OS

HOMEWORK WEEK 5

Q1. Given,

time to execute an instruction = 0.5 MS = 0.5×10^{-3} ms

In page fault,

process time to handle = 250 MS = 0.25 ms

disk time to read = 10 ms

= 10×10^{-3} s

a.) In 10^{-2} s → 1 page transfer

∴ 1 s → $\frac{1}{10^{-2}}$

ie 100 page transfer/s.

b.) Total disk transfer time = transfer time for non dirty page + transfer time for dirty page

= $\frac{2}{3} \times 10\text{ms} + \frac{1}{3} \times (10 \times 2)\text{ ms}$

here, 2 is for 2 disk transfers in dirty pages

(.25ms handle time can be omitted as it is very small in comparison to 10ms)

= $\frac{20}{3}\text{ ms} + \frac{20}{3}\text{ ms} = \frac{40}{3}\text{ ms} = 13.3\text{ ms}$

Now,

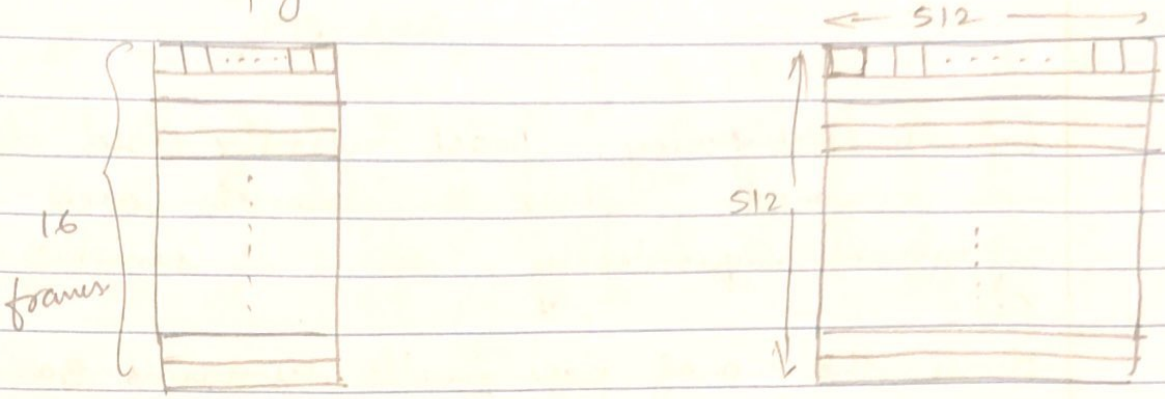
In 0.5×10^{-3} ms → 1 instruction

∴ 13.3 ms → $\frac{1}{0.5 \times 10^{-3}} \times 13.3$

= $\frac{1000 \times 13.3}{0.5}$

= 26600 instructions

Q.2: Given, 512 × 512 array of elements.
page size = 512 = frame size.



a) In FIFO, i.) In row major order. When 1st element referenced, page fault occurs and the complete row is placed on 1st frame. Rest 511 elements in that row are already present so, when they are refurenced, no page fault occurs.
Again when 1st element of 2nd row is referenced, page fault occurs (2 page faults till now) and 2nd row placed on 2nd frame. Rest 511 elements in that is already present so, no more page fault in elements in that row.
Similarly it happens for all the reest of the rows.
Resulting in total of 512 page faults.

ii.) In column major order - elements are referenced column wise. When 1st element refurenced the complete row (512 elements) is place in 1st frame. 1st page fault. When 2nd element column wise is referenced, it is not present in frame. So 2nd page fault occurs.
Similarly everytime each element is referenced sequencially column wise, page fault occurs.

Total no. of page fault = Total no. of elements
= 512 × 512
= 262,144

b.) In LRU policy, least recently used elements are removed. Since the elements are referenced sequentially, so it is similar to FIFO.
Hence, the no. of page faults remains same in LRU as there are in FIFO.

Q3. a.) a* b g a* d e a* b a* d e g d e

For OPTIMAL, the reference string is provided, so we know the future access of pages.

