

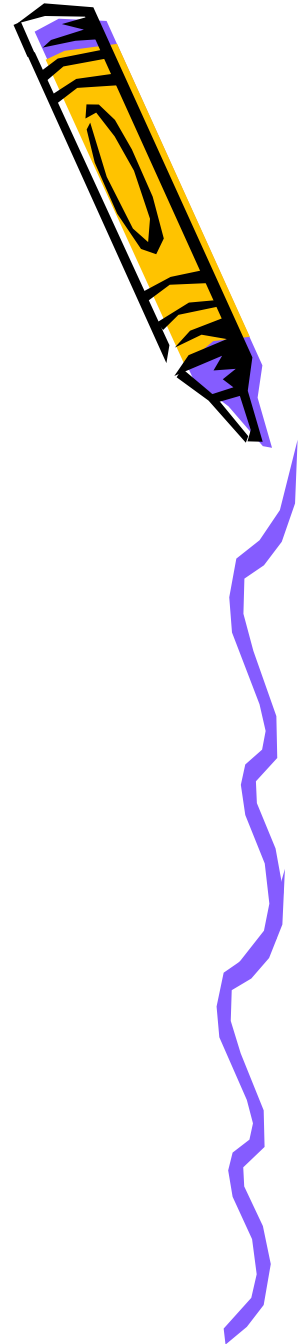
# Algoritma Pergantian Page (*Page Replacement*)

Heri Kurniawan  
OS-Gasal 2009/2010



# Tujuan Pembelajaran

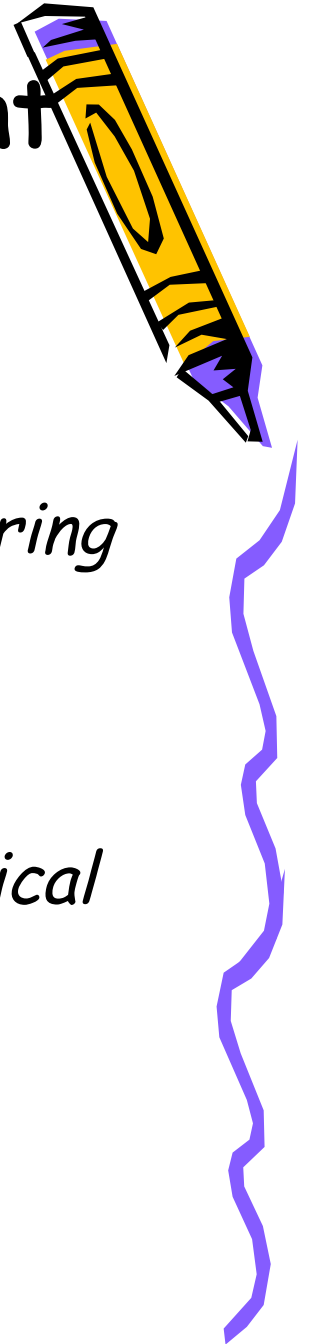
- Memahami algoritma pergantian page
  - FIFO
  - Optimal
  - Least Recently Used (LRU)
  - Least Recently Used (LRU) Approximation
  - Second-chance
  - Circular Queue (Algoritma Clock)
  - Enhanced Second-Chance
  - Counting-Based : LFU & MFU



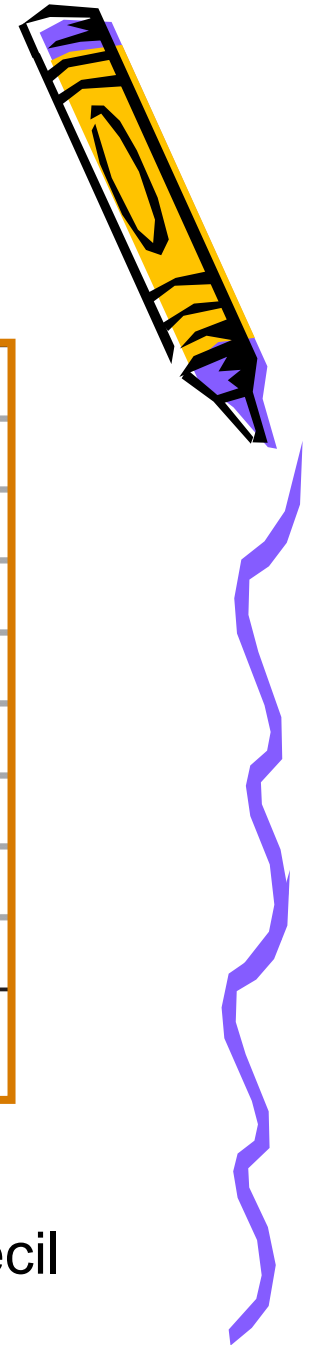
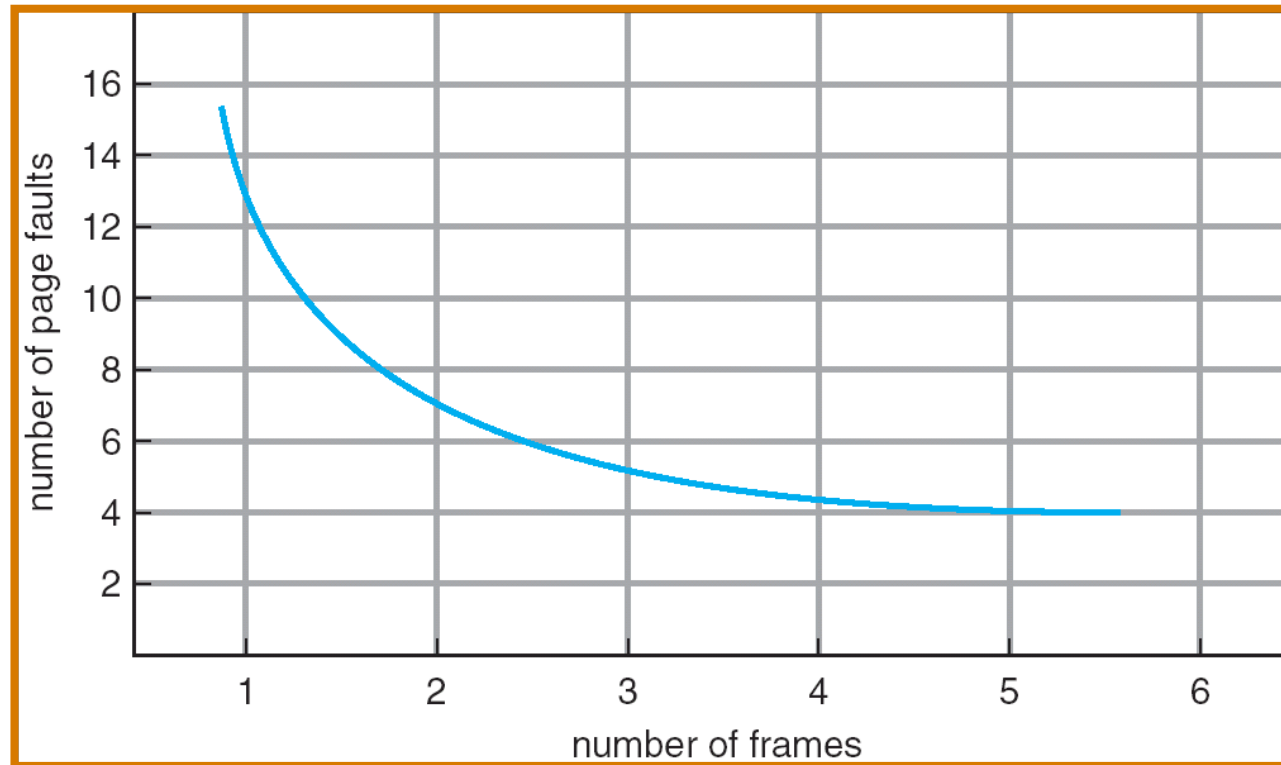
# Algoritma Page Replacement

- Tujuan  $\Rightarrow$  Mencari algoritma dengan *page fault rate* terkecil
- Evaluasi algoritma  $\rightarrow$  jalankan kumpulan *string* (reference string) yang merujuk kelokasi memori dan hitung *page fault* dari string tersebut.
- String menandakan nomor page, bukan *logical address*!
- Misalnya, referensi *string* nya

