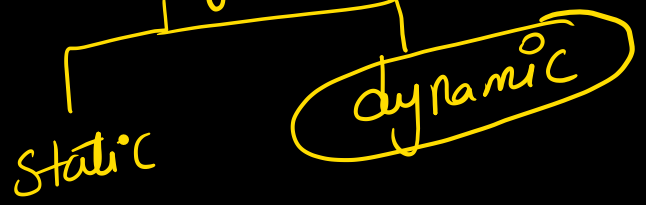
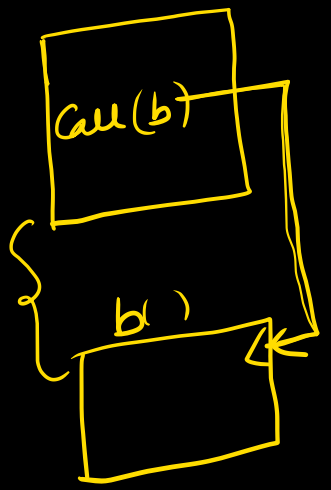
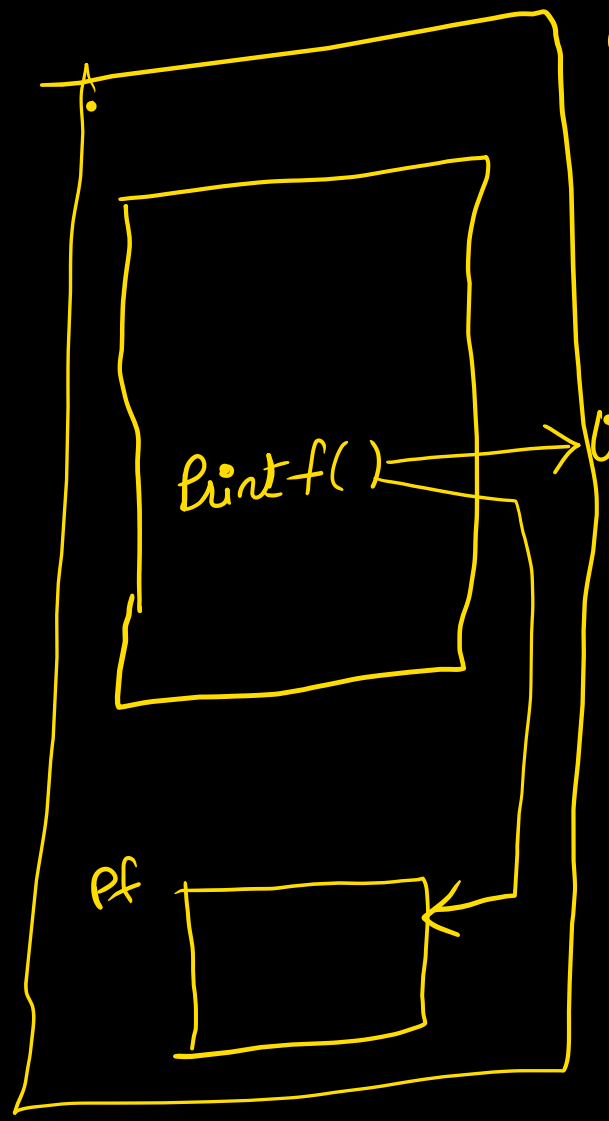
Linking.

