conditions :

- mutual exclusion
- Hold and wait
- No Premsation
- circular wait

Strategies for handling

- dead lock

deadlock avoidance – banker algorithm

Banker's algorithm :

consider the following ex of system. check whether the system is safe / not using bankers algorithm. determine the seq of Procus if the system is safe

alloc - max

(work) (max - alloc)

Procus	Alloc	max	Available	Need
	A B C	A B C	A B C	A B C
			2 3 0	
P_0	0 1 0	7 5 3	3 3 2	7 4 3
	3 0 2			
P_1	2 0 0	3 2 2	5 3 2	1 2 2 0 2 0
		9 0 2	7 4 3	6 0 0
P_2	3 0 2			
		2 2 2	7 4 5	0 1 1
P_3	2 1 1			4 3 1
		4 3 3	7 5 5	
	0 0 2			
P_4			10 5 7	

Note

- work and finish are two vectors
- fin max - how much each process can request for each resource

$-] = [2\ 3\ 0]$

$0\ 2] = [3\ 0\ 2]$

$1\ 0\ 2] = [0\ 2\ 0]$

- allocation -

- Available - NO of each resources currently available

- Need - NO of remaining resources needed.

Step 1 - minusing alloc - max

Step 2 - Banker's algorithm

Need ≤ work ⇒ (work = work + alloc)

Soln:

[7 4 3] ≤ [3 3 2] → the condition fails. could couldn't assign p0

[1 2 2] ≤ [3 3 2] → the condition is true p1 is assigned.

New work = [3 3 2] + [2 0 0]

= [5 3 2]

[6 0 0] ≤ [5 3 2] → condition is false p2 is not assigned.

[0 1 1] ≤ [5 3 2] → condition is true p3 is assigned.

NW = [5 3 2] + [2 1 1]

= [7 4 3]

[4 3 1] ≤ [7 4 3] → condition is true p4 is assigned

NW = [7 4 3] + [0 0 2]

= [7 4 5]

$[143] \leq [745]$ → condition is true P_0 is assigned

$NW = [745] + [010]$

$= [755]$

$[600] \leq [755]$ → condition is true P_2 is assigned

$NW = [755] + [302]$

$= [1057]$

Seq of process ⇒ P_1, P_3, P_4, P_0, P_2

ii) what will happen if pro P_1 req one additional instance of resource type A & 2 instance of resource type C

$\therefore$ request 1 $= \begin{matrix} A & B & C \\ [1 & 0 & 2] \end{matrix}$ (resource

Step 1 → $req_i \leq Need_i$

$[102] \leq [122]$

Step 2 → $req_i \leq available$

$[102] \leq [332]$

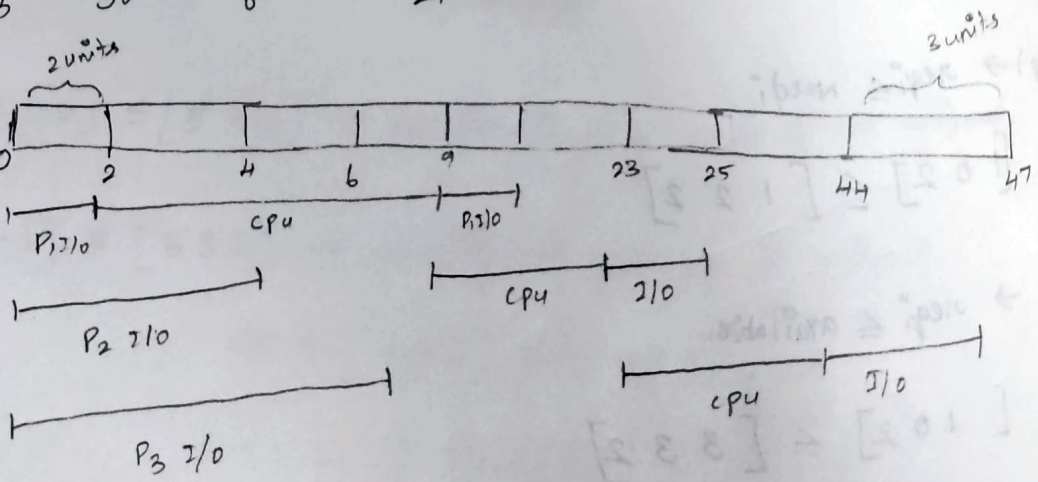
Step 3 → $avai = avai - req_i = [332] - [102] = [230]$

