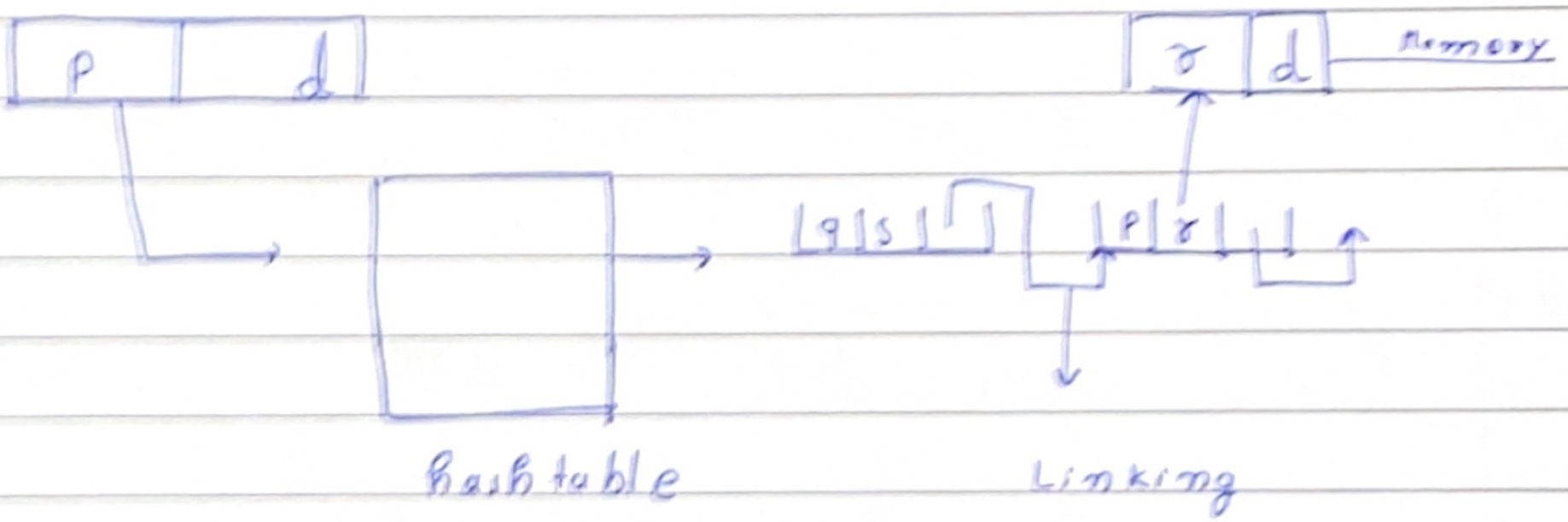
Hashed Page Table

Isme Jo page table hoti hai its in the form of hashing Yaani Jo page table hai

hashing kaise



Inverted Page Table

har ek process ka pagetable Alag hona chahiye jo page table Dhoondo Phir Page dhoondo Dikat hai

page table Size => No° of Entries x Entry

No. of pages = $\frac{2^{46}}{2^{13}}$ = 2^{33} => No. of pages

=> $2^{33} \times 2^{2} = \underline{\underline{2^{35}}}$

2^{35} Bytes => 2^{13} B (Frame Size)

1 Level

No° of frames Required = $\frac{2^{35}}{2^{13}} = 2^{22}$

Outer level table Size = $2^{22} \times 2^{2} = 2^{24}$

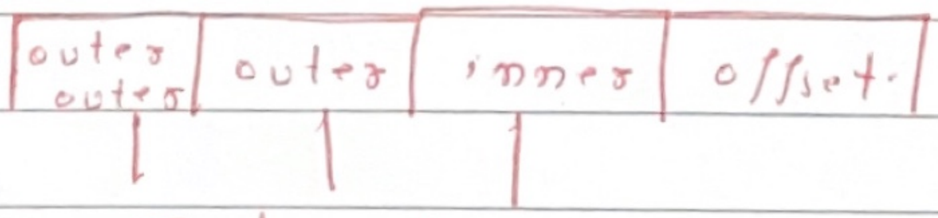
$2^{24} > 2^{13}$

2 Level

outer level table Size = 2^{24}

No. of frames req = $2^{24} / 2^{13} = 2^{11}$

$\underline{2^{11} \times 2^{2}}$ = Size = $\underline{\underline{2^{13}}} = 2^{13}$



3 levels.

Global PoC

Frame no°	Page no°	Process ID	
0	P_0	P_1	
1	P_1	P_2	→ P_1 but Diff Pid
2	P_2	P_1	
3	P_1	P_3	— P_1 "
4	P_3	P_2	
5	P_2	P_3	

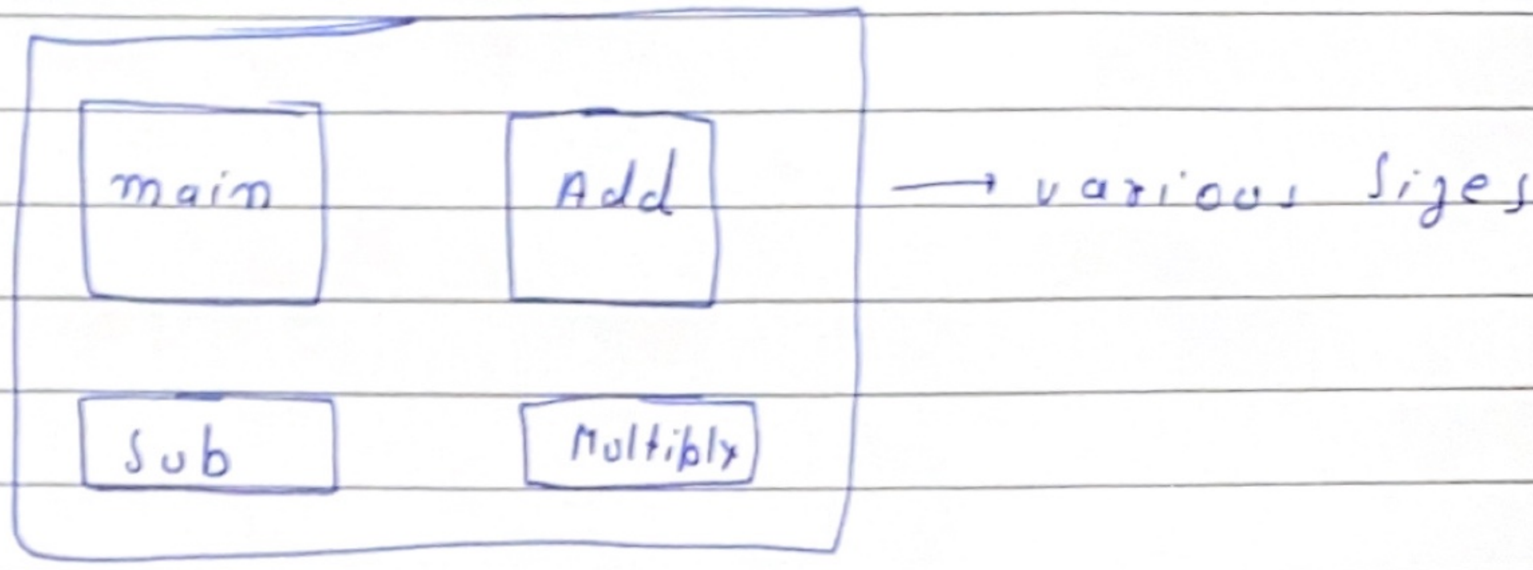
Searching Time is more as compared to Normal

