8.3

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
T_0	0	0	1	2	0	0	1	2	1	5	2	0
T_1	1	0	0	0	1	7	5	0				
T_2	1	3	5	4	2	3	5	6				
T_3	0	6	3	2	0	6	5	2				
T_4	0	0	1	4	0	6	5	6				

a:) What is the content of the matrix Need 2 0

Need Matrix (Max - Allocation)

	A	B	C	D
T_0	0	0	0	0
T_1	0	7	5	0
T_2	1	0	0	2
T_3	0	0	2	0
T_4	0	6	4	2

$< T_0, T_2, T_3, T_4, T_1 >$

(b) Is the system in a safe state 2 0 → Yes

① Need $[T_0] \leq$ Available then,

Ava = [0 0 1 2] + [1 5 2 0]

= 1 5 3 2

Finish $[T_0]$ = true.

② Need $[T_1] \leq$ Ava [0 7 5 0] ≤ 1 5 3 2 ✗

③ Need $[T_2] \leq$ Ava [1 0 0 2] ≤ 1 5 3 2 ✓

Ava = 1 3 5 4 + 1 5 3 2 = 2 8 8 6

~~1 0 0 2~~ ~~2 5 3~~

Finish (T_2) = True

(4) Need $[T_3] \leq Ava \Rightarrow 0020 \leq 2886$

$Ava = 2886 - 0632 = 2\ 14\ 11\ 8$

$Ava = [2\ 14\ 11\ 8]$

(5) Need $[T_4] \leq Ava \Rightarrow 0642 \leq 2\ 14\ 11\ 8$

$2\ 14\ 11\ 8 - 0\ 0\ 1\ 4 = 2\ 14\ 12\ 12$

$Ava = [2\ 14\ 12\ 12]$

(6) Need $[T_1] \leq Ava \Rightarrow [0\ 7\ 5\ 0] \leq [2\ 14\ 12\ 12]$

$2\ 14\ 12\ 12 + 1\ 0\ 0\ 0 = 3\ 14\ 12\ 12$

(C) If a request from Thread T_1 arrives for (0,4,2,0) can the request be grant immediately ?

Sol:- Yes, As $(0,4,2,0) \leq (1,5,2,0)$

Request Available

8.9

	Allocation	Need	Max	Avai
	A B C D	A B C D	A B C D	0 3 0 1
T_0	3 0 1 4	2 1 0 3	5 1 1 7	
T_1	2 2 1 0	1 0 0 1	3 2 1 1	
T_2	3 1 2 1	0 2 0 0	3 3 2 1	
T_3	0 5 1 0	4 1 0 2	4 6 1 2	
T_4	4 2 1 2	2 1 1 3	6 3 2 5	

finish [T_0] = false
finish [T_1] = false
⋮
finish [T_4] = false

$< T_2, T_1, T_3,$

Need	Available	Checking
2 1 0 3	0 3 0 1	Need ≤ Ava ✗
1 0 0 1	0 3 0 1	Need ≤ Ava ✗
0 2 0 0	0 3 0 1 3 1 2 1 3 4 2 2	Need ≤ Ava ✓ Finish [T_2] = True
4 1 0 2	3 4 2 2	Need ≤ Ava ✗
2 1 1 3	3 4 2 2	Need ≤ Ava ✗
2 1 0 3	3 4 2 2	Need ≤ Ava ✗
1 0 0 1	3 4 2 2 2 2 1 0 5 6 3 2	✓ Finish [T_1] = True
4 1 0 2	5 6 3 2 0 5 1 0 5 11 4 2	✓ finish [T_3] = True
2 1 1 3	5 11 4 2	✗
2 1 0 3	5 11 4 2	✗

Not a safe sequence
So system enters into dead locks.

(3)

	Allocation	Max	Need	Available
T_0	3 1 4 1	6 4 7 3	3 3 3 2	2 2 2 4
T_1	2 1 0 2	4 2 3 2	2 1 3 0	1
T_2	2 4 1 3	2 5 3 3	0 1 2 0	
T_3	4 1 1 0	6 3 3 2	2 2 2 2	
T_4	2 2 2 1	5 6 7 5	3 4 5 4	

Need	Available	checking
3 3 3 2	2 2 2 4	Need ≤ Ava X
2 1 3 0	2 2 2 4	Need ≤ Ava X
0 1 2 0	2 2 2 4 + 2 4 1 3	Need ≤ Ava ✓ finish [T_2] = True
2 2 2 2	4 6 3 7 4 1 1 0	Need ≤ Ava ✓ finish [T_3] = True
3 4 5 4	8 7 4 7 2 2 2 1	Need ≤ Ava ✓ finish [T_4] = True
3 3 3 2	10 9 6 8 3 1 4 1	Need ≤ Ava ✓ finish [T_0] = True
2 1 3 0	13 10 10 9 2 1 0 2	Need ≤ Ava finish [T_1] = True
	15 11 10 11	

$< T_2, T_3, T_4, T_0, T_1 >$

The system is in safe state & the safe sequence is $< T_2, T_3, T_4, T_0, T_1 >$.