1, 4, 1, 6, 1, 6, 1, 6, 1, 6, 1



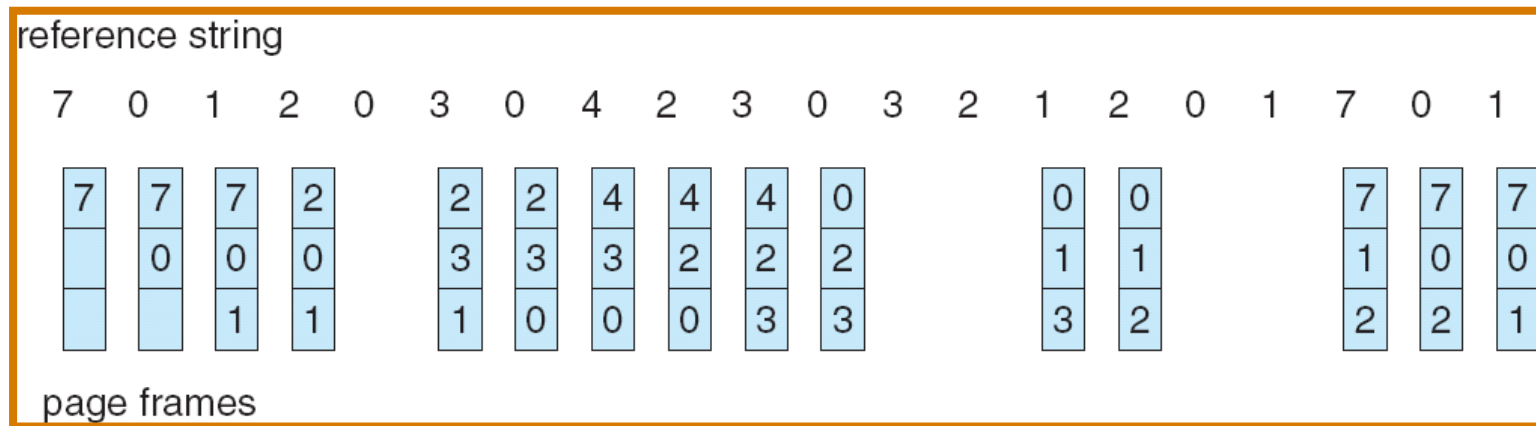
## Grafik Page Fault vs Jumlah Frame



Semakin besar jumlah frame, jumlah page fault semakin kecil

# Algoritma FIFO

Page yang menempati memori paling lama dipilih untuk diganti



Total page fault = 15

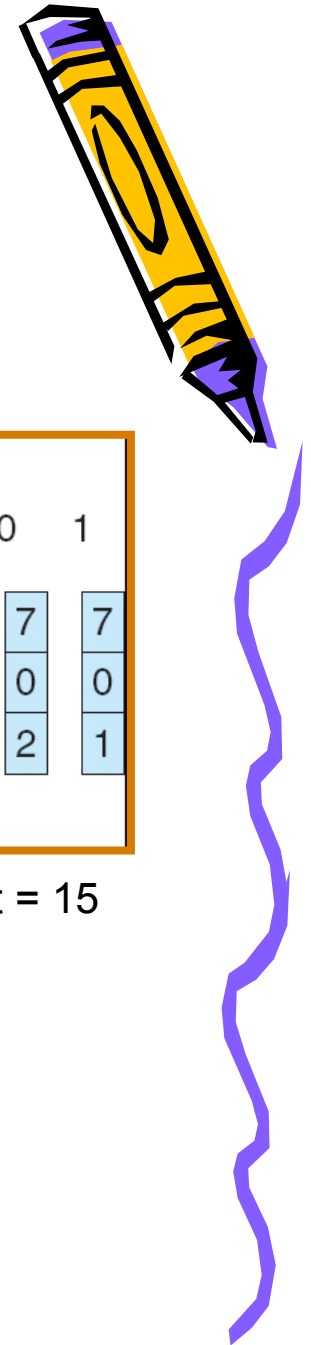
(7,0,1) page fault, dimasukkan kedalam frame

(7,0,1)→(2,0,1)

2 menggantikan 7 karena entri 7 paling tua

(2,0,1)→(2,3,1)

3 menggantikan 0 karena entri 0 paling tua



# FIFO

- FIFO

- Pros

- page lama berisikan inisialisasi modul yang tidak digunakan

- Cons

- Page lama berisikan inisialisasi variabel yang masih digunakan

- Akibat kesalahan penempatan

- Page Fault bertambah
  - Memperlambat Eksekusi



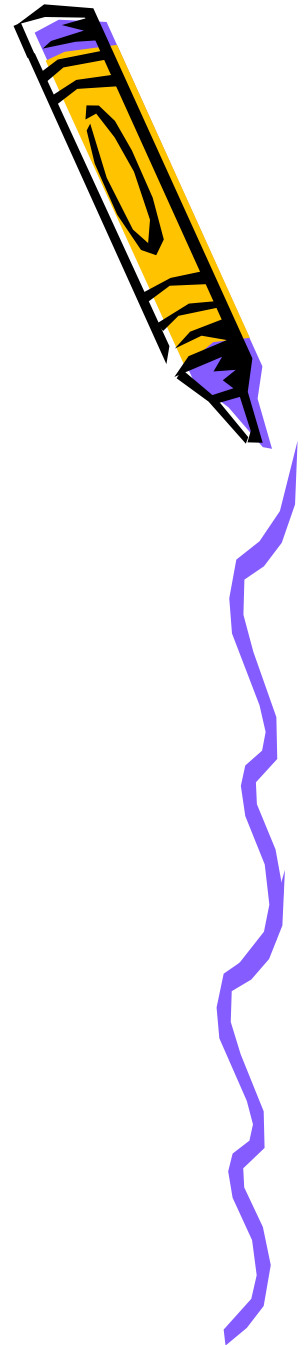
# FIFO

- Referensi : 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- 3 frame

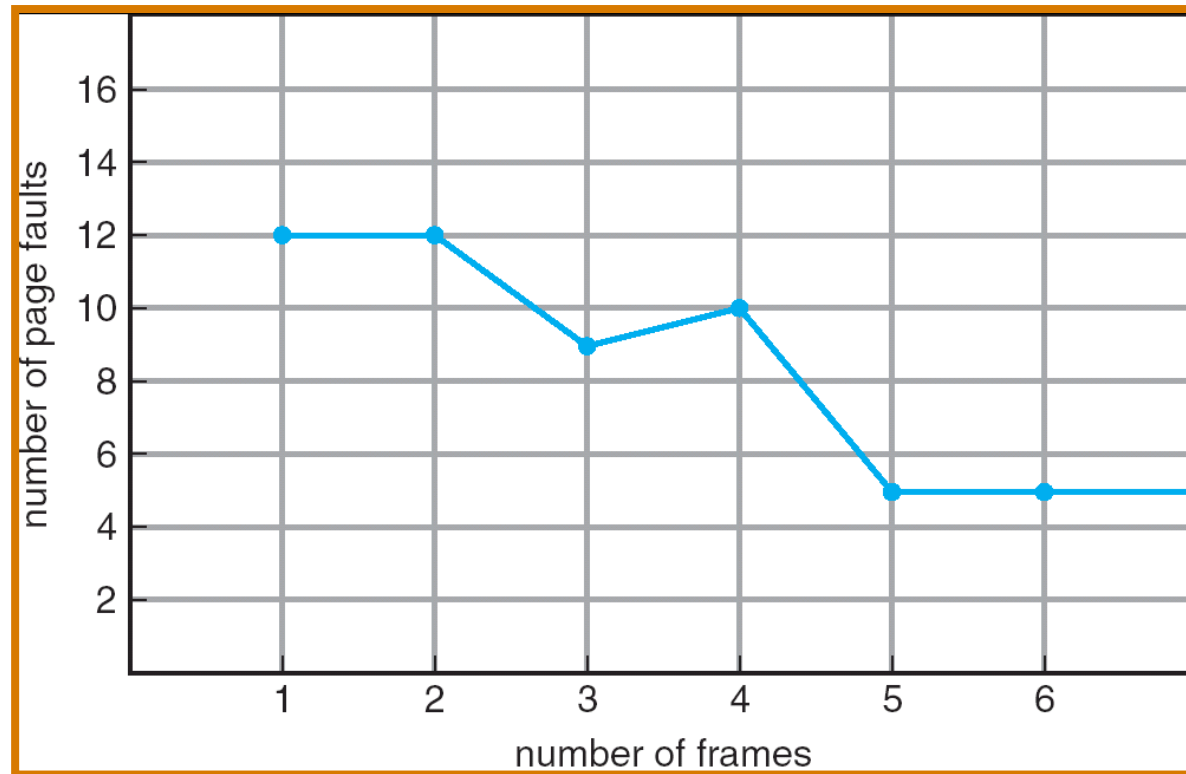
1	1	4	5	
2	2	1	3	9 page faults
3	3	2	4	

- 4 frames

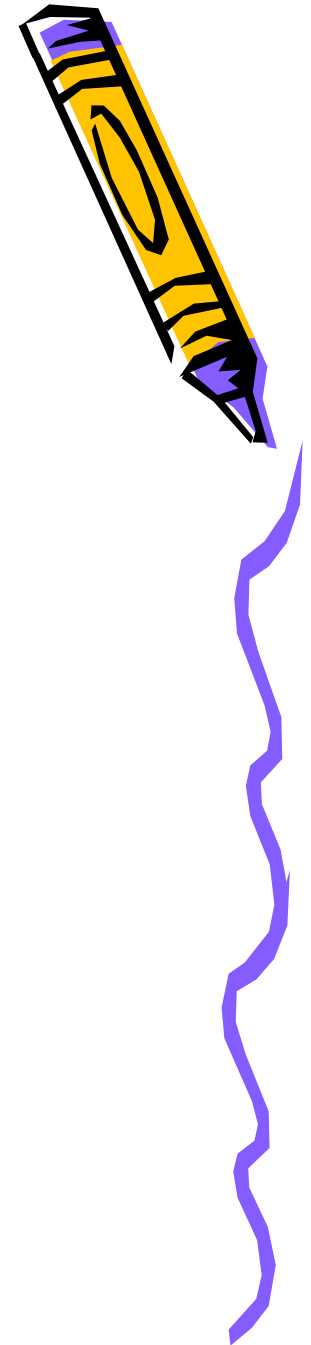
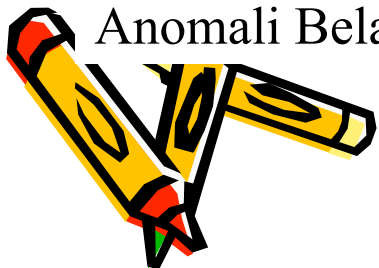
1	1	5	4	
2	2	1	5	10 page faults
3	3	2		
4	4	3		