Segmentation

In Segmentation hamare program ko segments mein divide kara jaata hai

Yaani jo Segments hote hai vo variable Size ke hote hai

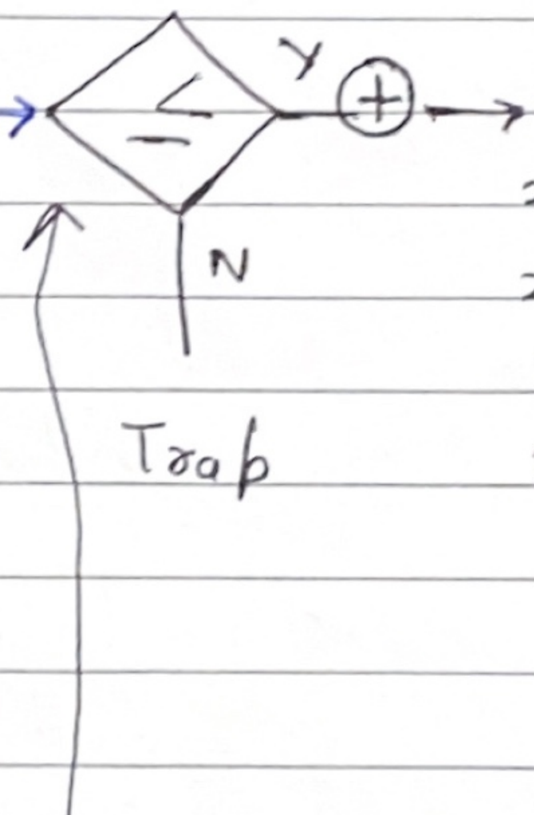
Add, main, → variable size posible hai Not in paging.

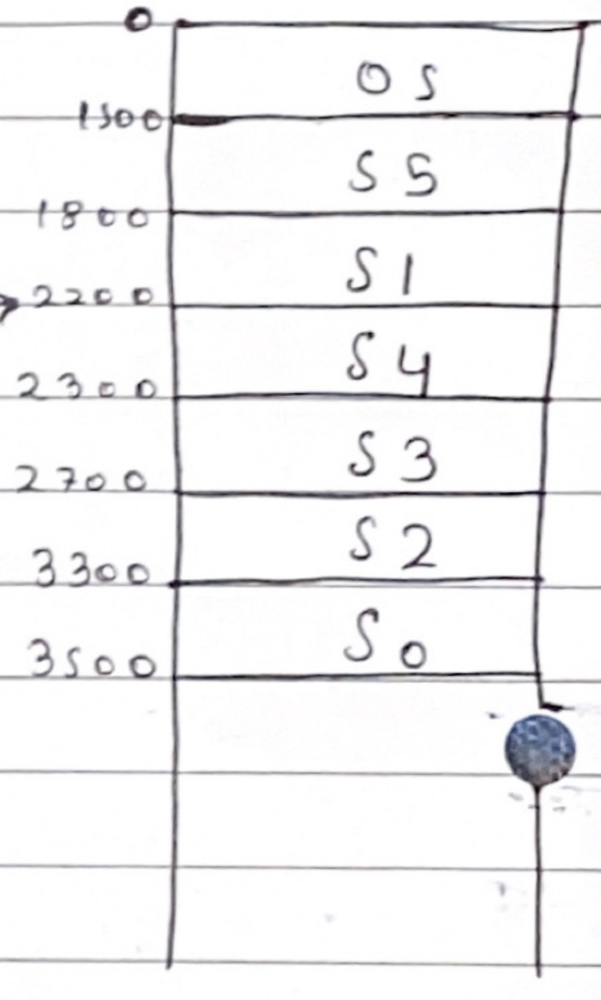


Segment table

	BA	Size
0	3300	200
1	1800	400
2	2700	600
3	2300	400
4	2200	100
5	1500	300

Segment table





CPU => | Segment no° | Segment offset |

↓

Ye value should always be less than size.

S1 → 400 bytes
off — 500 byte → Trap

CLASSMATE
Date :
Page :