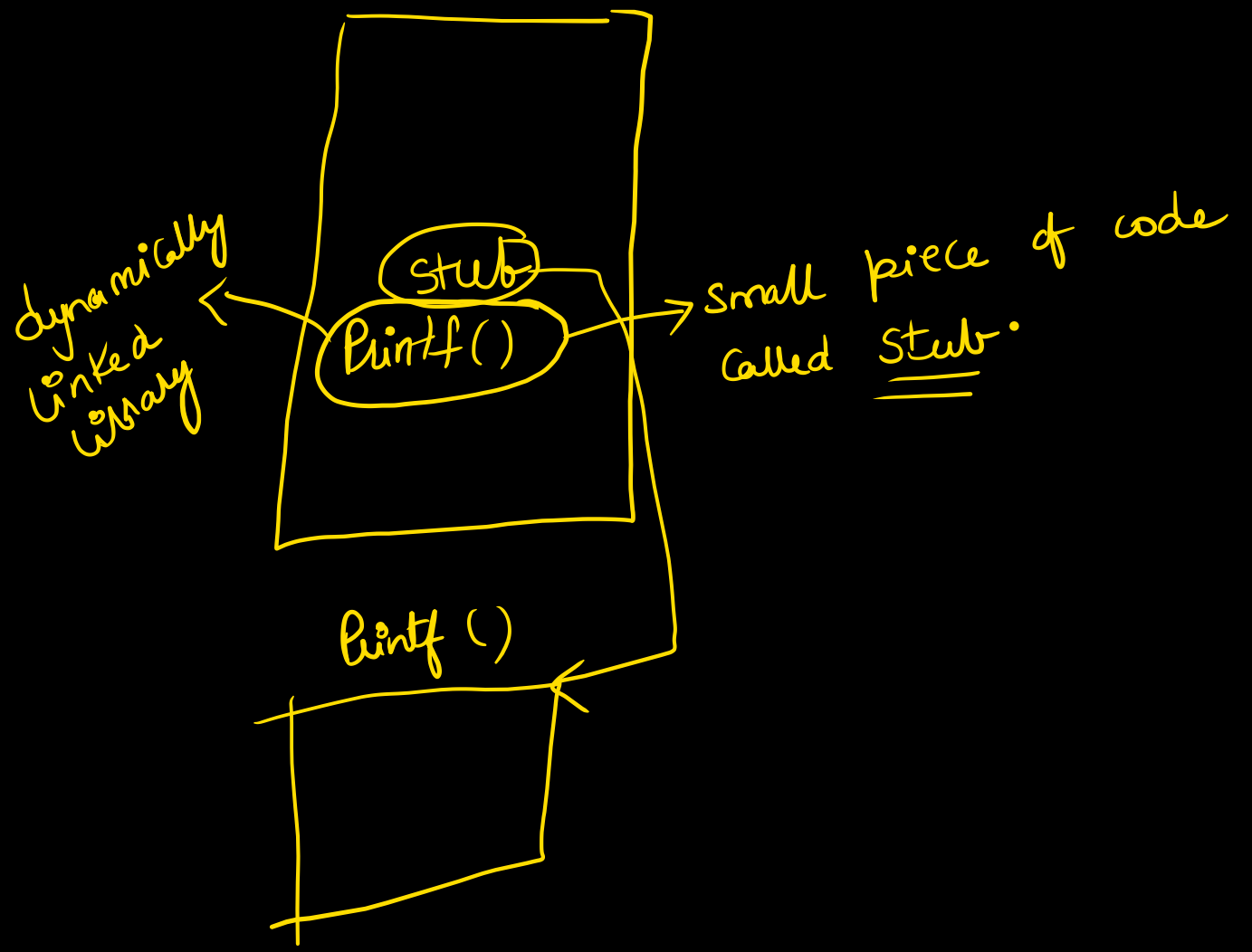






linked at Run time

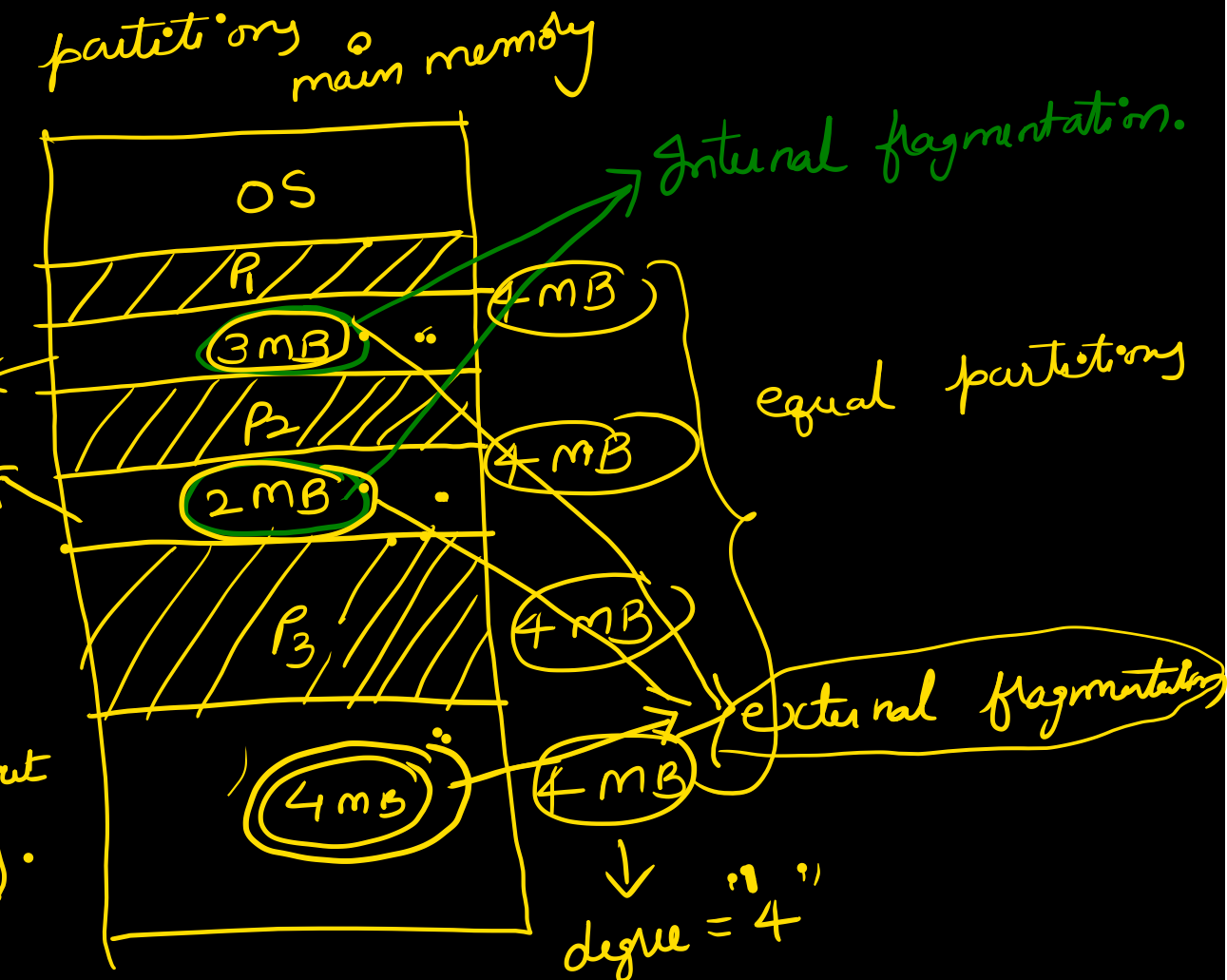
dynamic.



small piece of code called stub.

memory management:

1) Fixed partitioning with equal sized partitions



$P_1 \rightarrow$ 1MB.

Rule: only 1 process/partition.

$P_2 \rightarrow$ 2 MB

$P_3 \rightarrow$ 4 MB.

$P_4 \rightarrow$ 9 MB $\rightarrow$ X

4 + 3 + 2 $\rightarrow$ available but process has to be contiguous.

Ps > 4m X

Cannot be given to any process.

Disadv:

→ Process size is limited

→ Internal fragmentation

→ External fragmentation

→ Degree of multiprogramming is fixed

↓

no of processes that can be present in 'MM'

→ Fixed partitioning with unequal sized partitions:

$P_1 \rightarrow$ 0.5 MB

✓ $P_{10} \rightarrow$ 6 MB

Disadv: (i) Size of Bio is limited.

(ii) IF↓

(iii) EF✓

(iv) degree of MP → fixed

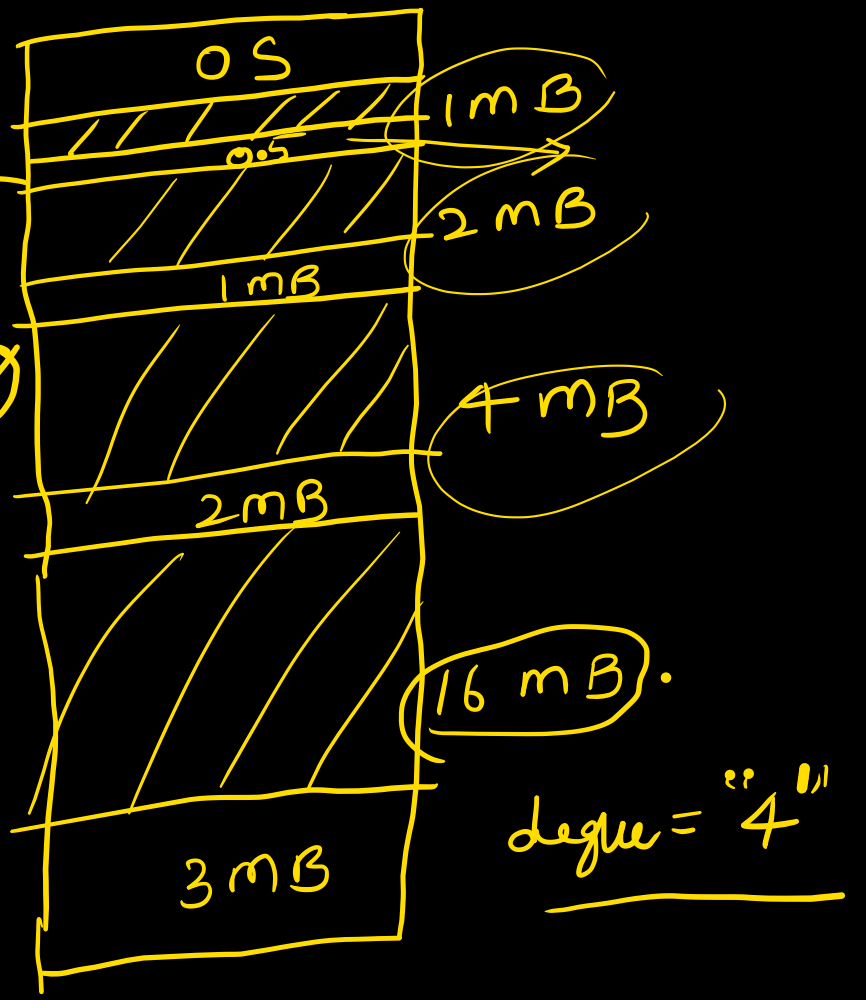
IF

IF↓

EF✓

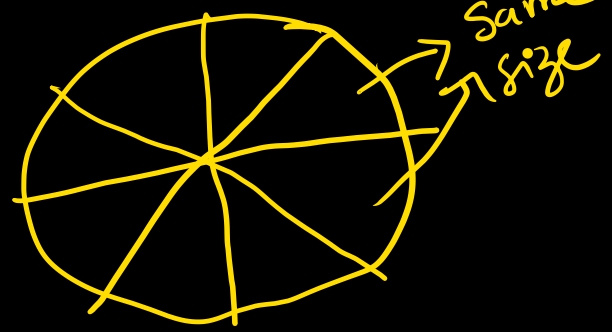
6.5 MB

16 MB

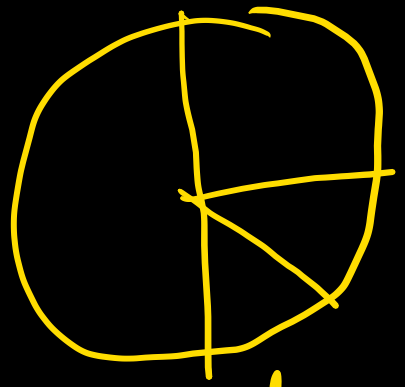


degree = "4"

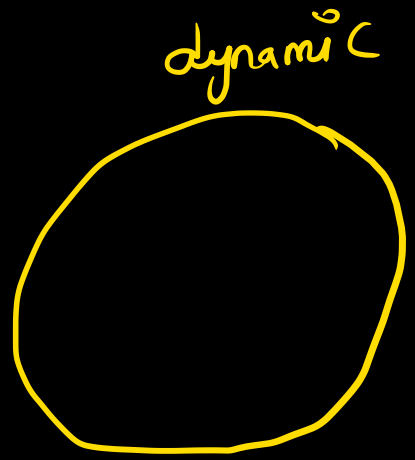
Dynamic partitioning:



Equal



unequal.