$allo = all_i + req_i = [200] + [102] = [302]$

$Need_i = Need_i - req_i = [122] - [102] = [020]$

Consider 3 processes all occuring at time 0 with total execution time of 10, 20, 30 units respectively. each process spends the 1st 20% of execution time doing i/p and o/p. in next 70% in doing computation and last 10% in i/p and o/p again. the OS uses shortest remaining compute time first scheduling algorithm and schedules a new process either when the running process gets blocked on inp and outp or when the running process finishes its compute burst. assume that all i/p o/p operations can be overlap as much as possible.. For what % does the cpu remains idle.

Pid	BT	I/O	Cpu exec	I/O
P_1	10	2	7	1
P_2	20	4	14	2
P_3	30	6	21	3



5 units is idle

$$\frac{5}{47} \times 100 = 10.638$$

$$= 10.64$$

Unit-3

Memory management

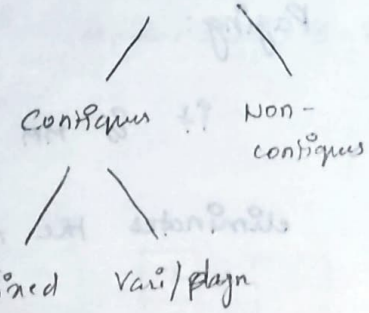
the prouss is stored in the secondary memory and when the prouss starts it fetch the data from secondary memory to main memory. the proussing starts that prouss is called swaping in and after the prouss finishes it will leave the os is called swapping out.

- To keep track of used memory space by processes.

contiques and non-contiques → if not prousued one after other then it is non-con

if one after the other

Memory manag-



In contiques
- fixed partitioning
- variable partitioning / dynamic

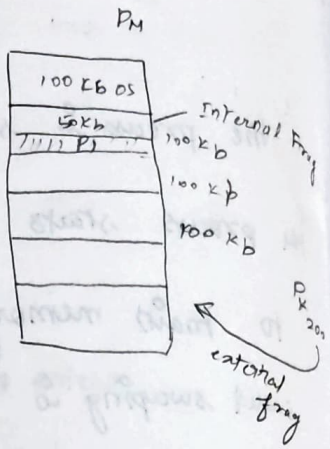
In fixed part the os will fix the size

If the memory

Internal fragmentation & external fragmentation :-

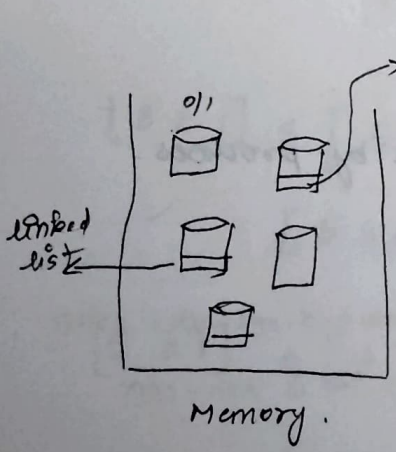
dynamic partitioning :

external fragmentation ✗



- bit map / bit vector
- linked list

} techniques to handle free space in OS



if it is 1 then it is occupied

if it is 0 then it is free

Paging :

Page is Stored in the MM

it is MM technique which follows fixed part teams and eliminates the need of contiques allocation.

Page no and page offset (address)