Non Contagious

Paging

Segmentation

- Ques

Segment no°	Base	Length
0	1219	700
1	2300	14
2	90	100
3	1327	580
4	1952	96

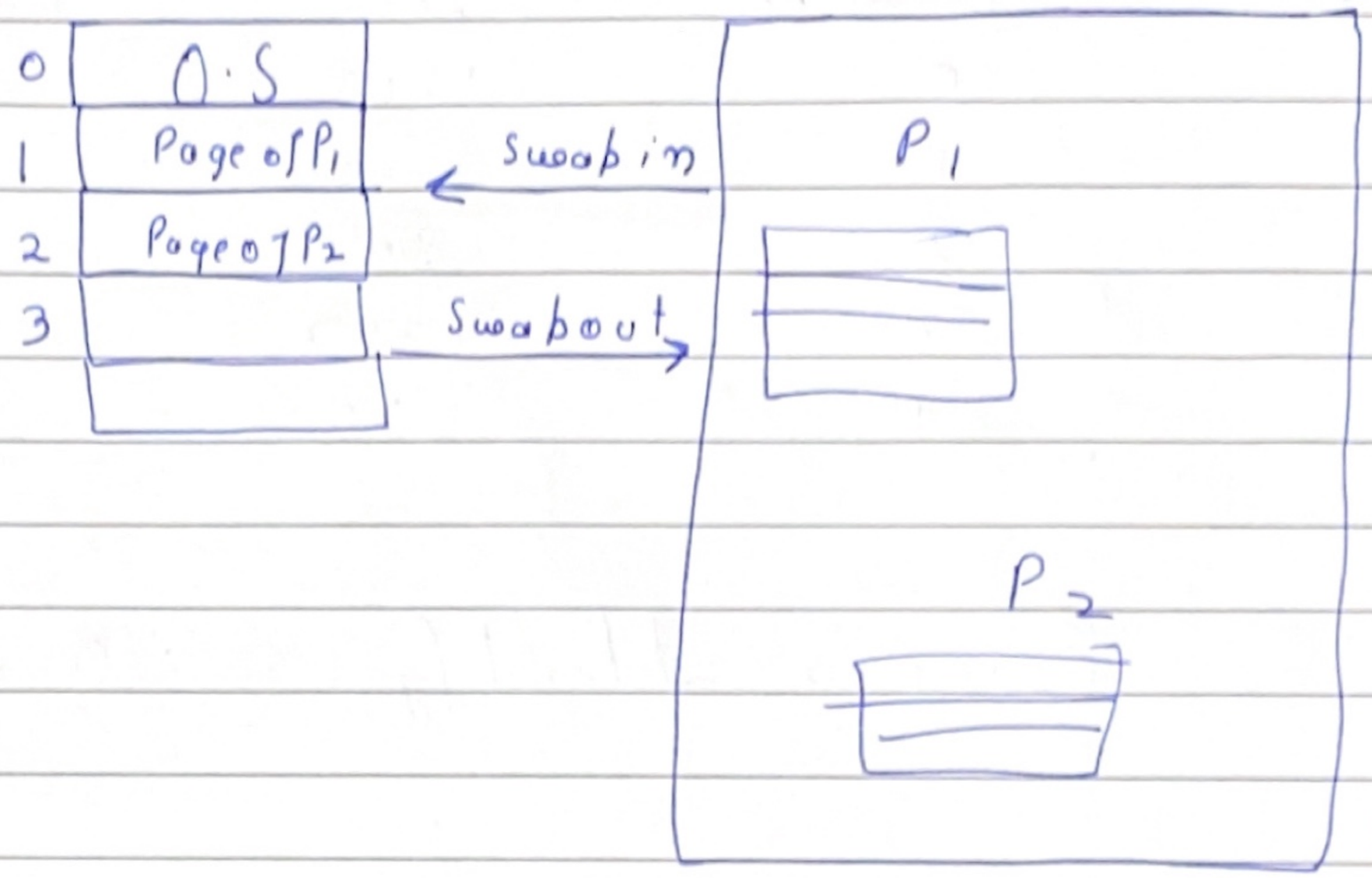
Error => Identify

0 , 430
1 , 11
2 , 100 =
3 , 425
4 , 95

Error 2 , 100 => [0 to 99]
↳ offset value

Virtual Memory

It gives an illusion to the user program that the size of process bigger than main memory can also Execute



kuch kuch part ate hai Swap in & Swap Out. hote hai process ka.

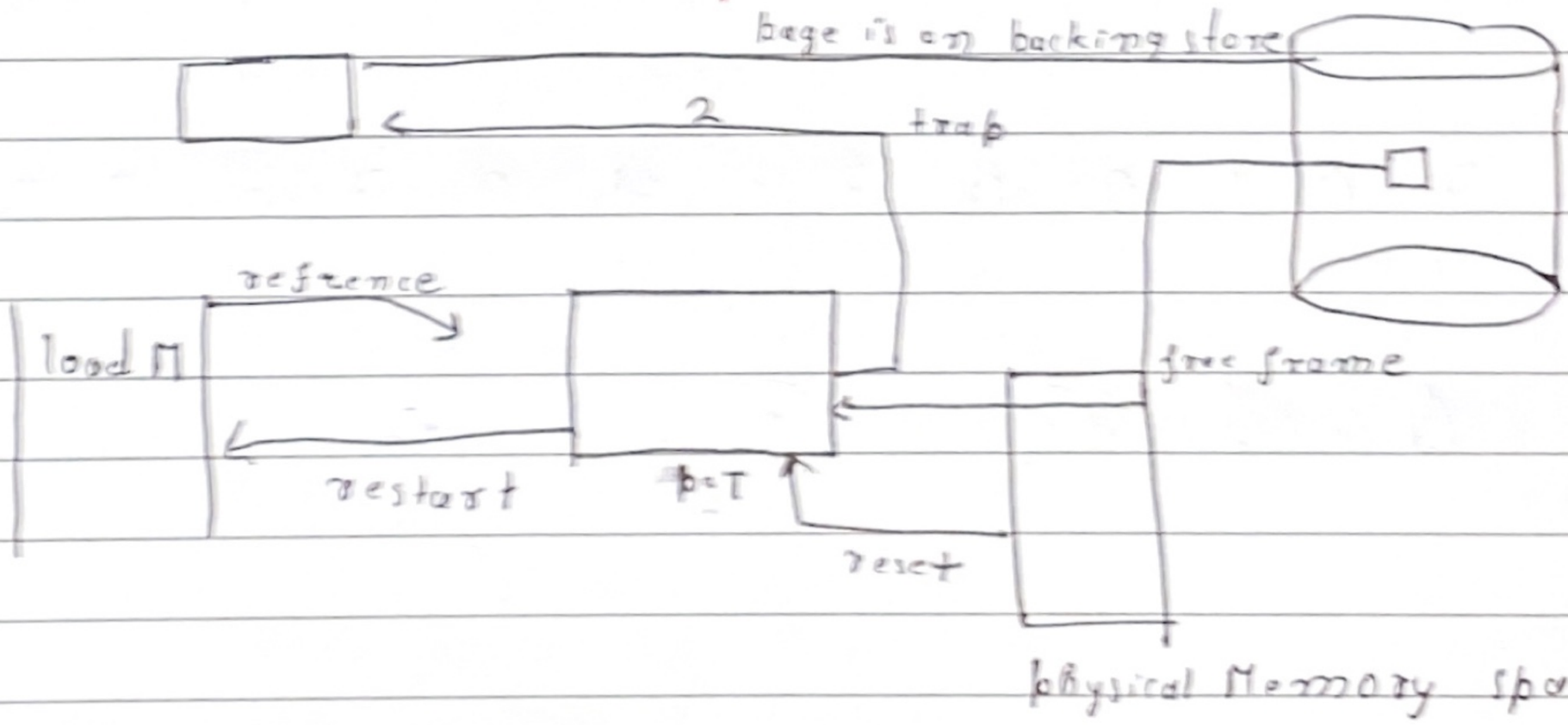
CPU → Address Generate karega
maanle usney page chahiye
1 but vo Invalid hai M.M
mai hai

Page fault hoga → Matlab RAM
mein include karo.

