11:54 AM Thu 14 Sep

Semester 6501

Home Insert Draw View Class Notebook

P.6.30 i — Process P_i ; j — Resource Type R_j;

$System_j$ = (10, 5, 7).

No more use.

Sol:

		$Allocation_i$	Max_i	$Need_i$
		A B C	A B C	A B C
③	P0	(0, 1, 0)	(7, 5, 3)	(6, 4, 3)
①	P1	(2, 0, 0)	(3, 2, 2)	(1, 2, 2)
④	P2	(3, 0, 2)	(9, 0, 2)	(6, 0, 0)
②	P3	(2, 1, 1)	(2, 2, 2)	(0, 1, 1)
⑤	P4	(0, 0, 2)	(4, 3, 3)	(4, 3, 1)

$\sum_j$ = (7, 2, 5)

Step 1: Compute initial $Available_j$

$Available_j = System_j - \sum_j (Allocation)$

= (10, 5, 7) - (7, 2, 5)

= (3, 3, 2).

Step 2: Compute & Check

$Max_i - Allocation_i$ = $Need_i$

[Checking]

	①	②	③	④	⑤
①	×	×	✓	-	-
②	✓	-	-	-	-
③					
④			×	✓	-
⑤			✓	-	-
③					✓

IF ($Need_i \leq Available_j$) THEN

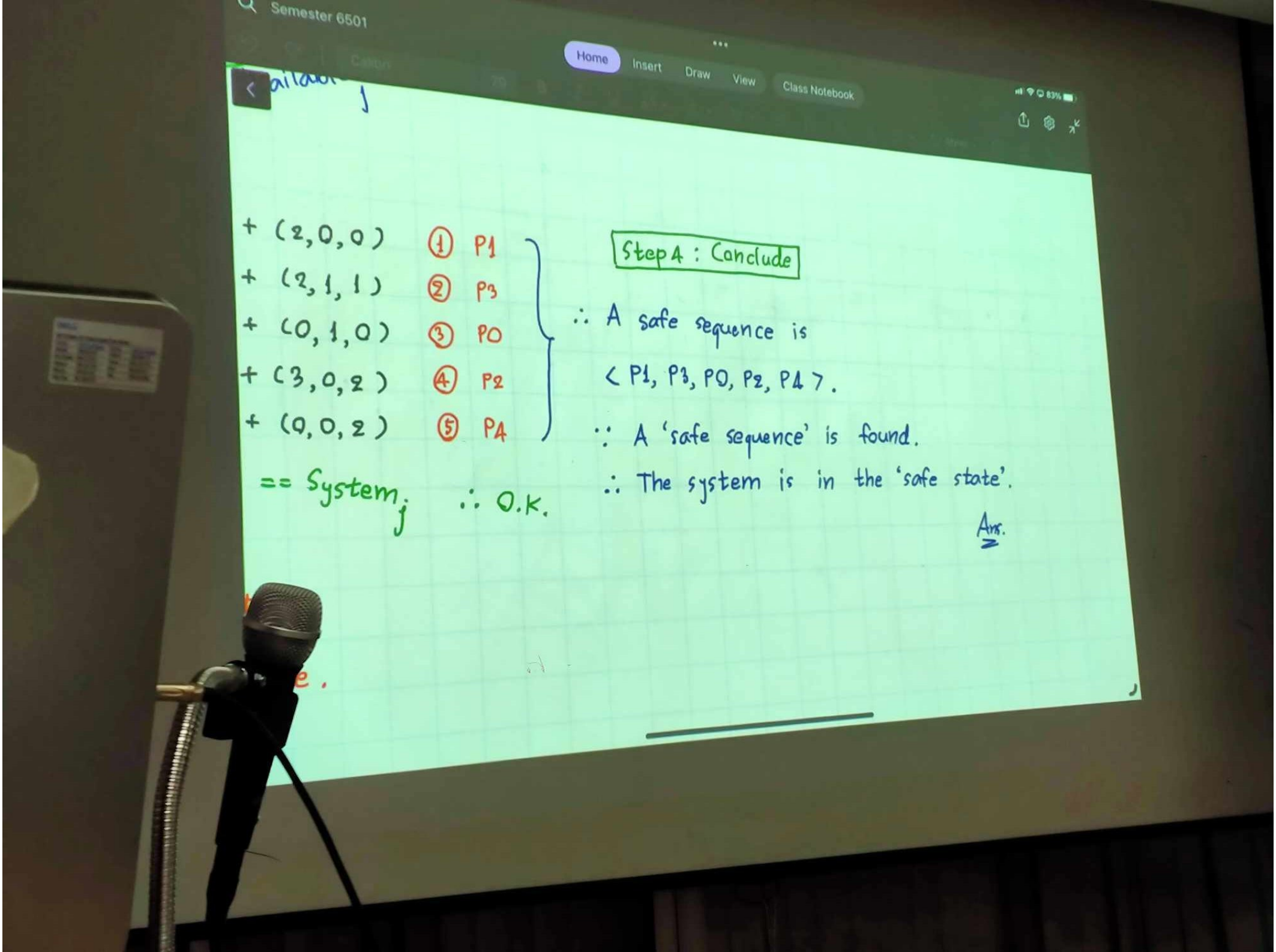
$Available_j = Available_j + Allocation_i$