[a*] [b] [g] a* [d] [e] a* b a* d e [g] d e

a*	a*	a*	a*	a*	a*	a*	a*	a*	a*	a*	a*	a*	a*
	b	b	b	b	b	b	b	b	b	(b)	g	g	g
		g	g	(g)	e	e	e	e	e	e	e	e	e
				d	d	d	d	d	d	d	d	d	d
IDR		IDR	IDR		IDR	IDR						IDR	

No. of DISK reads = 5
Page faults = 5
Disk write = 0

{ □ means page fault
DR - Disk Read

b.) FIFO

a* b g a* d e a* b a* d e g d e

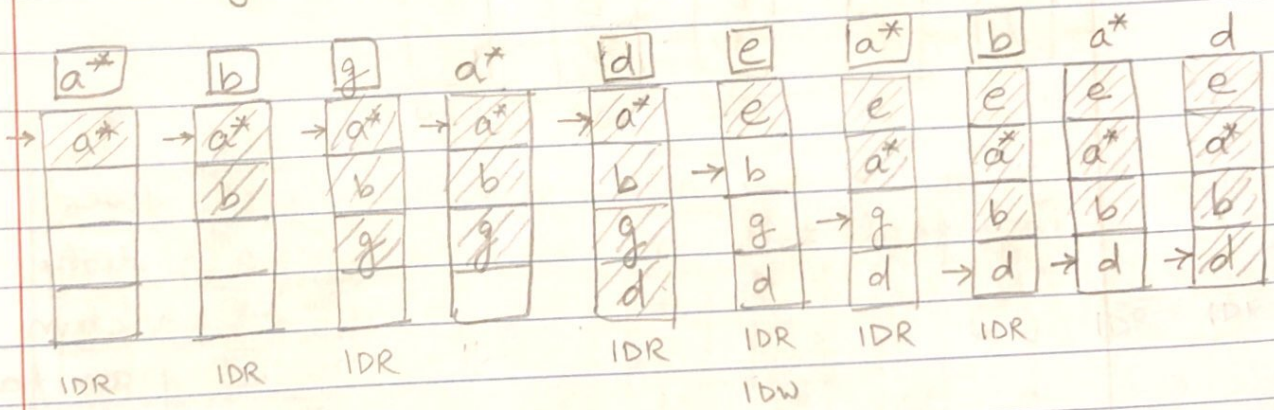
a*	a*	a*	a*	a*	e	e	e	e	e	e	e	d	d
	b	b	b	b	b	a*	a*	a*	a*	a*	a*	a*	e
		g	g	g	g	g	b	b	b	b	b	b	b
				d	d	d	d	d	d	d	g	g	g
IDR	IDR	IDR		IDR	IDR	IDR	IDR				IDR	IDR	IDR
					IDW								IDW

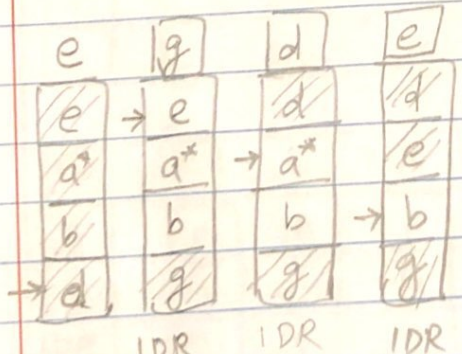
No. of page faults :- 10
Disk read = 16
Disk write = 2

