

(1)

	P1	P3	P5	P7	P8
23	Blocked	Blocked	Ready Suspended	Blocked	Ready
37	Suspended Ready	Ready Suspended	Blocked	Blocked	Ready
47	Ready	Ready	Ready	Blocked	Exited

(1)

	P1	P3	P5	P7	P8
23	Blocked read from disk unit 3	Blocked read from disk unit 2	Ready	Blocked write disk 3	Ready
37	Ready	Ready	Suspended	Blocked write disk 3	Ready
47	Ready	Ready	Ready	Blocked write disk 3	Exited

Assuming that only one process can execute at a time and for the unknown times (like 23, 37 & 47) we don't know if the process is executing, so we assume it to be in ready queue. Also, I assumed that a process is swapped therefore it was not being accessed frequently i.e. it was suspended.

②

a) LRU

Rank - 4

(A worst algo can be designed
by looking at future, so 5 not given)

b) FIFO

Rank - 2, Belady's anomaly observed.

c) Optimal replacement

Rank - 1

d) Second-chance replacement

Rank - 3

Assuming, as rank increase page fault rate increases.

③

a) LRU

No of Frames	No of faults
2	20 (All page faults)
2	18
3	16
4	9
5	8
6	8
7	7

b) FIFO

No. of frames	No. of faults
1	20
2	18
3	16
4	12
5	10
6	10
7	7

c) OPT

No. of frames	No. of faults
2	20
3	19
4	11
5	8
6	7
7	7

⑤ a) FCFS (First come first serve)

Head movements =

$$(26-26) + (37-26) + (100-37) + (100-14) + (88-14) \\ + (88-33) + (99-33) + (99-12)$$

$$= 442$$

$$26 \rightarrow 37 \rightarrow 100 \rightarrow 14 \rightarrow 88 \rightarrow 33 \rightarrow 99 \rightarrow 12$$

b) SSTF (Shortest seek time first)

head movements

$$\begin{aligned} &= (26-26) + (33-26) + (37-33) + (37-19) \\ &\quad + (19-12) + (88-12) + (99-88) + (100-99) \\ &= \underline{\underline{124}} \quad (26 \rightarrow 33 \rightarrow 37 \rightarrow 14 \rightarrow 12 \rightarrow 88 \rightarrow 99 \rightarrow 100) \end{aligned}$$

c) SCAN (Assuming 100 is last cylinder)

$$26 \rightarrow 33 \rightarrow 37 \rightarrow 88 \rightarrow 99 \rightarrow 100 \rightarrow 19 \rightarrow 12$$

head movements

$$\begin{aligned} &= (26-26) + (33-26) + (37-33) + (88-37) \\ &\quad + (99-88) + (100-99) + (19-12) \end{aligned}$$

= 162 (Assuming 100 is last & 0 is first cylinder)

d) C-SCAN

$$26 \rightarrow 33 \rightarrow 37 \rightarrow 88 \rightarrow 99 \rightarrow 100 \rightarrow 0 \rightarrow 12 \rightarrow 19$$

$$\begin{aligned} &= (26-26) + (33-26) + (37-33) + (88-37) \\ &\quad + (99-88) + (100-99) + (100-0) + (12-0) \\ &\quad + (19-12) \end{aligned}$$

$$= 188$$

- ⑥
- rotational speed (ω) = 15,000 rpm.
 - # of bytes per sector (n_b) = 512 bytes.
 - # sectors per track (n_s) = 400
 - # tracks = 1000 (n_t)
 - avg seek time = 7 ms (v)
 - File size(S) = 1 MB = 1,098,576 Bytes

a) No of sectors required (n_s)

$$= \frac{1,098,576}{512} = 2048 \text{ sectors.}$$

Transfer time

= Rotational delay + Total seek time

~~$$= \left(\frac{2048}{400} \times \frac{1}{\omega} \right) \text{ min} + \left(\frac{2048}{900} \right) \times$$~~