// Add P_i to the safe sequence.

Step 3: Update

$Available_j = Available_j + Allocation_i$

$Available_j$	
A B C	
(3, 3, 2) + (2, 0, 0)	① P1
(5, 3, 2) + (2, 1, 1)	② P3
(7, 4, 3) + (0, 1, 0)	③ P0
(7, 5, 3) + (3, 0, 2)	④ P2
(10, 5, 5) + (0, 0, 2)	⑤ P4
(10, 5, 7) == $System_j$	∴ O.K.


Semester 6501
Home
Insert
Draw
View
Class Notebook
+ (2,0,0) ① P1
+ (2,1,1) ② P3
+ (0,1,0) ③ P0
+ (3,0,2) ④ P2
+ (0,0,2) ⑤ P4
== System_j ∴ O.K.
Step 4 : Conclude
∴ A safe sequence is
< P1, P3, P0, P2, P4 >.
∵ A 'safe sequence' is found.
∴ The system is in the 'safe state'.
Ans.

Sol 2.

$\therefore$ Allocation$_1$ = Allocation$_1$ + Request$_1$

= (2, 0, 0) + (1, 0, 2)

= (3, 0, 2).

No more use.

Step 2 : Compute & Check

Max$_i$ - Allocation$_i$

Step 3 : Update

= Available$_j$

		Allocation$_i$	Max$_i$	Need$_i$
		A B C	A B C	A B C
④	P0	(0, 1, 0)	(7, 5, 3)	(7, 4, 3)
①	P1	(3, 0, 2)	(3, 2, 2)	(0, 2, 0)
⑤	P2	(3, 0, 2)	(9, 0, 2)	(6, 0, 0)
②	P3	(2, 1, 1)	(2, 2, 2)	(0, 1, 1)
③	P4	(0, 0, 2)	(4, 3, 3)	(4, 3, 1)

$\sum_j$ = (8, 2, 7)