□ - page fault
DR - Disk read
DW = Disk write
○ - victim

c.) Second Chance Algorithm

a* b g a* d e a* b a* d e g d e

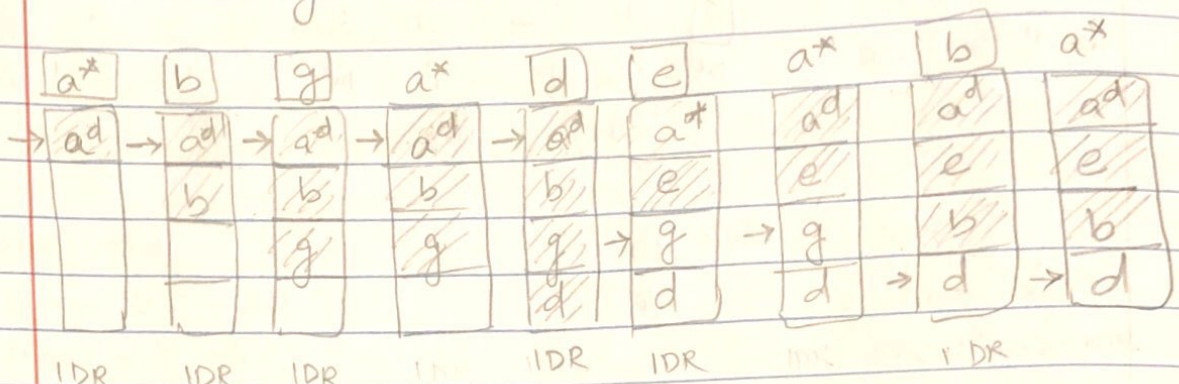


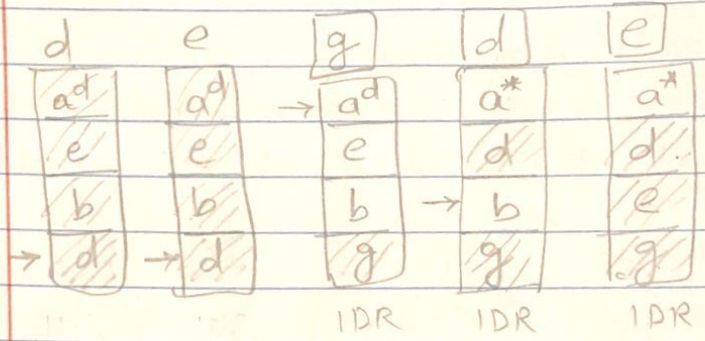


□ - page fault
→ - victim
▨ - used

Page fault = 10
Disk read = 10
Disk write = 2

d.) a^* b g a^* d e a^* b a^* d e g d e





Page fault = 9
DR = 9
DW = 0.

- ▨ used
- a^d dirty
- → victim
- □ page fault

Q4. b.) Optimal

A C B* D B* A E F B* F A G E F A

A	C	B*	D	B*	A	E	F	B*	F	A	G	E	F	A
A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
	C	C	C	C	C	E	E	E	E	E	E	E	E	E
		B*	B*	B*	B*	B*	B*	B*	B*	B*	B*	B*	B*	B*
			D	D	D	D	F	F	F	F	F	F	F	F

Page faults = 7

a.) LRU

A C B* D B* A E F B* F A G E F A

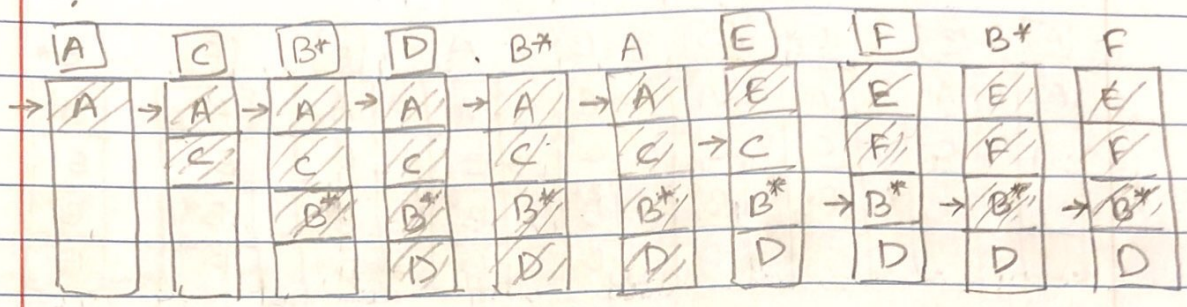
A	C	B*	D	B*	A	E	F	B*	F	A	G	E	F	A
A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
	C	C	C	C	C	E	E	E	E	E	E	E	E	E
		B*	B*	B*	B*	B*	B*	B*	B*	B*	B*	B*	B*	B*
			D	D	D	D	F	F	F	F	F	F	F	F

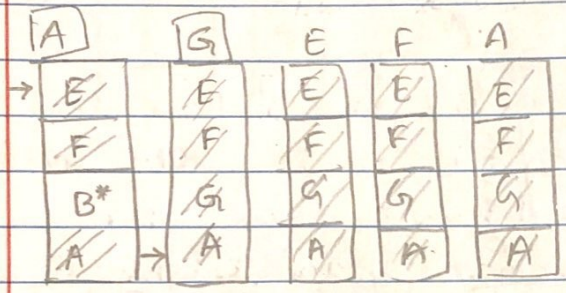
Page faults = 8

□ - page fault
→ - victim

c.) Second chance Algorithm

A C B* D B* A E F B* F A G E F A





D - page fault
▨ - used
→ - victim

Page fault = 8

Q5. Given,

length of page reference string = p

page number = n

No. of frames = m such that $n > m$

a.) Lower bound on no. of page fault = n

Since no. of frames is less than no. of pages referenced & all pages will atleast be accesed once, therefore, n no. of page fault will occur.

