FILE & DEVICE MANAGEMENT

→ Hard disk.

→ Tape.

File: Collection of logically related entities (records)

Attributes:

- Name
- Size
- Type
- Location
- Creation Date
- Password
- Owner
- Last modified Date
- Permissions, etc.

all attributes of file is called as file context & the file context will be stored in FCB (File context Block)

Types of file:

1) .DOC
2) .Txt
3) .obj
4) .exe
5) .class
6) .jpg
7) .c
8) .java
9) .Dll
10) .xls
11) .mp3

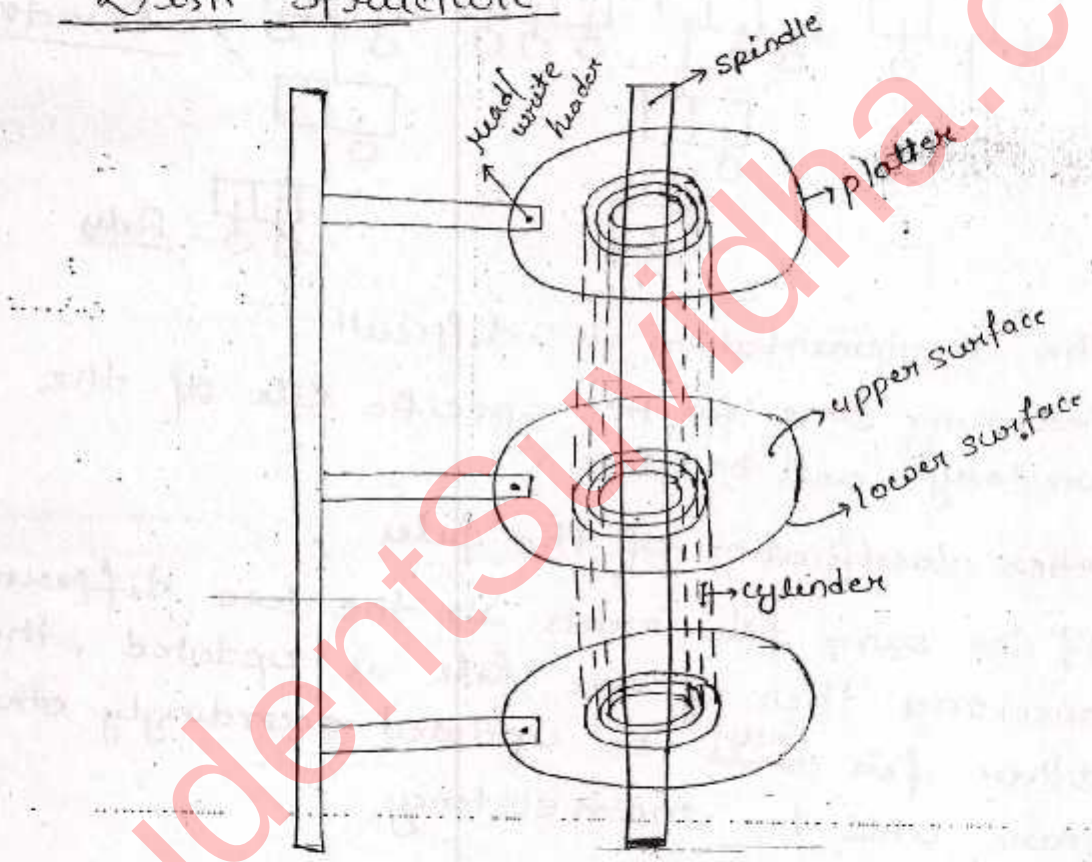
→ The implementation is difficult.

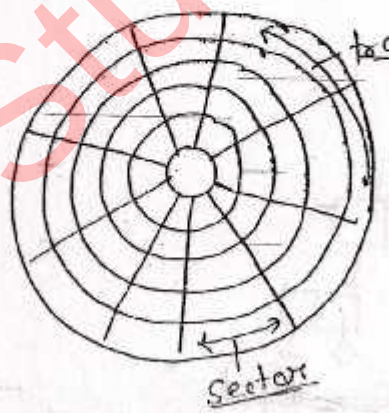
→ The searching time will be less.

→ Better classification of the files.

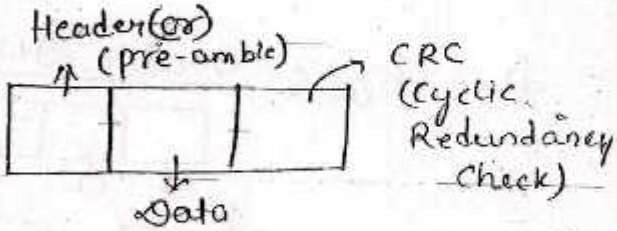
→ If the same file exists in two different directory, then if one file is updated then other file will be updated automatically by using the file sharing concept.

Disk Structure :-





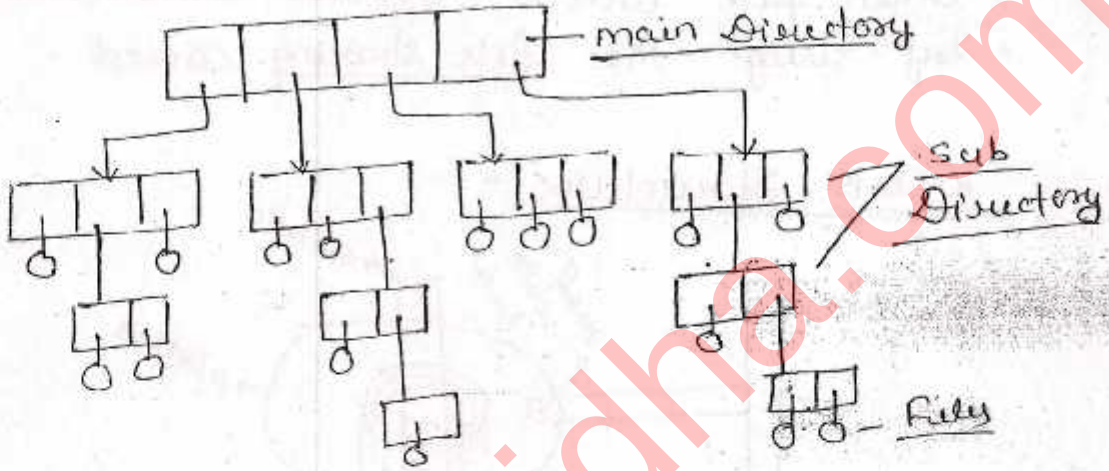
Surface



Sector or Disk Block

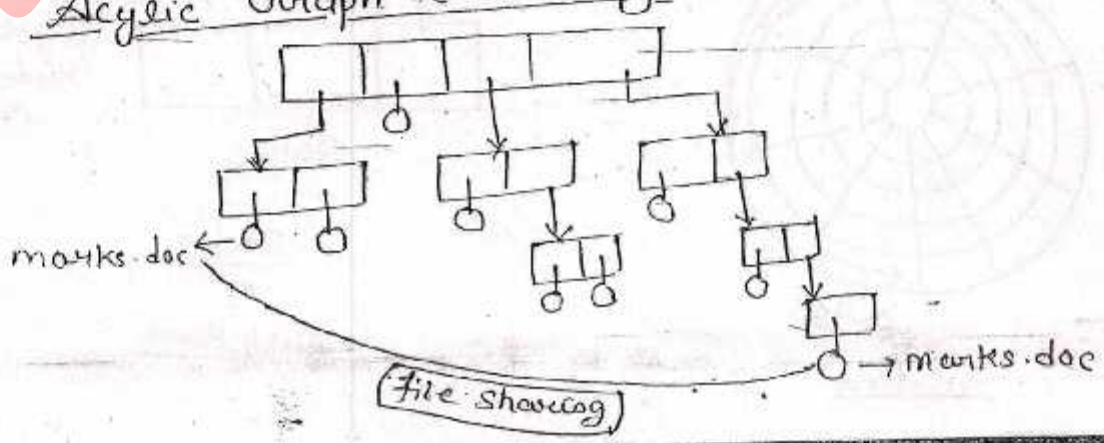
→ The implementation is easy.
→ Two files can't have the same name. (inconsistency)
→ Searching time will be more. (for specific file of the directory)

2) Multi level (tree level) Directory.



→ The implementation is difficult.
→ Searching time for the specific file of the directory will be less.
→ Better classification of the files.
→ If the same file exists in the two different directories then if one file is updated, the other file must be updated accordingly, otherwise there will be inconsistency.