P1	P3	P4	P0	P2
①	②	③	④	⑤
			✓	-
✓	-	-	-	-
				✓
	✓	-	-	-
		✓	-	-

Available$_j$

A B C

(2, 3, 0) + (3, 0,

(5, 3, 2) + (2, 1,

(7, 4, 3) + (0, 0,

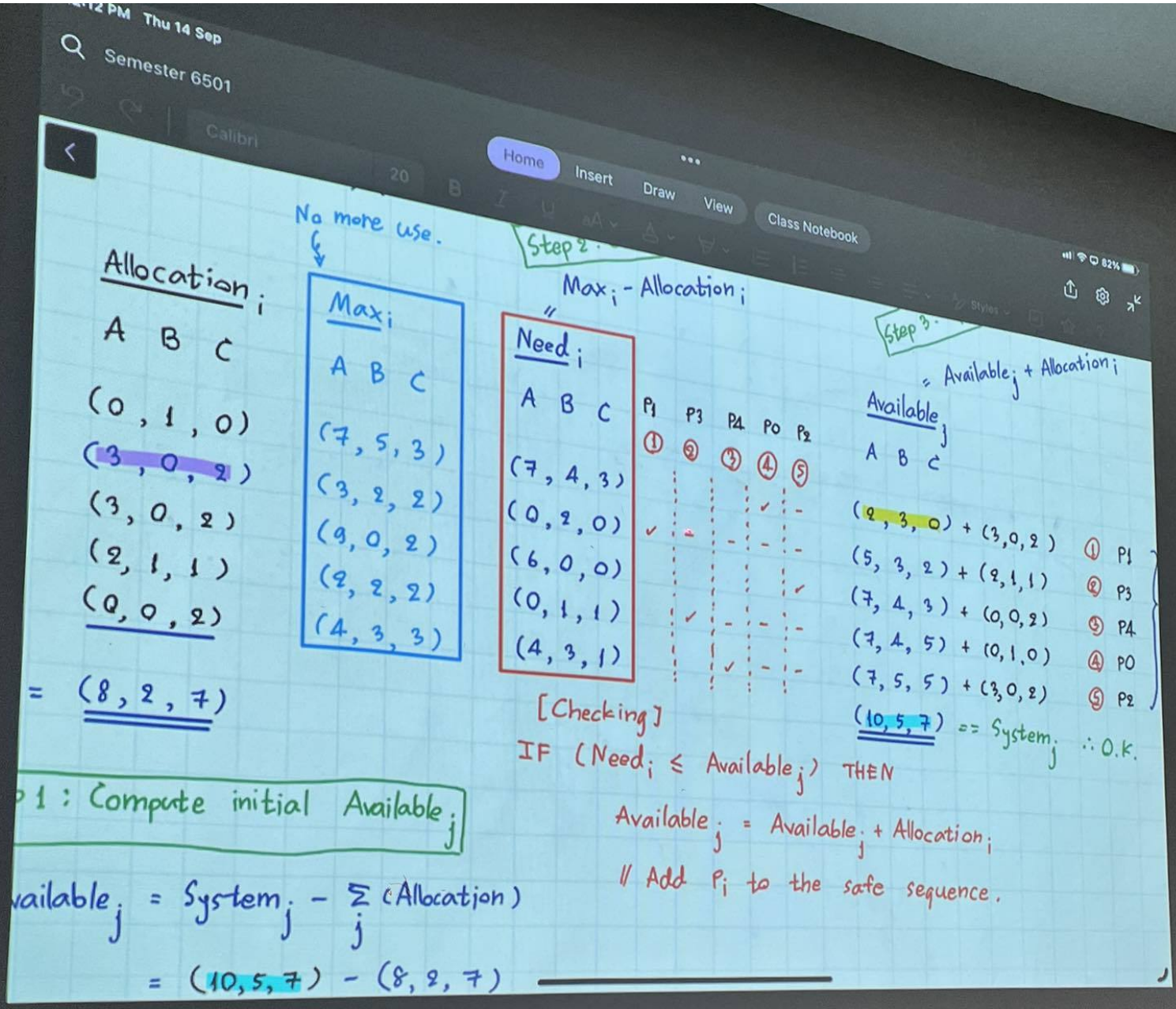
(7, 4, 5) + (0, 1,

(7, 5, 5) + (3, 0,

(10, 5, 7) == Syste

[Checking]

IF (Need$_i$ $\leq$ Available$_j$) THEN


Thu 14 Sep
Semester 6501
Home
Insert
Draw
View
Class Notebook
No more use.
Allocation$_i$
A B C
(0, 1, 0)
(3, 0, 2)
(3, 0, 2)
(2, 1, 1)
(0, 0, 2)
= (8, 2, 7)
Max$_i$
A B C
(7, 5, 3)
(3, 2, 2)
(9, 0, 2)
(2, 2, 2)
(4, 3, 3)
Step 2:
Max$_i$ - Allocation$_i$
Need$_i$
A B C
(7, 4, 3)
(0, 2, 0)
(6, 0, 0)
(0, 1, 1)
(4, 3, 1)
P1 P3 P4 P0 P2
① ② ③ ④ ⑤
[Checking]
IF (Need$_i$ ≤ Available$_j$) THEN
Available$_j$ = Available$_j$ + Allocation$_i$
// Add P_i to the safe sequence.
Step 3:
= Available$_j$ + Allocation$_i$
Available$_j$
A B C
(2, 3, 0) + (3, 0, 2) ① P1
(5, 3, 2) + (2, 1, 1) ② P3
(7, 4, 3) + (0, 0, 2) ③ P4
(7, 4, 5) + (0, 1, 0) ④ P0
(7, 5, 5) + (3, 0, 2) ⑤ P2
(10, 5, 7) == System$_j$ ∴ O.K.
1 : Compute initial Available$_j$
vailable$_j$ = System$_j$ - Σ$_j$ (Allocation)
= (10, 5, 7) - (8, 2, 7)