P_3 :- Need 7 3 3

Available
10 4 7
0 2 0
10 6 7

Finish[P_3] = true &

Available =

<P_1 P_4 P_0 P_2 P_3> Safe Sequence

Time Complexity :- $n \times \left(\frac{n}{m}\right)$

n → No. of Iterations for each Iteration

(2)

	Allocation	Max	Need	Available
				3 3 2
T_0	0 1 0	7 5 3	7 4 3	
T_1	2 0 0	3 2 2	1 2 2	
T_2	3 0 2	9 0 2	6 0 0	
T_3	2 1 1	2 2 2	0 1 1	
T_4	0 0 2	4 3 3	4 3 1	

Step-1 :- finish [T_0] = false. Work = 3 3 2

finish [T_4] = false.

[7 4 3] ≤ 3 3 2 ✗

Step-2 :- finish [T_0] = false && ~~[0 1 0]~~

T_1 = false && [1 2 2] ≤ 3 3 2 ✓

So allocate to T_1

Step-3 :- finish[T_1] = true;

Work = 3 3 2 + 2 0 0

W = 5 3 2

<T_1, T_3>

T_2 = false && 6 0 0 ≤ 5 3 2 ✗

T_3 = false && 0 1 1 ≤ 5 3 2 ✓

finish(T_3) = True W = 5 3 2 + 2 1 1 = 7 4 3

W = 7 4 3

T_4 = false && 431 ≤ 743

So, T_4 = True

Work = 743
+ 002
= 745

After I^st Iteration,

T_1, T_3, T_4 = true T_0, T_2 = false

T_0 = false && 743 ≤ 745 ✓

So, T_0 = true

W = 745
+ 010
= 755

T_2 = false && 600 ≤ 755 ✓

So, T_2 = true

W = 755
+ 302
= 1057

After II^nd Iteration all the resources are became true

So, safety List is

<T_1, T_3, T_4, T_0, T_2>

Safety sequence

~~Request - R~~

Resource — Request Algorithm

Steps:-

① If $Request_i \leq Need_i$, go to step 2. Otherwise raise an error condition, since the thread has exceeded its maximum claim.

② If $Request_i \leq Available$, go to step 3. otherwise T_i must wait, since the resources are not available.

③ Have the system pretend to have allocated the requested resources to thread T_i by modifying the state as follows:-

$$Available = Available + Request_i$$
$$Allocation = Allocation_i + Request_i$$
$$Need = Need - Request_i$$

Ex:-

	Allocation	Request	Available
			0 0 0
T_0	0 1 0	0 0 0	
T_1	2 0 0	2 0 2	
T_2	3 0 3	0 0 0	
T_3	2 1 1	1 0 0	
T_4	0 0 2	0 0 2	

→ firstly take a process which has request as 0 0 0

so, firstly take T_2 then,

$$Ava = 0\ 0\ 0 + 3\ 0\ 3 = 3\ 0\ 3$$

then we go for T_4 where $0\ 0\ 2 \leq 3\ 0\ 3$

$$\begin{matrix} 3 & 0 & 3 \\ 0 & 0 & 2 \end{matrix} \rightarrow 3\ 0\ 5$$

$T_0 \Rightarrow$ ~~0 1~~ 0 0 0 $\leq$ 3 0 5

Ava = 3 0 5
0 1 0
3 1 5

$T_1 \Rightarrow$ 2 0 2 $\leq$ 3 1 5 ✓

Ava = 3 1 5
2 0 0
5 1 5

$T_3 \Rightarrow$ 1 0 0 $\leq$ 5 1 5

Ava = ~~0 0 2~~ 1 0 0
~~5 1 5~~ 5 1 5
~~5 1 7~~ 6 1 5

$< T_2, T_4, T_0, T_1, T_3 >$

(or)

$< T_0, T_2, T_3, T_4, T_1 >$.

