

# Chapter 7

## — 例题与作业



- 根据资源、进程间的请求关系，画出Resource-Allocation Graph
- 根据资源分配图分析死锁出现的资源条件，或避免死锁的资源条件
- Deadlock Avoidance
  - 当每类资源只有一个资源实例，利用Resource-Allocation Graph Algorithm 判断有无死锁/系统是否安全
  - 当资源可有多个资源实例时，利用Banker Algorithm判断系统是否安全?(无死锁?)，进程的资源请求是否be granted
- Deadlock Detecting
  - 当每类资源只有一个资源实例，利用waiting graph判断有无死锁?
  - 当资源可有多个资源实例时，利用deadlock detecting algorithm 判断有死锁?

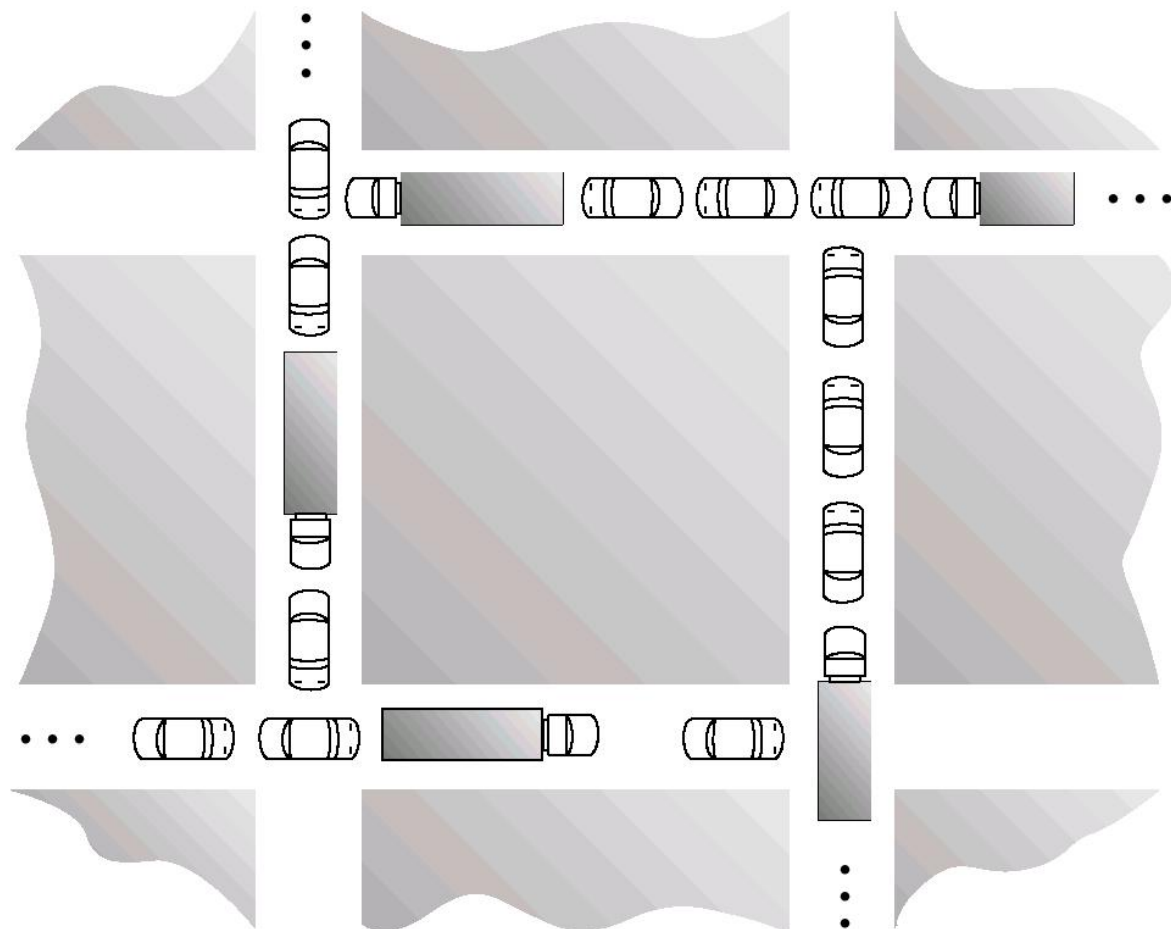
## Examples

- Resource Allocation Graph/Cycle and Deadlock
  - 例1. 道路交通与资源分配状态
  - 例2. 导致死锁发生的最大资源数/保证死锁不发生的最少资源数
  - 例3. Fig. 7.3, Fig. 7.4
  - 例4. 408选择题: 发生死锁的最少进程数
- Banker's Algorithm
  - 例5. /补充作业1
  - 例6.
- Deadlock Detection
  - 例7.
  - 补充作业2

