock Algorithm.
- Least Recently Used.
- Least frquently Used.
- Not Recently Used.
- ARC(adaptive replacement cache).
- CAR(Clock adaptive replacement).

First-In First-Out (FIFO)

- Pages in main memory are kept in a list
- First in first out is very easy to implement
- The fifo algorithm select the page for replacement that has been in memory the longest time

FIFO Example

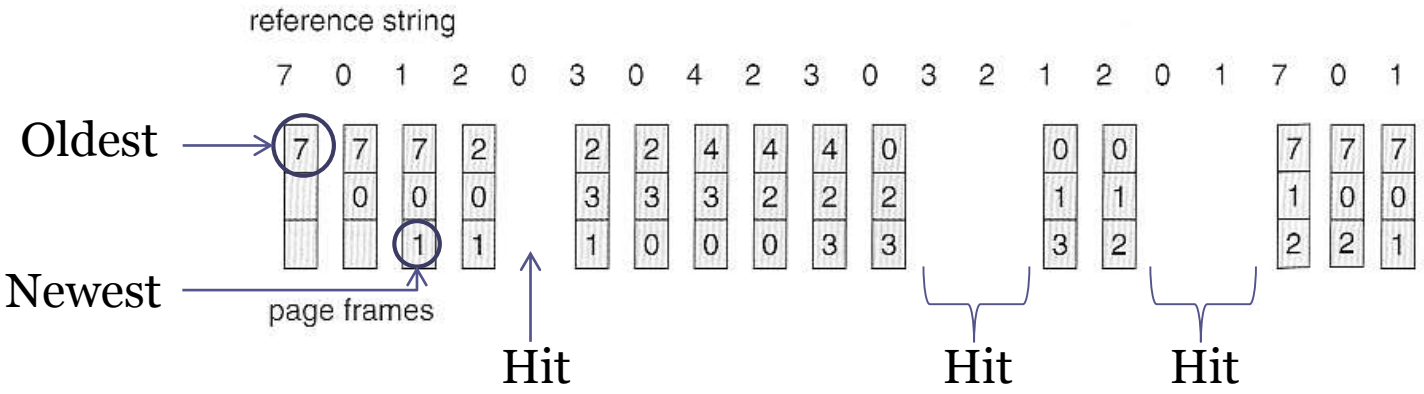


Fig: FIFO example

(FIFO)

- **Advantages**
  - FIFO is easy to understand.
  - It is very easy to implement.

- **Disadvantage**
- The oldest block in memory may be often used.

Optimal Replacement (OPT)

- The optimal policy selects that page for replacement that will not be needed for the longest time
- The OS keeps track of all pages referenced by the program
- This algorithm result is fewest number of page faults.
- **Advantages**
  - Optimal.
- **Disadvantages**
  - Impossible to implement (need to know the future) but serves as a standard to compare with the other algorithms.