3) Acylic Graph Directory.



Various Operations perform on the file

(i) Create
(ii) Open
(iii) write
(iv) Read
(v) Update
(vi) truncate
(vii) append
(viii) Save
(ix) Copy
(x) paste
(xi) close
(xii) Delete
(xiii) Re-name
(xiv) Save as

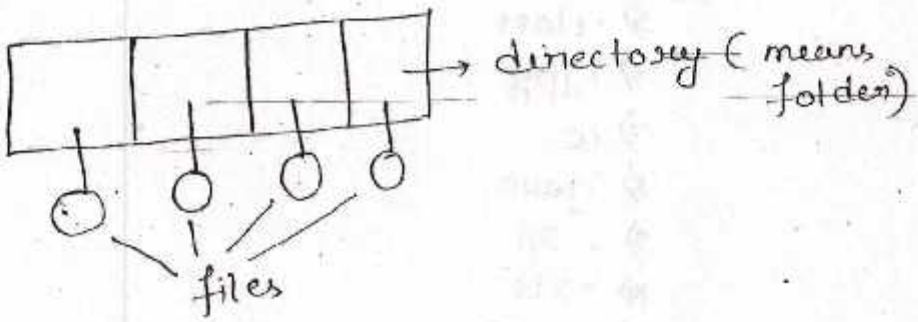
Various Access method

1) Sequential Access method
2) Random Access method

for the better classification of the files, the files will be stored on the directory.

Various Directory Structure

1) Single level Directory-



Q. consider a disk, which has 16 platters & every platter has two surfaces. Every surface has 1K tracks & every track is further divided into 512 sectors And every sector can store the 2KB data.
Then calculate

(i) what is the capacity of the disk?
(ii) How many bits are required to identify any particular ~~this~~ sector of the disk?

Sol: (i) → 1 platter – 2 surface
16 platter = 16 × 2 ie. 32 surface.

→ 1 surface – 1k tracks
32 surface – 32 k tracks.

→ 1 track – 512 sector
32K track – 32 × 512 × K sector.

Capacity of disk = $2^5 \cdot 2^9 \cdot 2^{10} \cdot 2^1 \cdot 2^{10}$ B
= 32 GiB

(ii) # of Sector in one disk = 16 × 2 × 1K × 512
= $2^4 \cdot 2^1 \cdot 2^{10} \cdot 2^9$
= (4+1+10+9) = 24 bits

Disk I/O Operation:

→ Seek time
→ Rotational latency → Transfer Rate
→ Transfer time

> → The read/write header can b never be outside the track, it will be pointing to any particular track of the surface.

→ Read/write header will be move in forward & backward direction. & Disk in one direction (either clockwise or anti-clockwise)

1) **Seek Time**: The amount of time taken to move the read/write header from its current position to the desired track is called as seek time.

2) **Rotational Latency**: The amount of time taken to rotate the track when the read/write header comes to exact position (~~track~~ Sector)

> The rotational latency is considered as
> $= \frac{1}{2}$ Rotation time

3) **Transfer Time**: The amount of time taken to transfer the required data.

— The transfer time depends on the rotational rate of the disk & the total size of the track.

→ The implementation is difficult

Transfer Rate-
The no. of bytes transferred for unique time is called as transfer rate of disk.

Q. Consider a disk system, which has an avg seek time of 30 ms. and the rotational rate of the disk is 360 rpm. Each track of the disk has 512 sectors. each of size 512 Bytes. Then calculate (continuous) 4 successive sectors?

a) What is the time taken to read

a) 0.32 sec b) 0.084 sec c) 0.046 sec d) 0.56 sec

ii) What is the data transfer rate?

a) 1336 KBPS b) 1436 KBPS c) 1536 KBPS d) 1636 KBPS

Sol:- Seek time = 30 ms $= 30 \times 10^{-9}$ sec.

Rotational Latency ⇒ 360 rotation = 60 sec

$\frac{1}{2}$ rotation $= \frac{\frac{1}{2} \times 60}{360}$

$= \frac{1}{12}$ sec.

$= 0.083$ sec.

Transfer time ⇒

In one rotation time, we can read the total size of the track & to read the required data how much time is required?

In 1 RT ← we can read total size of data

? ← To read required data.

∴ 1 rotation time $= \frac{1}{6}$ sec

total size of track $= 2^9 \cdot 2^9 B$

$= 2^8 \cdot 2^{10} B = 256 KB$

required data = 4 sector × 512 B

$= 2^2 \cdot 2^9 B = 2^1 \cdot 2^{10} B$

$= 2 KB$

256 KB ⟶ $\frac{1}{6}$ sec

2 KB ⟶ $\frac{\frac{1}{6} \times 2KB}{256 KB} = \frac{1 \times 2}{6 \times 256}$

$= \frac{1}{768}$

$= 0.0013$ sec.

ci) Time taken to read 4 successive sectors = RL + TT + ST

$= 30 \times 10^{-3} + 0.083 + 0.0013 + 0$

$= 0.030 \times 10^{-6} + 0.083 + 0.0013 + 0$

$= 0.084$ sec Ans.

cii) Data transfer rate-

In one rotation time ⟶ we can transfer total size of the track.

In one sec ⟶ how much data we will transfer.

$\frac{1}{6}$ sec ⟶ 256 KB

1 sec ⟶ $\frac{256 KB \times 1 sec \times 6}{sec}$

$= 1536 KB$ i.e. 1536 KBPS

Q. Consider a disk system which has an average seek time of 60 ms and the rotational rate of the disk is 3600 rpm. Each track of the disk has 256 ~~track~~ sector each of size 2 KB. Then calculate –

(i) What is the approximate time required to read 1200 random sectors.

a) 10sec b) 20sec c) 30 sec d) 40 sec

(ii) What is the data transfer rate?

a) 30 mBPS b) 40 mBPS c) 50 mBPS d) 60 mBPS

Sol:- Seek time = 60 ms = 60×10^{-9} sec.

Rotational latency ⇒ 3600 rotation — 60 sec

1 rotation — $\frac{60}{3600}$

$= \frac{1}{60}$ sec.

$RL = \frac{1}{2} \times RT$

$= \frac{1}{2} \times \frac{1}{60} = \frac{1}{120}$ sec.

Transfer time ⇒ 512 KB — $\frac{1}{60}$ sec

2 KB = 0.0000655 sec.

~~Total time~~ transfer time = $(RL + TT) \times 1200$

$= (0.0083 + 0.0000655) \times 1200$

Time required to read 1200 random sectors = ~~60×10^{-9} sec~~ $+ (0.0083 + 0.0000655) \times 1200$

= 10.038 sec.

10 sec

$\frac{1}{60}$ sec → 512 KB

1 sec → 60×512 KB

→ $60 \times 2^9 \cdot 2^{10}$ B

→ $15 \times 4 \times 2^9 \cdot 2^{10}$ B

→ $15 \times 2^2 \cdot 2^9 \cdot 2^{10}$ B

→ $30 \cdot 2^{20}$ B

→ 30 MBps.

→ Every sector has its own Ring Rotational latency and Transfer time & also seek time.

Q. How long does it take to load the 64 KB of a program from a disk, whose one rotation time is 20 ms & the avg seek time of the disk is 30 ms and the total size of the track is 32 KB. The paging is applied on the program & the program is divided into pages of size 2 KB. Assume that the pages are spread randomly over the disk.

a) 1230 ms b) 1320 ms c) 1420 ms d) 1520 ms.

Sol:-

Sol:-

ST = 30 ms

RL = 10 ms.

TT ⇒

32 KB — 20 ms

64 KB — $\frac{64 \text{ KB} \times 20 \text{ ms}}{32 \text{ KB}}$ = 40 ms

not required.

or

32 KB — 20 ms

2 KB — $\frac{2 \times 20}{32}$ = 1.25 ms.

$$\#\text{ of pages} = \frac{64KB}{2KB} = 32$$

Total time = $(ST + RL + TT) \times 32$

$= (30 + 10 + 1.25) \times 32$

$= 1320ms$

Q. How long does it take to load 128KB of a program from a disk, whose 1 rot^n time is 40ms & the avg seek time of the disk is 60ms & the total size of the track is 64KB. The paging is applied on the program. & the program is divided into 128 pages. and assume that the pages are spread randomly over the disk.

