1026

①

Proces	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

a) Need Matrix
(max - Allocation)

Proces	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

b) Safety Algorithm

```
⇒   if  need ≤ available
    then
        execute ~~resource~~ proces
        new available = available + allocation
    else
        don't execute & move forward to next process
```

② P_0 → need ≤ available

$(0,0,0,0) \leq (1,5,2,0)$ so, execute P_0

new available $= (1,5,2,0) + (0,0,1,2)$

$= (1,5,3,2)$

⊕ P_1 → need > available
(0,7,5,0) > (1,5,3,2) So, don't execute P_1

⊕ P_2 → need ≤ available
(1,0,0,2) ≤ (1,5,3,2) So, execute P_2

new available = available + allocation
= (1,5,3,2) + (1,3,5,4)
= (2,8,8,6)

⊕ P_3 → need ≤ available
(0,0,2,0) ≤ (2,8,8,6) So, execute P_3

new available = available + allocation
= (2,8,8,6) + (0,6,3,2)
= (2,14,11,8)

⊕ P_4 → need ≤ available
(0,6,4,2) ≤ (2,14,11,8) So, execute P_4

new ~~allo~~ available = (2,14,11,8) + (0,0,1,4)
= (2,14,12,12)

Again,

⊕ P_1 → need ≤ available
(0,7,5,0) ≤ (2,14,12,12) So, execute P_1

new available = (2,14,12,12) + (1,0,0,0)
= (3,14,12,12)

⊕ Safe sequence;
⟨P_0, P_2, P_3, P_4, P_1⟩

c) If Process P_1 arrives for (0,4,2,0)

Resource Request Alg.

① If request ≤ need , goto step 2
else error

② If request ≤ available goto step 3
else wait

③ Available = Available - Request
Allocation = Allocation + Request
Need = Need - Request

④ Check now state is safe or not ?

⇒ Need for P_1 (0,7,5,0)

① Request ≤ Need ; (0,4,2,0) ≤ (0,7,5,0)

② Request ≤ Available ; (0,4,2,0) ≤ (1,5,2,0)

③ Available = Available - Request
= (1,5,2,0) - (0,4,2,0)
= (1,1,0,0)

Allocation = Allocation + Request
= (1,0,0,0) + (0,4,2,0)
= (1,4,2,0)

Need = Need - Request
= (0,7,5,0) - (0,4,2,0)
= (0,3,3,0)

④ Now, new snapshot will be

	Allocation				Max				Available			
Process	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	1	0	0
P_1	1	4	2	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

Need Matrix

Process	A	B	C	D
P_0	0	0	0	0
P_1	0	3	3	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

Again, doing safety algorithm

P_0 ⇒ need ≤ available ; (0,0,0,0) ≤ (1,1,0,0) so, execute P_0
new available = (1,1,0,0) + (0,0,1,2) = (1,1,1,2)

P_1 ⇒ (0,3,3,0) > (1,1,1,2) don't execute P_1

P_2 ⇒ (1,0,0,2) ≤ (1,1,1,2) execute P_2
new available = (1,1,1,2) + (1,3,5,4) = (2,4,6,6)

P_3 ⇒ (0,0,2,0) ≤ (2,4,6,6) execute P_3
new available = (2,4,6,6) + (0,6,3,2) = (2,10,9,8)

P_4 ⇒ (0,6,4,2) ≤ (2,10,9,8) execute P_4
new available = (2,10,9,8) + (0,0,1,4) = (2,10,10,12

P_1 ⇒ (0,3,3,0) ≤ (2,10,10,12) execute P_1
new available = (2,10,10,12) + (1,4,2,0) = (3,14,12,12)

< P_0, P_2, P_3, P_4, P_1 >