# FIFO-Anomaly Belady



Anomali Belady: frame bertambah  $\Rightarrow$  page fault bertambah





# Algoritma Optimal

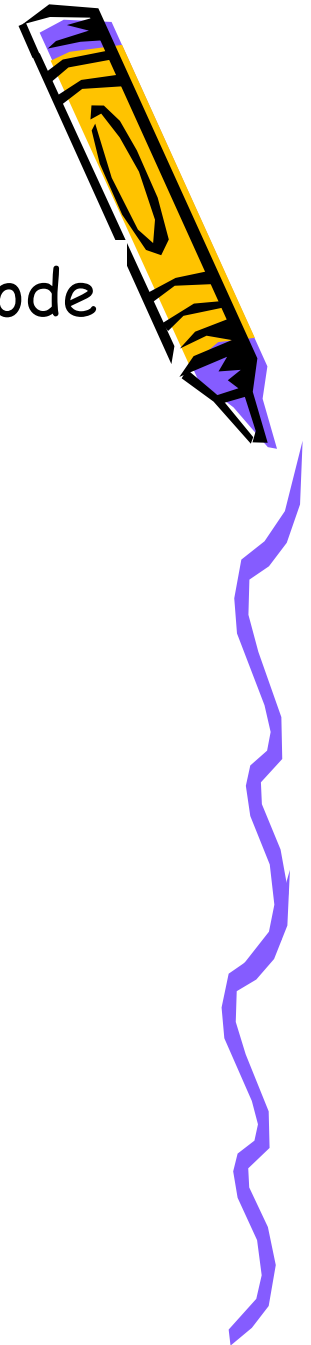
- Ganti page yang tidak akan digunakan pada periode berikutnya dengan waktu gilir yang terlama.
- 4 frame

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

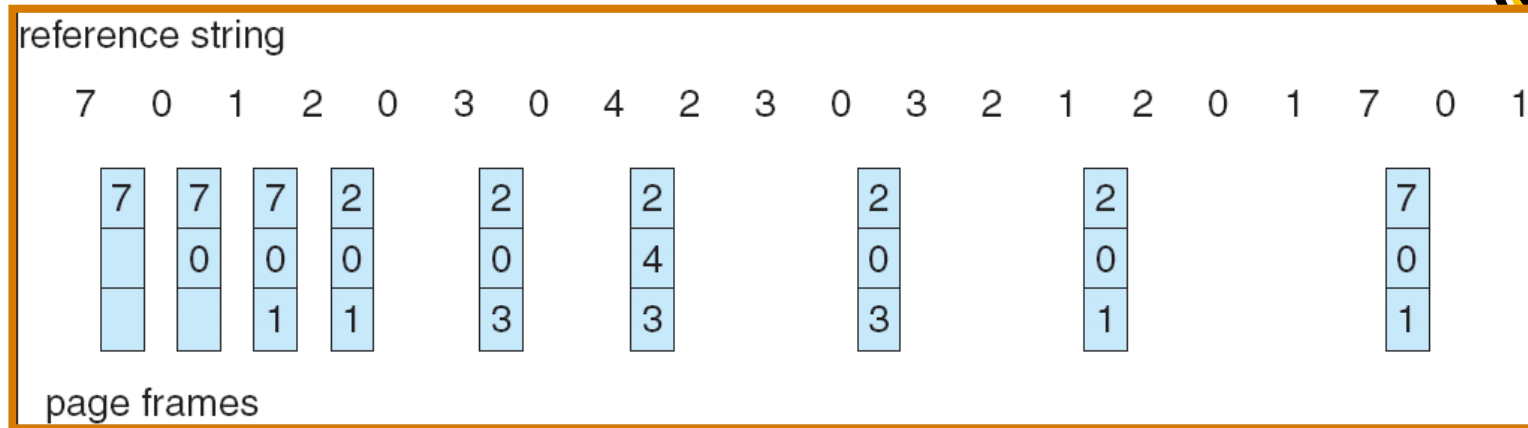
1	4
2	
3	
4	5

6 page faults

- Mempunyai jumlah *page fault* paling rendah
- Bagaimana cara memprediksinya?
- Sulit untuk diimplementasikan



# Algoritma Optimal



Total page fault = 9

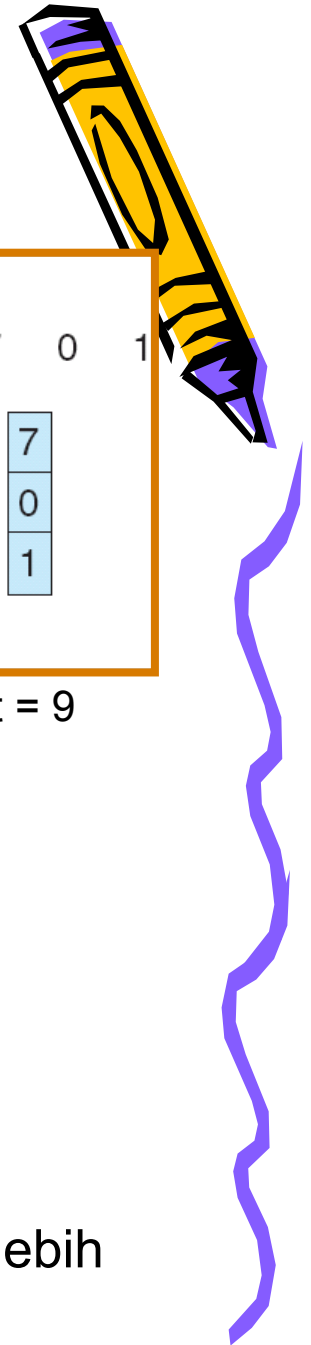
(7,0,1) page fault, dimasukkan kedalam frame

(7,0,1) → (2,0,1)

2 menggantikan 7 karena 7 tidak akan digunakan hingga referensi ke 18

(2,0,1) → (2,0,3)

3 menggantikan 1 karena waktu referensi kembali untuk 1 lebih lama diantara 2 dan 0



# Algoritma Least Recently Used (LRU)

- Perbedaan FIFO & OPT
  - FIFO menggunakan waktu dimulainya sebuah page dipindahkan ke memori
  - OPT menggunakan waktu sebuah page akan digunakan kembali dimasa mendatang
- LRU → cari page yang mempunyai waktu terlama yang belum pernah digunakan diantara page lain



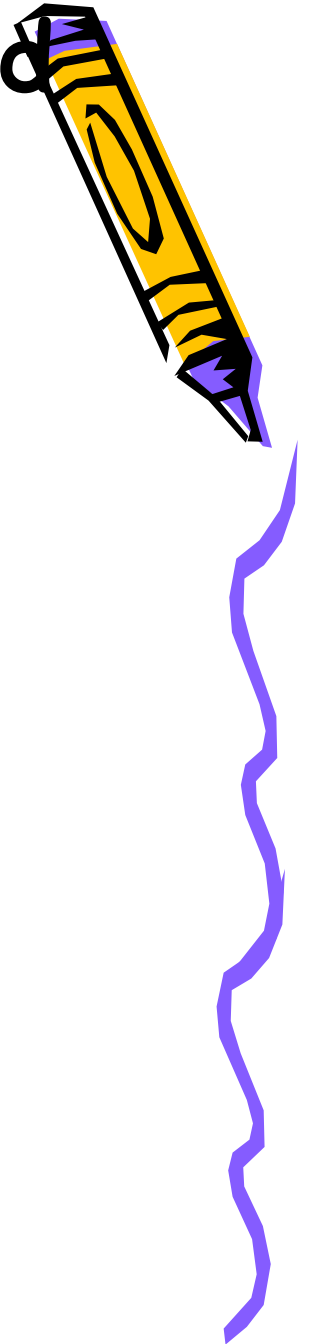
# Algoritma Least Recently Used (LRU)