Sol: ST = 60ms

RL = 20ms.

$$\#\text{ of page} = \frac{128KB}{P.S} = 128$$

$$\Rightarrow P.S = \frac{128KB}{128} = 1KB$$

TT $\Rightarrow$ 64KB — 40ms

for one page ... 1KB — $\frac{1KB \times 40ms}{64KB} = \frac{40}{64}$ ms

Total time = $\left(60 + 20 + \frac{40}{64}\right) \times 128$

$= 10320ms$

$= 10.32$ sec.

Q Consider a disk with the following specification.

1) No. of surfaces = 8
2) inner diameter = 4cm
3) outer diameter = 12cm
4) Inter track distance = 0.2mm
5) No. of sectors/track = 20
6) Sector size = 4KB
7) Rotational Rate = 3600 rpm

Q(i) What is the capacity of the disk?

(a) 128MB (b) 256MB (c) 512MB (d) 1GB

(ii) What is the data transfer rate?

(a) 1800 KBps (b) 2800 KBps (c) 3800 KBps (d) 4800 KBps

Sol:- (i) Capacity of disk =>

Outer diameter - inner diameter = (12-4) = 8cm

take radius =

(outer r. - inner)

(6-2) = 4cm.

No. of tracks = $\frac{4cm}{0.02cm} = \frac{400}{2} = 200$

~~No. of tracks = $\frac{8cm}{0.2mm} = \frac{8cm}{0.02cm}$~~

~~wrong.~~

~~$= \frac{800}{002}$~~

~~$= 400$~~

Capacity of disk = $200 \times 8 \times 20 \times 4KB \approx 128MB$

$= 2 \times 100 \times 8 \times 20 \times 4KB$

$= 2^1 \cdot 25 \cdot 2^2 \cdot 2^3 \cdot 2^1 \cdot 10 \cdot 2^2 \cdot 2^{10} B$

$= 2^1 \cdot 25 \cdot 2^2 \cdot 2^3 \cdot 2^1 \cdot 2^1 \cdot 5 \cdot 2^2 \cdot 2^{10}$

$= 125 \cdot 2^{21}$

$= 125 \cdot 2 \cdot 2^{20} B$

1 cm = 10mm

(ii) Total size = 4KB × 20 = 80KB

∴ RR = 3600 rpm

RL ⇒ 3600 r — 60 sec

1 r — $\frac{60}{3600}$ = $\frac{1}{60}$ sec.

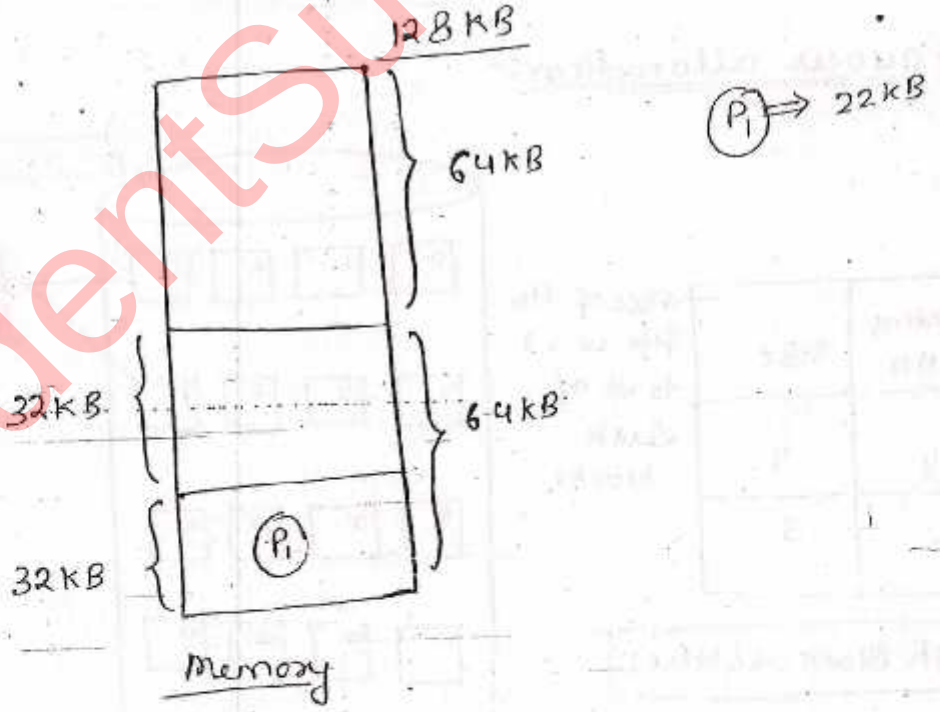
RL = $\frac{1}{120}$ sec = 0.083

$\frac{1}{60}$ sec — 80KB

1 sec — $\frac{1 \text{ s} \times 80\text{KB} \times 60}{1 \text{ s}}$

= 4800KB Ans.

Buddy System:



- In Buddy system, initially the m/m will be single continuous m/m free block.
- Whenever the request by the process comes in, the m/m will be divided into two half Blocks.
- If the request by the process is too small then the lower block of the m/m is further divided into two half blocks again.
- In the buddy system, the m/m will allocated from lower blocks to higher blocks.

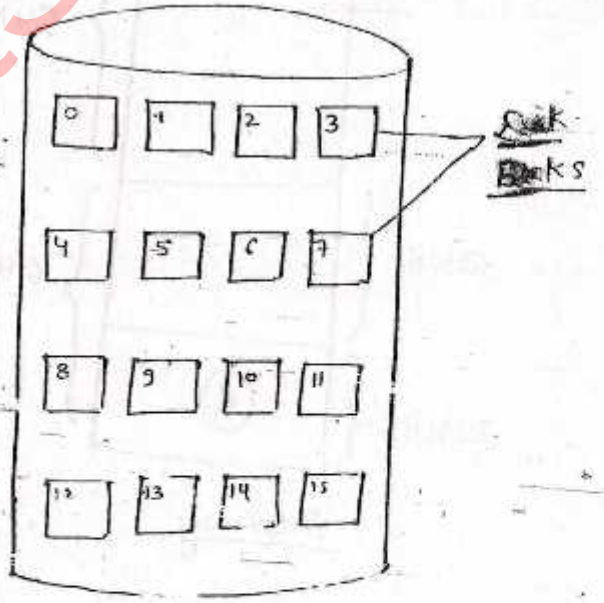
Disk Space Allocation Methods

1) Contiguous allocation:-

file	Starting DBA	Size
abc.doc	4	4
doc	i	3

Size of the file w.r.t to # of disk blocks

DBA (Disk Block Address)



- file abc.doc having size = 4. Disk block (4, 5, 6, 7) are allocated to this file. (4 = starting)

- file xy.doc having size = 3. Disk block (10, 11, 12) are allocated. (10 = starting DBA)

→ In the contiguous allocation, disk blocks are allocated to the file in a continuous manner.

→ Increasing the file size is not possible always. (bcz next block may or may not free)

→ It is suffering from the external fragmentation. (if disk block 8 is free & if we want to allocate this block to file abc.doc, by increasing size but we can't do it bcz it is not contignous & suffers from external fragmentation)

→ The internal fragmentation exist in the last disk block of the file.

→ It supports both sequential and the random access of the file. (using DBA we can jump on required loc.)

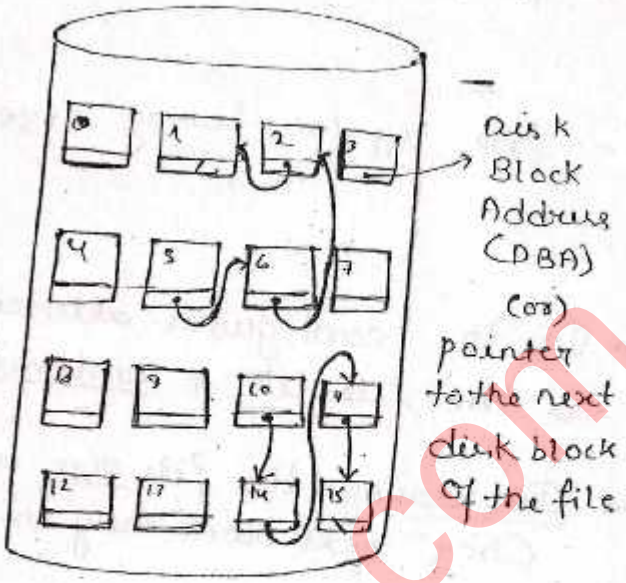
2) Non-contiguous allocation:-

→ Increasing the file size is always possible, if the free disk block is available.