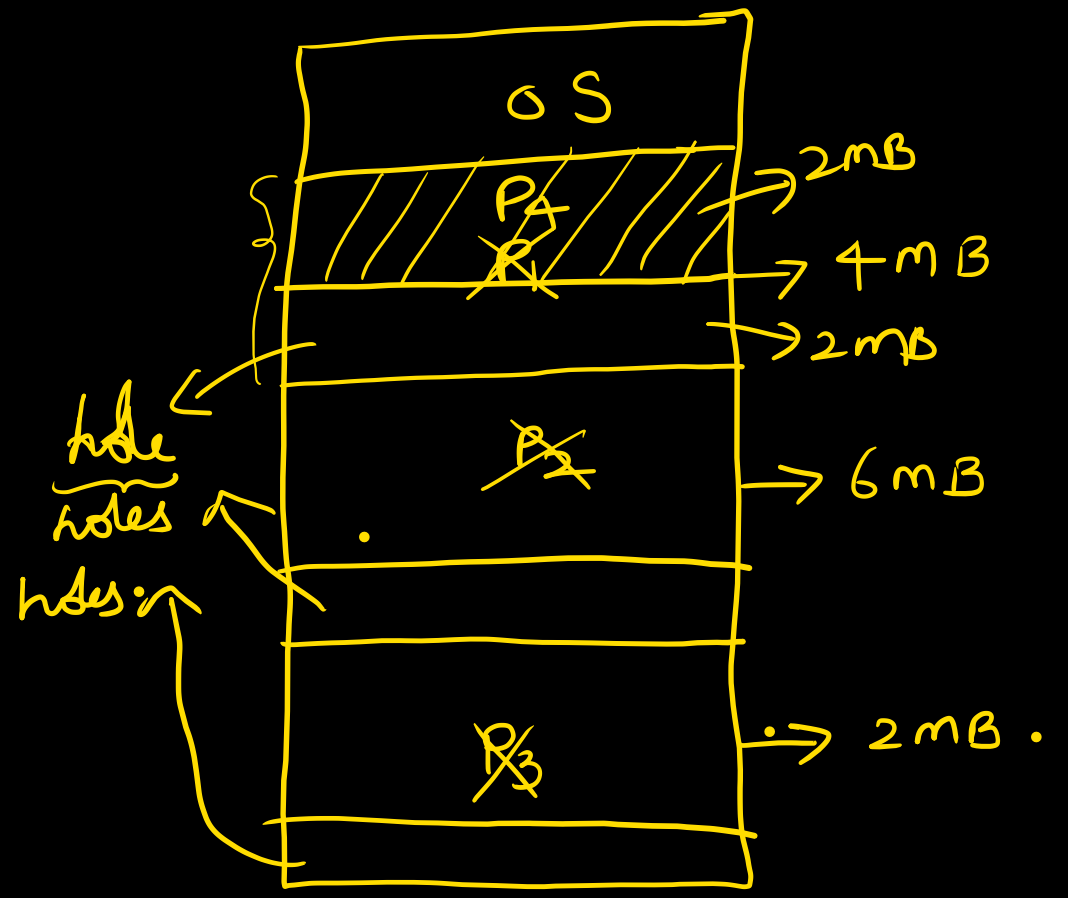
NO cutting the cake.

$P_1 \rightarrow 4$ mB

$P_2 \rightarrow 6$ mB

$P_3 \rightarrow 2$ mB

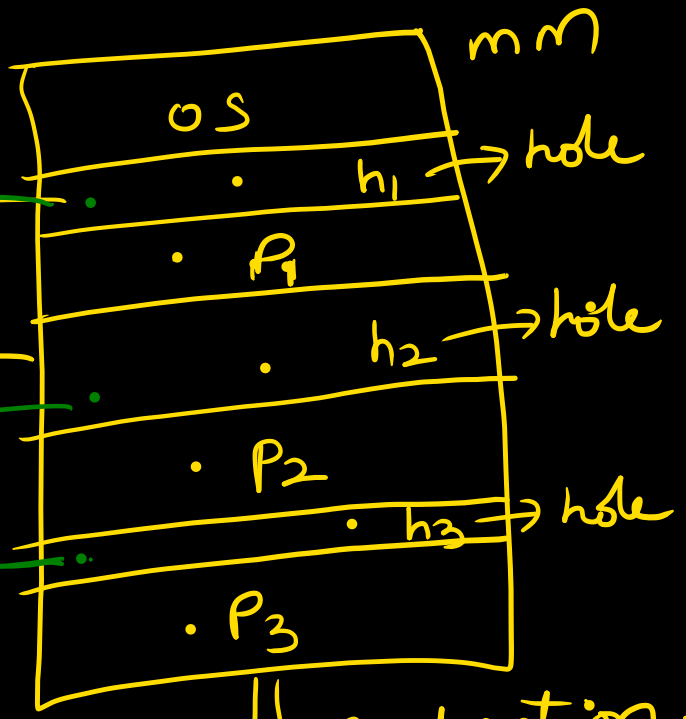
$P_4 \rightarrow 2$ mB.

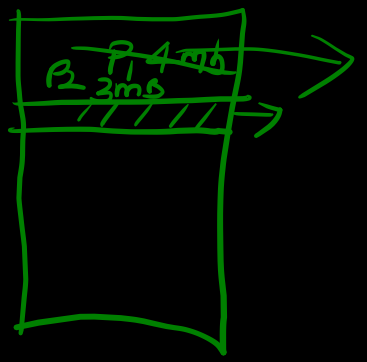


NO IF X

External fragmentation

→ because of holes.

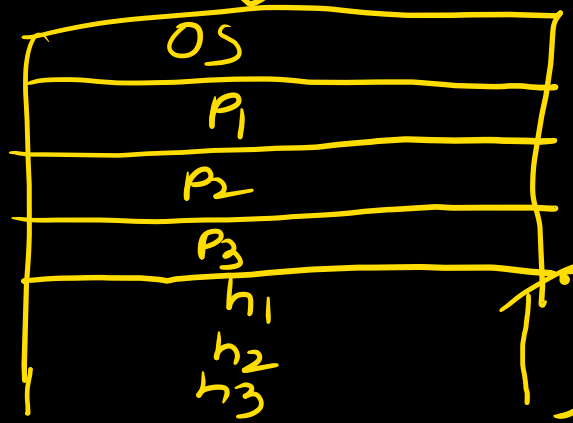




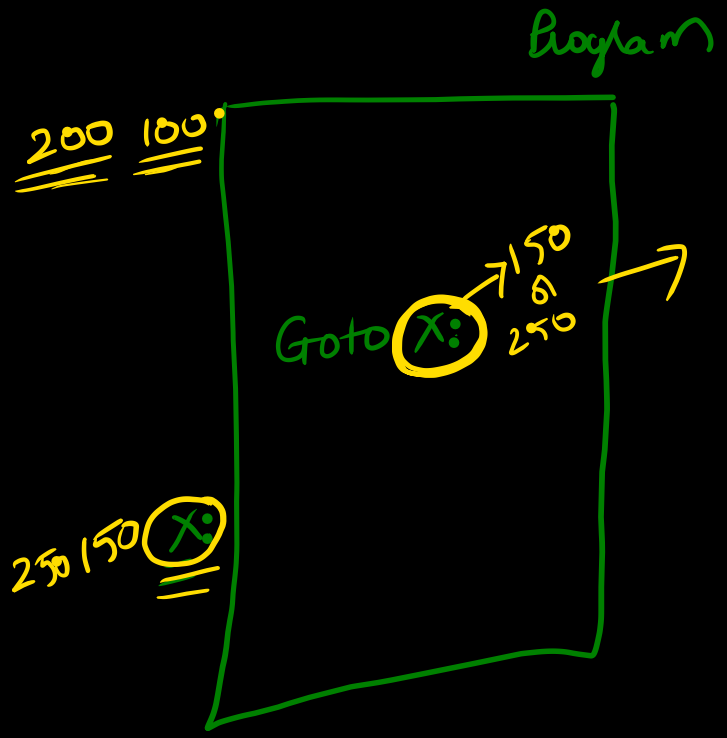
Process can be non contiguous.