Home Insert Draw View Class Notebook

Check

$_i$;

P4 P0 P2

④ ⑤

Step 3: Update

= Available$_j$ + Allocation$_i$

Available$_j$

A B C

(2, 3, 0) + (3, 0, 2) ① P1

(5, 3, 2) + (2, 1, 1) ② P3

(7, 4, 3) + (0, 0, 2) ③ P4

(7, 4, 5) + (0, 1, 0) ④ P0

(7, 5, 5) + (3, 0, 2) ⑤ P2

(10, 5, 7) == System$_j$ ∴ O.K.

Step 4: Conclude

∵ $< P_1, P_3, P_4, P_0, P_2 >$ is a 'safe sequence'.

∴ The system is in a 'safe state'.

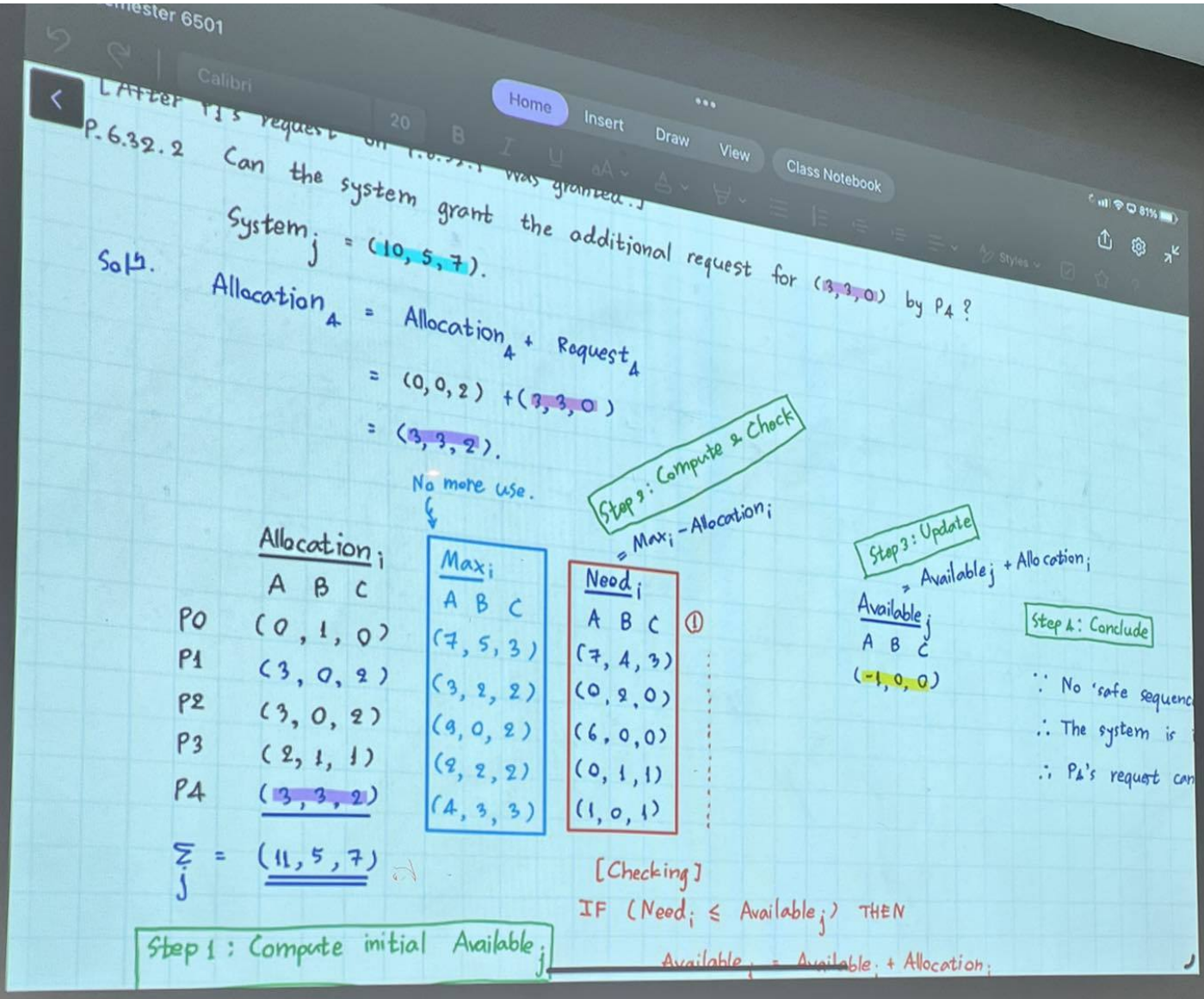
∴ P_1's request can be granted.

