Lesson 4

Bankers Algorithm

Numerical Solution

Question 1

Process	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P0	0	0	1	2	0	0	1	2	1	5	2	0
P1	1	0	0	0	1	7	5	0				
P2	1	3	5	4	2	3	5	6				
P3	0	6	3	2	0	6	5	2				
P4	0	0	1	4	0	6	5	6				

- Find the need matrix?
- Is the system is in safe state? If yes find the safe sequence?
- If P1 request (0, 4, 2, 0). Can the request be granted immediately?

Solution:

Process	Allocation				Max				Available				Need				Sequence
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	
P0	0	0	1	2	0	0	1	2	1	5	2	0	0	0	0	0	P0
P1	1	0	0	0	1	7	5	0	1	5	3	2	0	7	5	0	P2
P2	1	3	5	4	2	3	5	6	2	8	8	6	1	0	0	2	P3
P3	0	6	3	2	0	6	5	2	2	14	11	8	0	0	2	0	P4
P4	0	0	1	4	0	6	5	6	2	14	12	12	0	6	4	2	P1
									3	14	12	12					

For need matrix:

Need = max – allocation

For P0 :

need = 0 0 1 2 – 0 0 1 2 = 0 0 0 0

Similarly do for other process

Step 2:

At first

Since for p0: need(0 0 0 0)<=available(1 5 2 0) P0 is executed first

New available = old available +allocation

New available = 1 5 2 0 + 0 0 1 2 = 1 5 3 2

At second

Since for P2: need(1 0 0 2) <= available (1 5 3 2) so p2 is execute at second)

New available = old available + allocation

New available = 1 5 3 2 + 1 3 5 4

New available = 2 8 8 6

At third:

Since for P3: need (0 0 2 0)< = available (2 8 8 6) P3 is executed

New available = old available +allocation

New available = 2 8 8 6+0 6 3 2

New available = 2 14 11 8

At fourth:

Since for P4: need (0 6 4 2) < = available (2 14 11 8) P4 is executed at fourth

New available = old available + allocation

New available = 2 14 11 8 + 0 0 1 4

New available = 2 14 12 12

At last:

Since P1: need (0 7 5 0) <= available (2 14 12 12) P1 is executed

New available = 2 14 12 12 + 1 0 0 0

New available = 3 14 12 12

Since all the process are executed, system is in safe state. Safe sequence is P0 => P2=> P3=> P4=> P1

If P1 request (0, 4, 2, 0). Can the request be granted immediately?

P1 requested 0 4 2 0:

Request of P1 = 0 4 2 0

Need of P1= 0 7 5 0

Available of P1= 1 5 3 2

Step 1: request <=need

0 4 2 0 < = 0 7 5 0 true

(Note: If in case step 1 will be false write resource cannot be granted)

Step 2: request < = available

0 4 2 0 < = 1 5 3 2 true

(note: if step 1 is true but step 2 is false then write resource cannot be granted)

Step 3:

New available = available – request

New available = 1 5 3 2 – 0 4 2 0

New available = 1 1 1 2

Allocation = old allocation + request

Allocation = 1 0 0 0 + 0 4 2 0

Allocation = 1 4 2 0

Need = need – request

Need = 0 7 5 0 – 0 4 2 0

Need = 0 3 3 0

Process	Allocation				Max				Available				Need				Sequence
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	
P0	0	0	1	2	0	0	1	2	1	1	1	2	0	0	0	0	P0
P1	1	4	2	0	1	7	5	0	1	1	2	4	0	3	3	0	
P2	1	3	5	4	2	3	5	6					1	0	0	2	
P3	0	6	3	2	0	6	5	2					0	0	2	0	
P4	0	0	1	4	0	6	5	6					0	6	4	2	

After this find out safe sequence using safety algorithm (same as first one)

At first

P0 need(0 0 0 0)<=Available(1 1 1 2) P0 is executed

New available = available + allocation

New available = 1 1 1 2 + 0 0 1 2

NEW available =1 1 2 4

Question2:

Consider a system with four process having following need and allocation matrix of resources. Use banker algorithm to find safe state that have total of (6, 11) unit resource.

Process	Allocation		Max		Need remain	
	R1	R2	R1	R2	R1	R2
P1	2	4			0	5
P2	0	2			1	2
P3	2	3			0	4
P4	1	0			2	5

Solution

Process	Allocation		Max		Available		Need remain	
	R1	R2	R1	R2	R1	R2	R1	R2
P1	2	4	2	9	1	2	0	5
P2	0	2	1	4			1	2
P3	2	3	2	7			0	4
P4	1	0	3	5			2	5

Total R1 = 6, R2=11

Allocated R1 = 2+0+2+1 = 5, R2 = 4+2+3+0 = 9

Remaining available = total – allocated

For R1:

Remaining available = total R1 – allocated R1

Remaining available = 6 – 5 = 1

For R2:

Remaining available = total R2 – allocated R2

Remaining available = 11 – 9 = 2

For calculating maximum:

Need = max – allocation

Max = need + allocation

For P1:

Max for R1 = need for R1 + allocated for R1

Max for R1 = 0+2 = 2

Max for R2 = need for R2 + allocated for R2

Max for R2 = 5+ 4 = 9

Similarly calculated for P2 P3 and P4

Process	Allocation		Max		Available		Need remain		Sequence
	R1	R2	R1	R2	R1	R2	R1	R2	
P1	2	4	2	9	1	2	0	5	P2
P2	0	2	1	4	1	4	1	2	P3
P3	2	3	2	7	3	7	0	4	P4
P4	1	0	3	5	4	7	2	5	P1

For safe sequence:

At first:

P2's need (1 2)< = available (1 2), so P2 is executed first

New available = available + allocation

New available = 1 2 + 0 2 = 1 4

At second :

P3's need (0 4) <= available (1 4), so P3 is executed second.

New available = available + allocation

New available = 1 4 + 2 3 = 3 7

At third:

P4's need (2 5) < = available (3 7)

, so p4 is executed at third.

New available = available + allocation

New available = 3 7 + 1 0 = 4 7

At last:

P1's need (0 5) <= available(4 7), so p1 is executed last

New available = available + allocation

New available = 4 7 + 0 5 = 4 12

Since all the process are executed, system is in safe state. Safe sequence is P2 => P3=> P4=> P1