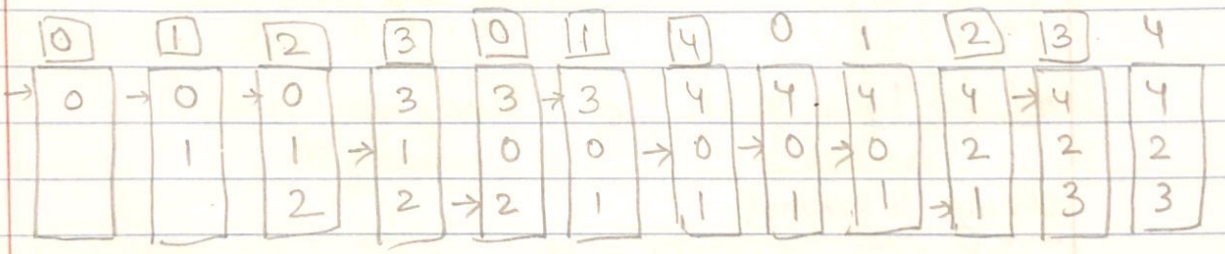
b.) Upper bound on no. of page fault = p.

Maximum no. of page fault will occur when each and every page accesed has page fault. i.e. the no. of referenced page string.

Q.6.) Belady's anomaly

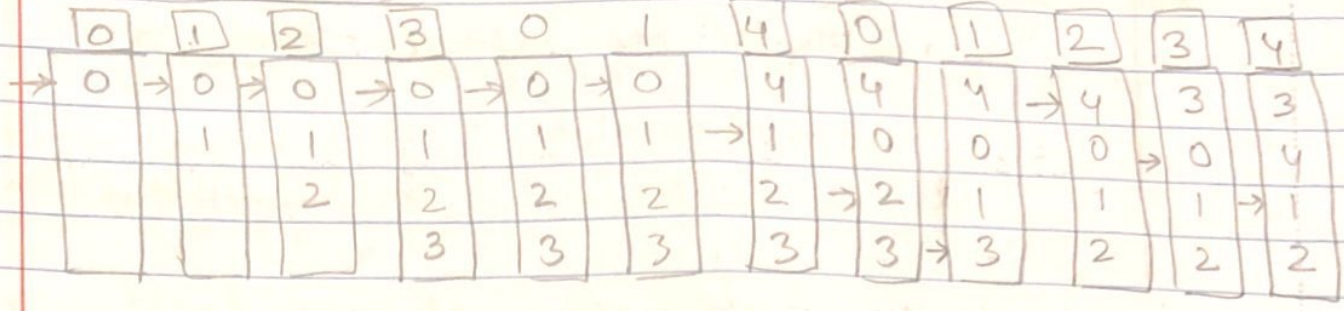
String - 0 1 2 3 0 1 4 0 1 2 3 4

a.) FIFO - Using 3 Frames



No. of page faults = 9

FIFO - 4 frames



No. of page fault = 10.

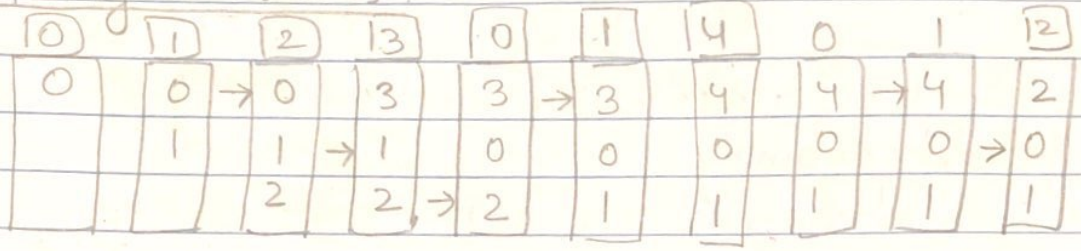
We see, the no. of pagefaults with 4 frames > no. of page fault with 3 frames.