→ There is no external fragmentation.

→ The internal fragmentation may exist in the last disk block of the file.

file	Starting DBA	Ending DBA
abc.doc	5	1
xyz.doc	10	15



Disk

→ It supports only the sequential access of the file.

→ There is an overhead of maintaining the pointer in every disk block.

→ The pointer of any disk block is lost, then the file will be truncated.

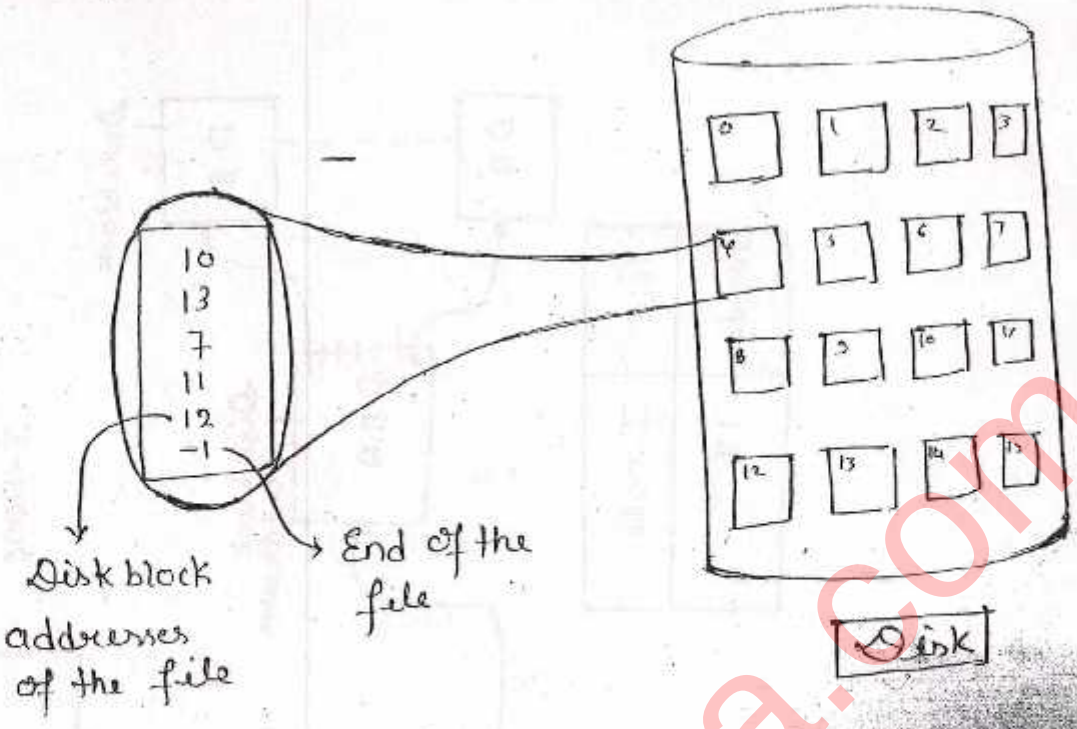
→ The ending DBA confirms that the total file is accessed.

3) Index allocation :-

→ In the index allocation, one disk block is use to just to store the DBA(s) of the file.

→ Every file is associated with its own index node.

→ If the file is very very large then one disk block may not be sufficient to store the DBA(s) of the file.



File	Index Node
abc.doc	4

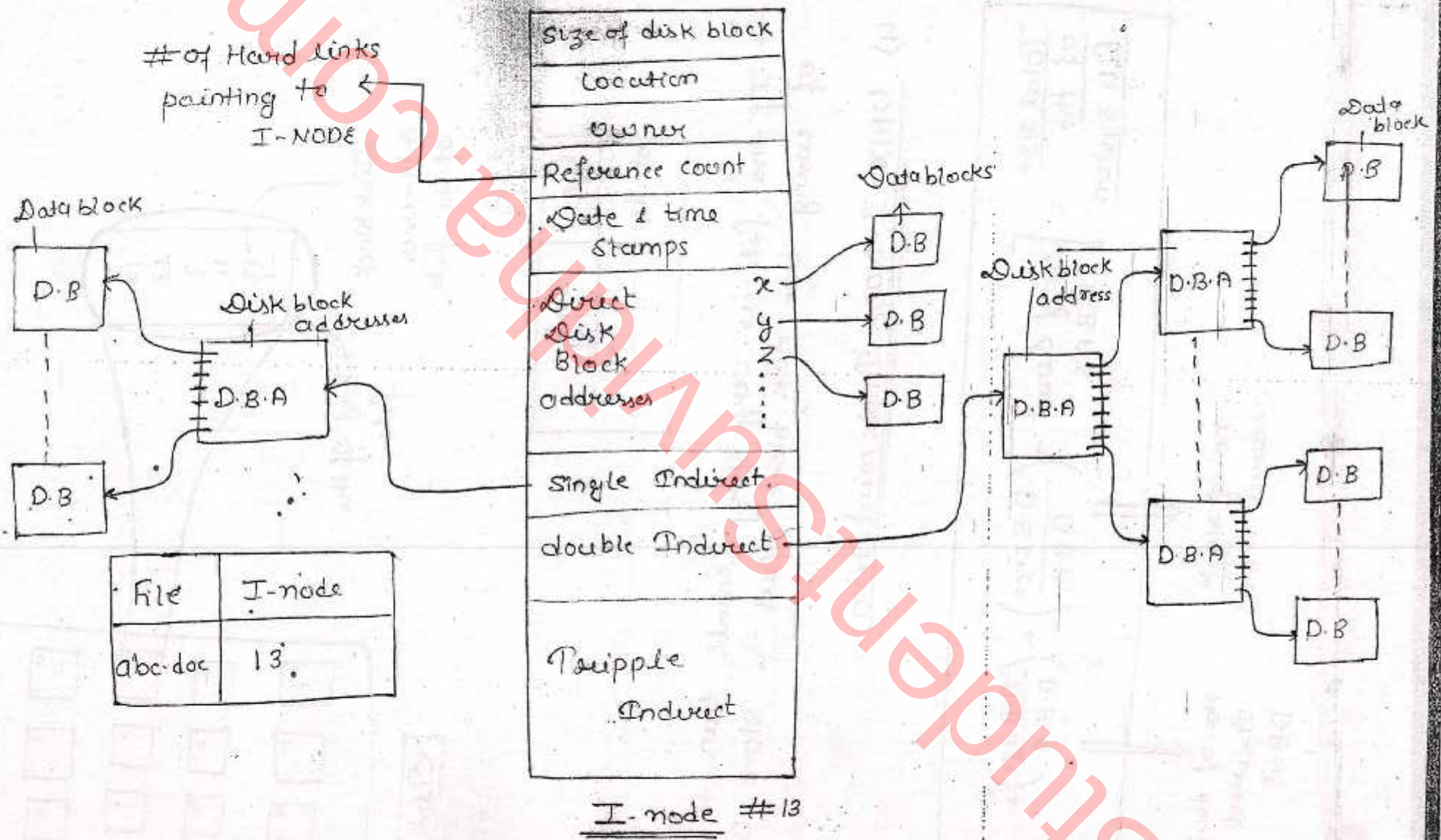
→ If the file is very very small, then its waste of using one disk block, just to store DBA(s).

4) UNIX - I-NODE Implementation:

$$\text{Total size of the File system} = \left[\#\text{ of Direct D.B.A's} + \left(\frac{\text{D.B size}}{\text{D.B.A}}\right) + \left(\frac{\text{D.B size}}{\text{D.B.A}}\right)^2 + \cdots\right] * \text{DB size}$$

no. of single indirect D.B.A's

no. of double Indirect D.B.A's



→ all are disk block (some store data & some addresses).

→ I-NODE is w.r.t. every file and every file will have its own I-NODE

→ I-NODE is associated with the various disk blocks, in some disk blocks we are storing the data & in some disk blocks we are storing the addresses.