Recovery from Deadlocks

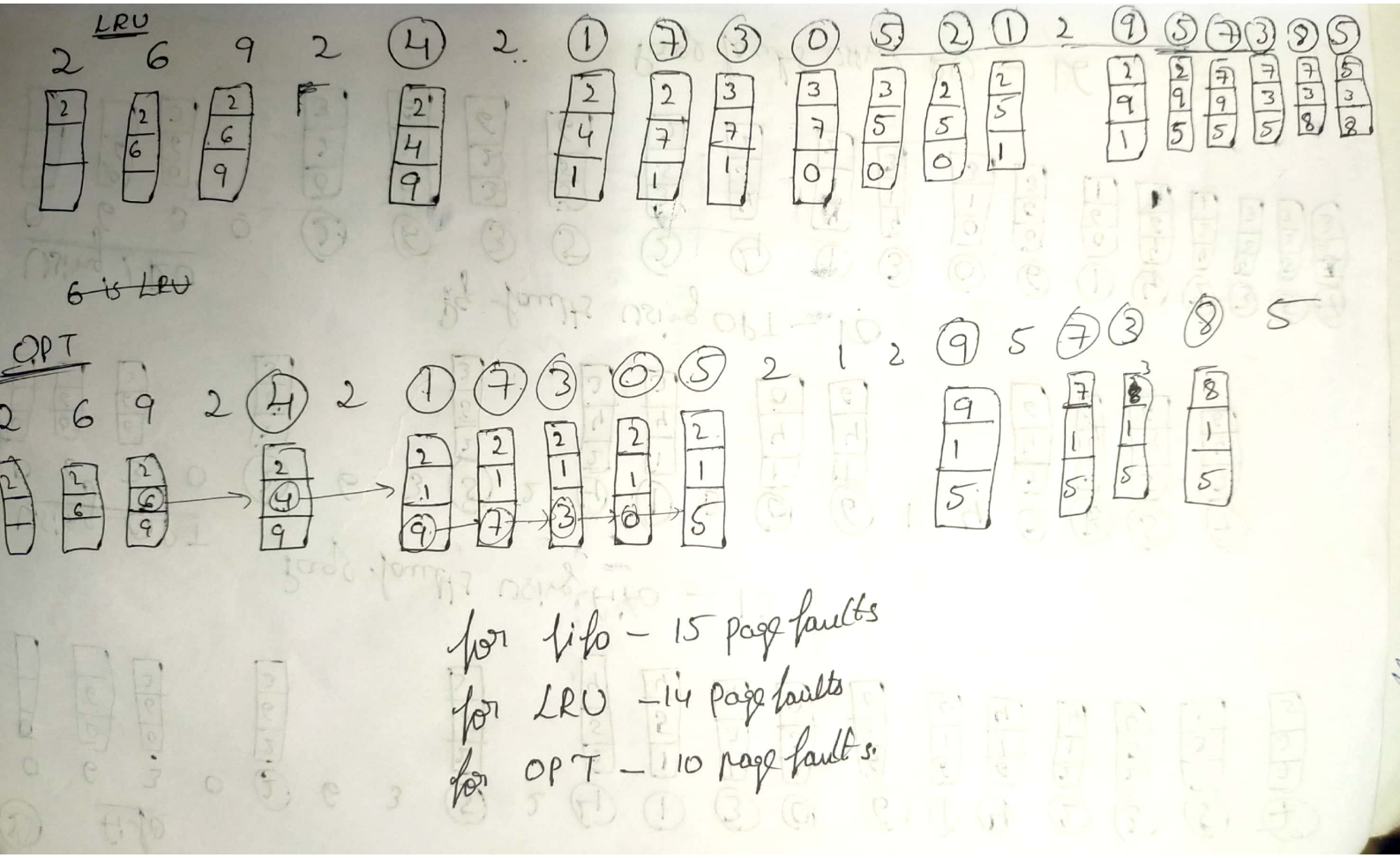
* Process & Thread Termination
  - (i) Abort all deadlocked processes. [Delete whole cycles]

FIFO, LRU, OPT Replacement Algorithms

(1).

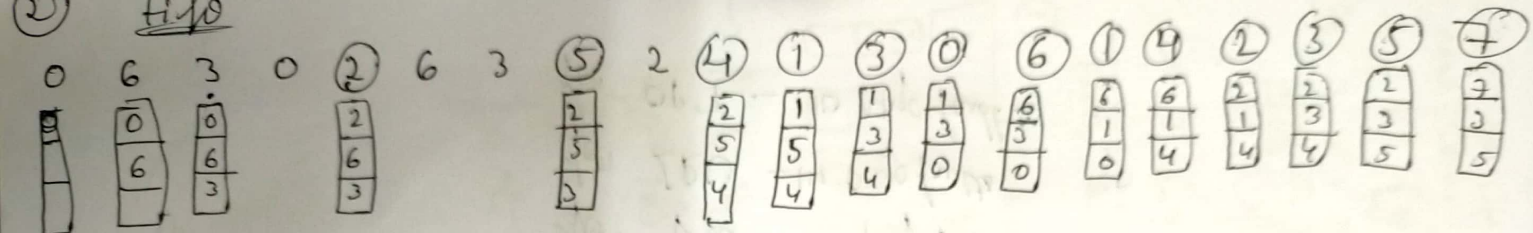
Reference string Using FIFO

2	6	9	2	4	2	1	7	3	0	5	2	1	2	9	5	7	3	8	5
2	2	2	7	4	4	4	7	7	7	5	5	5	5	5	9	9	3	3	3
	6	6	6	6	2	2	2	3	3	3	2	2	2	2	5	5	5	8	8
		9	9	9	9	1	1	1	0	0	0	1	1	1	1	7	7	7	5