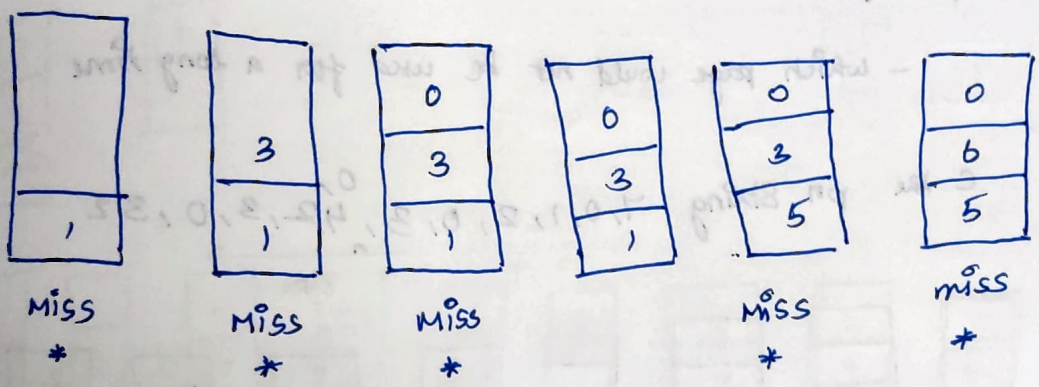
if the page looking for the page is main memory and if it not has free page page fault occurs.

Page replacement algorithms:

- FIFO

main me

Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames. Find the number of page faults and page fault ratio

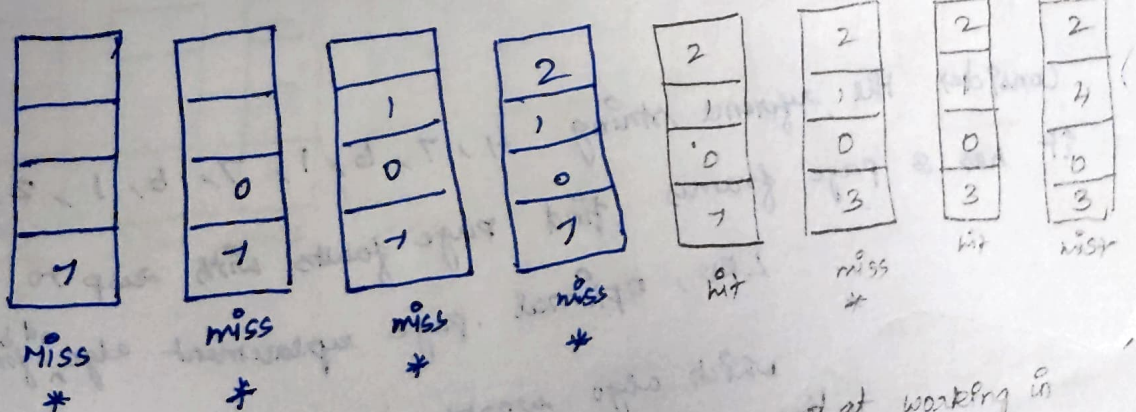


Page fault = 5

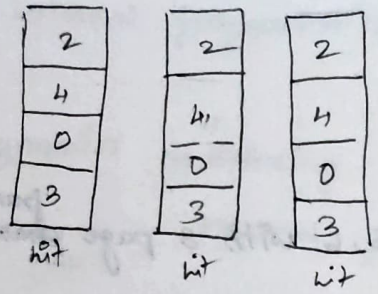
Page fault ratio = 5/6

- LRU (Least recently used)

Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames find the no of page fault



id at working in

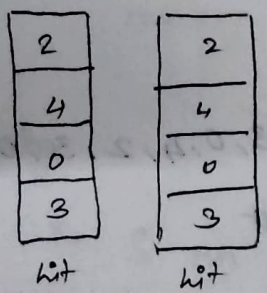


Page fault = 6

- Optimal

- which page could not be used for a long time

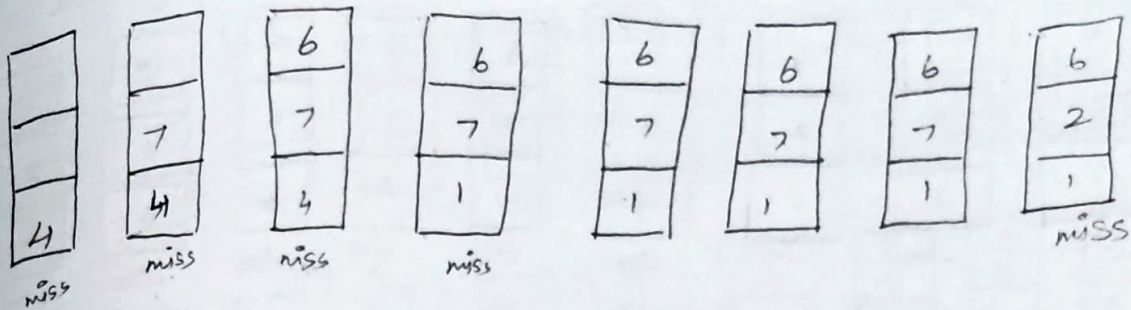
C the pn string 7, 0, 1, 2, 0, 3, 4, 2, 3, 0, 3 2