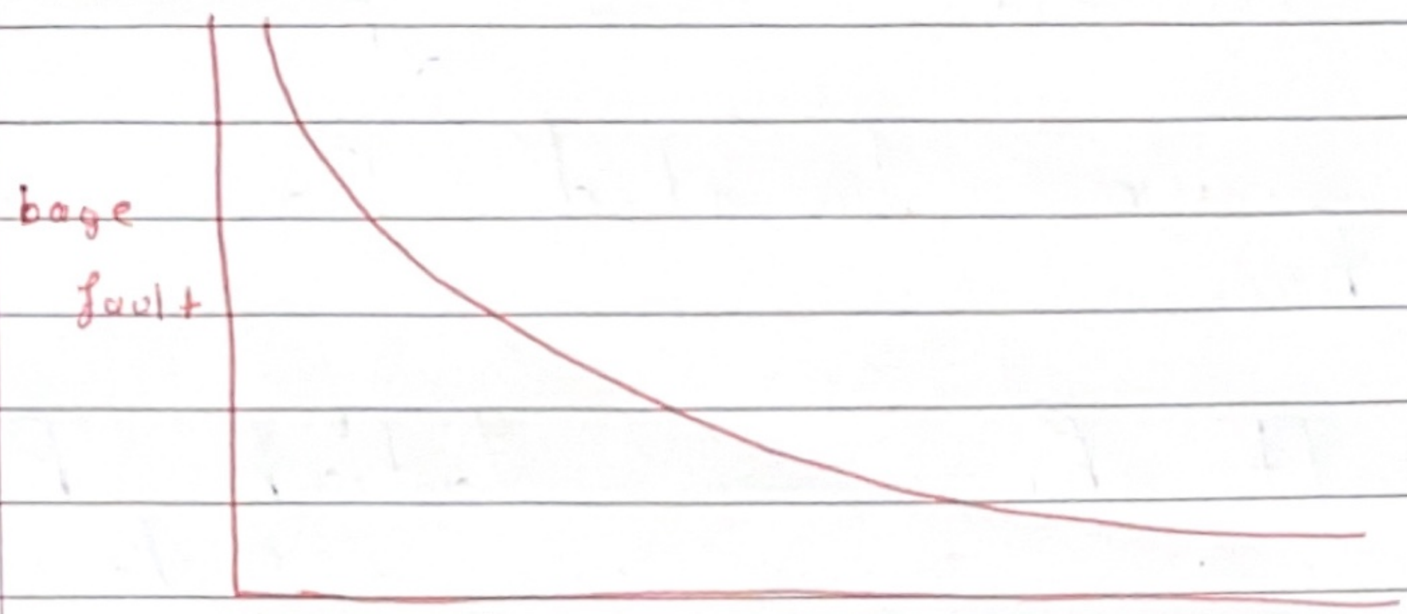
Demand paging

In this CPU jab koi aise page
demand kare jo available na
ho M.M mein tab vo Virtual
Space se aata hai M.M mein

Steps



Graph of Page fault Vs No° of frame



Page fault ke liye Algorithm

a) FIFO (First In First Out):

Adding more frames can cause more page faults ⟶ Beladay's Anomaly.

⟶ 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

⟶ Que Size = 3.

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1

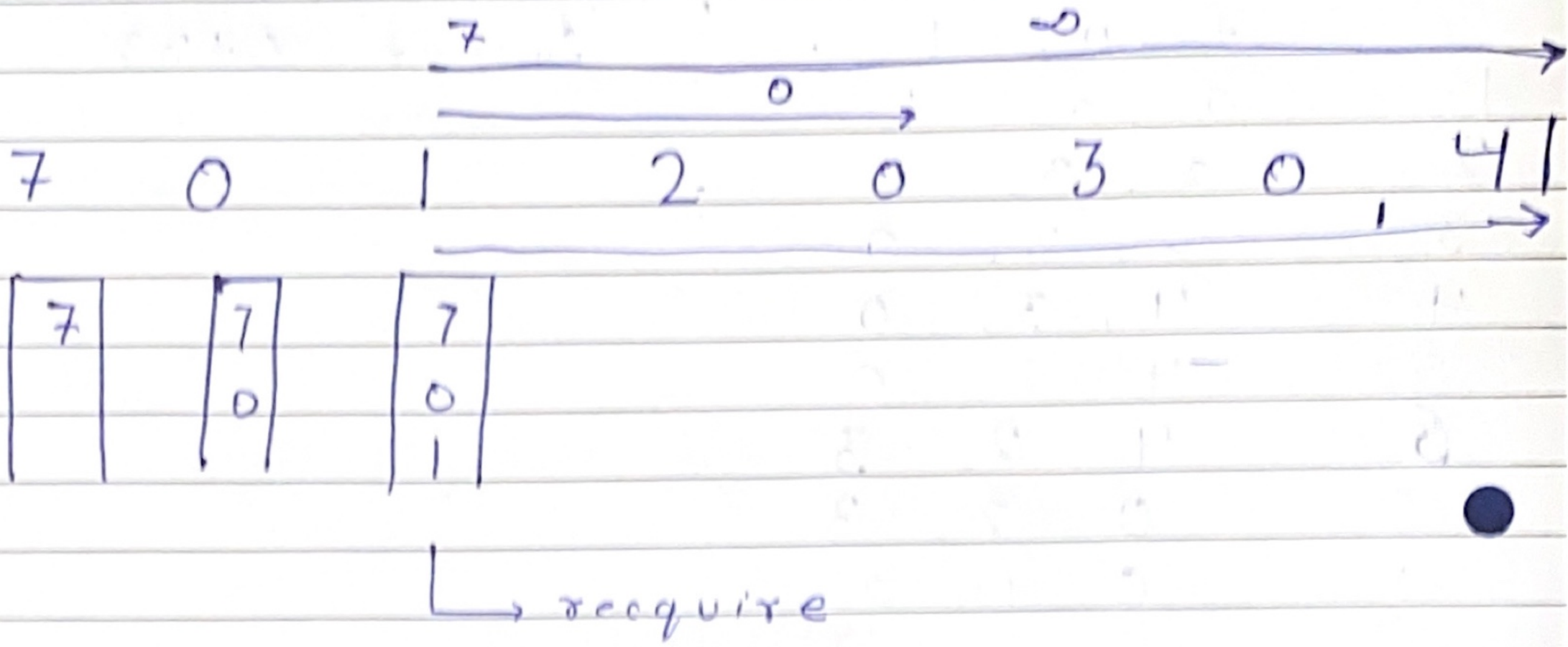
7, 0, 1 ⟶ Insert (+3)

1	2, 0, 1
2	2, 3, 1
3	2, 3, 0
4	4, 3, 0
5	4, 2, 0
6	4, 2, 3
7	0, 2, 3
8	0, 1, 3
9	0, 1, 2
10	7, 1, 2
11	7, 0, 2
12	7, 0, 1

15 page faults.