- Paging
- Segmentation

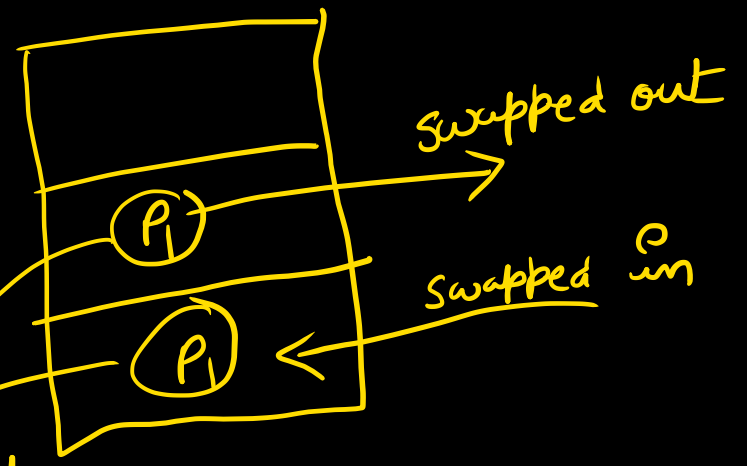
Compaction → Time consuming process and costly.



→ NO external fragmentation

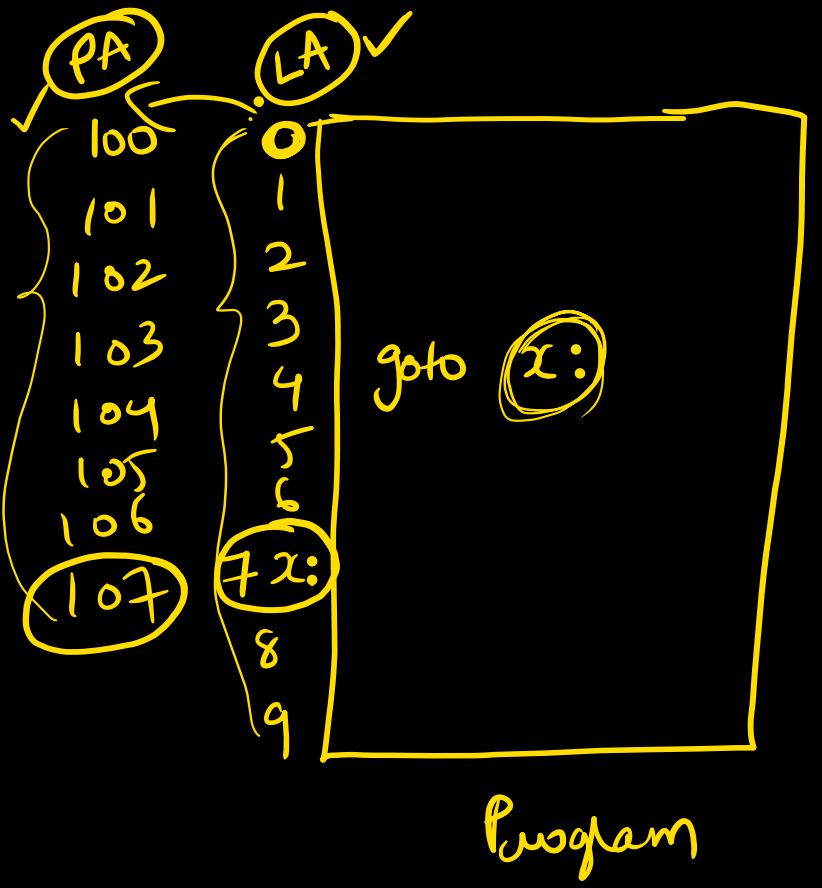


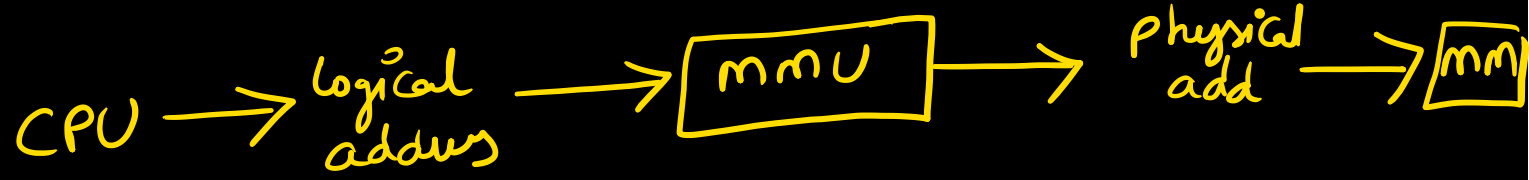
loader will know the exact add.



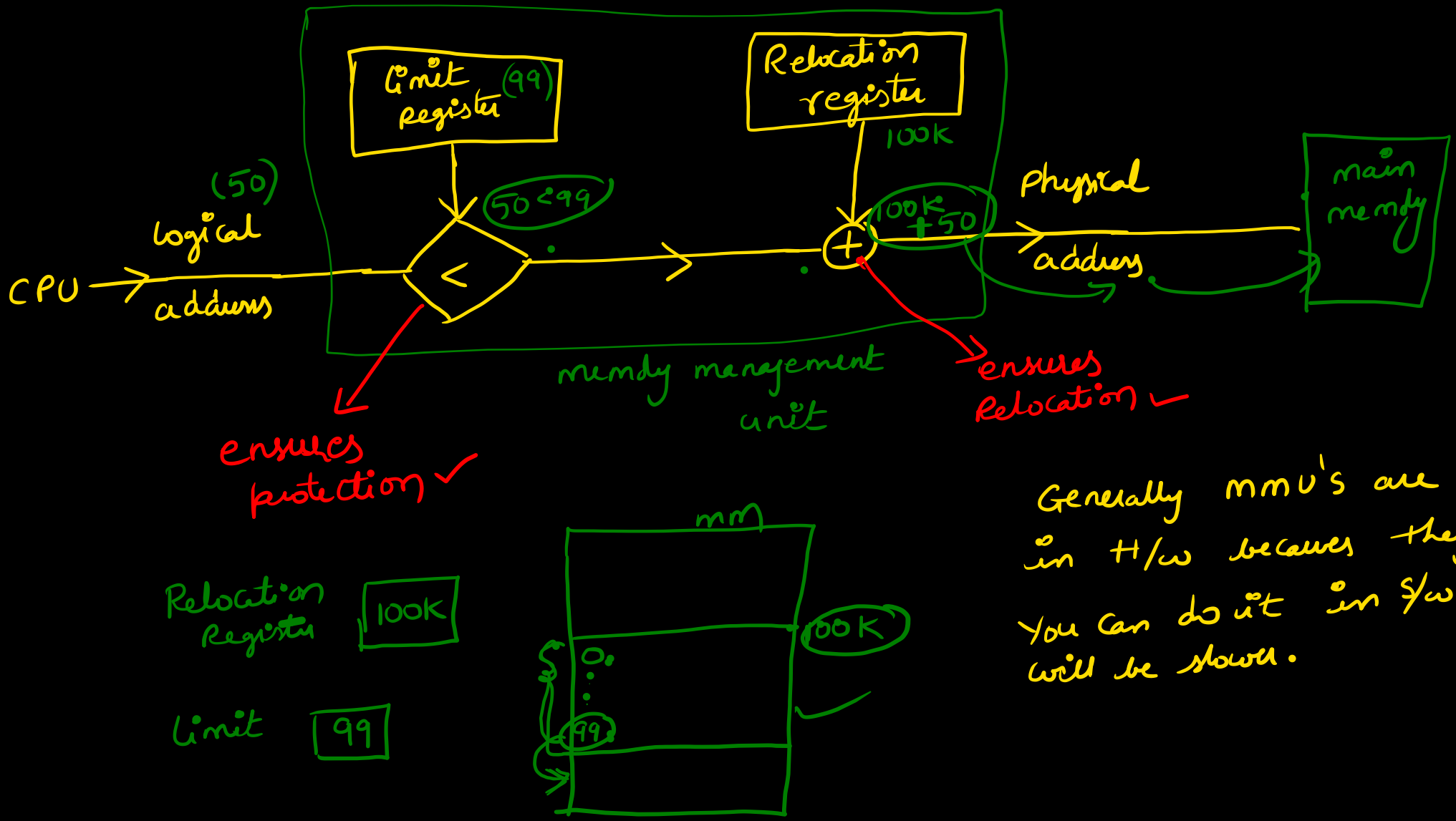
loader can't help if program in loader multiple times at different locations.