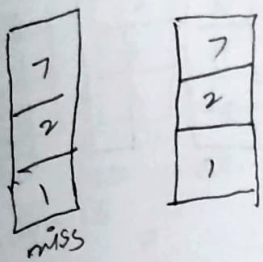


hit

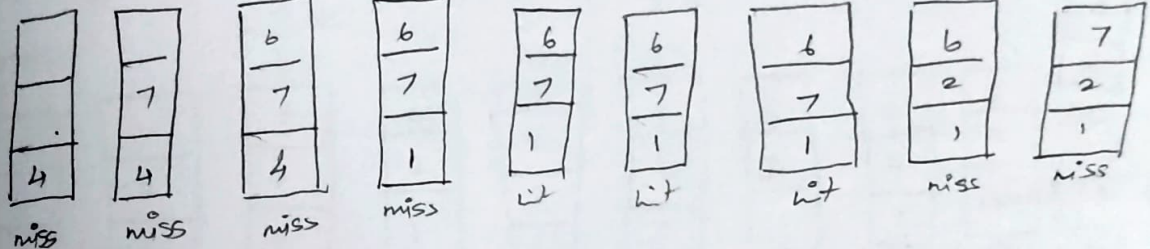
Page fault = 6

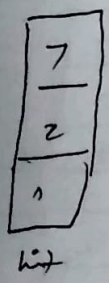
Consider the reference string 4, 7, 6, 1, 7, 6, 1, 2, 7, 2 it has 3 page frames find page faults with resp to FIFO, LRU, optimal. Page replacement algo and find out which algo works good.