7.5, 7.7, 7.11

补充作业1、2

## 例1. 道路交通与资源分配状态



Traffic Deadlock for Exercise 7.4

## 例1. 道路交通与资源分配状态

- 利用进程-资源模型描述 Fig.7.9
  - 进程—汽车，资源—道路，  
进程执行——汽车前进  
资源单位—道路长度，  
资源实例—一个单位长度的道路
  - 进程执行（由k时刻进展至k+1时刻）：车辆前行一个单位长度
  - 道路：4个路口A，B，C，D，长度为一个单位（对应一个资源实例）
  - 路口间4条直道：长度为4个单位，对应4个资源实例
  - 小车：长度为1，每前进一步需要使用2个道路资源实例，然后再释放一个道路资源实例
  - 大车：长度为2，每前进一步需要使用3个道路资源实例，然后再释放一个道路资源实例
- Available, Allocation, Need, MAX, by means of Banker algorithm

- 某系统有K台互斥使用的同类设备，n=3个并发进程分别需要使用2、3、4台设备，可确保系统发生死锁的**最大设备数目K**为多少？

- 答案

- 资源种类数m=1，K个资源实例
- 导致发生死锁的

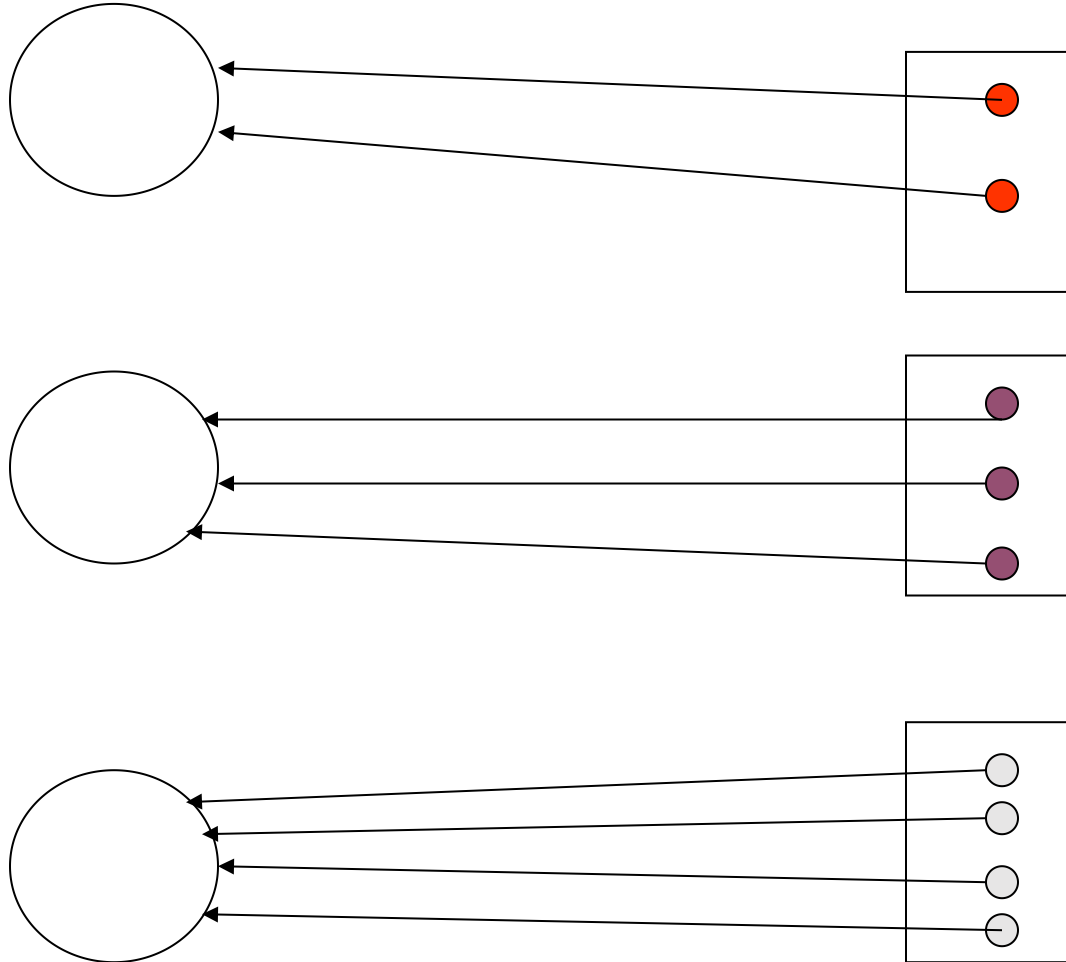
$$K = \text{所有进程的设备资源总需求} L - \text{进程总数} n \\ = (2 + 3 + 4) - 3 = 6$$