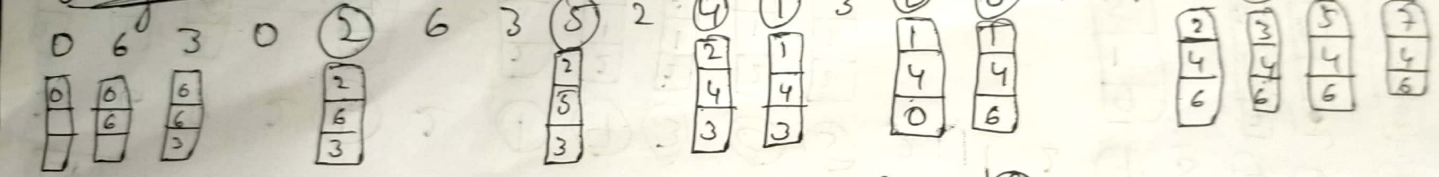
for fifo – 15 page faults
for LRU – 14 page faults
for OPT – 10 page faults.

(2) Fifo



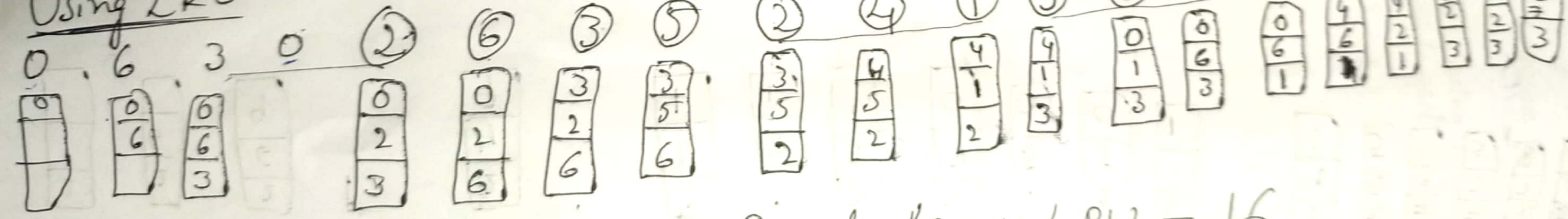
Page faults using fifo – 13

Using OPT



Pg faults using OPT – 10

Using LRU

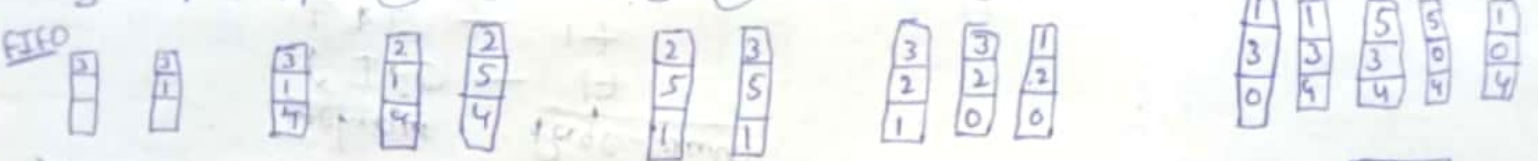


Page faults using LRU – 16

① Reference String

3 1 4 ② ⑤ 4 ① ③ 5 ② ⑥ ① 1 0 2 ③ ④ ⑤ ⑥ ①

FIFO

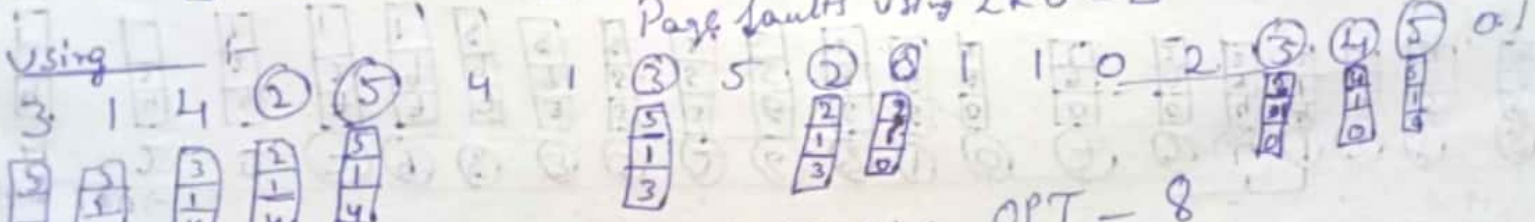


Page faults using fifo = 12

LRU

Page faults using LRU = 13

Using



Page faults using OPT - 8