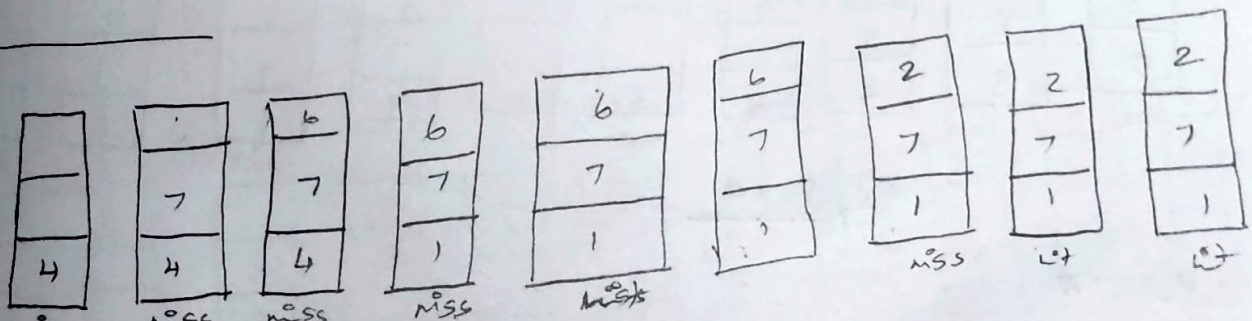


Page f = 6





Page f = 6

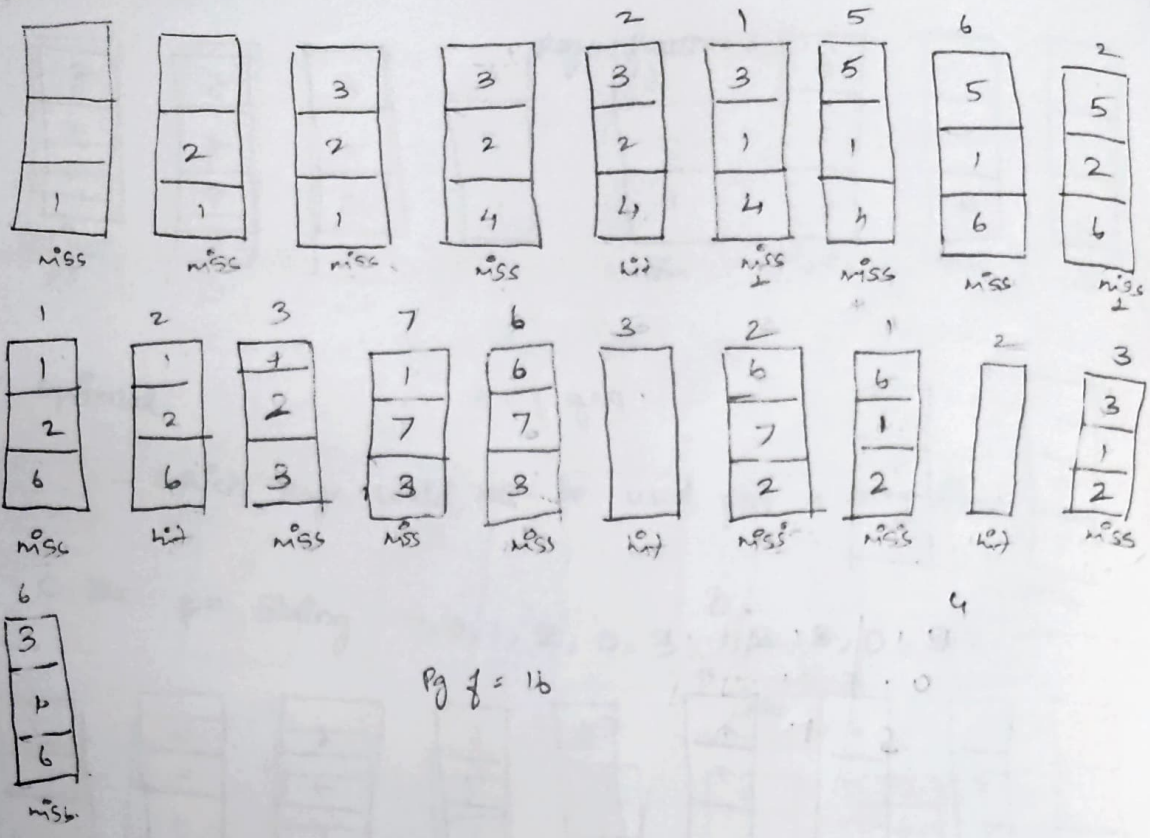


Page f = 5

∴ the optimal page replacement was good at working in this scenario.