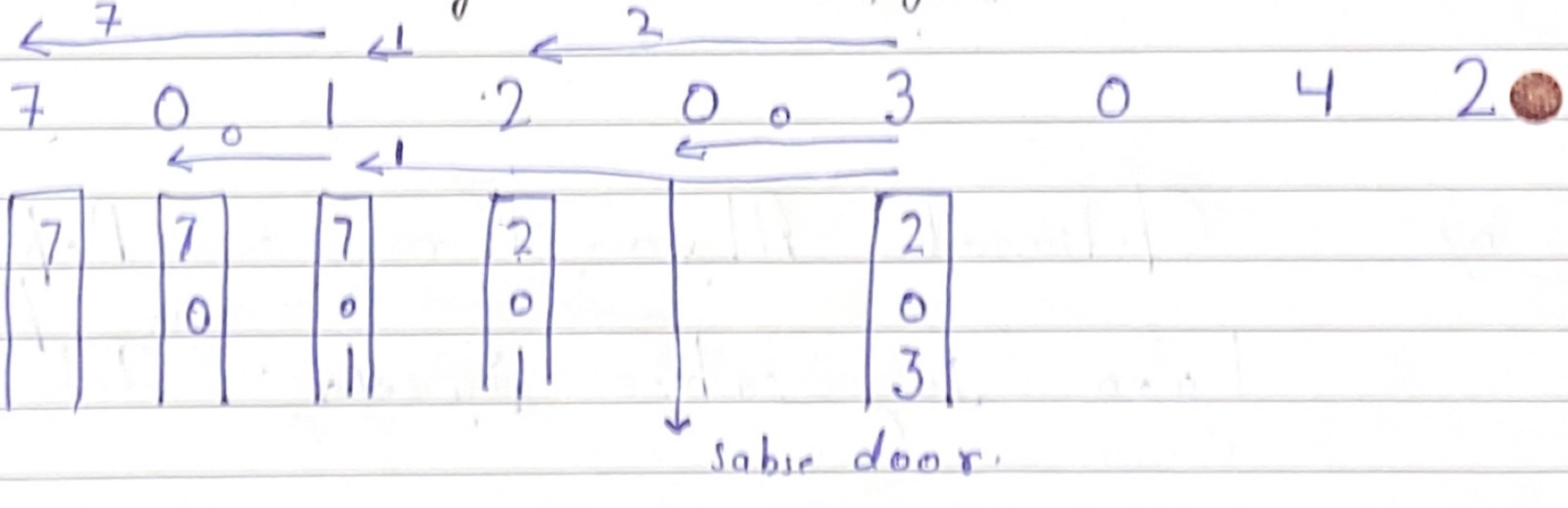
b> Optimal Algorithm => Replace uss page karo jo sabse jyada late use mein hai.

Ex ⟶



7 ka use sabse baad mein hai
toh. replace 7

c) Least Recently Used Algorithm



Allocation of Frames:

FIXED — Equally no° of frames divide kardo

or

Allocate Acc to Size of process.

S_i = Size of process P_i

$\sum s_i = S$

m = total no° of frame

a_i = allocation for P_i

$$P_i = \frac{s_i}{S} \times m$$

Global Vs Local Allocation

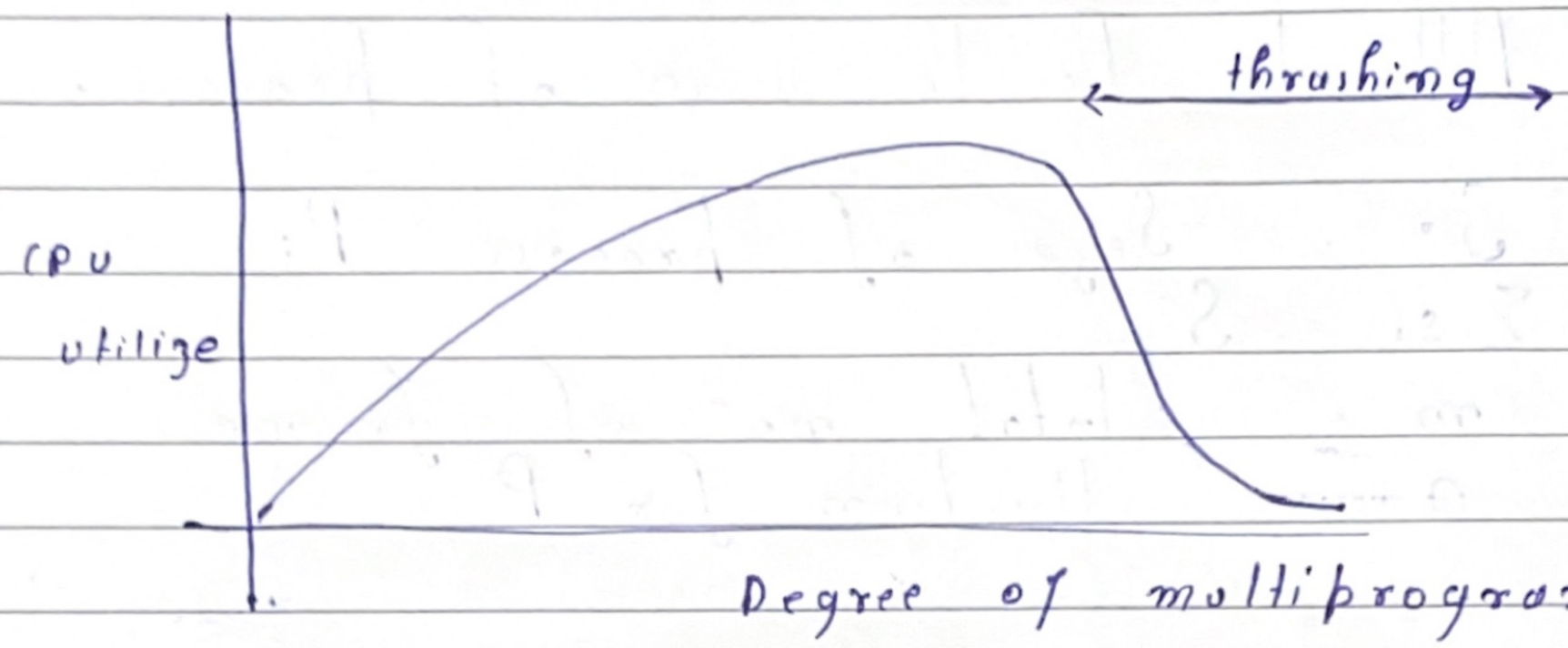
Global => process kisi bhi process ka replace kar sakti hai

Local => process Replace karega sirf apni allocated frames mein.

Thrashing

A process doesnot have Enough pages toh pagefault Jyada ho jaate hai.