OPTIMAL Example

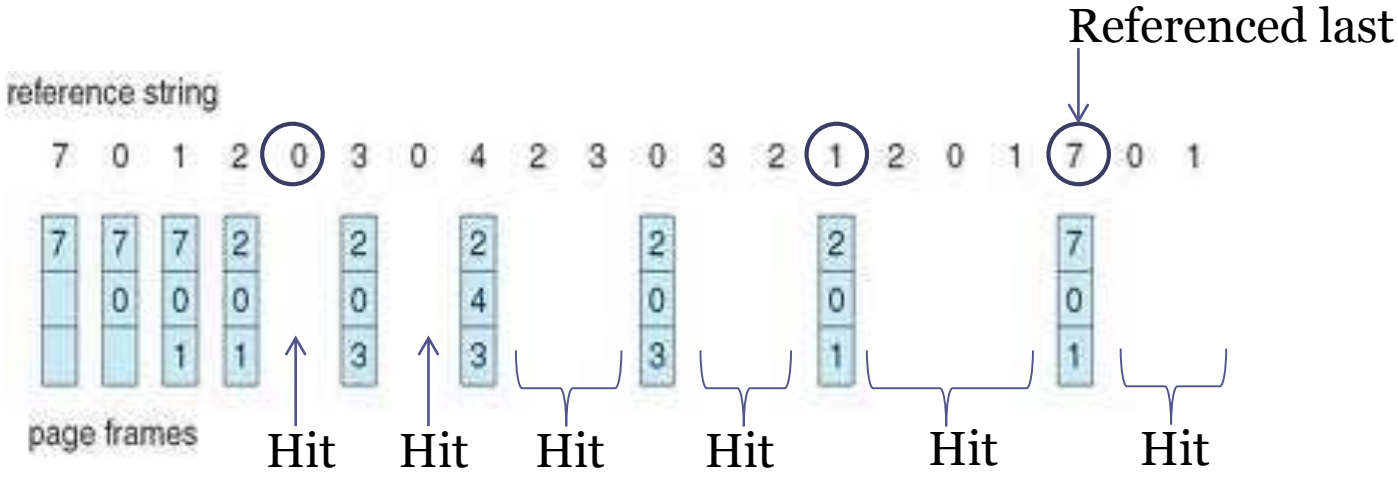
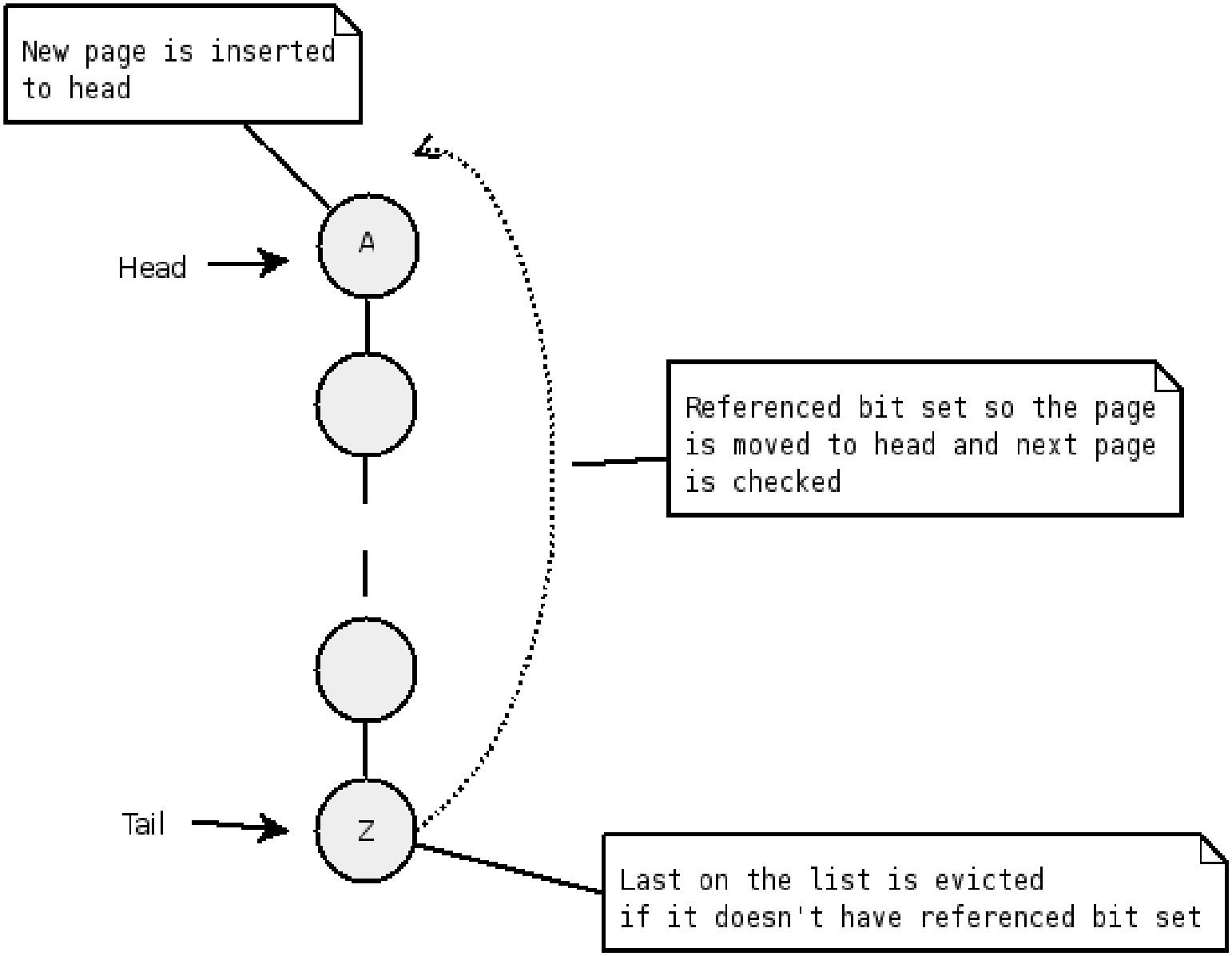


Fig: OPTIMAL example

Second chance alg.

- Pages kept in a linked list
  - Oldest is at the front of the list
- Look at the oldest page
  - If its “referenced bit” is 0…
    - Select it for replacement
  - Else
    - It was used recently; don’t want to replace it
    - Clear its “referenced bit”
    - Move it to the end of the list
  - Repeat
- What if every page was used in last clock tick?
  - Select a page at random


New page is inserted
to head
A
Head
Referenced bit set so the page
is moved to head and next page
is checked
Tail
Z
Last on the list is evicted
if it doesn't have referenced bit set

Second chance alg

Advantages

- Better performance than FIFO.

Disadvantages

- Not practical (evicted block may be needed in a short time later).