$$= \left(\frac{2048}{400} \times \frac{1}{\omega} \right) \text{ min} + \left(\left[\frac{2048}{400} \right] \times v \right) \text{ ms}$$

$$= (5.12 \times (15,000)^{-1}) \text{ min} + (5 \times 9) \text{ ms}$$

$$= (0.0205) \text{ s} + (0.02) \text{ s}$$

$$= 0.0405 \text{ seconds.}$$

- b) Average access time is the total seek time = 0.02 seconds
- c) Rotational delay = 0.0205 seconds

d) Total time to read 1 sector

$$= \left(\frac{1}{400} \times \frac{1}{\omega} \right) \text{ min}$$

$$= \left(\frac{1000}{400} \times \frac{60}{15,000} \right) \text{ ms}$$

$$= 0.01 \text{ ms}$$

(Assuming disk read head was positioned at the sector, only rotational latency is involved)

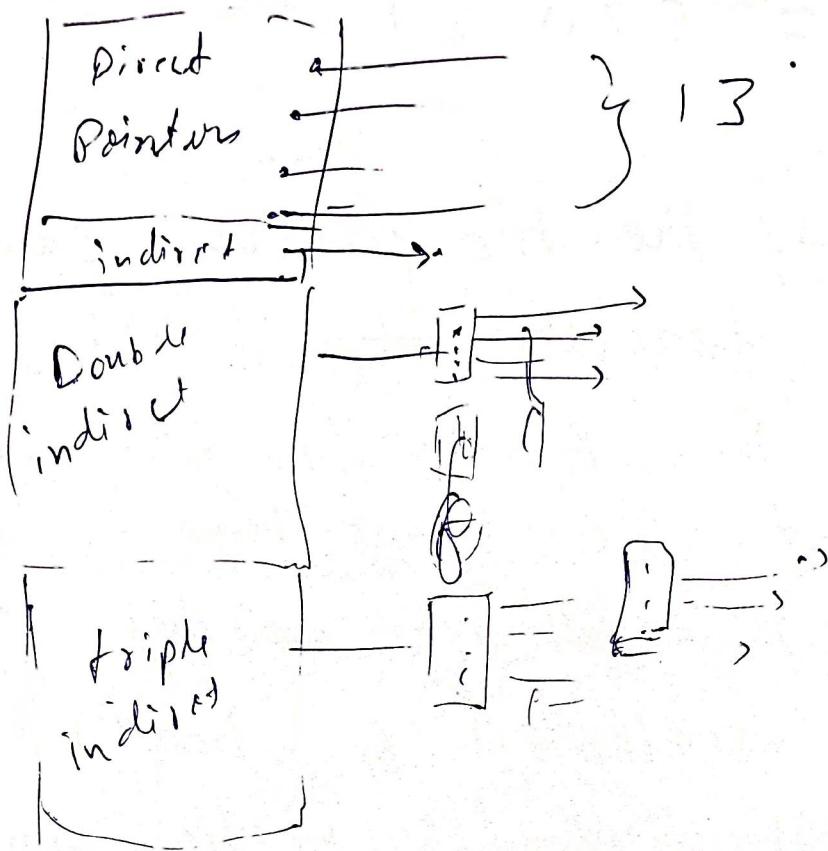
e) Total time to read one track.

$$= \left(\frac{1}{\omega} \right) \text{ min}$$

$$= \left(\frac{1000 \times 60}{15,000} \right) \text{ ms}$$

$$= 4 \text{ ms}$$

i-node ONIX



$$\text{One block} = 8 \text{ KB } (\Delta)$$

Total size of file the i-node can handle

$$= (\text{Due to direct pointers}) + (\text{Due to indirect pointers} \times \text{Direct pointers})$$

$$+ (\text{Due to double indirect} \times \text{Due to indirect}) \\ \times \text{Due to direct pointers}$$

$$+ (\text{Due to triple indirect} \times \text{Due to double indirect} \\ \times \text{indirect} \times \text{Direct pointers})$$

$$= 13s + (13 \times 13s)s + (13 \times 13 \times 13)s + (13)^9 s$$

$$= 13(1 + 13 + 13^2 + 13^3)s = 30,940 s$$

$$\begin{aligned}
 &= 30,990 \times 8 \text{ KB} \\
 &= 247,520 \text{ KB}
 \end{aligned}$$