Ans.

lable$_j$) THEN

Available$_j$ + Allocation$_i$

to the safe sequence.


Home
Insert
Draw
View
Class Notebook
P.6.32.2 Can the system grant the additional request for (3, 3, 0) by P_4 ?
System j = (10, 5, 7).
Sol:
Allocation4 = Allocation4 + Request4
= (0, 0, 2) + (3, 3, 0)
= (3, 3, 2).
No more use.
Step 2: Compute & Check
= Maxi - Allocationi
Step 3: Update
= Available j + Allocationi
Step 4: Conclude
Allocationi
A B C
P0 (0, 1, 0)
P1 (3, 0, 2)
P2 (3, 0, 2)
P3 (2, 1, 1)
P4 (3, 3, 2)
Σ j = (11, 5, 7)
Maxi
A B C
(7, 5, 3)
(3, 2, 2)
(9, 0, 2)
(2, 2, 2)
(4, 3, 3)
Needi
A B C
(7, 4, 3)
(0, 2, 0)
(6, 0, 0)
(0, 1, 1)
(1, 0, 1)
①
Available j
A B C
(-1, 0, 0)
∵ No 'safe sequenc
∴ The system is
∴ P4's request can
[Checking]
IF (Needi ≤ Available j) THEN
Available j = Available j + Allocationi
Step 1 : Compute initial Available j

= (3, 3, 2).

No more use.

Allocation$_i$	A	B	C
P0	0	1	0
P1	3	0	2
P2	3	0	2
P3	2	1	1
P4	3	3	2
$\sum_j$ =	11	5	7

Max$_i$	A	B	C
	7	5	3
	3	2	2
	9	0	2
	2	2	2
	4	3	3

Step 2: Compute Need$_i$

$= \text{Max}_i - \text{Allocation}_i$

Need$_i$	A	B	C
	7	4	3
	0	2	0
	6	0	0
	0	1	1
	1	0	1

①

Step 3: Update

$= \text{Available}_j + \text{Allocation}_i$

Available$_j$	A	B	C
	-1	0	0

Step 4: Conclude

∵ No 'safe sequenc

∴ The system is

∴ P_4's request can

[Checking]

IF ($\text{Need}_i \leq \text{Available}_j$) THEN

$\text{Available}_j = \text{Available}_j + \text{Allocation}_i$

// Add P_i to the safe sequence.