The Clock Algorithm

- Variant of FIFO & LRU
- Keep frames in circle
- On page fault, OS:
- Checks reference bit of next frame
- If reference bit = 0, replace page, set bit to 1
- If reference bit = 1, set bit to 0, advance pointer to next frame

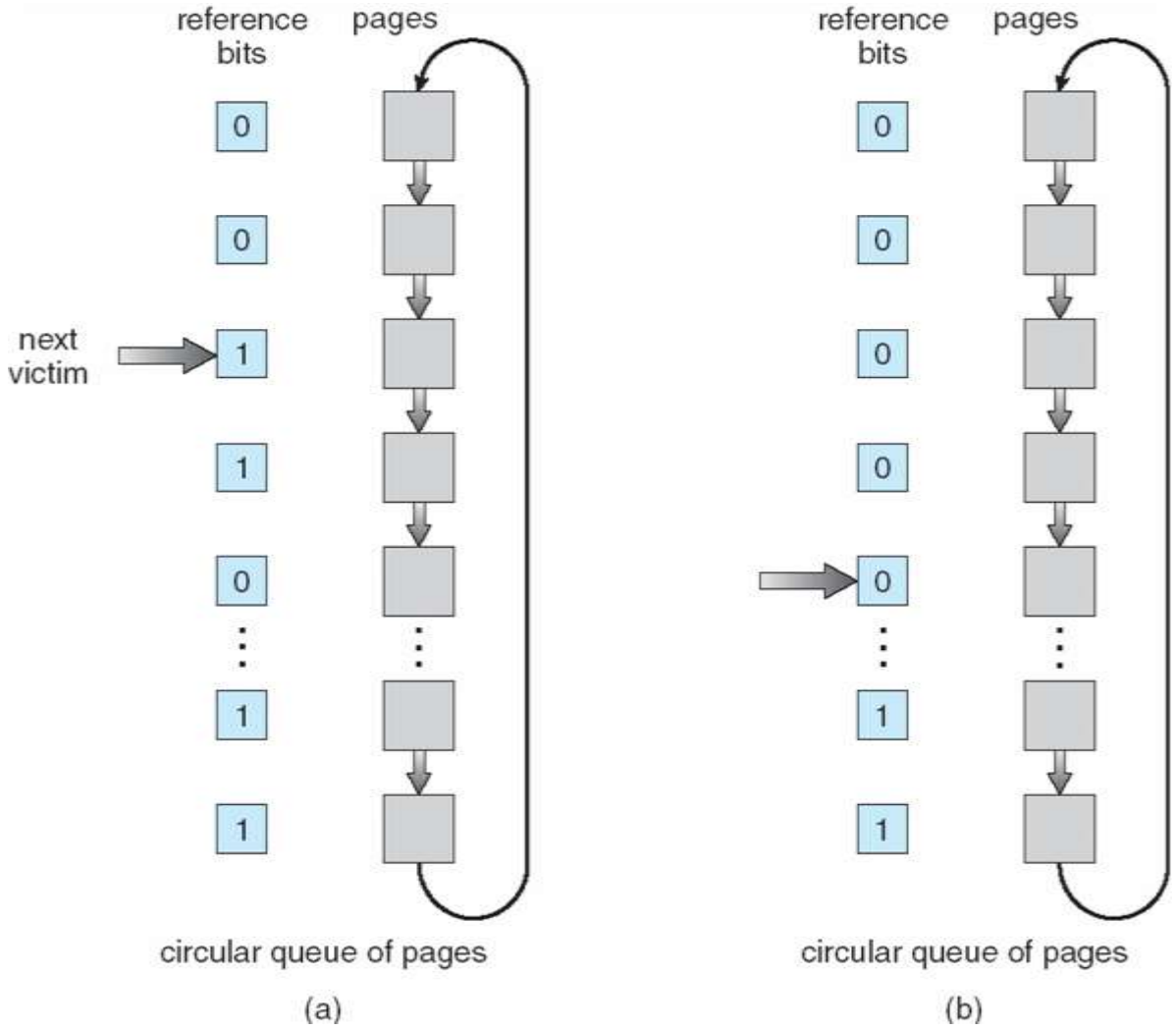
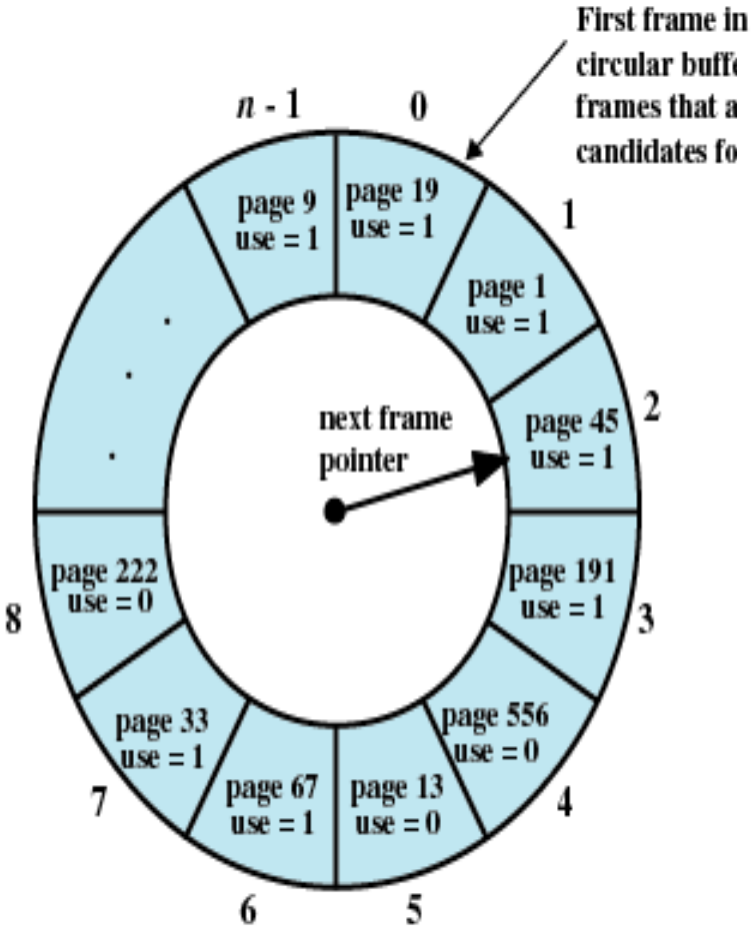
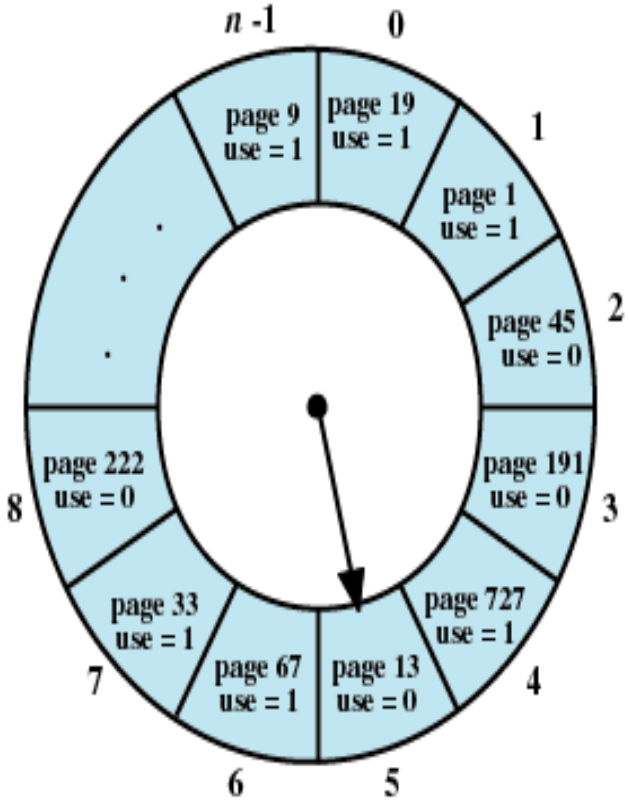