→ Depending on the size of the file, the file will be stored only in one particular place, may be in the Direct DBA's or may be in the single indirect or may be in the double indirect or may be in the tripple indirect & so on.

$$\frac{\text{Disk block size (D.B.size)}}{\text{Disk block Address (D.B.A)}} = \#\text{ of Disk Block Addresses possible to store in one Disk Block}$$

$$\text{Max}^{m} \text{ Double Indirect} = \left(\frac{\text{D.B.size}}{\text{D.B.A}}\right)^2 \times \text{D.B.size}$$

$$\text{Max}^{m} \text{ Tripple Indirect} = \left(\frac{\text{D.B.size}}{\text{D.B.A}}\right)^3 \times \text{D.B.size}$$

Q. consider the UNIX-I-NODE 12 Direct D.B.P's, 1-single indirect, 1-Double indirect & 1-tripple indirect. The D.B.A requires 32-bits & the D.B size is 1KB. Then what is the max^m file size possible.

a) 4GB

b) 8GB

c) 16GB

d) 32GB

Sol:- max^m file size - consider only tripple indirect

$$= \left(\frac{\text{D.B.size}}{\text{D.B.A}}\right)^3 \times \text{D.B.size}$$

$$\left(\frac{1KB}{32}\right)^3 \times 1KB$$

$$= \left(\frac{1KB}{4B}\right)^3 \times 1KB$$

$$= \frac{1K \times 1K \times 1K}{4 \times 4 \times 4} \times 1KB$$

$$= \frac{1G \times 1024}{64} B$$

$$= 16GB \text{ Ans}$$

65/86.

Sol:

$$\left(\frac{\text{D.B size}}{\text{D.B.A}}\right)^3 \times \text{D.B.size}$$

$$\left(\frac{512B}{4B}\right)^3 \times 512\,B$$

$(128)^3 \times 512\,B \Rightarrow 2^{21} \cdot 2^9\,B = 1\,GB$ Ans.

Pg 2.66
Q.10

Sol:- DB size = 1024 B.

# of Disk Block address = $\frac{DBS}{DBA}$

$$128 = \frac{1024B}{x}$$

$$\Rightarrow x = \frac{1024B}{128} = \frac{2^{10}B}{2^7} = 2^3 B$$

$$\text{max}^m = \left(\frac{DB\ size}{DBA}\right)^3 \times DB\ size$$

$$= \left(\frac{2^{10}B}{2^3 B}\right)^3 \times 2^{10}B$$

$$= (2^7)^3 2^{10}B$$

$$= 2^{21} 2^{10} B = 2^1 GB = 2\ GB\ Ans.$$

Gate 2003

Q. A processor uses 2-level page tables for virtual to physical address translation.:

The page tables for the both level are stored in the main m/m. Virtual & physical addresses are both 32 bits wide. The m/m. is byte addressable. for virtual to physical address translation the 10 most significant bits of virtual address are used as index into the first level while the next 10 bits are used as index into 2-nd level page table. The 12- least significant bits of virtual address are used as offset with in the page. Assume that the page table entries in both the levels of page tables are 4-Bytes wide. Further the processor has a translation Look aside buffer with the hit-ratio of 96%. The TLB contains frequently used virtual page no's and the corresponding physical page no's. The processor also has a physically addressed cache with the hit ratio of 90%. The main m/m access time is 10nsec, cache access time is 1 nsec and TLB access time is also 1 nsec.

Q. What is the EMAT? (To the nearest 0.5ns)

3.8ns

a) 1.5ns

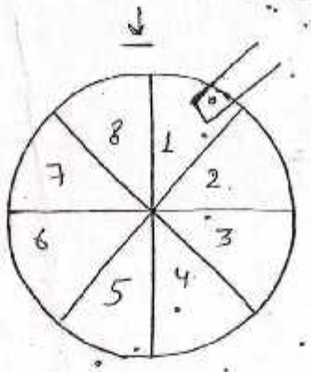
b) 2ns

c) 3ns

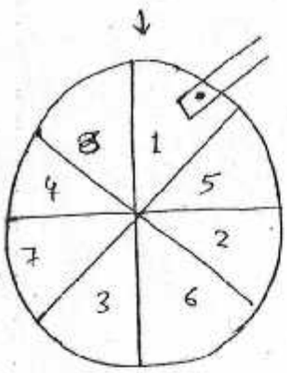
d) 4ns

- PT will never be stored in the cache.
- cache saves data pages only.

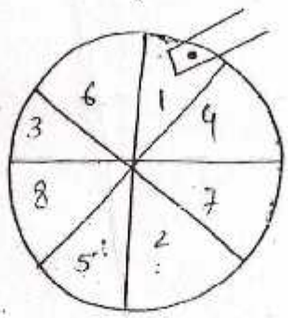
Disk Interleaving



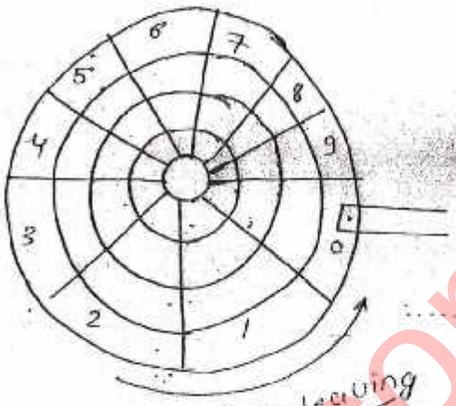
no interleaving disk



single interleaving disk



double interleaving disk



no interleaving disk

prob^{m} :- in 4th diagram, - read/write header has to read/write continuous sectors

- but problem is that speed of rotation of disk is very high and read/write header is not able to catch next sector.

∴ has to wait for completion of one rotation.

- read/write header is not able to catch the next sectors bcz header is busy in processing & calculating CRC & transferring data from previous read sector.

Sol^n:- interleaving makes header to comfortably calculate CRC, transferring data and be able to catch the sector.

Transfer time

No interleaving disk: one rotation time is needed to read all '8' sectors of the track.

Single interleaving disk: two rot^n time is required to read all '8' sectors of the track.

double interleaving disk: 2.75 rot^n time is required to read all '8' sectors of the track.

Qu:- Consider the above double interleaving disk, whose track is divided into 8 sectors each of size 2KB. The avg. seek time of the disk is 30 msec. and the rotational rate of the disk is 3600 rpm. The rotational latency is still considered as half rotation time.

Then calculate -

a) How much time is required to read all '8' sectors of the track using double interleaving disk.

~~a)~~ 84ms b) ~~74ms~~ 45ms c) 64ms d) 54ms

b) What is the time diff^n to read all the 8-sectors of the track if it is non-interleaving disk.

a) 55ms ~~b)~~ 29ms c) 37ms d) 17ms.

c) What is the data transfer rate using double interleaving disk.

a) 149KBPS b) 249 KBPS ~~c)~~ 349KBPS d) 449KBPS

Sol:- no. of sector = 8

each of size = 2KB

seek time = 30ms

Double interleaving

3600 r — 60 sec

$1r - \frac{60}{3600} = \frac{1}{60} s \approx 16.7 ms$

$R.L = \frac{1}{2} \times 16.7 ms = 8.35 ms$

T.T. = 2.75 x rotation time

= 2.75 x 16.7 ms = 45.925 ms.

Total time = (30 + 8.35 + 45.925) = 84ms (a)

(b) no interleaving

T.T = 1r = 16.7ms.