∴ we need memory mangment H/W.


PA
LA
100
101
102
103
104
105
106
107
0
1
2
3
4
5
6
7
8
9
goto x:
x:
Program


CPU
logical addrss
mmu
physical add
mm

5 min break



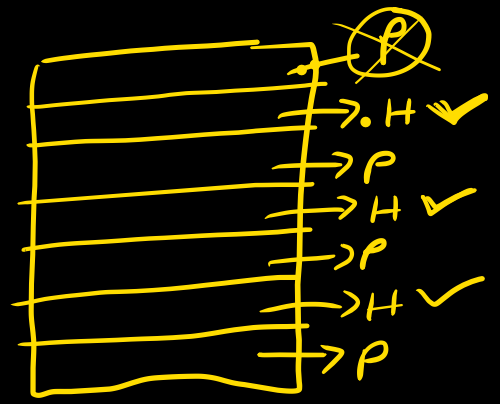
Generally MMU's are implemented in H/w because they are faster. You can do it in S/w also but it will be slower.

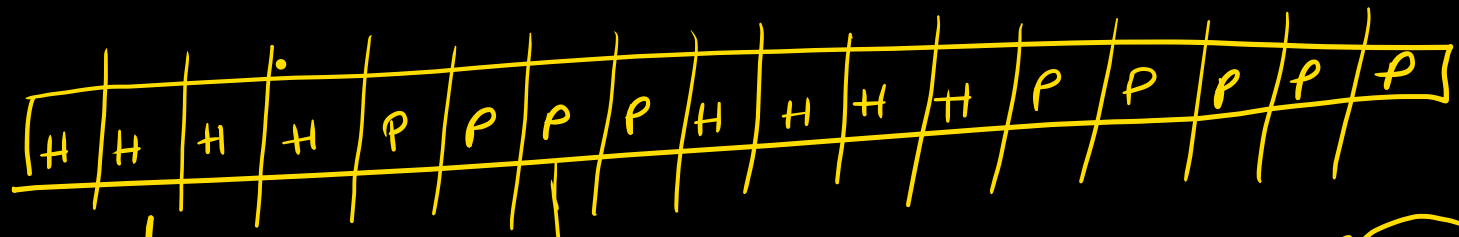
advantage of Dynamic partitioning:

(i) Degree of mp is dynamic (not fixed)
(ii) NO limitation on size of process
(iii) NO internal fragmentation

Disadvantage:

→ allocation and deallocation of memory is complex because of holes.
→ external fragmentation.





1 Dimensional list of memory

allocation unit → any size (4B is most popular)

2B 16B 32B

32 bit

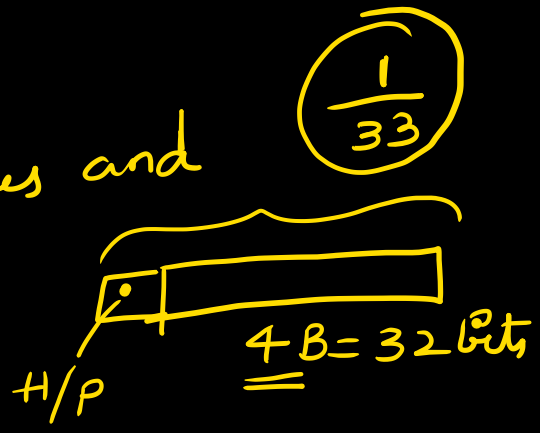
Bit map: one bit for each allocation unit.

→ Keep track of holes and processes.