Itne Jyada ho Jaye ki koi process execute hee na ho paye



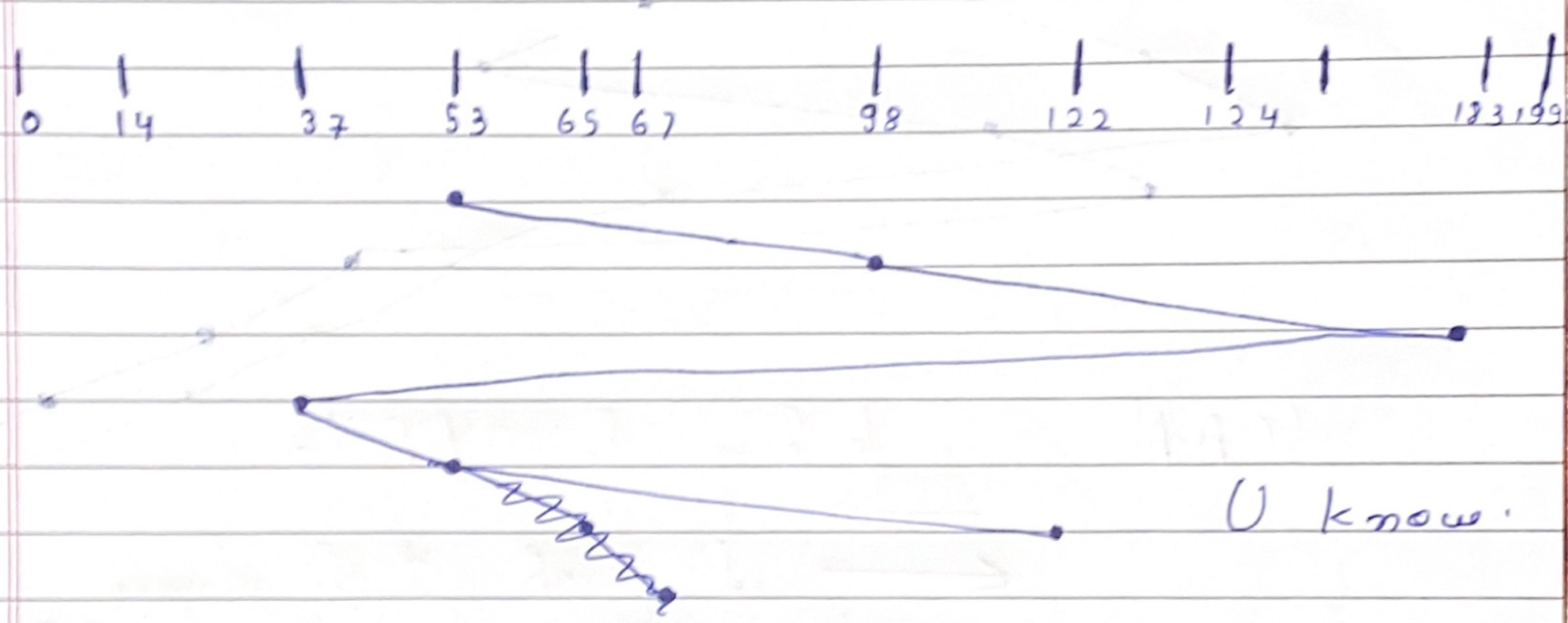
No. of process ↑↑ frames per process
↓↓ ⟶ page fault ↑↑↑.

Disk Scheduling

FCFS

Jaise Order mein aa rahi hai Vaise hee.