Step 1 : Compute initial Available$_j$

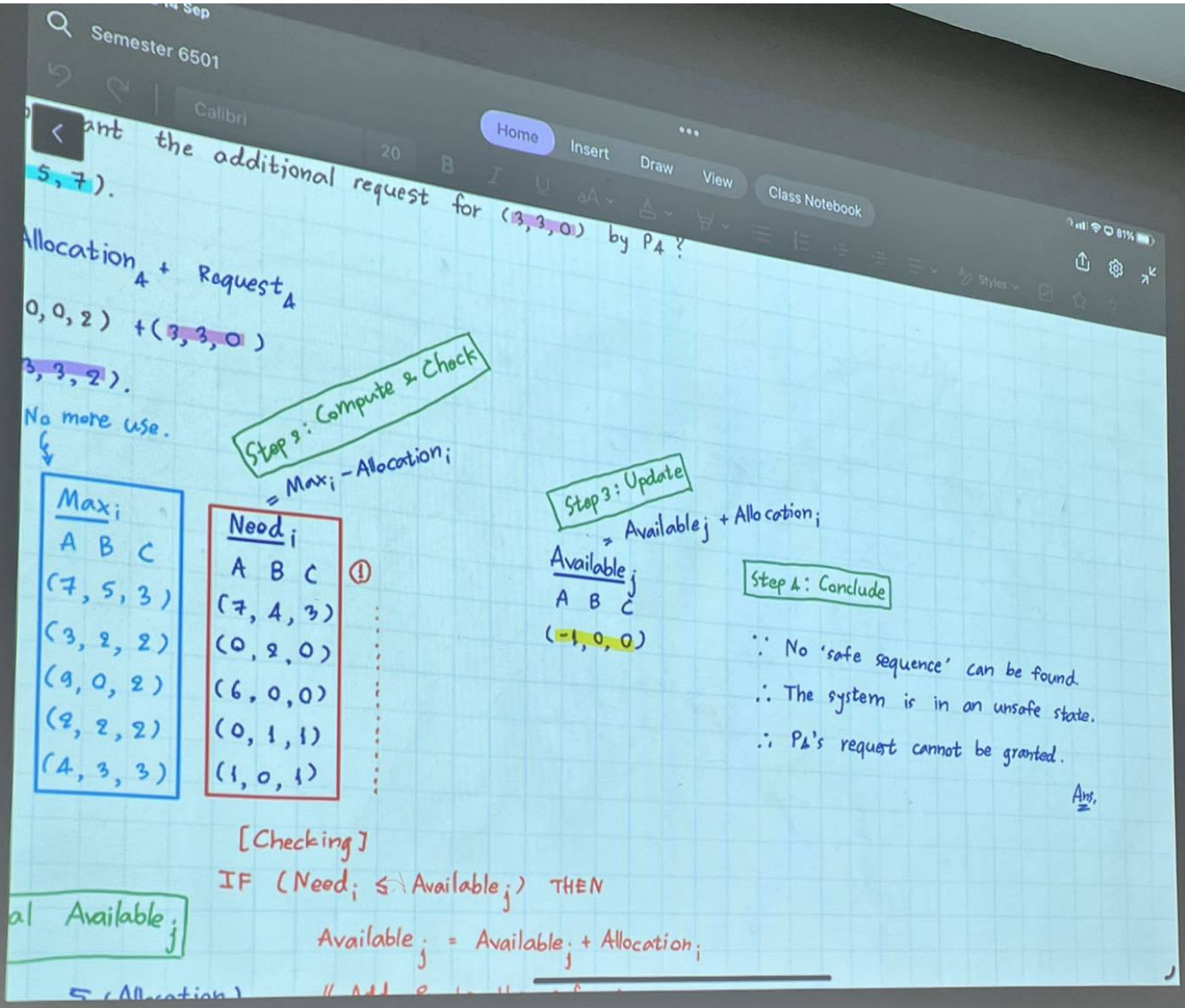
$\text{Available}_j = \text{System}_j - \sum_j (\text{Allocation})$

$= (10, 5, 7) - (11, 5, 7)$

$= (-1, 0, 0)$.

[After P_1's request on P.6.39.1 was granted and P_4's request on P.6.39.2 was not granted.]

P.6.39.3 Can request for (0, 2, 0) be granted?


Semester 6501
Home
Insert
Draw
View
Class Notebook
ant the additional request for (3, 3, 0) by P4 ?
5, 7).
Allocation4 + Request4
0, 0, 2) + (3, 3, 0)
3, 3, 2).
No more use.
Step 2: Compute & Check
= Maxi - Allocationi
Maxi
A B C
(7, 5, 3)
(3, 2, 2)
(9, 0, 2)
(2, 2, 2)
(4, 3, 3)
Needi
A B C
(7, 4, 3)
(0, 2, 0)
(6, 0, 0)
(0, 1, 1)
(1, 0, 1)
Step 3: Update
= Availablej + Allocationi
Availablej
A B C
(-1, 0, 0)
Step 4: Conclude
∵ No 'safe sequence' can be found.
∴ The system is in an unsafe state.
∴ P4's request cannot be granted.
Ans.
[Checking]
IF (Needi ≤ Availablej) THEN
Availablej = Availablej + Allocationi

12:19 PM Thu 14 Sep

Semester 6501

Home Insert Draw View Class Notebook

[After P_1's request on P.6.39.1 was granted and P_4's request on P.6.32.2 was not granted.]