- 确保不发生死锁的最少资源实例总数为：

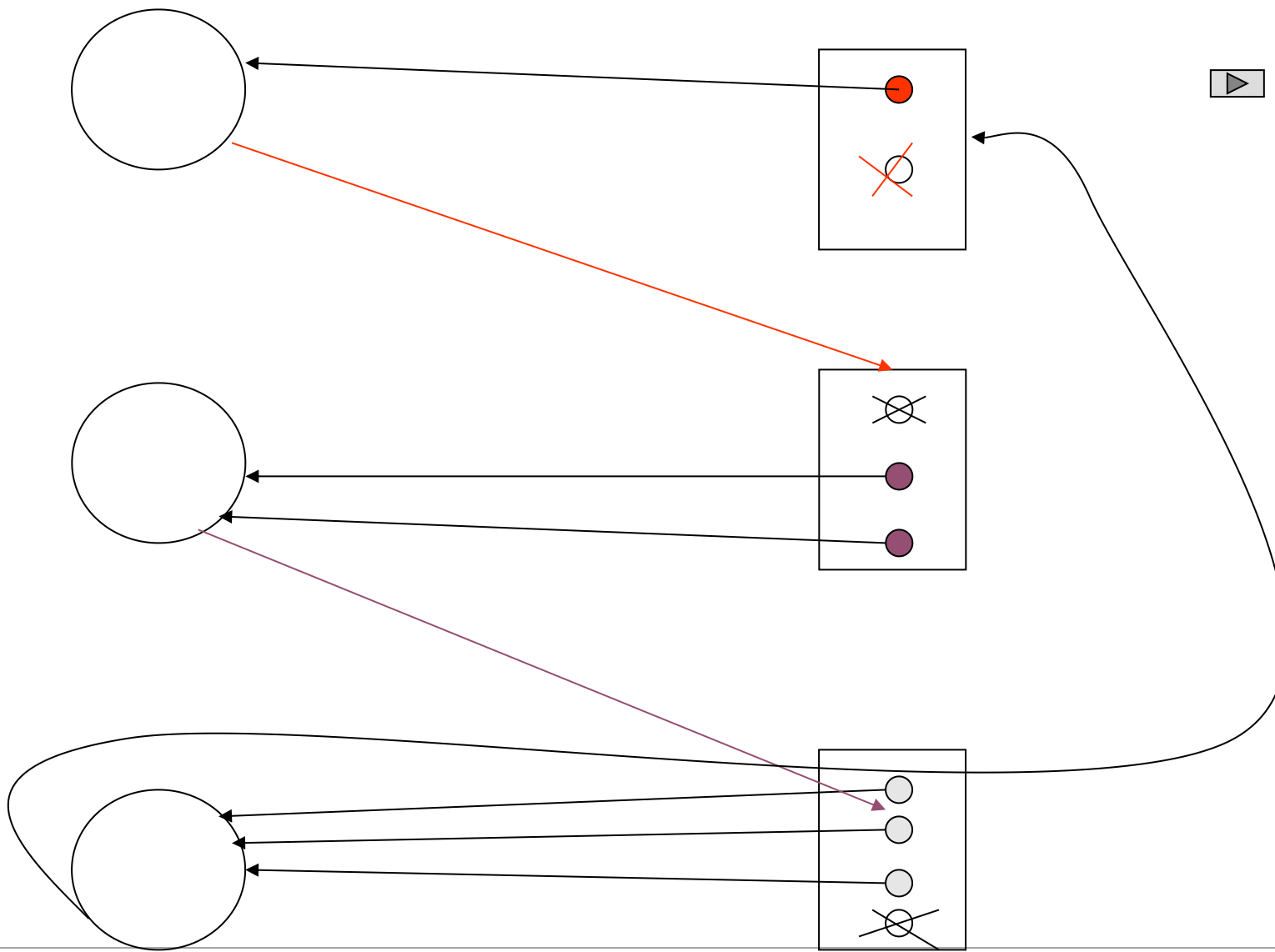
$$K+1 = \text{所有进程的设备资源总需求} L - (\text{进程总数} n - 1) \\ = (2 + 3 + 4) - (3 - 1) = 7$$