Total time = (30 + 8.35 + 16.7) = 55ms

Time diff = (84 - 55) = 29ms

(c) Data Transfer Rate

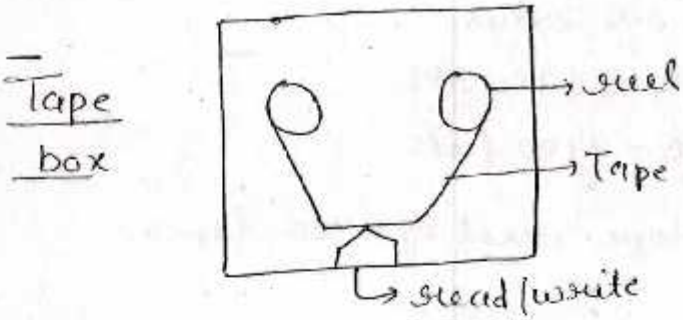
16KB — 45.925m

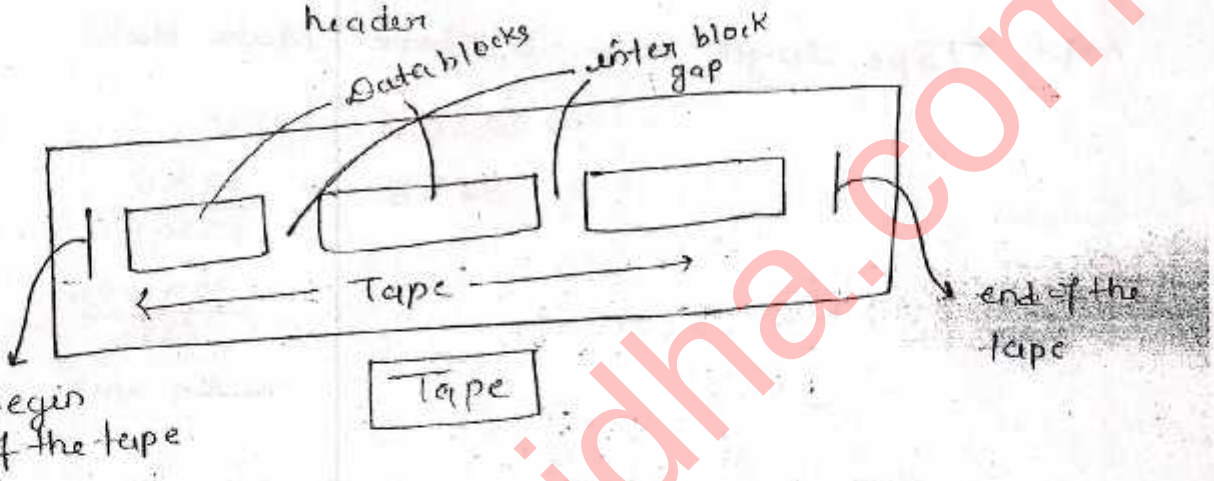
⇒ 45.925 ms — 16KB

1s — $\frac{16000}{45.925}$ KBps

≈ 348 KBps.

Tape





Consider $1K = 10^3$ only for tape

→ The tape is the cheapest secondary storage device generally used to take the large data back-up.

→ The tape is divided into various data blocks and each data block is seperated by inter-block gap.

→ The recording density in the tape is measured as BPI (bytes per inch)

4-6 Pg

Data block size = 32 KB
inter block gap = 0.4 inches
recording density = 6250 BPI
tape length = 2400 feets

# of B stored on tape reel of 2400 ft. = ?

Sol:- Tape length req. to store 1 data block

⇒ 6250 B — 1 inch

32 KB — $\frac{32\text{ KB}}{6250\text{ B}}$

$= \frac{32 \times 1000}{6250}$

= 5.1 inches

tape length req. to store 1 data block including gap

= (5.1 + 0.4) inches

= 5.5 inches

data blocks possible to store on given tape reel:

$= \frac{2400 \times 12}{5.5}$

= 5236

Capacity of the tape = 5236 x 32 KB

= 167552 KB

= 167.552 K. KB

= 167 MB

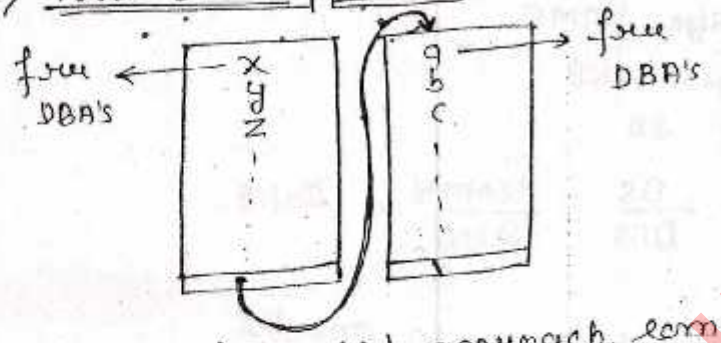
DISK FREE SPACE MANAGEMENT

Disk of size = 20MB
Disk Block size = 1KB
D.B.A = 16 bits

#f disk blocks (sectors) available on disk = $\frac{\text{Disk size}}{\text{Disk Block size}}$

$= \frac{20MB}{1KB} = 20K$

1) Free List Approach:



- In the free list approach, some disk blocks are used just to store, the free disk block addresses.

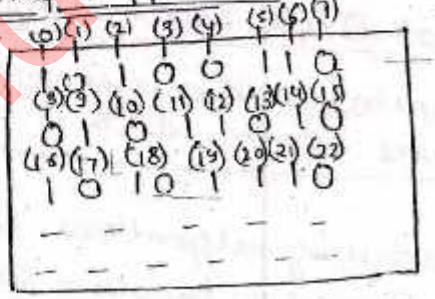
- #f disk block addresses possible to store in one disk block

$= \frac{\text{DB Size}}{\text{D.B.A}} = \frac{1KB}{16 \text{ bits}} = \frac{2^{10} \cdot B}{2B} = 512$

512 stored in 1 DBlock

20K stored in $\frac{20K \times 1DB}{512} = \frac{20 \times 2^{10}}{2^9} = 40$ DBlock

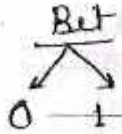
2) Bit-map Approach:



Bit-map (1 Disk Block)

free disk blocks – 1, 3, 4, 7, 8, ...

occupied disk block – 0, 2, 5, 6, 9, ...



0 – DBlock is free

1 – DBlock is occupied

- In Bit-map approach, only one binary bit is used to represent the free disk blocks.

#of DBlocks possible to map in one Disk block.

Disk block size = 1KB = 1K x 8 bits = 8K bits

In one Disk block we can map 8K Disk Blocks

8K — 1 DB

20K — $\frac{20K \times DB}{8K} = \frac{20}{8} DB = 2.5 DB \approx 3 DB$

13-pg64

Disk size = 40MB

DB size = 2KB

DBA = 3B

#of DB = $\frac{DS}{DBS} = \frac{40MB}{2KB} = 20K$

#of DBlocks possible to map in one DB

DBS = 2KB x 8 = 16 K bits

16K — 1 DB

20K — $\frac{20K \times 1DB}{16K} = \frac{20}{16} DB \approx 2$ Disk B

DISK SCHEDULING

Track requests = 98, 183, 37, 122, 14, 124, 65, 67.

Goal of Disk Scheduling

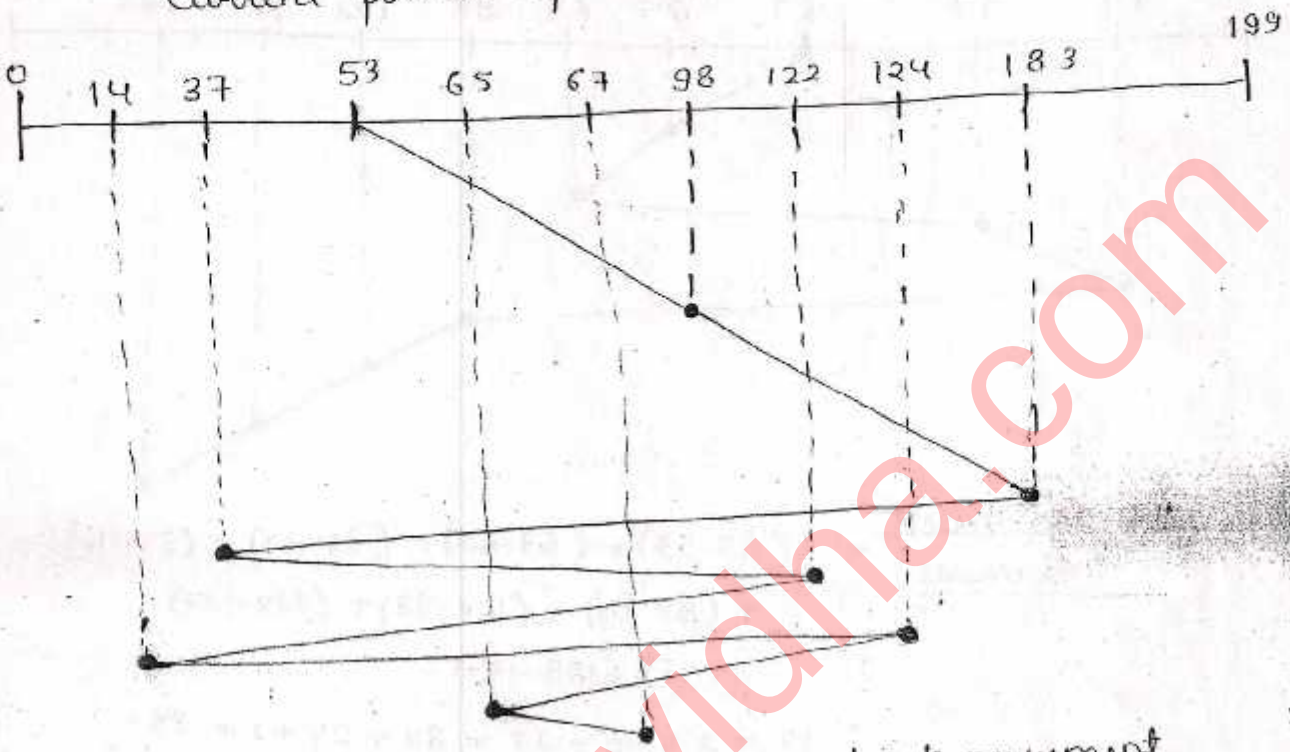
To minimize the average seek time of the disk.