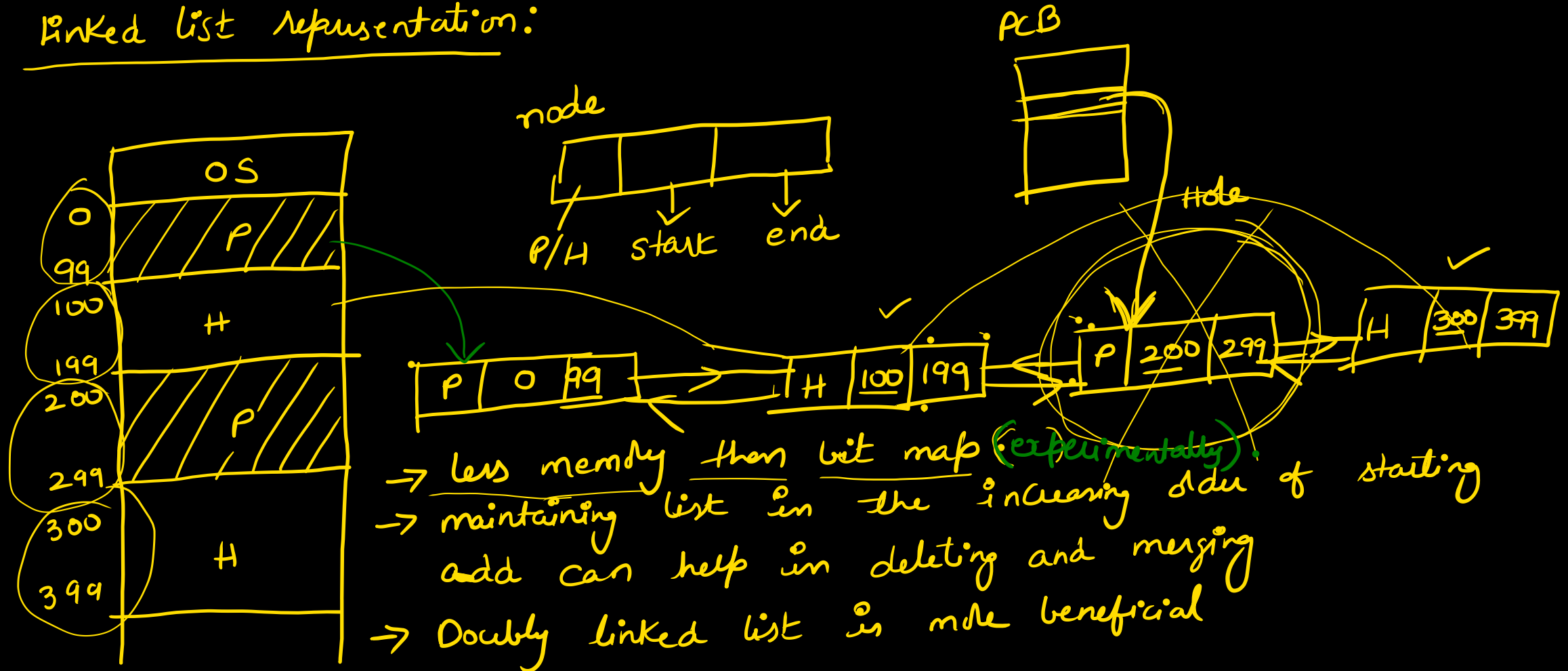
32bits

4B → 32 bits — 1 bit

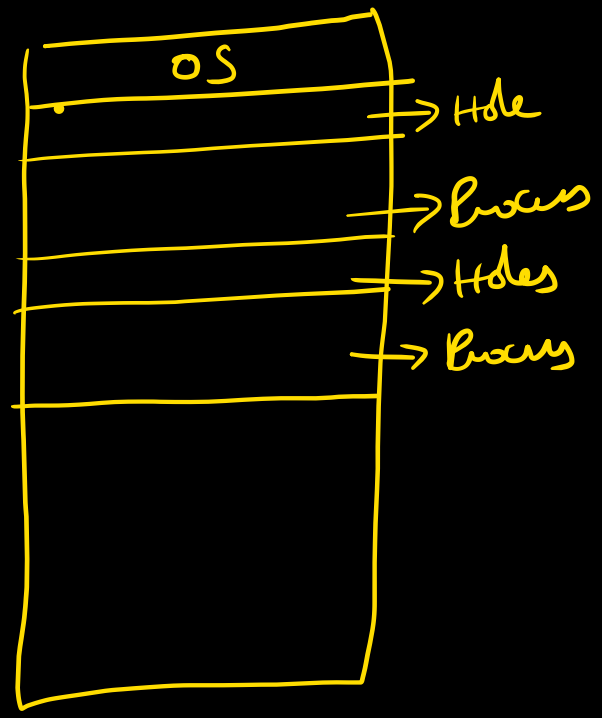
$\frac{1}{33}$ fraction of memory wasted



Linked list representation:



→ less memory than bit map (experimentally).
→ maintaining list in the increasing order of starting add can help in deleting and merging
→ Doubly linked list is more beneficial



$P_1 \rightarrow 4$ mB → which hole will be alloted.

(i) First fit

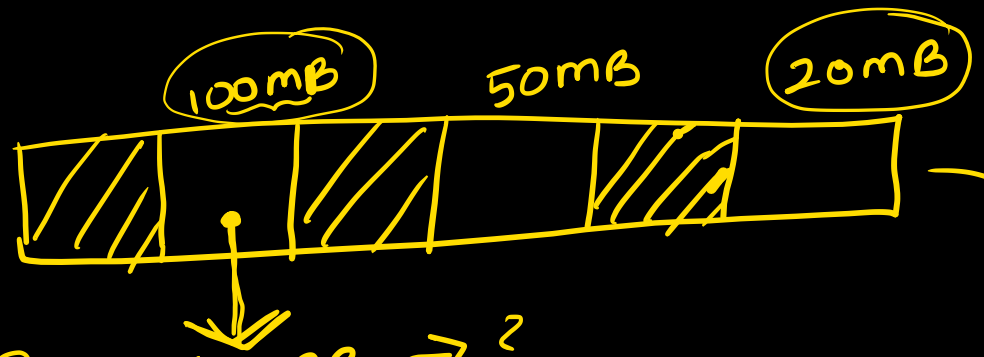
(ii) Next fit

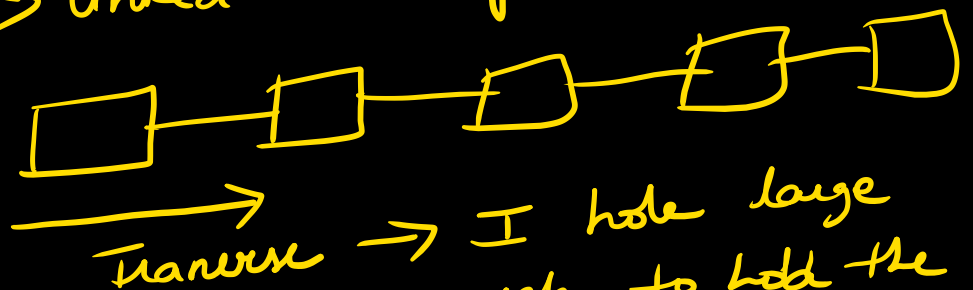
(iii) Best fit

(iv) Worst fit

(v) Quick fit

First fit



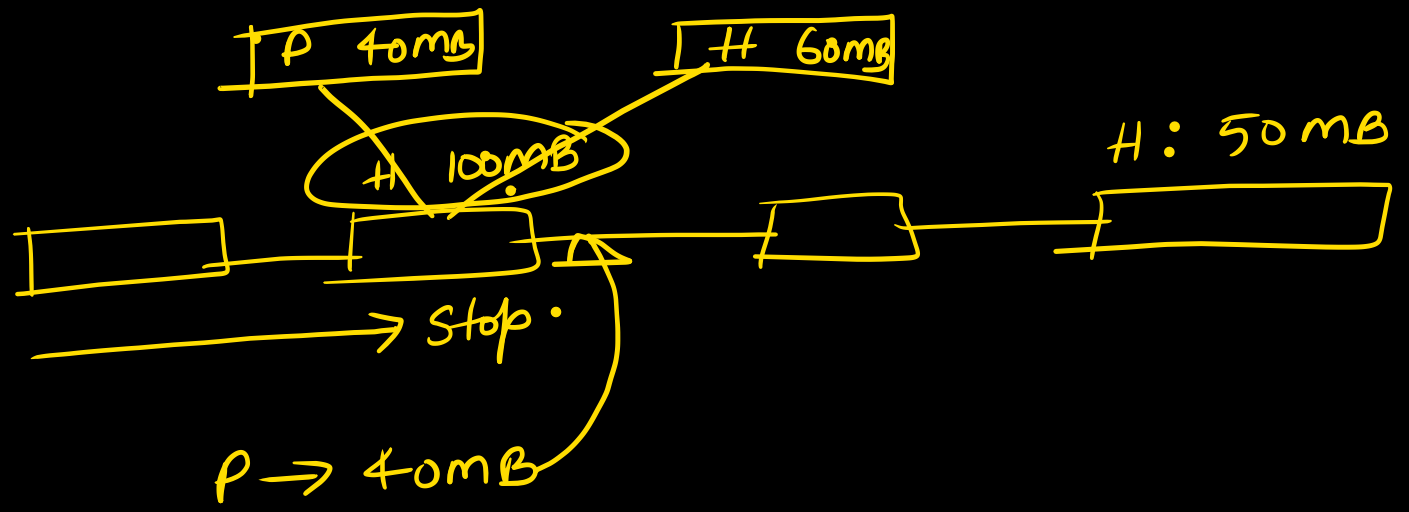


Simple to implement.

C programming (malloc) uses first fit.

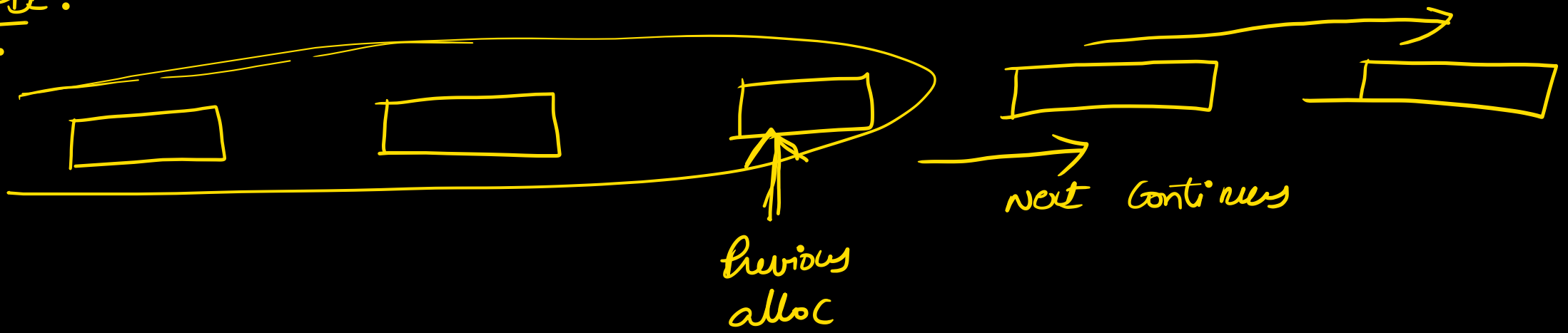
First fit is the best algo.