P.6.32.3 Can request for (0,2,0) by P_0 be granted?

System$_j$ = (10, 5, 7).

Sol:

Allocation$_0$ = Allocation$_0$ + Request$_0$

= (0, 1, 0) + (0, 2, 0)

= (0, 3, 0).

No more use.

	Allocation$_i$	Max$_i$	Need$_i$
	A B C	A B C	A B C
P0	(0, 3, 0)	(7, 5, 3)	(7, 2, 3)
P1	(3, 0, 2)	(3, 2, 2)	(0, 2, 0)
P2	(3, 0, 2)	(9, 0, 2)	(6, 0, 0)
P3	(2, 1, 1)	(2, 2, 2)	(0, 1, 1)
P4	(0, 0, 2)	(4, 3, 3)	(4, 3, 1)
$\sum_j$	(8, 4, 7)		

Step 1: Compute initial Available$_j$

Available$_j$ = System$_j$ − $\sum_j$ (Allocation)

= (10, 5, 7) − (8, 4, 7)

= (2, 1, 0).

Step 2: Compute & Check

= Max$_i$ − Allocation$_i$

[Checking]

IF (Need$_i$ ≤ Available$_j$) THEN

Available$_j$ = Available$_j$ + Allocation$_i$

// Add P_i to the safe sequence.

Step 3: Update

= Available$_j$ + Allocation$_i$

Available$_j$

A B C

(2, 1, 0)

Step 4: Conclude

∵ No 'safe sequence' can be found.

∴ The system is in an unsafe state.

∴ P0's request cannot be granted.

Ans.

OK?

Semester 6501

Calibri 20

Home Insert Draw View Class Notebook

= Available j + ...

lable j

B C

(, 0)

Step 4: Conclude

∴ No 'safe sequence' can be found.

∴ The system is in an unsafe state.

∴ P0's request cannot be granted.

Ans,

N

- Allocation:

11:34 AM Mon 18 Sep

Semester 6501

Home Insert Draw View Class Notebook

6.42 Is deadlock detected in the system for the sequence $\langle P_0, P_2, P_3, P_1, P_4 \rangle$? (① ② ③ ④ ⑤) OK?

System$_j$ = (7, 2, 6).

Sol^n.

Step 2: Check (Need$_i$)

Step 3: Update Available$_j$ = Available$_j$ + Allocation$_i$

		Allocation$_i$ A B C	Request$_i$ A B C	P0 P2 P3 P1 P4	Available$_j$ A B C
①	P0	(0, 1, 0)	(0, 0, 0)	✓ - - - -	(0, 0, 0) + (0, 1, 0) ① P0
④	P1	(2, 0, 0)	(2, 0, 2)		(0, 1, 0) + (3, 0, 3) ② P2
②	P2	(3, 0, 3)	(0, 0, 0)	✓ - - - -	(3, 1, 3) + (2, 1, 1) ③ P3
③	P3	(2, 1, 1)	(1, 0, 0)		(5, 2, 4) + (2, 0, 0) ④ P1
⑤	P4	(0, 0, 2)	(0, 0, 2)	✓ - - - -	(7, 2, 4) + (0, 0, 2) ⑤ P4
$\sum_j$		(7, 2, 6)		[Checking]	(7, 2, 6) => System$_j$ ∴ O.K.

IF (Request$_i$ ≤ Available$_j$) THEN

Step 4: Conclude

∵ $\langle P_0, P_2, P_3, P_1, P_4 \rangle$ is a 'finish sequence'.

∴ All processes finished its tasks and 'no deadlock' was detected in the system.

Ans.

Step 1: Compute initial Available$_j$

Available$_j$ = Available$_j$ + Allocation$_i$

// Add P_i to the finish sequence.

Available$_j$ = System$_j$ - $\sum_j$ (Allocation)

= (7, 2, 6) - (7, 2, 6)

= (0, 0, 0).

11:46 AM Mon 18 Sep

Semester 6501

Home Insert Draw View Class Notebook

6.43 P_2 requests 1 more C.
Find out if the system is in a deadlock state.

Find a 'finish sequence'.

$System_j = (7, 2, 6)$.