- Contoh :

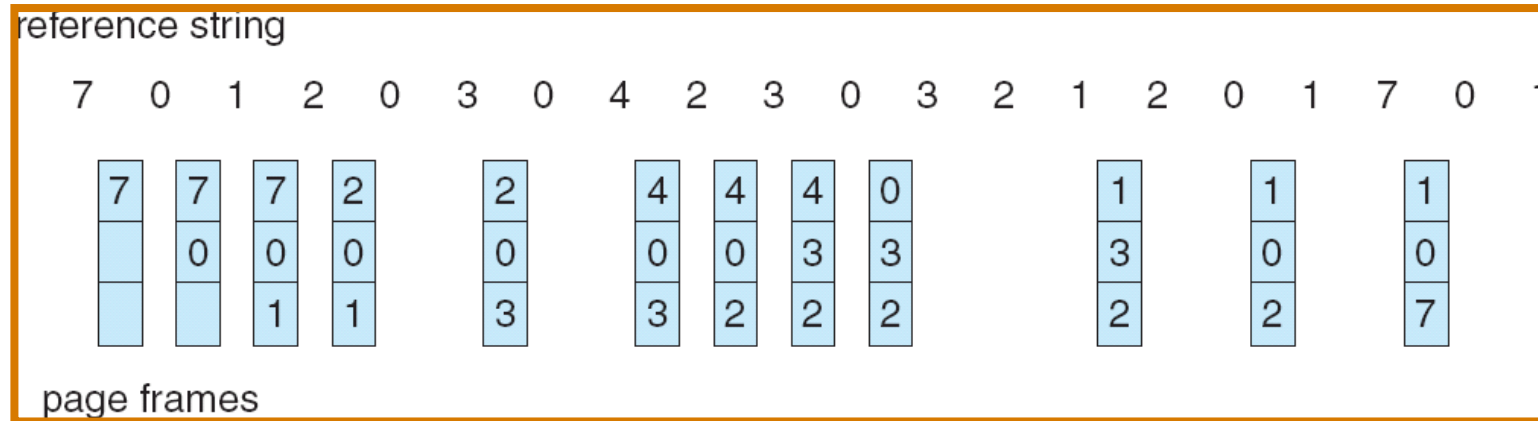
- Referensi string:

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

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



# Algoritma Least Recently Used (LRU)



Total page fault = 12

(7,0,1) page fault, dimasukkan kedalam frame

(2,0,1) -> (2,0,3)

3 menggantikan 1 karena 1 menempati periode waktu paling lama belum digunakan

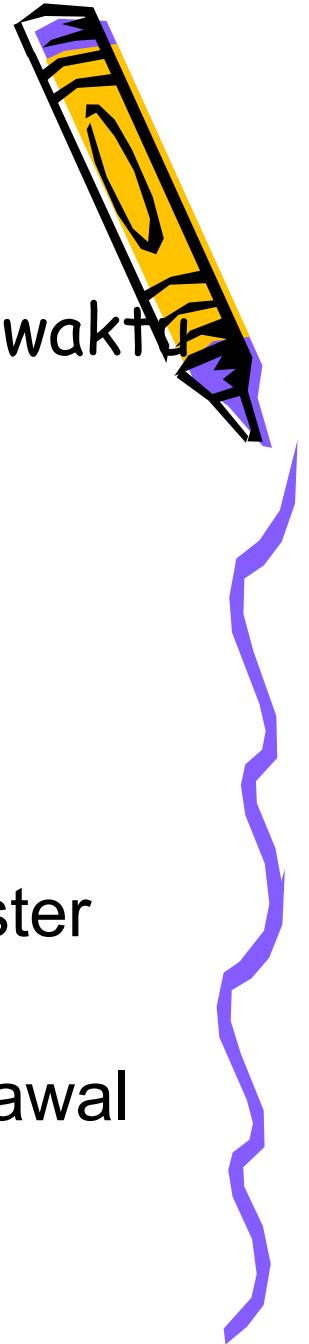
(2,0,3) -> (4,0,3)

4 menggantikan 2 karena 2 menempati periode waktu paling lama belum digunakan



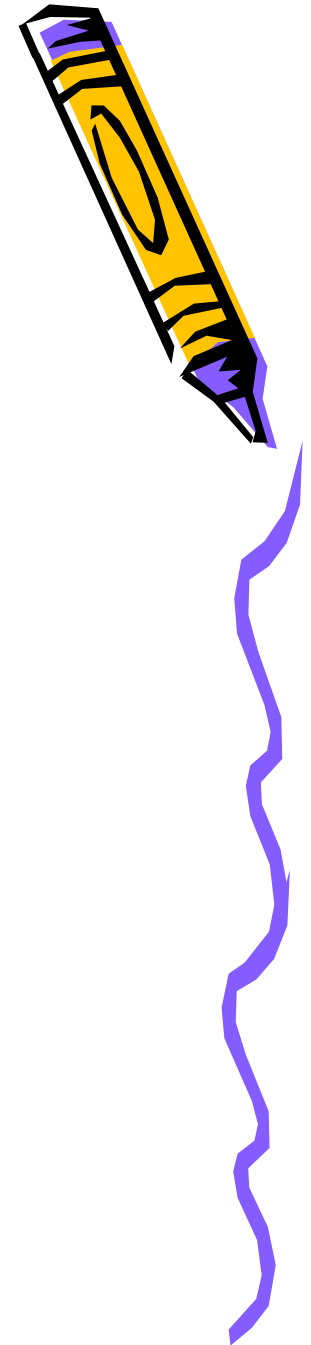
# Algoritma Least Recently Used (LRU)

- Pengaturan penggunaan frame berdasarkan waktu terlama
  - Clock Counter
  - Stack
- Clock Counter
  - Setiap entri page punya field *time-of-use*
  - Jika ada referensi ke suatu page, nilai register clock ditempatkan ke field *time-of-use*
  - Ganti page yang mempunyai waktu paling awal



# LRU dengan counter clock

page:	7	faults:	1	frames:	7	-1	-1	time:	1		
	0		2		7	0	-1		1	2	
	1		3		7	0	1		1	2	3
	2		4		2	0	1		4	2	3
	0		4		2	0	1		4	5	3
	3		5		2	0	3		4	5	6
	0		5		2	0	3		4	7	6
	4		6		4	0	3		8	7	6
	2		7		4	0	2		8	7	9
	3		8		4	3	2		8	10	9
	0		9		0	3	2		11	10	9
	3		9		0	3	2		11	12	9
	2		9		0	3	2		11	12	13
	1		10		1	3	2		14	12	13
	2		10		1	3	2		14	12	15
	0		11		1	0	2		14	16	15
	1		11		1	0	2		17	16	15
	7		12		1	0	7		17	16	18
	0		12		1	0	7		17	19	18
	1		12		1	0	7		20	19	18



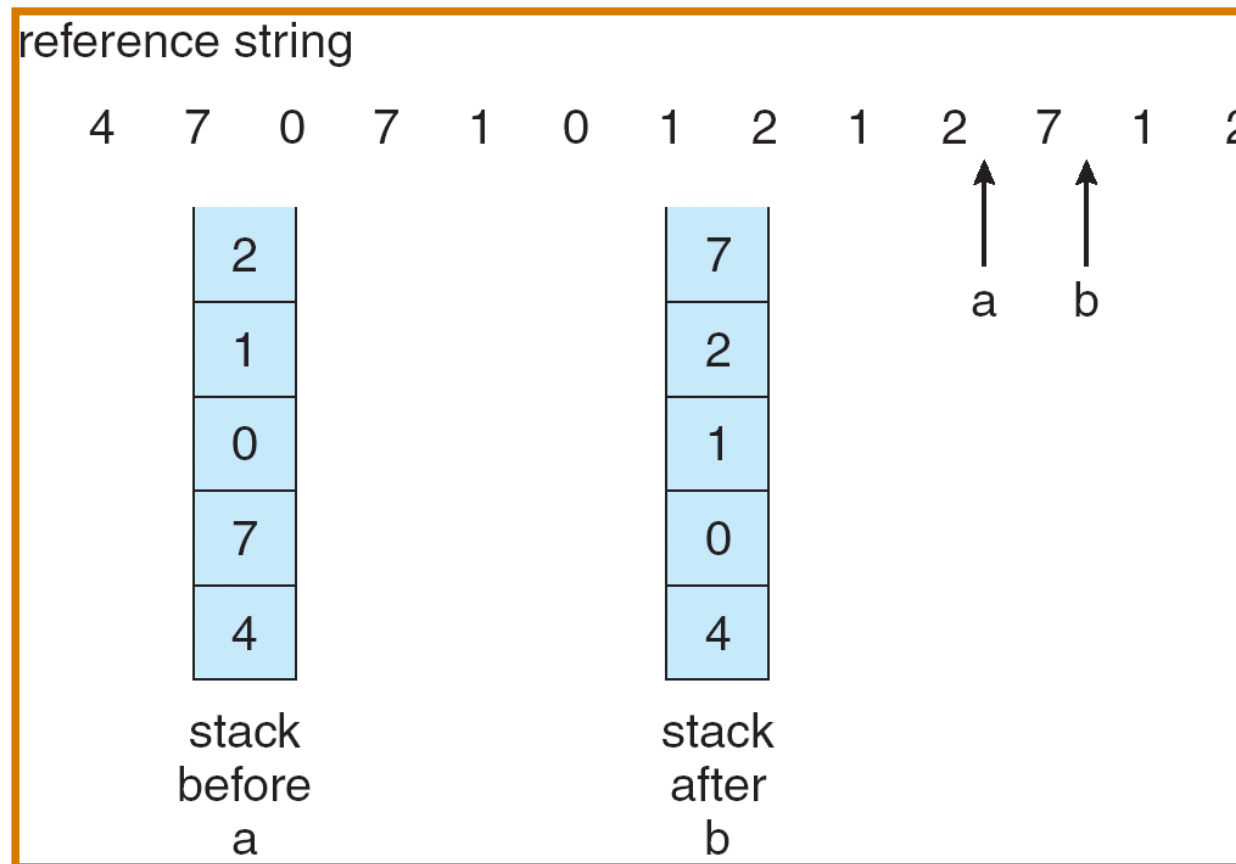
# Algoritma Least Recently Used (LRU)