④ (a) Yes, the two process can be blocked forever if foo() gains lock on S & bar() gets lock on R at the same time. Now, foo() will be waiting for R to be unlocked & bar() will be waiting for S to be unlocked forever. Neither of them would get any lock on both S & R & thus block each other forever.

(b) No, if a single process will not be postponed while other is running. Because if one process is running this means it has lock to both semaphore S & R & the other process is waiting. When the locked are released - The other process starts executing. So, they ~~will~~ may

both get blocked, but, one process can't indefinitely postpone the other process.

- ⑧ As described in reflection attack an attacker takes advantage of a challenge-response authentication and makes the target to answer his own challenge & sends the target response of correct authentication.

To guard against such an attack, we have to use a better authentication mechanism like using public & private keys. The attackers can't simply use ~~us~~ to ~~to~~ make the target to authenticate them. Also, the function used to process the keys are very complex & non-invertible, so it will provide more secure authentication.

Other simple method can be that user will not accept a challenge that ~~is~~ the user has sent itself. Requiring need of a large number of challenges.

Yet, another protection strategy can be to use different protocols at sender & receiver side. So, some sender side's response will not be accepted on the receiver side.

② a) LRU (Least Recently used)

It is not the worst because we can devise an algo which results in max page faults by looking at future.

~~Given above rank order is~~

Rank = 4

b) FIFO (First in first out)

It is a frequently used algo, but, not as good as OPT

Rank = 2

c) OPT

It is the best algo which looks at future of page requests & makes optimal page replacement.

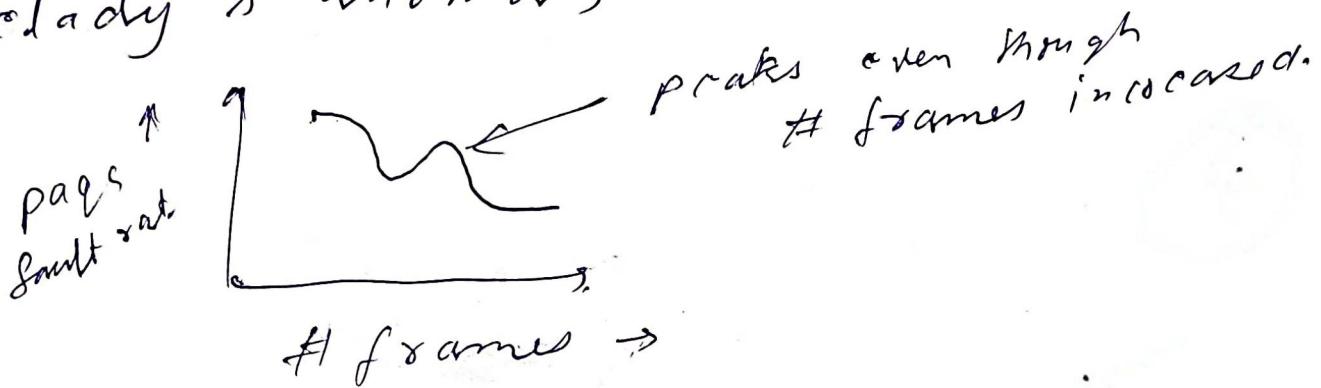
Rank = 1

d) Second-change

It is an improved version of LRU

$$\therefore \text{Rank} = 3$$

Brady's anomaly



Bentley's anomaly is observed in FIFO.

Note :- FIFO can also perform worse than LRU in certain scenarios, since the performance of these algo (except OPT) is heavily dependent on the actual sequence of requests.