Disk scheduling algorithm will decide, read/write header — go on which track if many requests for read. — then disk scheduling is required when many track requests will occur.

1) FCFS (First come First Serve)

Assuming 200 tracks from 0-199

Current position of read/write header = 53



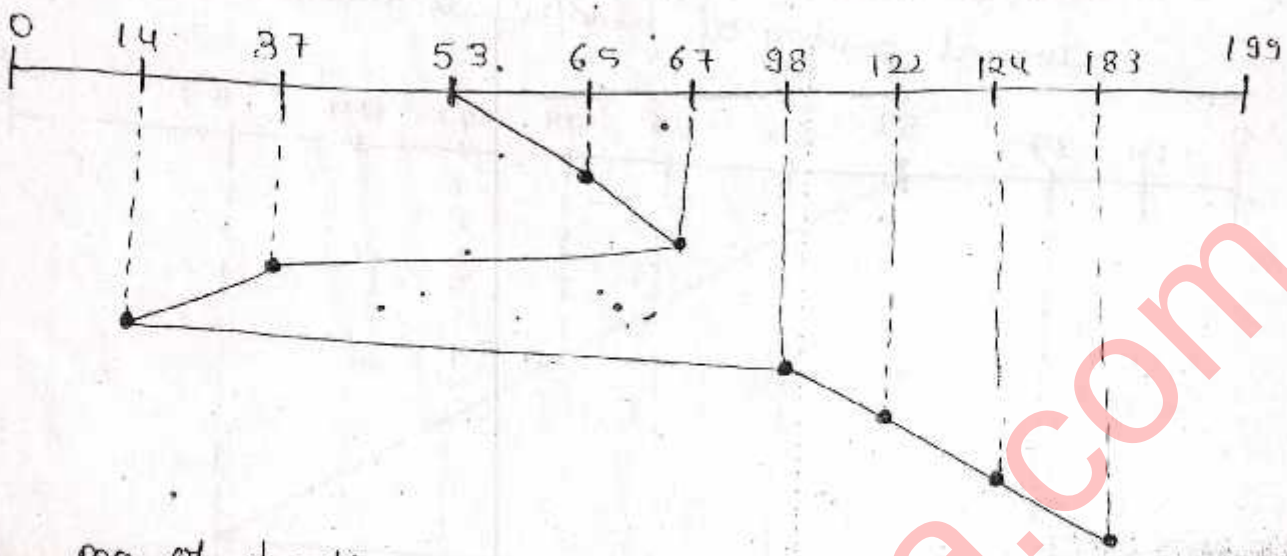
In this, we have to find out total track movement made by the read/write header.

no. of track movements = (98-53) + (183-98) + (183-37)
+ (122-37) + (122-14) + (124-14)
+ (124-65) + (67-65)

= 640 Ans.

2) SSTF (Shortest Seek Time First) or Nearest Track Next

Track request - 98, 183, 37, 122, 14, 124, 65, 67



no. of track movements = (65-53) + (67-65) + (67-37) + (37-14) + (98-14) + (122-98) + (124-122) + (183-124)

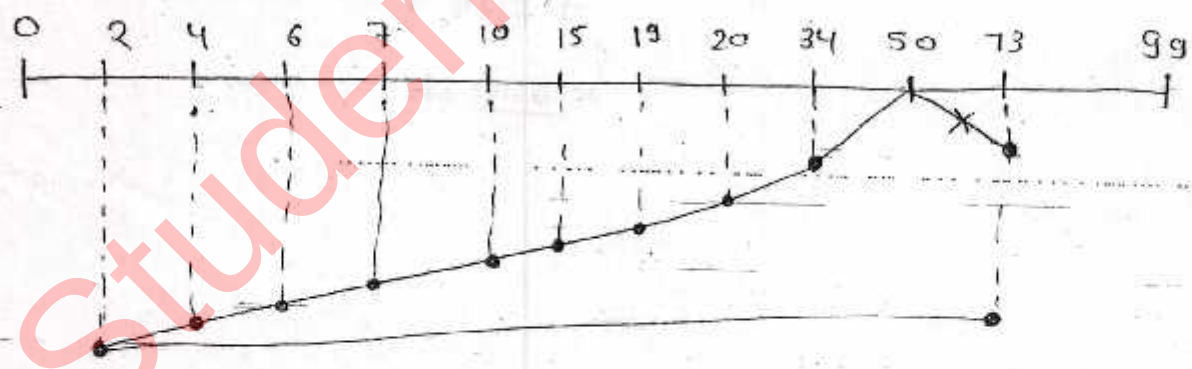
= 12 + 2 + 30 + 23 + 84 + 24 + 2 + 59

= 14 + 53 + ~~118~~ + 26 + 59 + 84

= ~~67~~ + ~~142~~ + ~~59~~ 67 + 85 + 84

= ~~209~~ + ~~59~~ 152 + 84 = 236 Ans.

Q.1 Track req - 4, 34, 10, 7, 19, 73, 2, 15, 6, 20

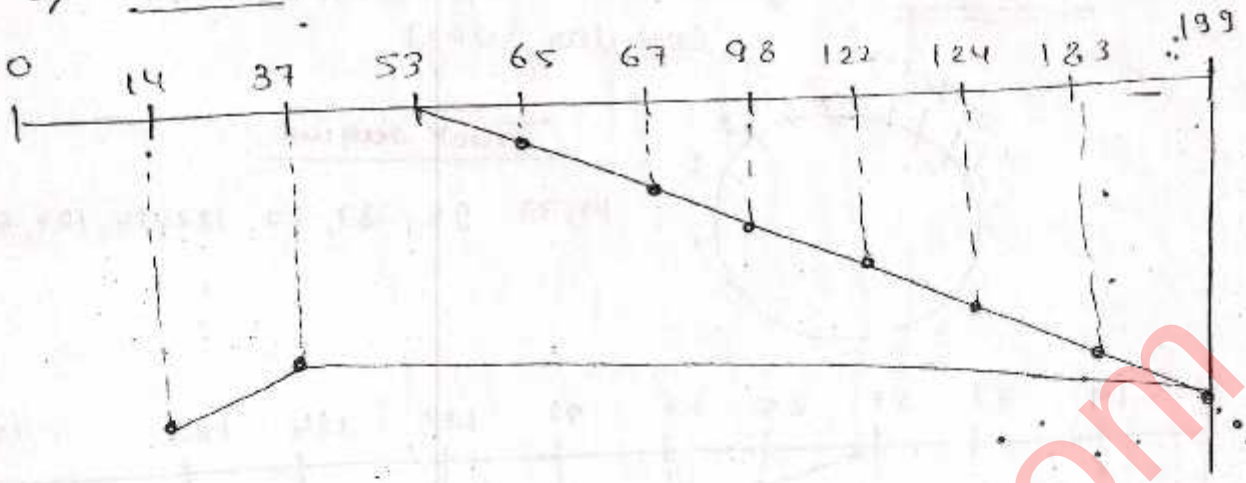


no. of track movement = (50-34) + (34-20) + (20-19) + (19-15) + (15-10) + (10-7) + (7-6) + (6-4) + (4-2) + (73-2)

= 16 + 14 + 1 + 4 + 5 + 3 + 1 + 2 + 2 + 71

= 30 + 18 + 71 = 119 Ans

3) SCAN (elevator):-



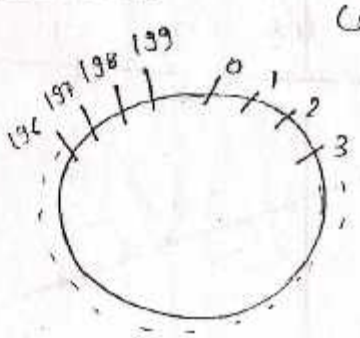
no. of track movement = (65-53) + (67-65) + (98-67) + (122-98)
+ (124-122) + (183-124) + (199-183) + (199-37)
+ (37-14)