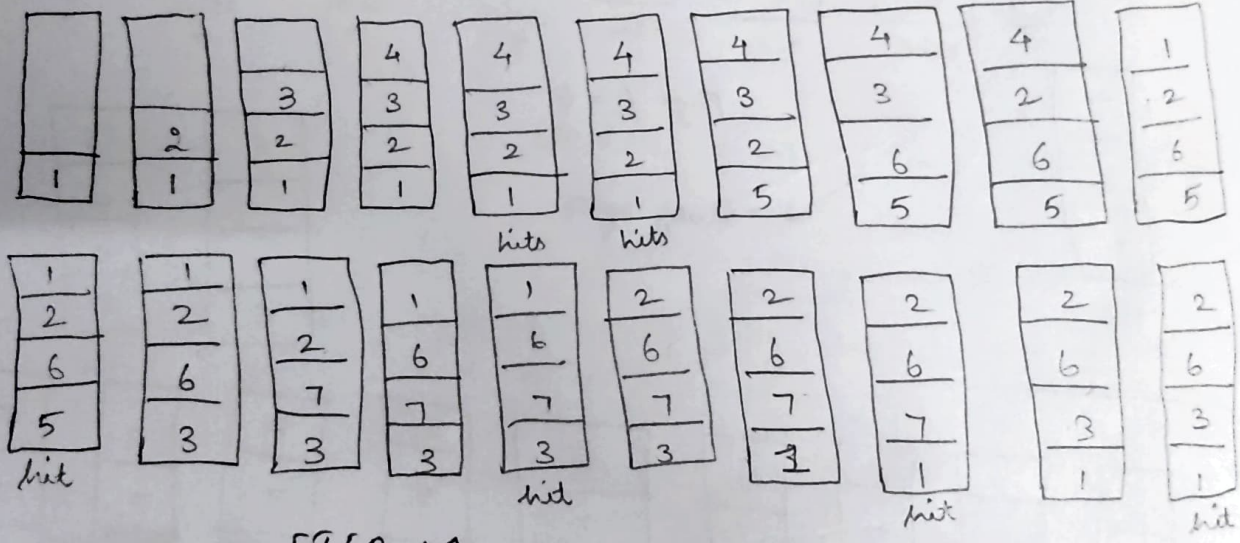
yes to

FIFO



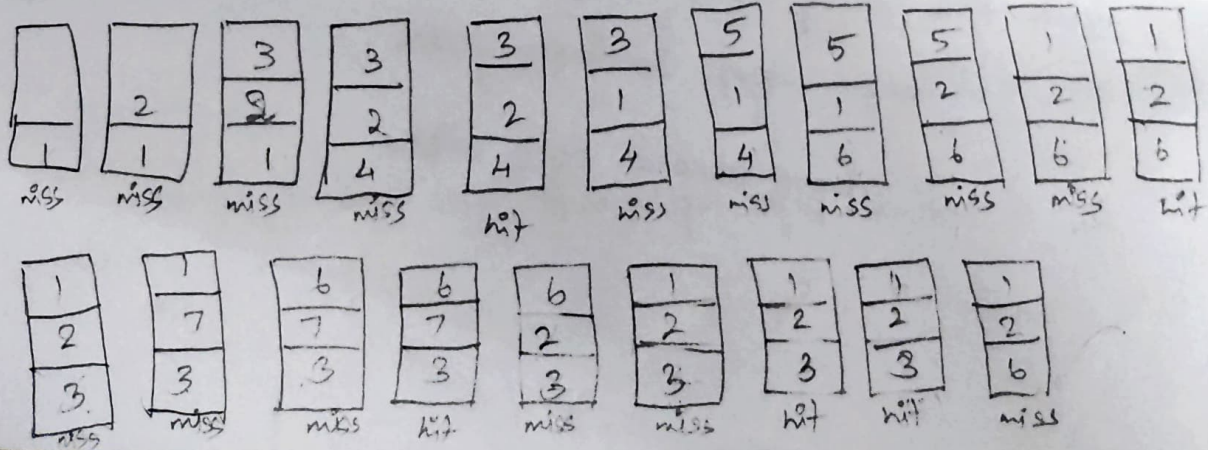
Pg f = 16

using 4 frames

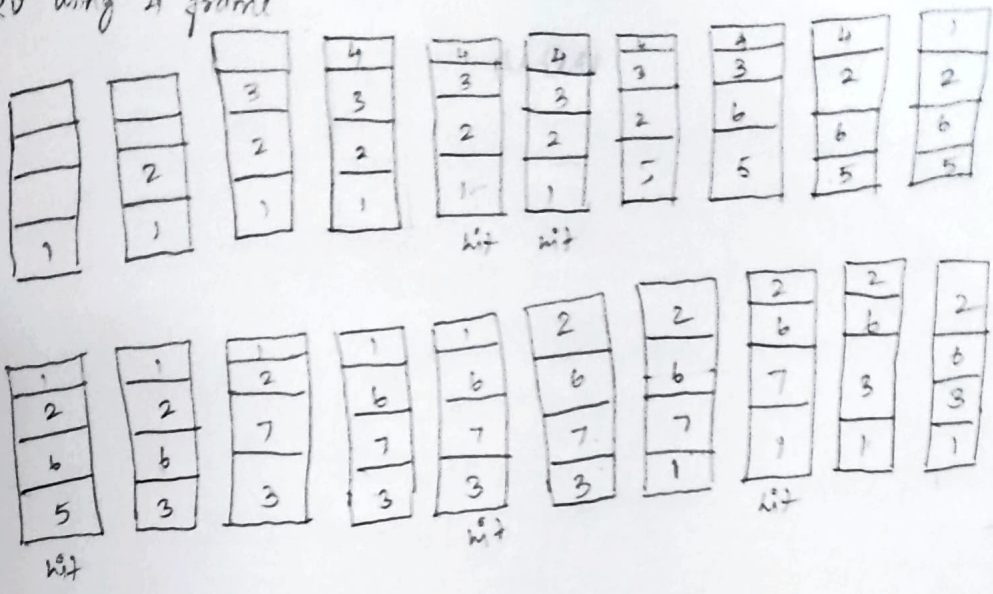


FIFO = 14