First fit:



First is the best

next fit:



First fit is best

Best Fit:

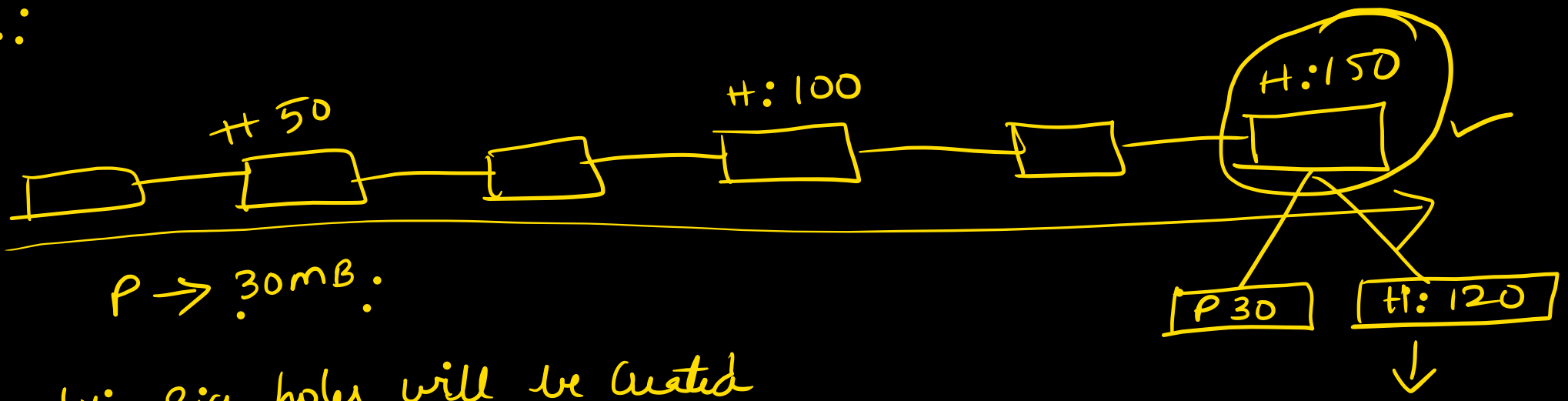
P → 30mB

Disadv:

1) This is time Consuming.
2) very small holes will be created which will be useless.

First is best one.

Worst Fit:



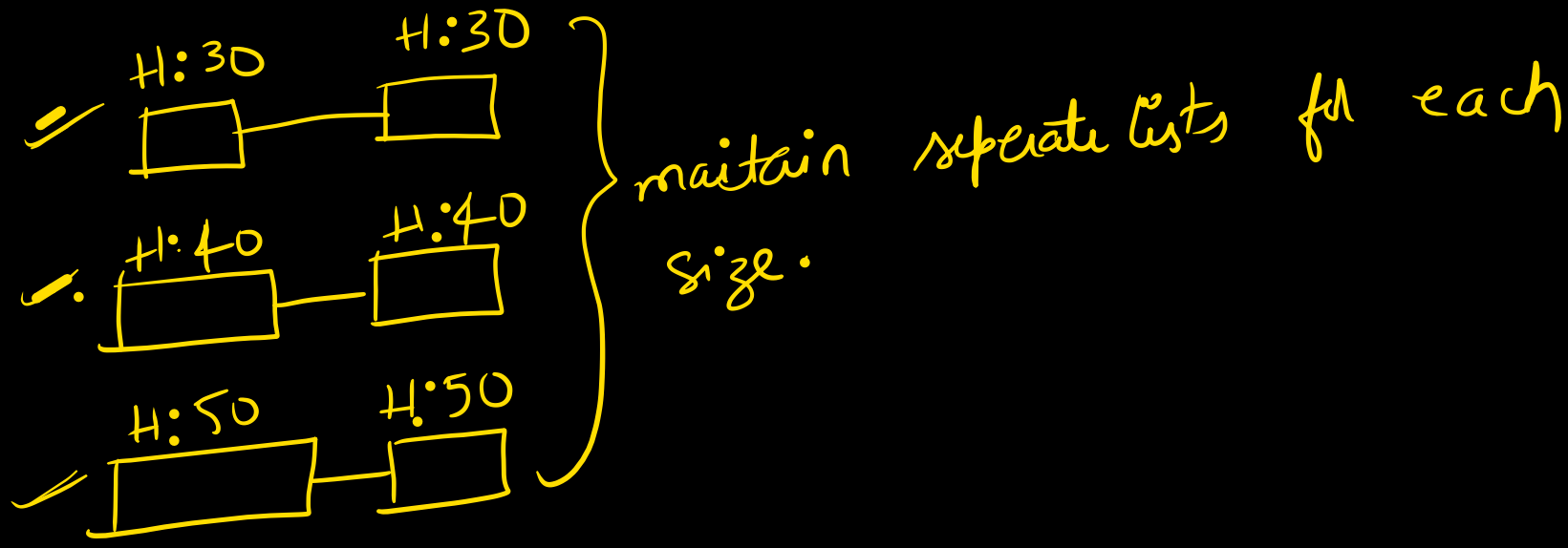
P → 30MB:

adv: Big holes will be created

dis: entire scan is required.

First is best.

Quick Fit:

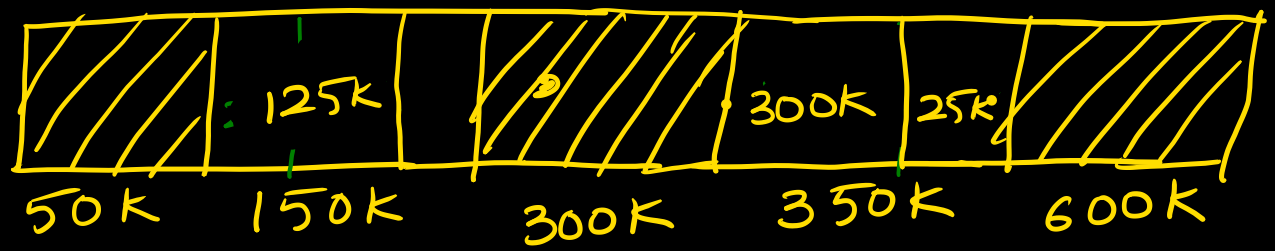


maintain seperate lists for each size.

~~Did~~ Dis: maintanence.

not used.

Gate 16) Process requests 300K, 25K, 125K, 50K.



a) Either FF & BF

b) FF but not BF

c) BF but not FF

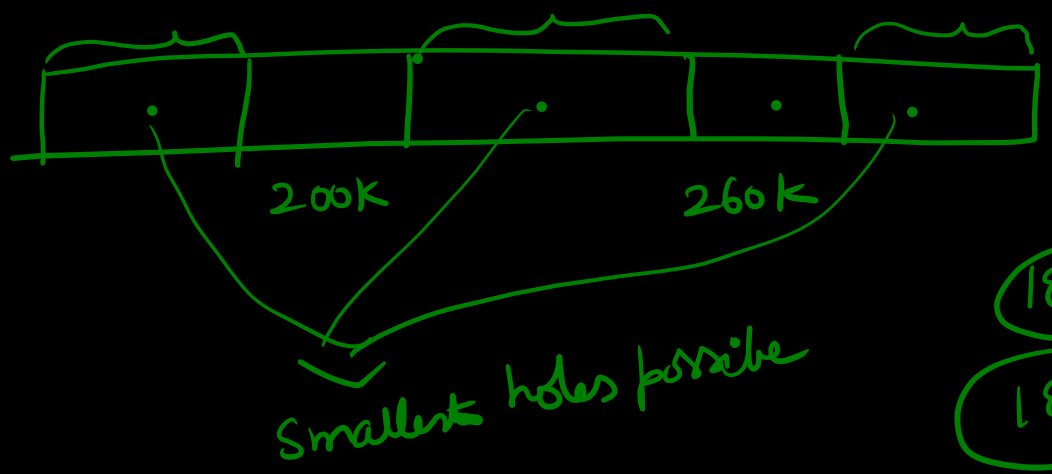
d) None.