reference bits
pages
0
0
next victim
1
1
0
1
1
circular queue of pages
(a)
reference bits
pages
0
0
0
0
0
1
1
circular queue of pages
(b)

Fig: CLOCK



(a) State of buffer just prior to a page replacement



(b) State of buffer just after the next page replacement

Figure 8.16 Example of Clock Policy Operation

The Clock Algorithm

- **Advantages**
- Easy to implement.

Disadvantages

- Not practical (evicted block may be needed in a short time later).
- Not scan resistant.

Least Recently Used (LRU)

- The least recently used page replacement algorithm keeps track page uses over a short period of time.

LRU Example

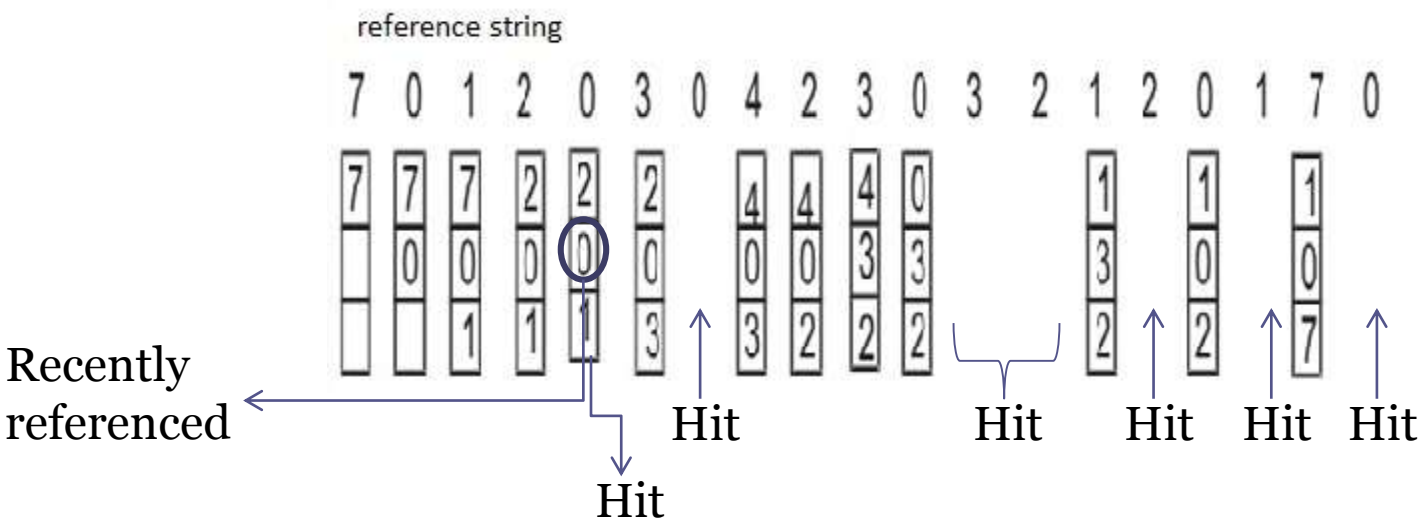
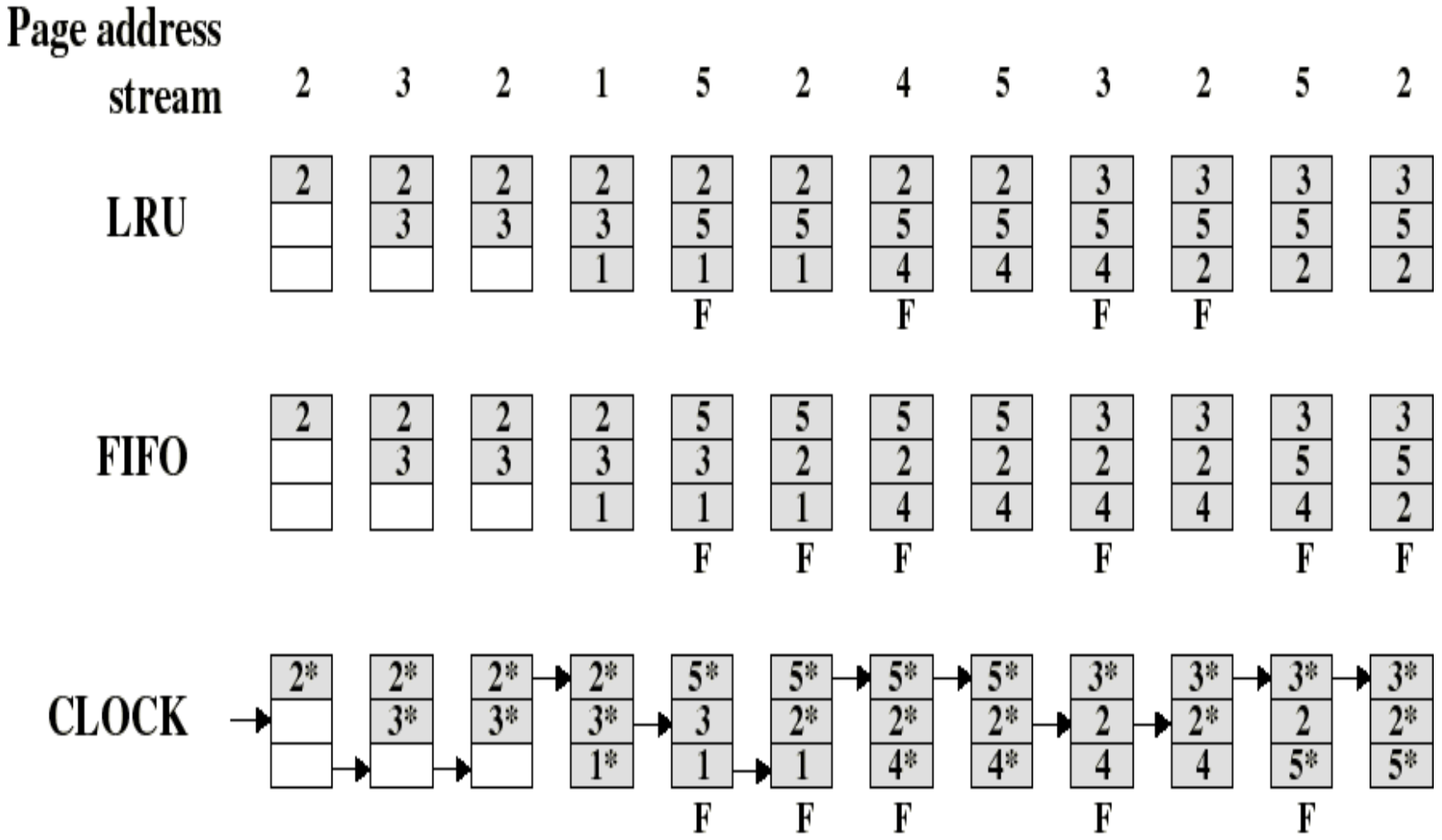