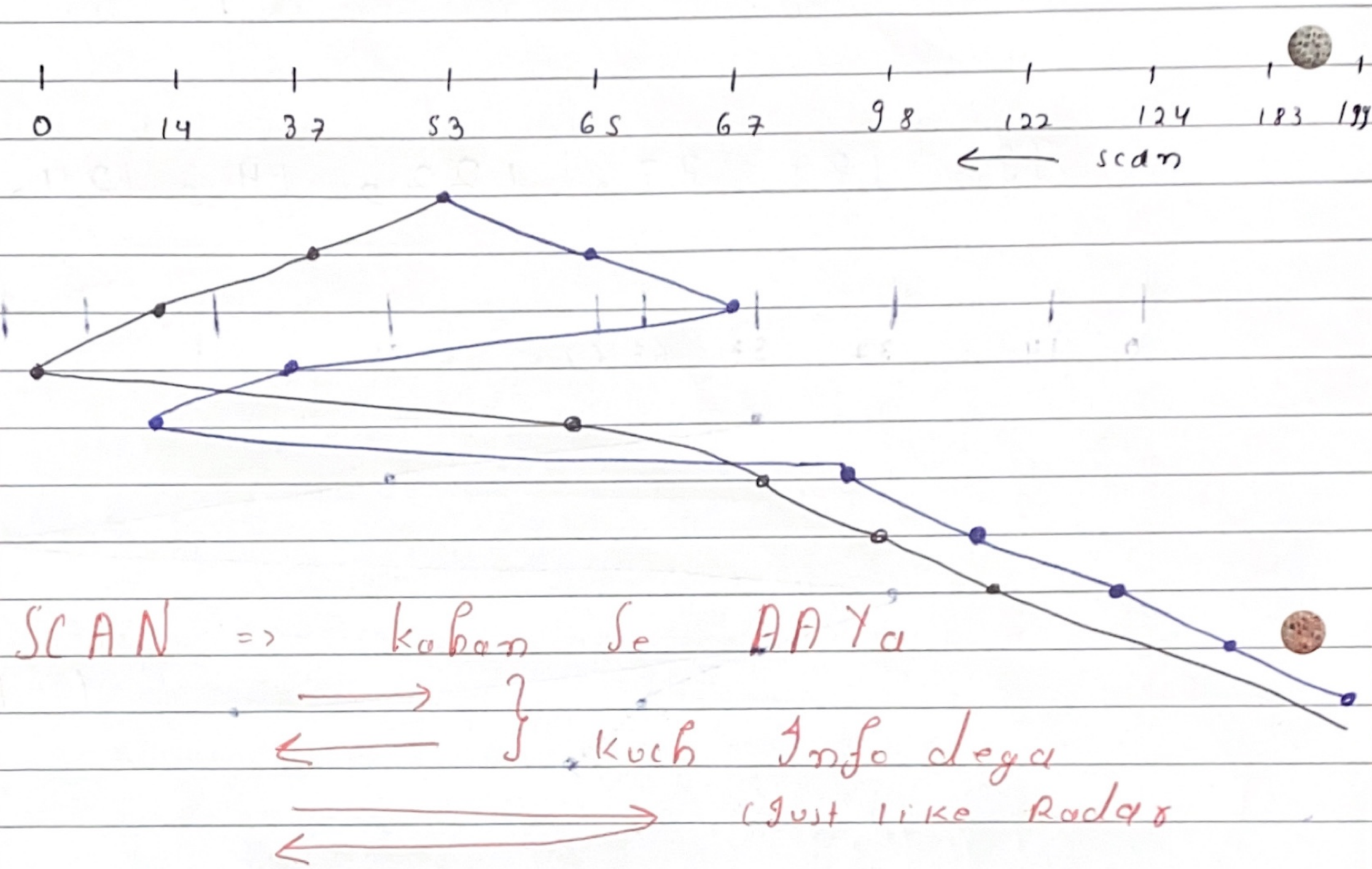
98, 183, 37, 122, 14, 124, 65, 67



SSTF

Jo sabse paas hai usko pehle dekho

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

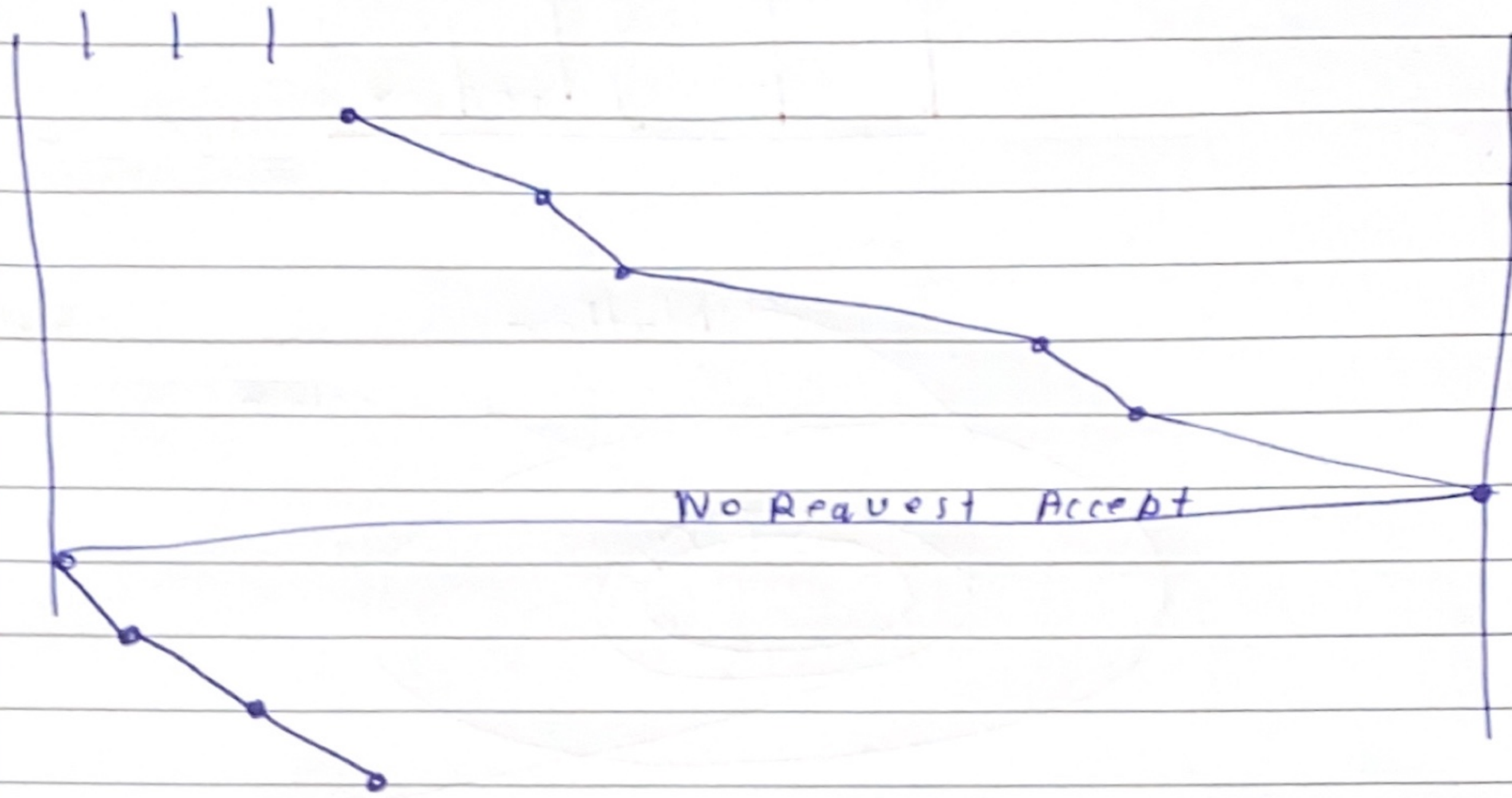


SCAN => kahan se AAya

⟶ } kuch Info dega

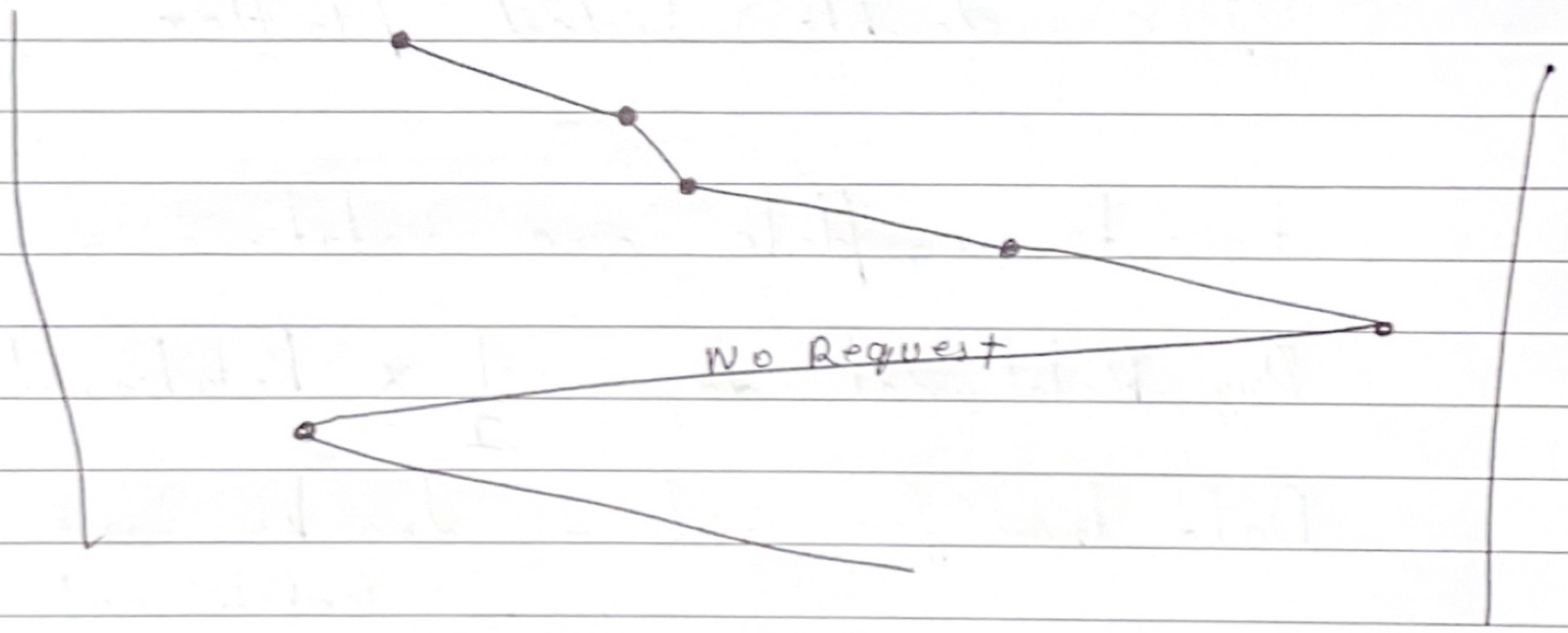
⟵ } (Just like Radar

CSCAN => Jaise hee End pe pahuchega Vis Start pe AAyega without koi Request fulfill kare then Again