FF BF

300K, 50K, 125K, 25K

Gate16 A 1000 KB memory is managed using variable partitions but no compaction. It currently has two partitions of sizes 200 KB and 260 KB respectively. The smallest allocation request in KB that could be denied is for

a) 151 b) 181 c) 231 d) 541



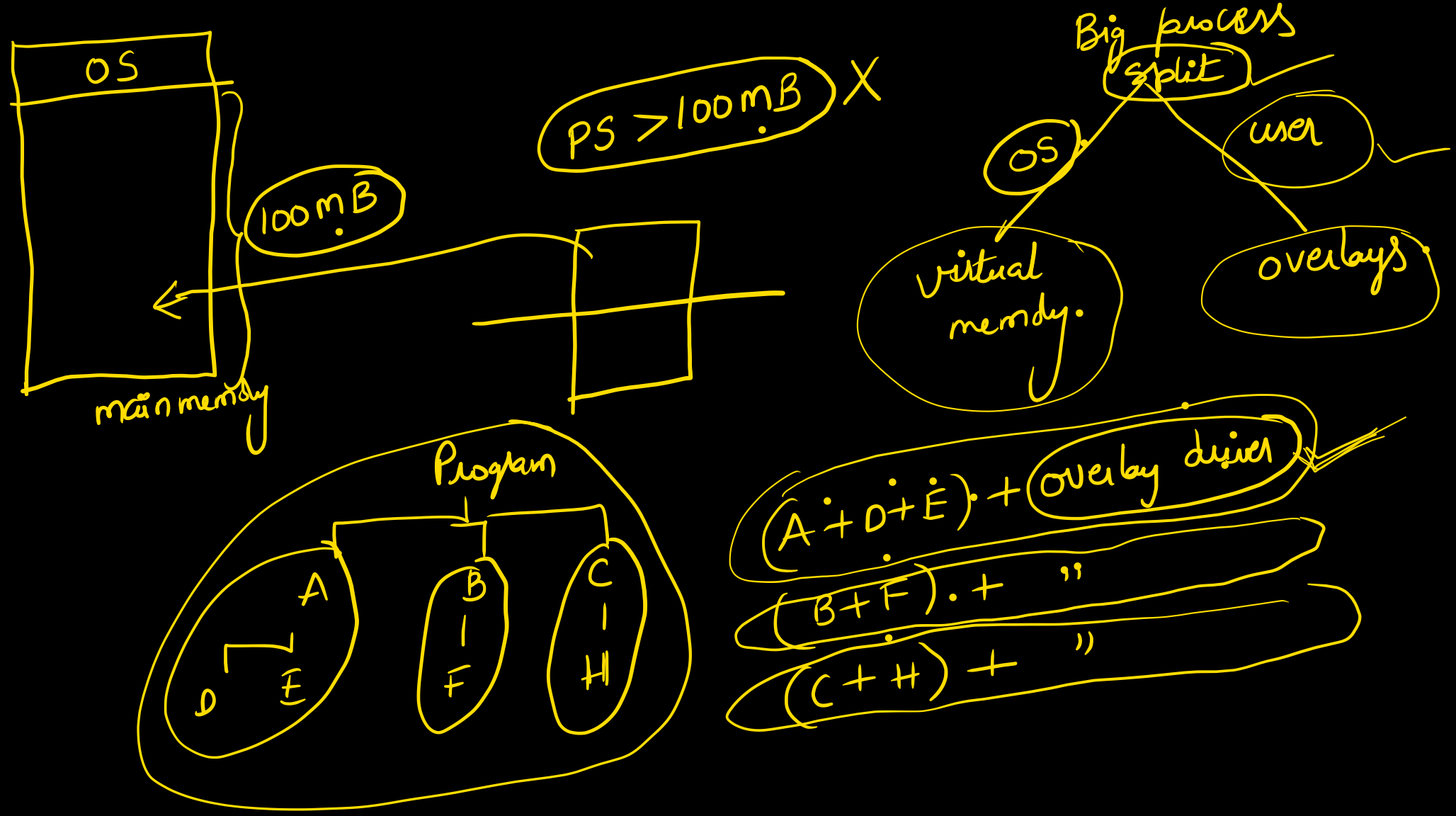
$3H + 200 + 260 = 1000$

$3H = 540$

$H = 180$

180

181


OS
100mB
main memory
PS > 100mB X
Big process
split
OS
user
virtual memory
overlays
Program
A
B
C
D
E
F
H
(A+D+E) + overlay driver
(B+F) + "
(C+H) + "

Consider a 2 pass assembler

Pass 1: 70 KB ✓

Pass 2: 80 KB ✓

symbol table : 30 KB ✓

Common Routine: 20 KB ✓

Total memory : 200 KB

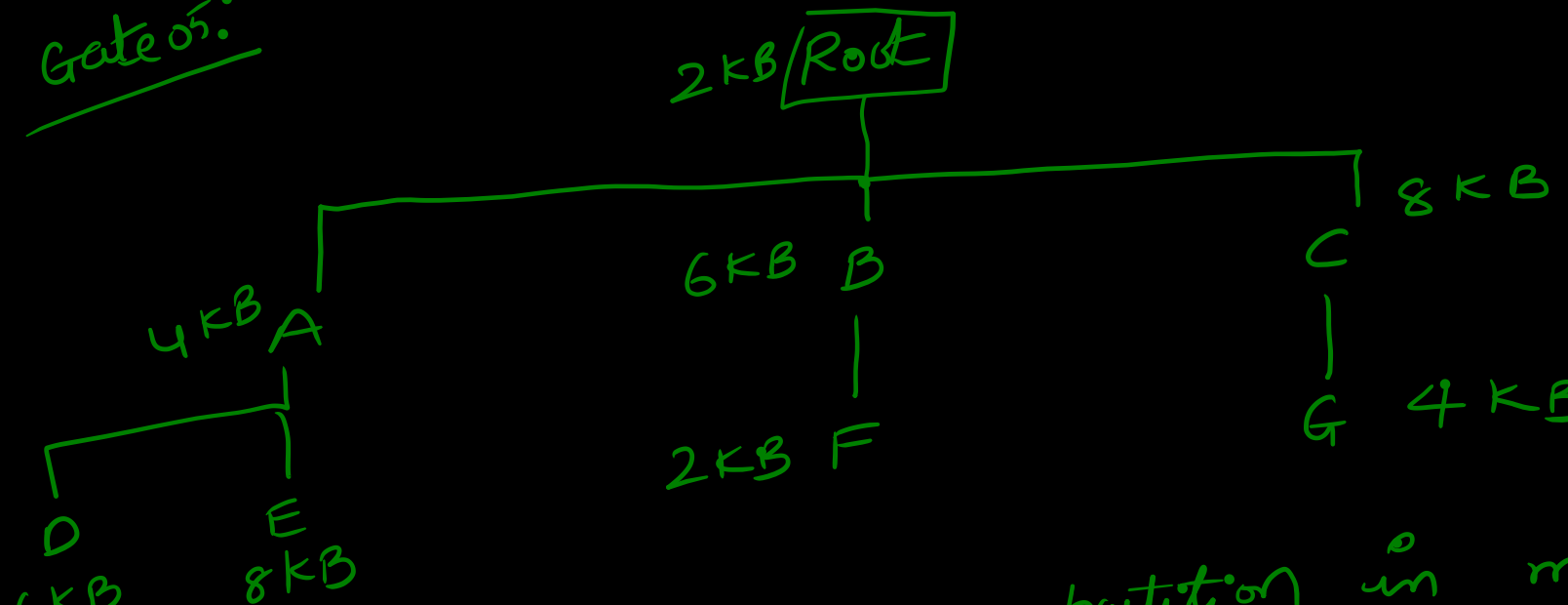
But at anytime only one pass will be in use and both passes always need ST and CR. If overlay driver is 10 KB, then what is the minimum partition size required?

Pass 1 requirement = 70 + 30 + 20 + 10 = 130 KB ✓

Pass 2 req = 80 + 30 + 20 + 10 = 140 KB ✓

140 KB ✓

Gate 05:



what will be size of partition in memory required to load and run this program?

R + A + D

R + A + E

R + B + F

R + C + G · 14 KB · ✓