LRU = 15



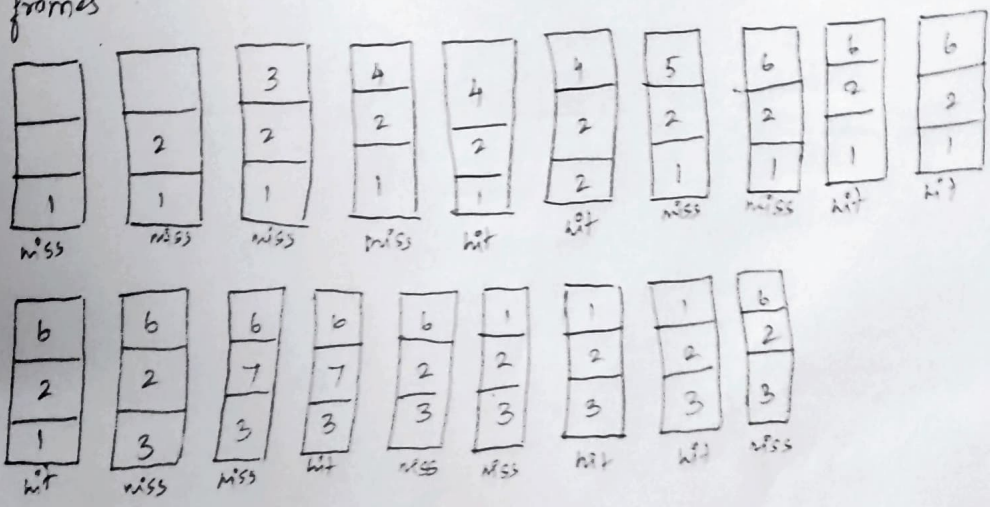
LRU using 4 frame



Pf = 15

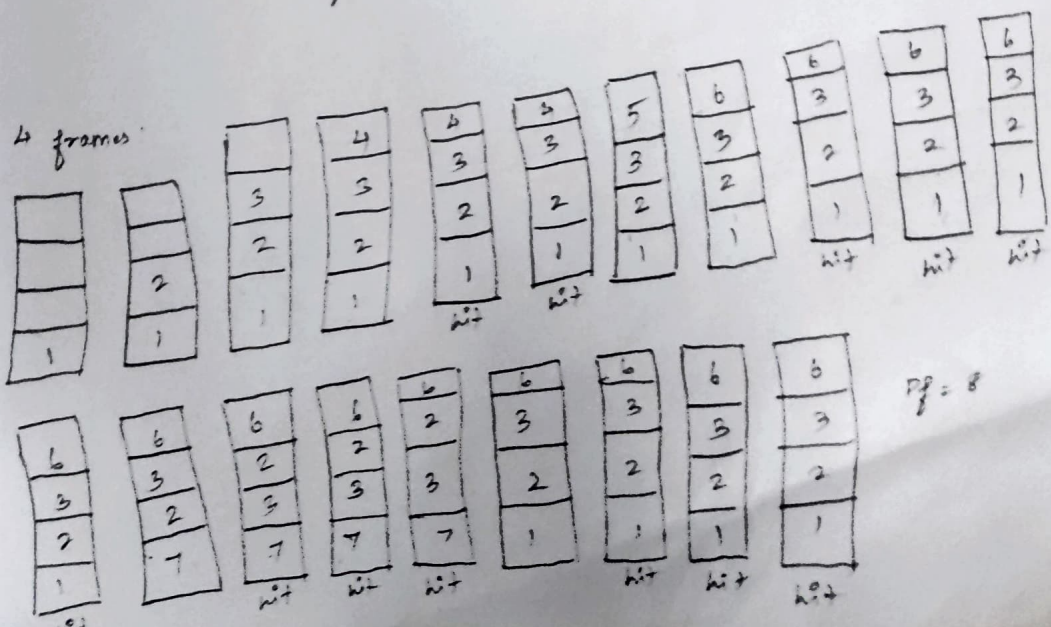
Optimal page replacement

3 frames



Pf = 11

4 frames



Pf = 8