= 12 + 2 + 31 + 24 + 2 + 59 + 16 + 162 + 23

= 14 + 55 + 61 + 201 = 331

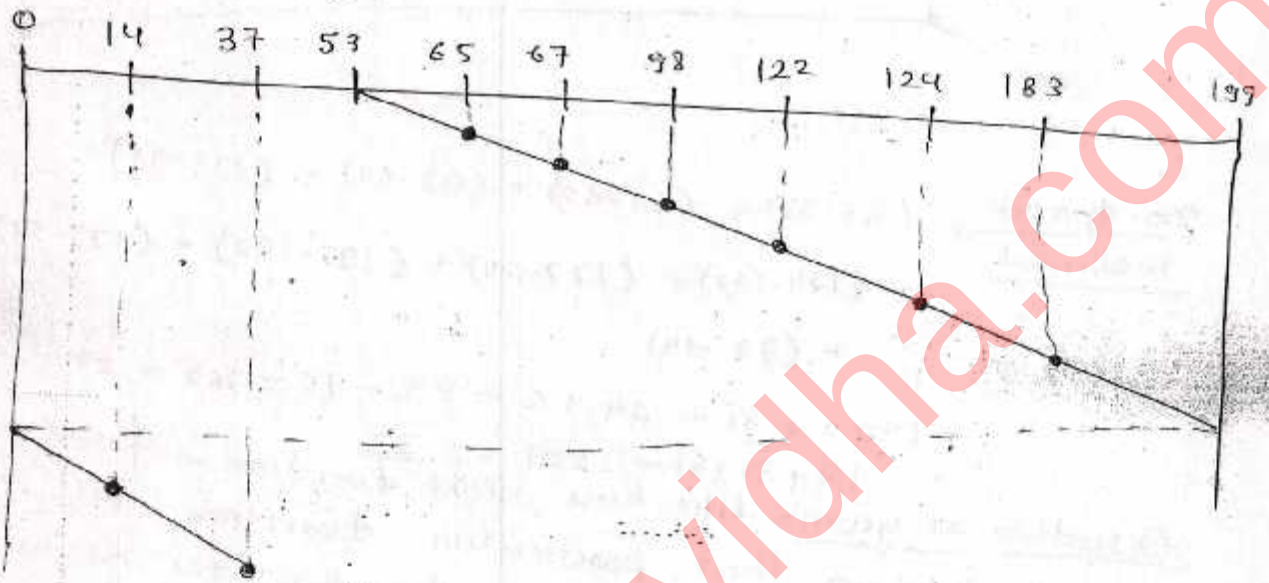
Direction is given - like here right direction : first move towards that particular direction till end & move in opposite direction for required track. (not upto 0).

4) C-SCAN (Tracks are assumed to be in circular way)



Track request

14, 37 98, 183, 37, 122, 14, 124, 65, 67



#of track movement = (65-53) + (67-65) + (98-67)
+ (122-98) + (124-22) + (183-124)
+ (199-183) + (199-0) + (14-0)
+ (37-14)

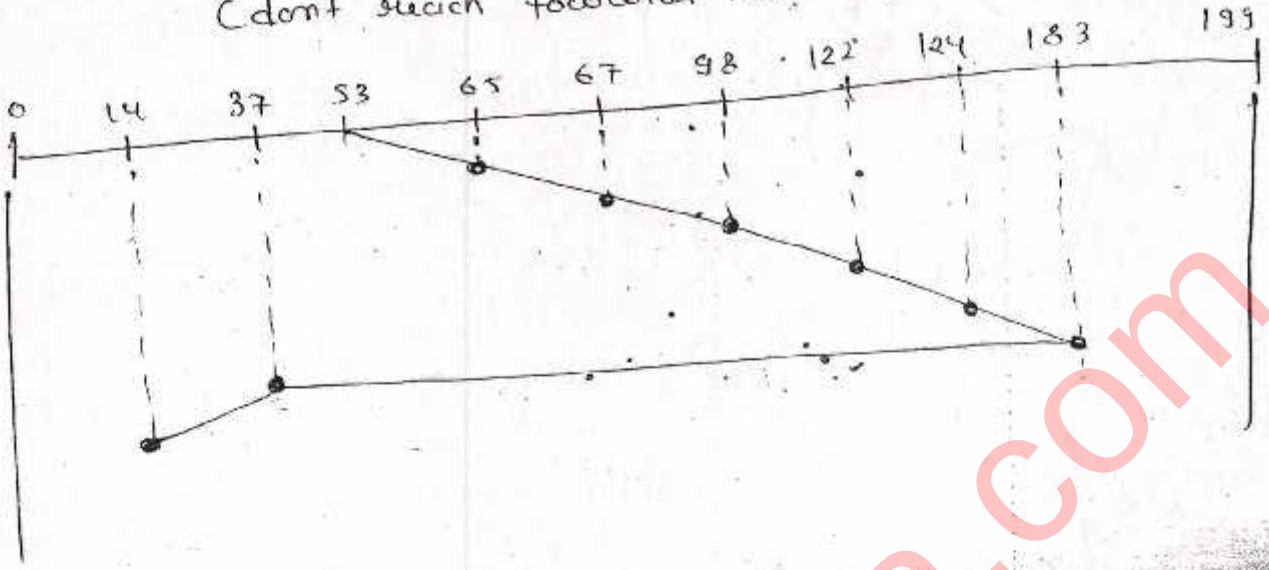
= 12 + 2 + 31 + 24 + 2 + 59 + 16 + 199 + 14 + 23

= 14 + 55 + 61 + 215 + 37

= 69 + 276 + 37

= 382 Ans

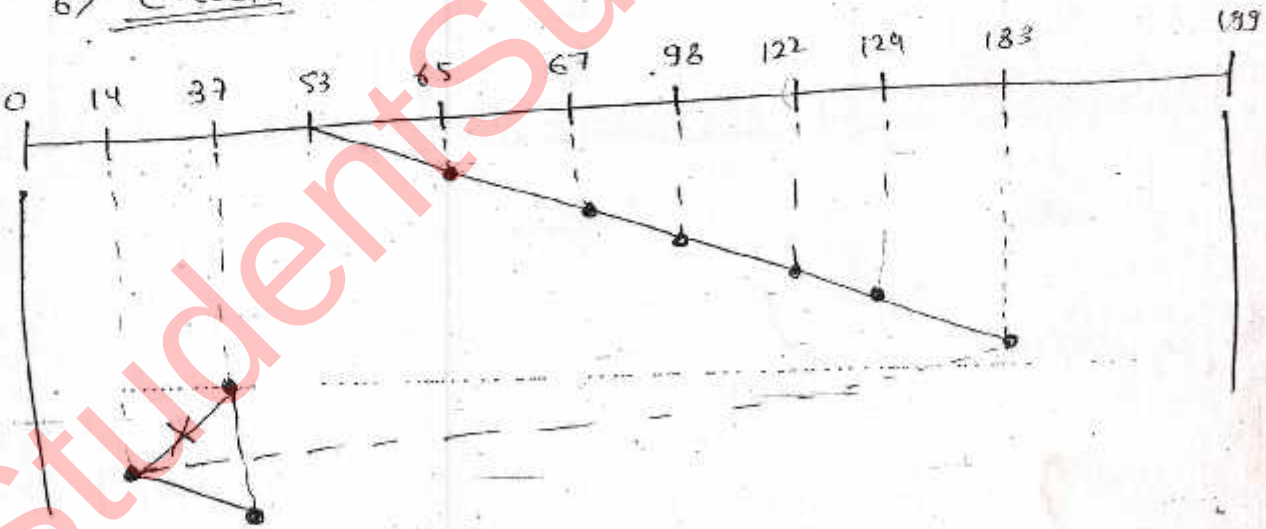
5) Look - (look for the last pending request in the direction of read/write header (don't reach towards both end)



no. of track movement = 12 + 2 + 31 + 24 + 2 + 59 + 146 + 23

= 299 Ans

6) C-look -



no of track movement = 12 + 2 + 31 + 24 + 2 + 59 + 169 + 23

= 14 + 55 + 61 + 192

= 69 + 253

= 322 Ans.