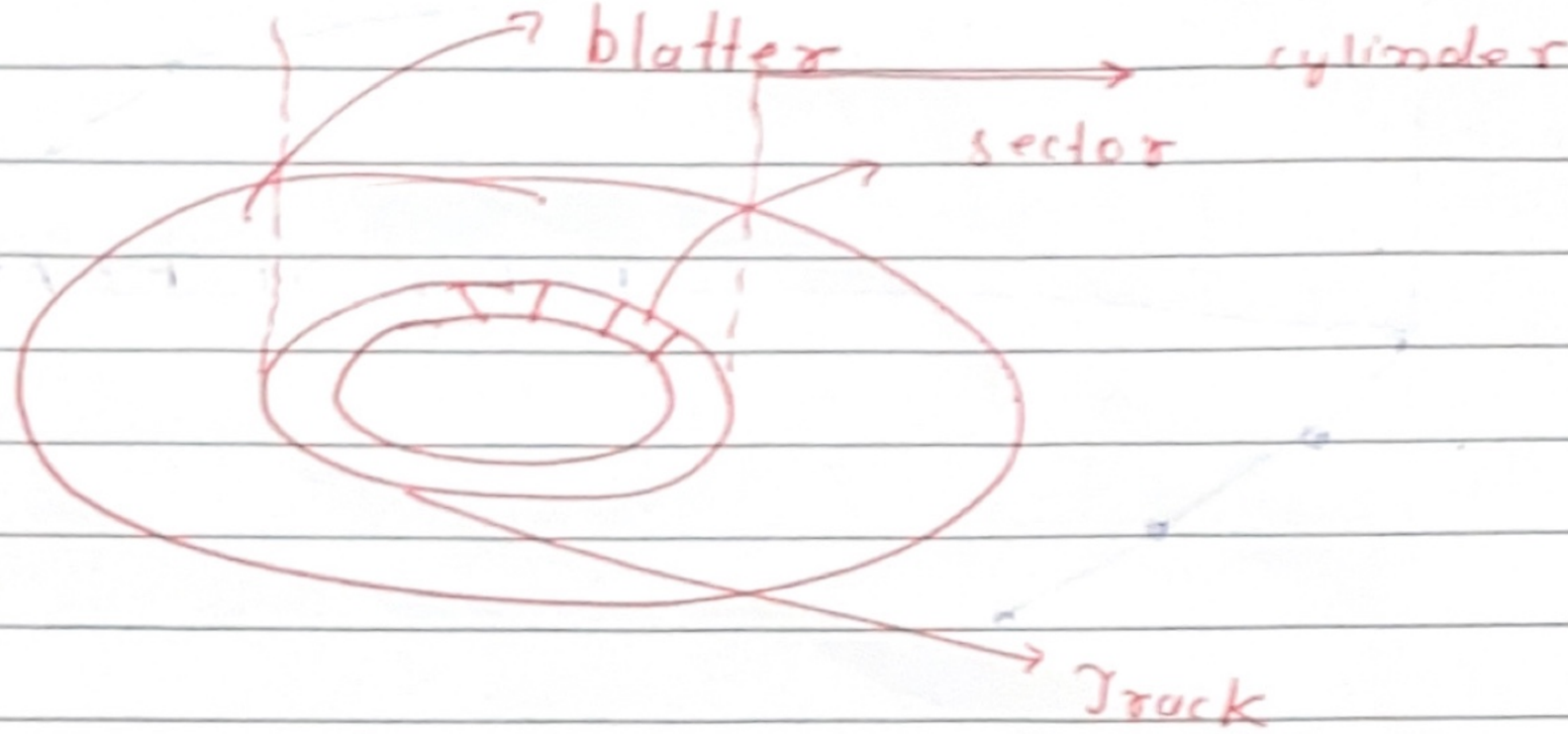


C Look

limit fo hoti hai yo hoti hai request ke according



Disk Structure



double sided platter yaani upar neeche sector hai.

Sector Size = 512 Bytes
50 Sector per track
2000 tracks per surface
five double sided platter
} Given.

time to complete one rotation = Rotational delay.

Avg Rotational day = $\frac{1}{2}$ x Rotational delay

Data transfer rate = $\frac{\text{Tracks Size}}{\text{Rotational delay}}$

Dead Lock

Har eh process resource ko => request
use
release

4 Conditions of Dead Lock

- Mutual Exclusion => Eh process hee resource ko eh samay be use karegi
- Hold and wait => A process holding alleast one resource & waiting to acquire additional resource holding
- No Preemption => Yani eh process resource tabhi release karegi jab uska task ho jayega
- Circular Wait => Set of Process hold & wait eh circular order mein kar rahi hogi

$P_i \longrightarrow R_j$ => P_i requesting R_j

$R_j \longrightarrow P_i$ = R_j is holded by P_i

Ensure that System Never a DeadLock

→ DeadLock Prevention => Deadlock mein jaao then recover karo

→ DeadLock Avoidence => Pretend Deadlock never occur in system. preffered method in UNIX

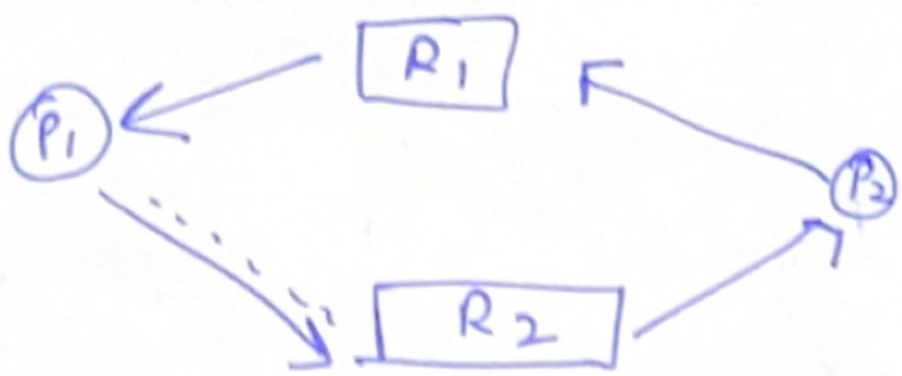
Safe State

Jab koi process resource ki demand kare toh os ye decide kare agar allocate kardiya toh System state Safe mein rahega

Toh agar ek Aisa sequence ho $<P_1 \cdots P_n>$ of All process that sabki need satisfy ho jaaye . Safe state.

Resource ⟶ Single Instance

Graph



Multiple Instances of Same Algo

Allocate			Max			Available		
A	B	C	A	B	C	A	B	C
0	1	0	7	5	3	3	3	2
2	0	0	3	2	2	2	0	0
3	0	2	9	0	2	5	3	2
2	1	1	2	2	2	2	1	1
0	0	2	4	3	3	7	4	3
						0	0	2
						7	4	5
						0	1	0
						7	5	5
						3	0	2
						10	5	7

Need

P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

$\{P_1, P_3, P_4, P_0, P_2\}$

Bankers Algo

Memory Management

Contiguous => process request memory & a block is assigned acc to its Request

30 KB ⟶ hai but continous Nahi hai ⟶ ✡.

External
Internal
] ⟶ Fragmentation Theory.

Fixed ⟶ partitioning
Variable ⟶ partioning
] ⟶ Theory

Theory by Yourself