- Stack
  - Setiap ada referensi page, pindahkan page ke posisi paling atas
  - Page yang paling sering digunakan (*most recently used*) berada diposisi atas.
  - Page yang paling jarang digunakan (*least recently used*) berada diposisi bawah.
  - Umumnya berbentuk *double linked-list*

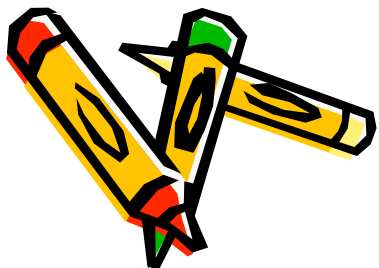




# Algoritma Least Recently Used (LRU)

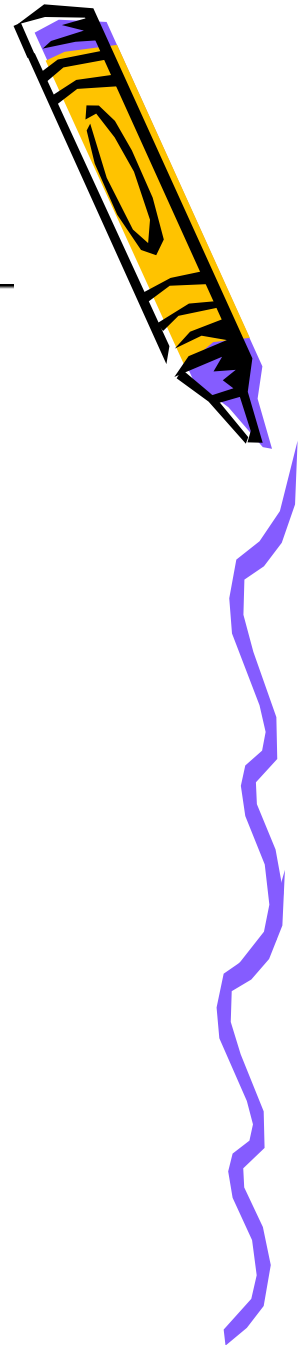


Penggunaan stack



# LRU - Stack

page:	7	faults:	1	frames:	7	-1	-1	stack:	7		
	0		2		7	0	-1		7	0	
	1		3		7	0	1		7	0	1
	2		4		2	0	1		0	1	2
	0		4		2	0	1		1	2	0
	3		5		2	0	3		2	0	3
	0		5		2	0	3		2	3	0
	4		6		4	0	3		3	0	4
	2		7		4	0	2		0	4	2
	3		8		4	3	2		4	2	3
	0		9		0	3	2		2	3	0
	3		9		0	3	2		2	0	3
	2		9		0	3	2		0	3	2
	1		10		1	3	2		3	2	1
	2		10		1	3	2		3	1	2
	0		11		1	0	2		1	2	0
	1		11		1	0	2		2	0	1
	7		12		1	0	7		0	1	7
	0		12		1	0	7		1	7	0
	1		12		1	0	7		7	0	1

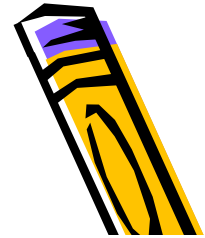


# Aproksimasi algoritma LRU

- Beberapa hardware tidak mendukung LRU, namun mensupport reference bit
- Reference bit digunakan untuk mengimplementasikan algoritma LRU
- Awalnya semua reference bit adalah 0
- Bit referensi (*reference bit*) -> untuk menandakan page sedang digunakan (read /write) atau tidak
  - Nilai awal bit=0 → page tidak digunakan
  - Nilai bit= 1 → page sedang digunakan (direferensi)
  - Ganti page yang mempunyai nilai *reference bit*= 0



## Aproksimasi algoritma LRU (cont.)



- Terjadi interrupt dalam interval 100 msec
- Setiap interval, geser bit paling kiri ke kanan
- Bit referensi (8bit)
  - 8 bit shift-register → history penggunaan page untuk 8 periode
  - 00000000 → page belum pernah digunakan/direference selama 8 periode
  - 11111111 → page digunakan selama 8 periode
  - 11000100 ?
  - 01110111 ?
- Page yang diganti adalah page yang mempunyai bit paling kecil
- Penggunaan Reference bit yang lain ? → lihat algoritma second-chance



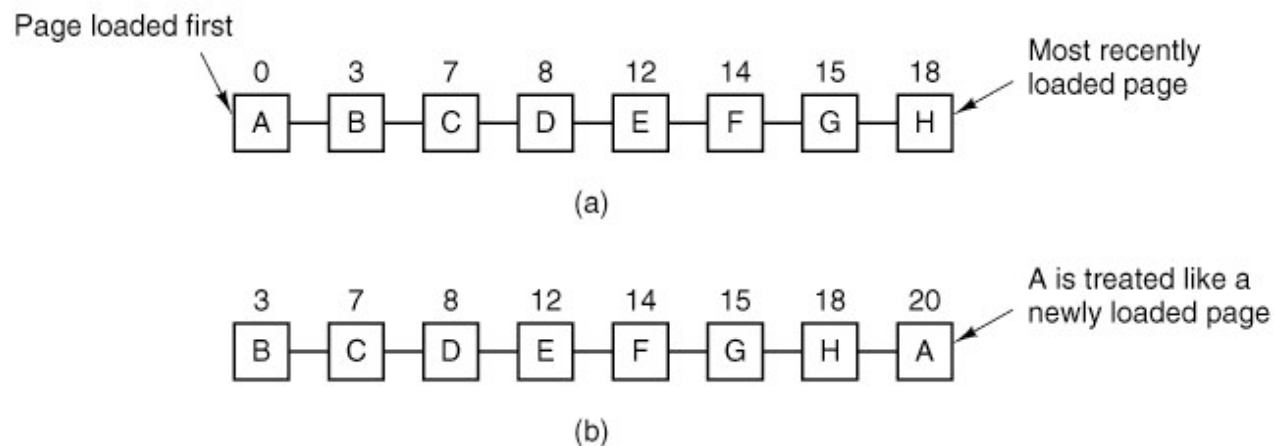
# Algoritma Second-Chance

- Algoritma *Second-Chance*
  - Modifikasi dari algoritma FIFO
    - Menghindari pergantian *Old page* yang direferensi
    - Mencari *old page* yang jarang direferensi
  - Menggunakan bit referensi (*reference bit*)
    - Nilai bit = 0, page diganti
    - Nilai bit = 1
      - Ubah *arrival time* = *current time*
      - Ubah nilai bit = 0
  - Jika page selalu direferensi, maka page tak pernah dihapus



# Algoritma Second-Chance

- Page yang terpilih (*victim page*), ditimpa dengan page baru pada posisi tersebut
- Jika reference bit semua adalah 0, algoritma second chance menyerupai FIFO