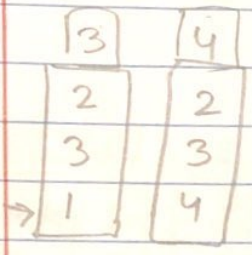
So, FIFO suffers from Belady's anomaly.

b.) LRU

0 1 2 3 0 1 4 0 1 2 3 4

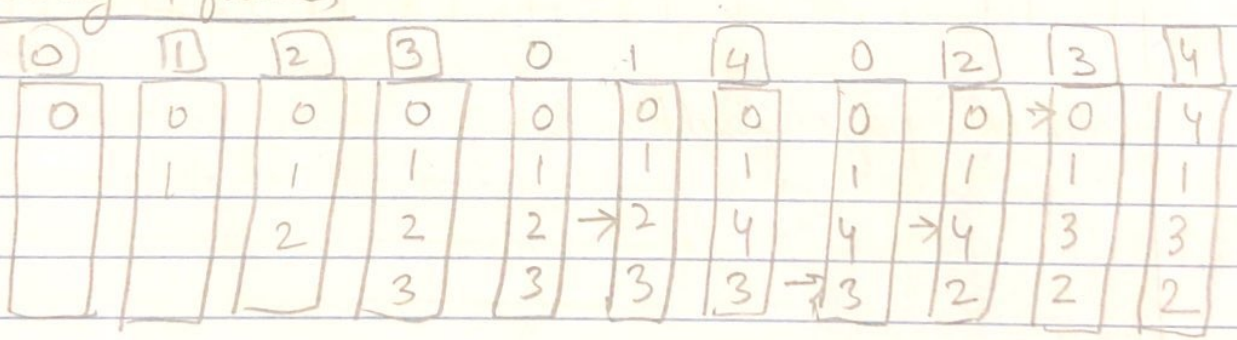
Using 3 Frames





No. of page fault = 10

Using 4 frames



No. of page faults = 8.

In case of LRU we see that,

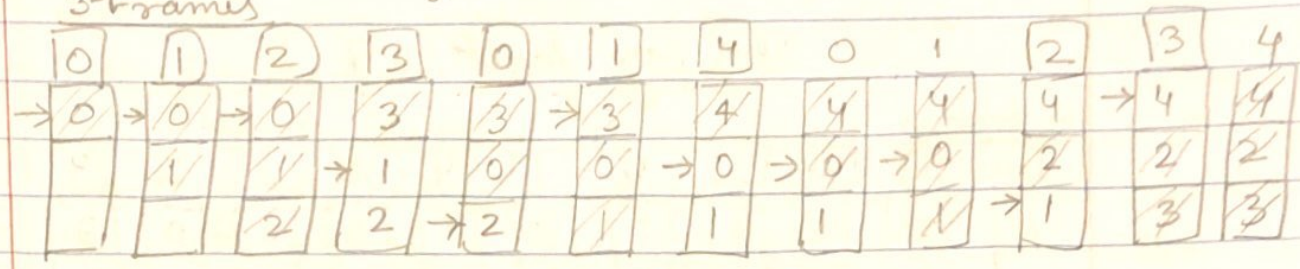
page fault 4 frames < page fault 3 frames.

∴ It doesnot suffer from Belad'ys Anomaly.