Sol:

$Request_2 = Request_2 + (0, 0, 1)$.
$= (0,0,0) + (0,0,1) = (0,0,1)$.

Step 2 : Check

	Allocation$_i$ (A B C)	(Need$_i$) Request$_i$ (A B C)
P0	(0, 1, 0)	(0, 0, 0)
P1	(2, 0, 0)	(2, 0, 2)
P2	(3, 0, 3)	(0, 0, 1)
P3	(2, 1, 1)	(1, 0, 0)
P4	(0, 0, 2)	(0, 0, 2)
$\sum_j$	(7, 2, 6)	

[Checking]

IF (Request$_i \leq$ Available$_j$) THEN

Available$_j$ = Available$_j$ + Allocation$_i$

// Add P_i to the finish sequence.

Step 1: Compute initial Available$_j$

Step 3: Update

Available$_j$ = Available$_j$ + Allocation$_i$

Available$_j$ (A B C)	
(0, 0, 0) + (0, 1, 0)	① P0
(0, 1, 0)	

Step 4 : Conclude

∵ Only <P0> finished its task.

∴ P1, P2, P3, P4 are in deadlock.

Deadlock is detected in the system.

Ans.

P. 6.11 Assume P_3 requests resources A=1, B=2, C=3.

Check what deadlock state of the system will be if the resource allocating sequence is $\langle P_0, P_2, P_3, P_1, P_4 \rangle$.

Sol^n.

$System_j = (7, 2, 6)$.

↑ Check if this is a finish sequence.

$Request_i = (1, 2, 3)$.

Step 2: Check ($Need_i$)

		Allocation_i	Request_i	P0	P2	P3
		A B C	A B C	①	②	③
①	P_0	(0, 1, 0)	(0, 0, 0)	✓	-	-
	P_1	(2, 0, 0)	(2, 0, 2)			
②	P_2	(3, 0, 3)	(0, 0, 0)		✓	-
	P_3	(2, 1, 1)	(1, 2, 3)			×
	P_4	(0, 0, 2)	(0, 0, 2)			
	$\sum_j$	(7, 2, 6)				

[Checking]

Step 3: Update $Available_j = Available_j + Allocation_i$

$Available_j$		
(0, 0, 0) + (0, 1, 0)	①	P_0
(0, 1, 0) + (3, 0, 3)	②	P_2
(3, 1, 3) Not enough B for P_3		

Step 4: Conclude

∵ $\langle P_0, P_2, P_3, P_1, P_4 \rangle$ is not a 'finish sequence'.

∴ The system has deadlock where P_1, P_3, P_4 are in the deadlock.

Ans.

Step 1: Compute initial $Available_j$

$Available_j = System_j - \sum_j (Allocation)$

$= (7, 2, 6) - (7, 2, 6)$

$= (0, 0, 0)$.

IF ($Request_i \in Available_j$) THEN

$Available_j = Available_j + Allocation_i$

// Add P_i to the finish sequence.

Semester 6501

Calibri 20

Home Insert Draw View Class Notebook

Styles

49%

Deadlock

- **Prevention** : (p. 6.15) 4 conditions must not occur simultaneously.
- **Avoidance**
  - [Single instance] (p. 6.22) Resource-Allocation Graph
    - Cycle(s) : 'Unsafe state'
      - [
      - [A
    - No cycle : 'Safe state'
  - [Multiple instances] (p. 6.30) Banker's Algo. : 'Safe sequence' = 'Safe
- **Detection & Recovery**
  - [Single instance] (p. 6.38) Wait-for Graph : Cycle(s) = Deadlock.
  - [Multiple instances] (p. 6.42) Detection Algo. : 'Finish sequence' = '

11:26 AM Thu 21 Sep

Semester 6501

Calibri 20

Home Insert Draw View Class Notebook

must not occur simultaneously.

) Resource-Allocation Graph

- Cycle(s) : 'Unsafe state'
  - [Single instance] Surely deadlocked.
  - [Multiple instances] Possibly deadlocked (may / may not).
- No cycle : 'Safe state'

) Banker's Algo. : 'Safe sequence' = 'Safe state'.

) Wait-for Graph : Cycle(s) = Deadlock.

2) Detection Algo. : 'Finish sequence' = 'No deadlock'.