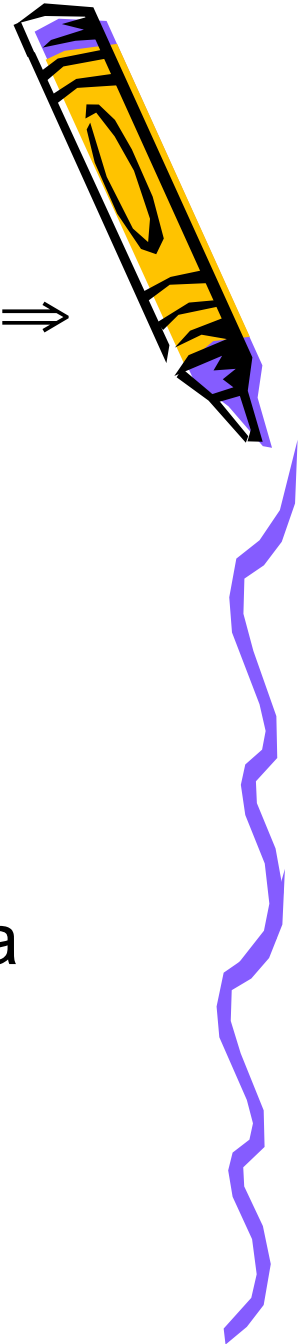


a. Deretan page dalam bentuk FIFO

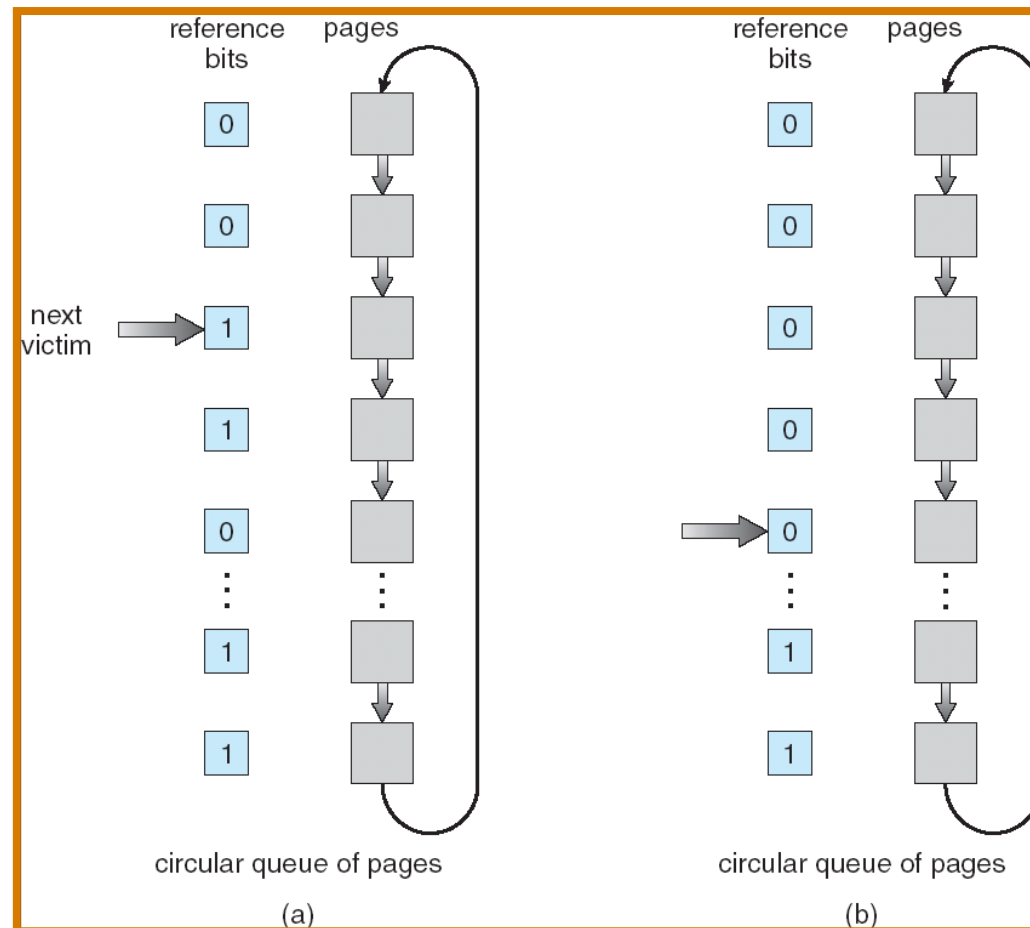
b. Page A dipindah dari posisi *arrival time* (0) ke posisi *current time* (20)

## • *Circular Queue (Algoritma Clock)*

- Model lain dari Algoritma *Second-Chance*  $\Rightarrow$  Circular Queue (Algoritma *clock*)
  - Page dalam bentuk lingkaran
  - Jika :
    - Nilai bit = 0, ganti page
    - Nilai bit = 1
      - Ubah nilai bit=0
      - Pointer bergerak ke page berikutnya searah jarum jam

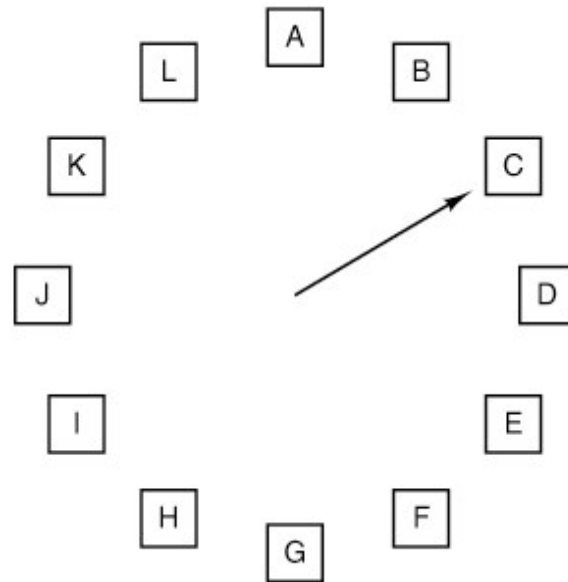


# • Circular Queue (Algoritma Clock)





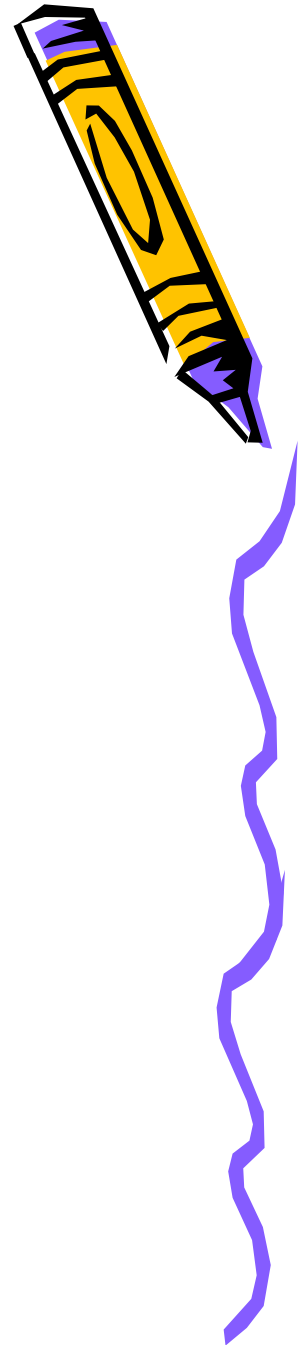
# Circular Queue (Algoritma Clock)



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

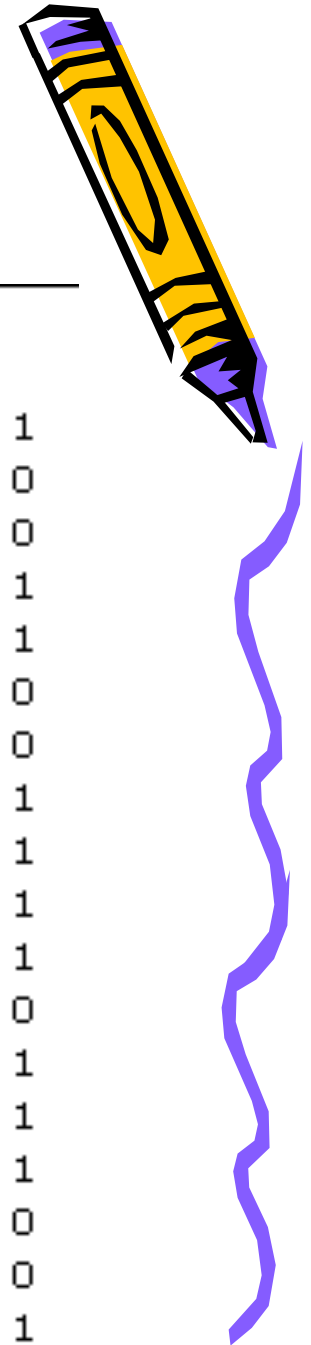
- R = 0: Evict the page
- R = 1: Clear R and advance hand

Algoritma Clock



# Circular Queue (Algoritma Clock)

page:	7	faults:	1	frames:	7	-1	-1	ref:	1
	0		2		7	0	-1		1 1
	1		3		7	0	1		1 1 1
	2		4		2	0	1		1 0 0
	0		4		2	0	1		1 1 0
	3		5		2	0	3		1 0 1
	0		5		2	0	3		1 1 1
	4		6		4	0	3		1 0 0
	2		7		4	2	3		1 1 0
	3		7		4	2	3		1 1 1
	0		8		4	2	0		0 0 1
	3		9		3	2	0		1 0 1
	2		9		3	2	0		1 1 1
	1		10		3	1	0		0 1 0
	2		11		3	1	2		0 1 1
	0		12		0	1	2		1 1 1
	1		12		0	1	2		1 1 1
	7		13		0	7	2		0 1 0
	0		13		0	7	2		1 1 0
	1		14		0	7	1		1 1 1