Fig: LRU example

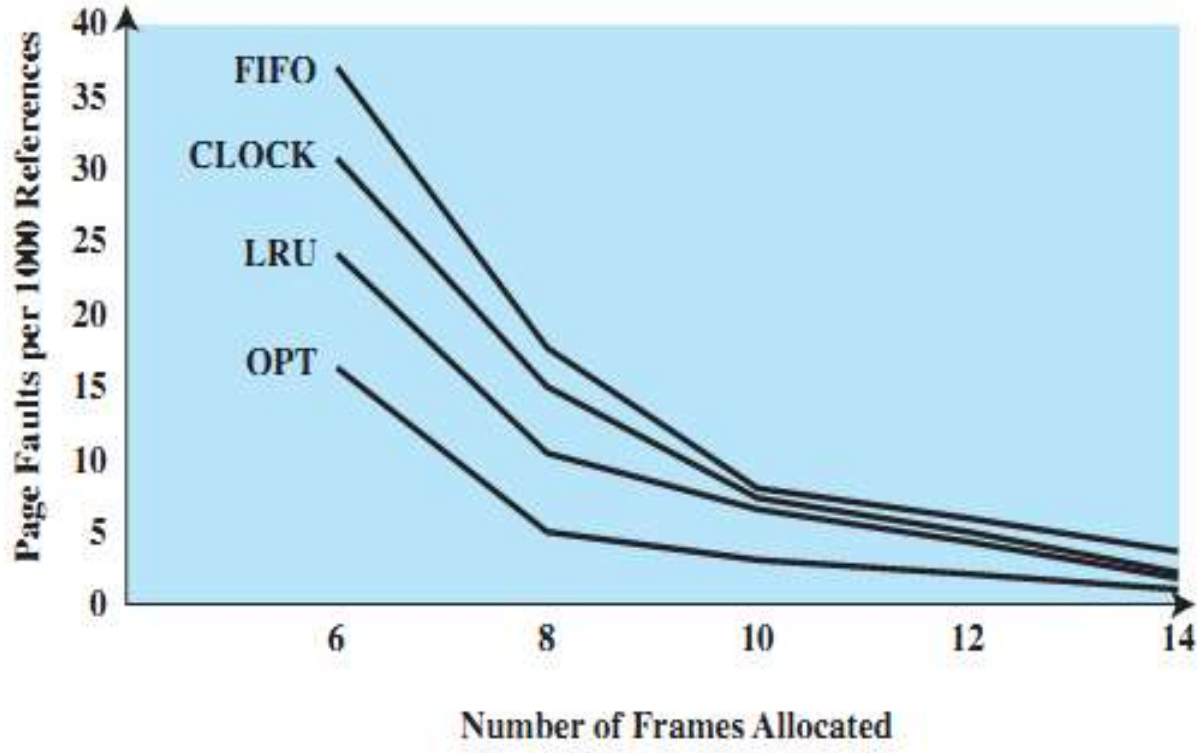
LRU

- **Advantages**
  - LRU page replacement algorithm is quiet efficient.
- **Disadvantages**
  - Implementation difficult. This algorithm requires keeping track of what was used when, which is expensive if one wants to make sure the algorithm always discards the least recently used item.

Comparison of Clock with FIFO and LRU (1)



Fixed-Allocation, Local Page Replacement



Least-Frequently-Used Algorithm (LFU)

- The Least-Frequently-Used (LFU) Replacement technique replaces the least-frequently block in use when an eviction must take place.
- Software counter associated with each block, initially zero is required in this algorithm.
- The operating system checks all the blocks in the cache at each clock interrupt.
- The R bit, which is '0' or '1', is added to the ounter for each block. Consequently, the counters are an effort to keep track of the frequency of referencing each block.
- When a block must be replaced, the block that has the lowest counter is selected for the replacement.

(LFU)

Advantages

- Frequently used block will stay longer than (fifo)

Disadvantages

- Older blocks are less likely to be removed , even if they are on longer frequently used because this algorithm never forgets anything.
- Newer blocks are more likely to be replaced even if they are frequently used.
- Captures only frequency factor.

The Not Recently Used Page Replacement Alg.