0 1 2 3 0 1 4 0 1 2 3 4

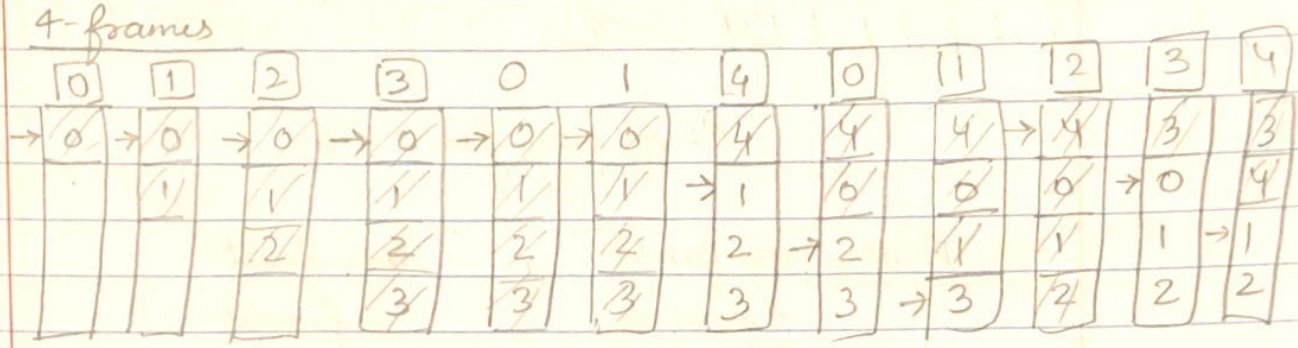
(•) Second chance Algorithm

3-Frames



No. of page faults = 9.

4-frames



No. of page fault = 10

We see that,

No. of page fault (4 frames) > No. of page faults (3 frames)

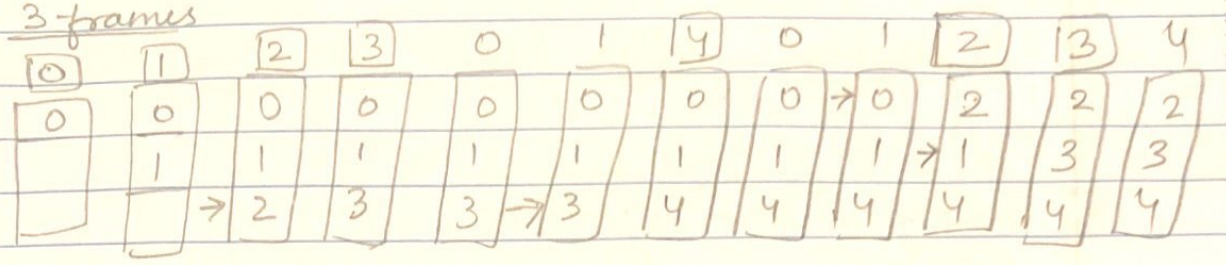
Second chance algorithm suffers Belady's anomaly.

{ □ - pagefaults
☑ - used
→ - victim next

⊙ OPTIMAL

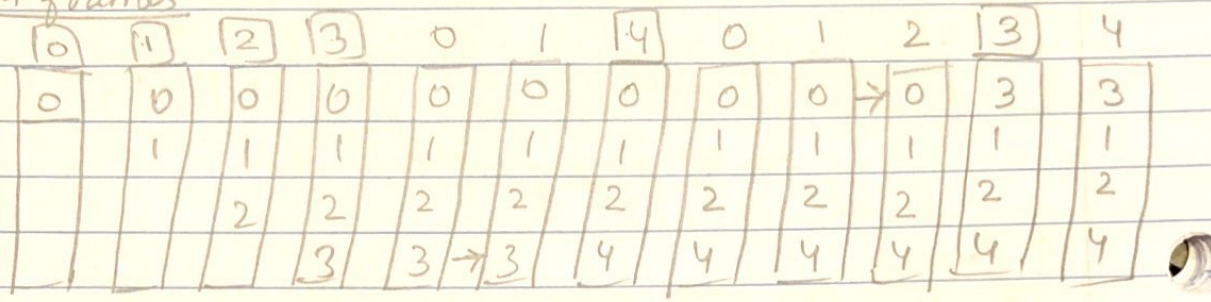
0 1 2 3 0 1 4 0 1 2 3 4

3 frames



No. of page faults = 7.

4 frames



No. of page fault = 6.

For optimal, page fault (4 frames) < Page fault (3 frames).

∴ It doesnot suffer from Belady's anomaly.

Q7. The suggested trick seems to be good. Writing page into the device back is very costly. It takes a lot of time. The code page is already in the disk and can be accessed from there when required. This would save OS time & disk space. But, if the binary file gets changed while the program is running in the physical memory, that will cause problem because different program will get loaded when required.

If the binary file is locked so that user can't make changes while program is running in the main memory, the suggested trick can work.