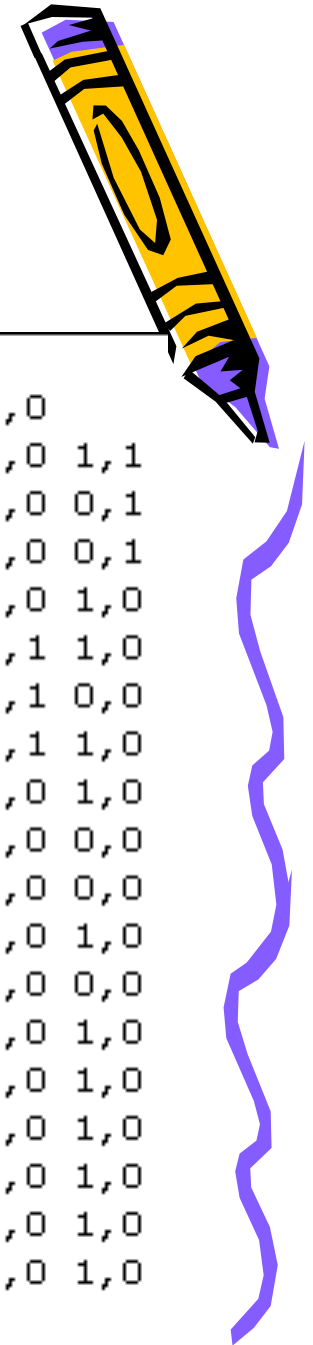


# Algoritma Enhanced Second-Chance

- Referenced (R) = Read / Write
- Modify (M) = Write
- (R,M) = (Referenced bit, Modify bit)
  - Kelas 0 (0,0) : tidak digunakan dan tidak dimodifikasi
  - Kelas 1 (0,1) : tidak digunakan namun dimodifikasi
  - Kelas 2 (1,0) : digunakan dan tidak dimodifikasi
  - Kelas 3 (1,1) : digunakan dan dimodifikasi
- Prioritas pergantian page, mulai dari kelas yang paling rendah



# Algoritma Enhanced Second- Chance (Cont.)



rpage:	7	faults:	1	frames:	7	-1	-1	ref,dirty:	1,0
	0		2		7	0	-1		1,0 1,0
wpage:	1		3		7	0	1		1,0 1,0 1,1
	2		4		2	0	1		1,0 0,0 0,1
	0		4		2	0	1		1,0 1,0 0,1
	3		5		2	0	3		1,0 0,0 1,0
wpage:	0		5		2	0	3		1,0 1,1 1,0
	4		6		4	0	3		1,0 0,1 0,0
	2		7		4	0	2		1,0 0,1 1,0
	3		8		4	3	2		0,0 1,0 1,0
	0		9		0	3	2		1,0 1,0 0,0
	3		9		0	3	2		1,0 1,0 0,0
	2		9		0	3	2		1,0 1,0 1,0
	1		10		0	1	2		0,0 1,0 0,0
	2		10		0	1	2		0,0 1,0 1,0
	0		10		0	1	2		1,0 1,0 1,0
	1		10		0	1	2		1,0 1,0 1,0
	7		11		0	1	7		0,0 0,0 1,0
	0		11		0	1	7		1,0 0,0 1,0
	1		11		0	1	7		1,0 1,0 1,0



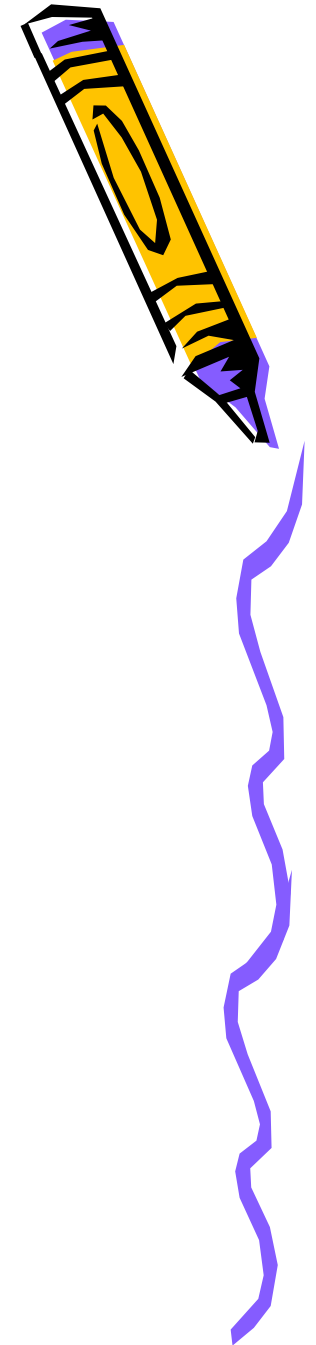
# Algoritma Counting-Based

- Rekam counter dari jumlah referensi masing-masing page
- **Algoritma LFU (Least Frequently Used):** ganti page yang mempunyai jumlah referensi paling kecil
- **Algoritma MFU (Most Frequently Used):** ganti page yang mempunyai jumlah referensi paling besar.



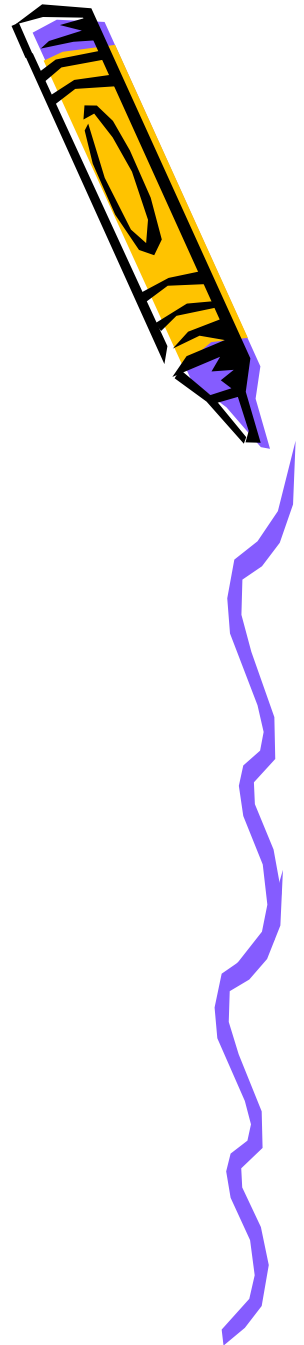
# LFU

page:	7	faults:	1	frames:	7	-1	-1	count:	1		
	0		2		7	0	-1		1	1	
	1		3		7	0	1		1	1	1
	2		4		2	0	1		1	1	1
	0		4		2	0	1		1	2	1
	3		5		2	0	3		1	2	1
	0		5		2	0	3		1	3	1
	4		6		4	0	3		1	3	1
	2		7		4	0	2		1	3	1
	3		8		3	0	2		1	3	1
	0		8		3	0	2		1	4	1
	3		8		3	0	2		2	4	1
	2		8		3	0	2		2	4	2
	1		9		3	0	1		2	4	1
	2		10		3	0	2		2	4	1
	0		10		3	0	2		2	5	1
	1		11		3	0	1		2	5	1
	7		12		3	0	7		2	5	1
	0		12		3	0	7		2	6	1
	1		13		3	0	1		2	6	1