This algorithm requires that each page have two additional status bits 'R' and 'M' called reference bit and change bit respectively. The reference bit(R) is automatically set to 1 whenever the page is referenced. The change bit (M) is set to 1 whenever the page is modified. These bits are stored in the PMT and are updated on every memory reference.

- When a page fault occurs, the memory manager inspects all the pages and divides them into 4 classes based on R and M bits.
  - **Class 1: (0,0)** – neither recently used nor modified - the best page to replace.
  - **Class 2: (0,1)** – not recently used but modified - the page will need to be written out before replacement.
  - **Class 3: (1,0)** – recently used but clean - probably will be used again soon.
  - **Class 4: (1,1)** – recently used and modified - probably will be used again, and write out will be needed before replacing it.

NRU (Not Recently Used)

- Advantages?
  - It is easy to understand.
  - It is efficient to implement.
  - Much better behavior than FIFO
  - Frequently used pages are much more likely to stay
- Disadvantages?
  - Behavior sucks near clock cycles

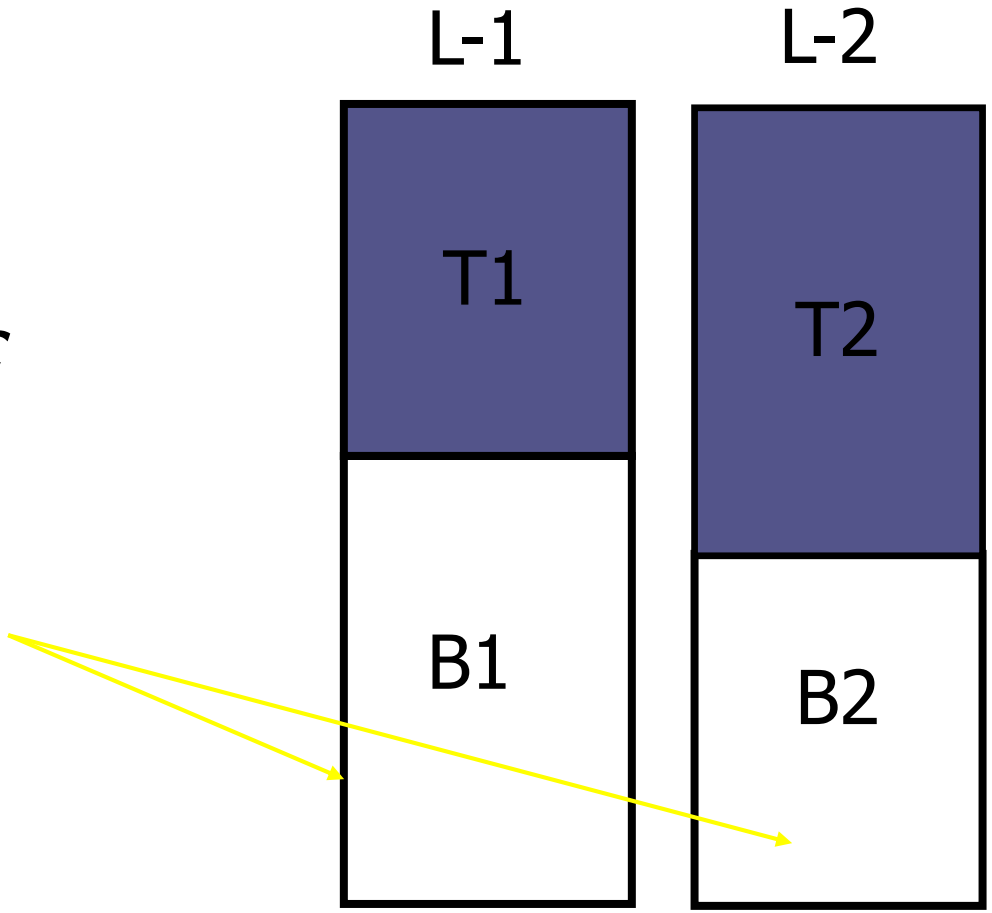
ARC(adaptive replacement cache)

- ARC is used (4) linked lists:
- T1, for recent cache entries.
- T2, for frequent entries, referenced at least twice.
- B1, *ghost* entries recently evicted from the T1 cache, but are still tracked.
- B2, similar *ghost* entries, but evicted from T2.
- T1 and B1 together are referred to as L1, a combined history of recent single references. Similarly, L2 is the combination of T2 and B2.
- Note that the *ghost* lists only contain metadata (keys for the entries) and not the resource data itself.

ARC

$|T1| + |T2| = c$

"Ghost caches"
(not in memory)



ARC

advantages

- self – tuning
- Low overhead
- Capture both frequency and recency

Disadvantages

- Slow.
- complex.
- Can not implemented in the real world.