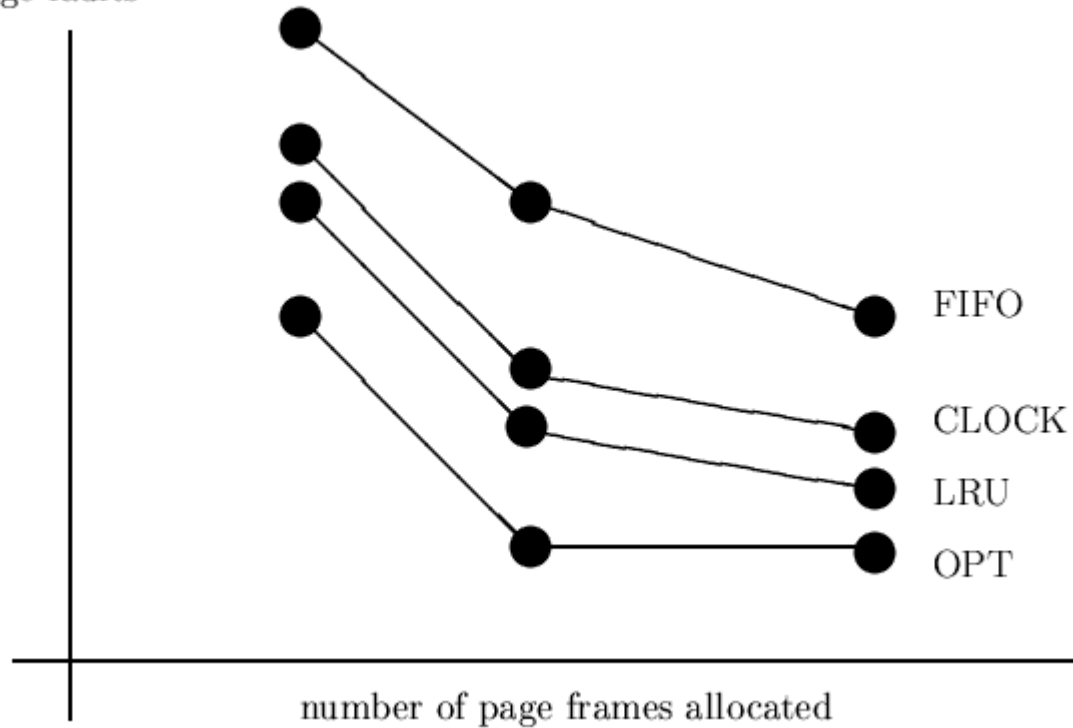
# MFU

page:	7	faults:	1	frames:	7	-1	-1	count:	1			
	0		2		7	0	-1		1	1		
	1		3		7	0	1		1	1	1	
	2		4		2	0	1		1	1	1	
	0		4		2	0	1		1	2	1	
	3		5		2	3	1		1	1	1	
	0		6		2	3	0		1	1	1	
	4		7		4	3	0		1	1	1	
	2		8		4	2	0		1	1	1	
	3		9		4	2	3		1	1	1	
	0		10		0	2	3		1	1	1	
	3		10		0	2	3		1	1	2	
	2		10		0	2	3		1	2	2	
	1		11		0	1	3		1	1	2	
	2		12		0	1	2		1	1	1	
	0		12		0	1	2		2	1	1	
	1		12		0	1	2		2	2	1	
	7		13		7	1	2		1	2	1	
	0		14		7	0	2		1	1	1	
	1		15		7	0	1		1	1	1	

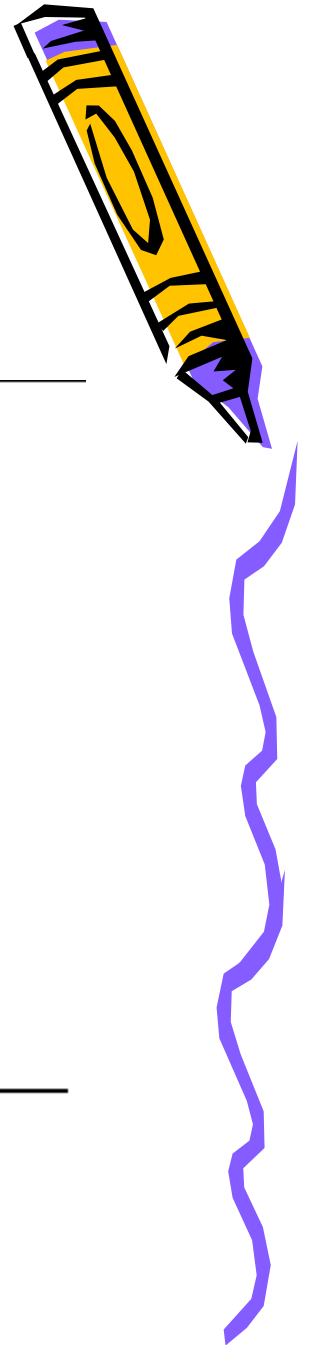


# Kinerja Algoritma Pergantian Page

number of page faults



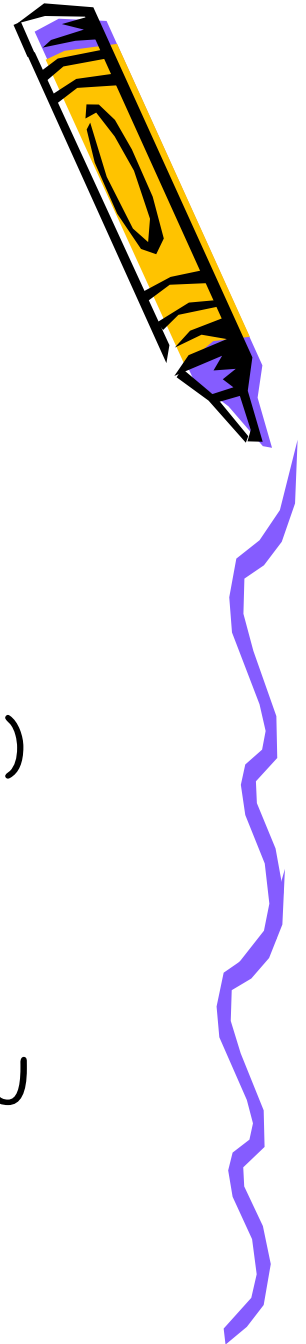
number of page frames allocated



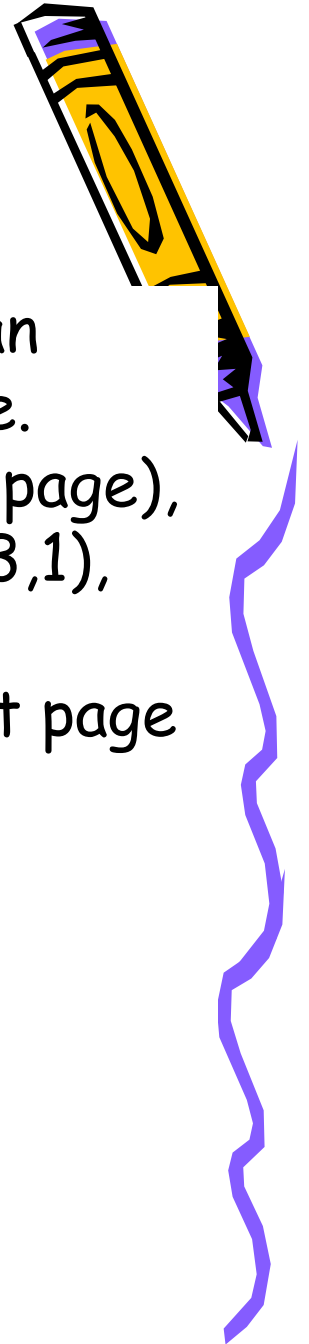


# Pertanyaan

- Bagaimana alur proses page replacement?
- Bagaimana algoritma FIFO bekerja?
- Bagaimana algoritma Optimal bekerja?
- Bagaimana algoritma Second Chance bekerja?
- Bagaimana algoritma LRU (Least Recently Used) bekerja?
- Bagaimana algoritma enhanced second chance bekerja?
- Bagaimana algoritma counting based LFU & MFU bekerja?



# latihan



- Sebuah proses mempunyai 3 frame menggunakan algoritma LRU untuk melakukan pergantian page. Antrian page mempunyai format (waktu, nomor page), secara berturut-turut adalah (0,4),(1,7),(2,4),(3,1),(4,7),(5,2),(6,9),(7,1),(8,7),(9,9),(10,4). Secara berturut-turut, bagaimanakah posisi frame saat page fault ke 6 dan 7?

A. 2,7,1 dan 2,7,9

B. 2,1,9 dan 7,1,9

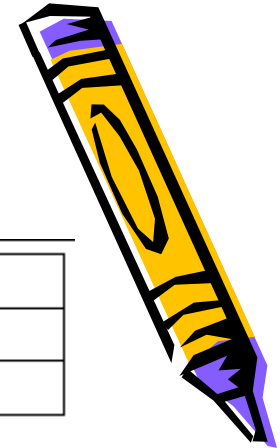
C. 2,7,9 dan 2,1,9

D. 7,1,9 dan 7,4,9

E. 4,7,1 dan 2,7,1



# Latihan



Frame	0	1	2	3	4	5	6	7
Page	17	32	41	5	7	13	2	20
Reference Bit	1	0	0	0	0	1	1	0

- Berdasarkan tabel diatas, diketahui frame 0 berisikan page 17, frame 1 berisikan page 32, dan seterusnya. Diasumsikan sistem menggunakan algoritma clock untuk melakukan pergantian page. Pada baris ketiga terdapat reference bit yang digunakan sistem saat ini. Reference bit diset 1 jika page dipindahkan ke memori.
- Saat ini posisi pointer berada pada frame ke 3. Frame ke 4 belum dieksekusi sama sekali. Jika terjadi permintaan page dengan referensi string sebagai berikut : 32, 14, 15, 2, 18, Jawablah pertanyaan berikut:

Isilah kolom berikut:

Frame	0	1	2	3	4	5	6	7
Page								
Reference Bit								



Berapa jumlah page yang diganti ?

Page mana saja yang diganti ?

Pada frame berapa, posisi terakhir dari pointer clock ?