Clock with Adaptive Replacement (AR)

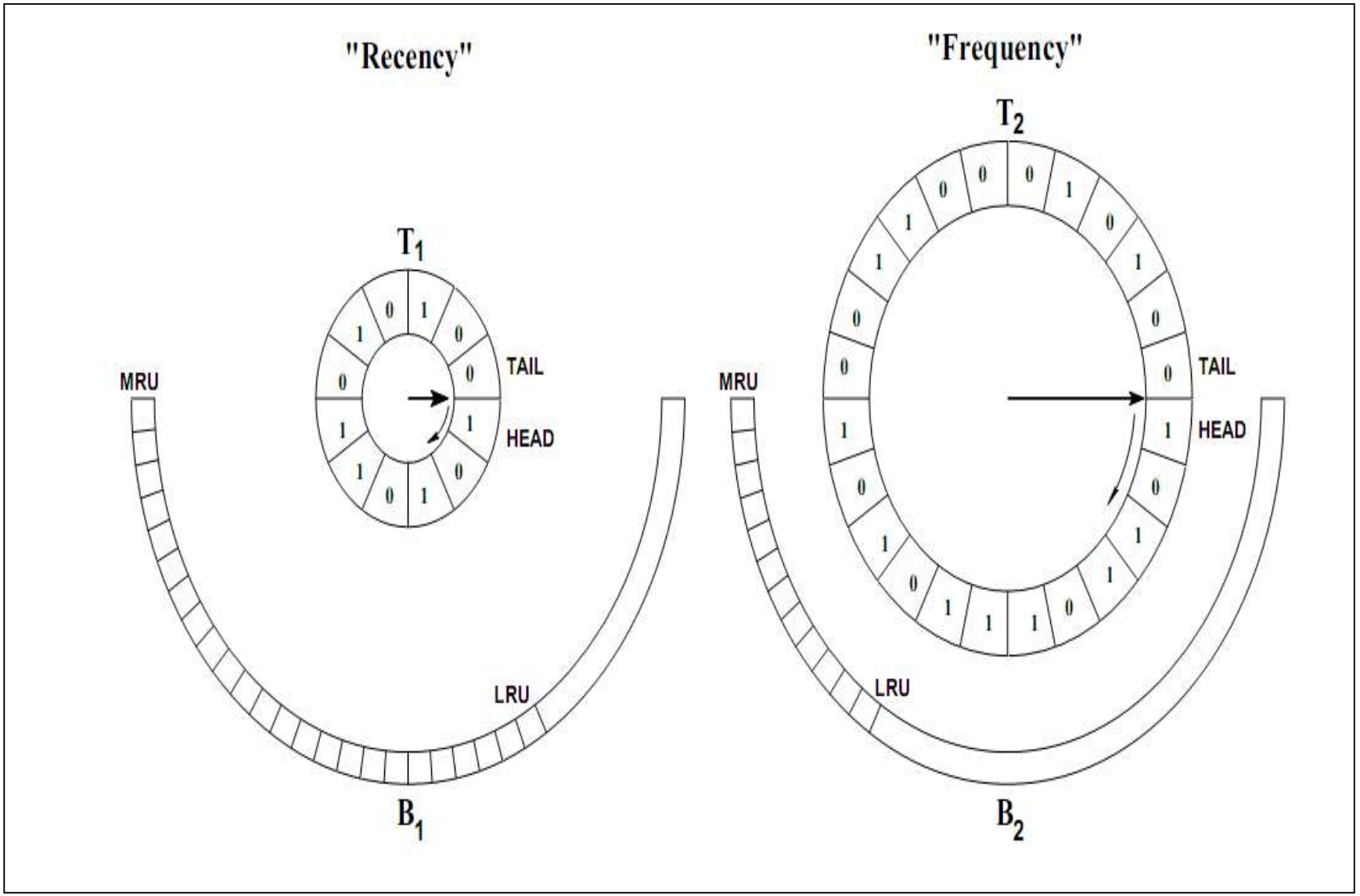
- Combines the advantages of Adaptive Replacement Cache (ARC) and CLOCK.
- Like ARC, It uses 4 doubly linked lists. 2 clocks T1 and T2 and 2 simple LRU lists B1 and B2.
- T1 clock stores pages based on 'recency' or 'short term utility' whereas T2 stores pages with 'frequency".
- T1 and T2 contain those pages that are in the cache, while B1 and B2 contain pages that have recently been evicted from T1 and T2 respectively.
- New pages are inserted in T1 or T2.
- If there is a hit in B1 size of T1 is increased and similarly if there is a hit in B2 size of T1 is decreased.
- The adaptation rule used has the same principle as that in ARC.

Advantages

- scan resistant.
- self – tuning
- Low overhead
- Capture both frequency and recency
- faster than ARC algorithm.

Disadvantages

- complex.
- Can not implemented in the real world(as hardware).


"Recency"
"Frequency"
T_1
T_2
MRU
MRU
TAIL
TAIL
HEAD
HEAD
LRU
LRU
B_1
B_2


Thank
You