## 01 Example 2 Resources and Deadlock

- $L=9$ ,  $K=9$ ,  $n=3$ 个进程的设备需求完全满足, 进程无等待



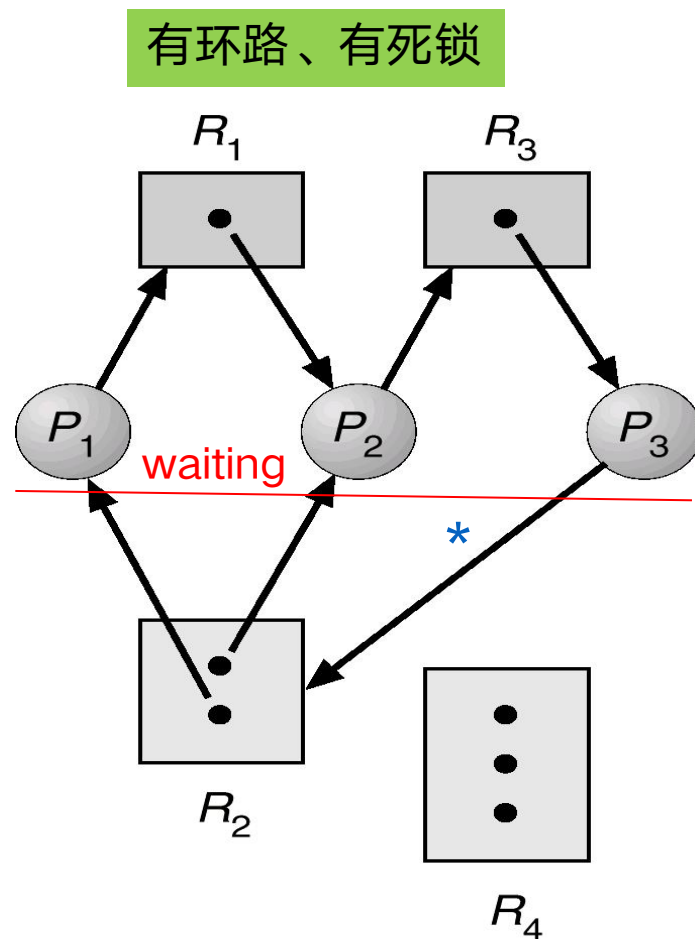
●  $L=9$ ,  $K=6$ ,  $n=3$ 个进程的需求不满足, 每个进程缺少1个资源, 形成循环等待





## ● Fig. 7.3

- 进程数目 $n=3$
- 有访问需求的资源总类 $m=3$ ,
- $R_1, R_2, R_3$ 的资源实例总数  
 $= (1 + 2 + 1) = 4$
- 3个进程对资源的总需求  
 $L = (2 + 3 + 2) = 7$
- $K = L - n = 7 - 3 = 4$   
, 等于目前资源实例总数, 发生死锁



F.g.7.3

## ● Fig. 7.4

- 进程数目  $n=4$
- 资源总类  $m=2$
- 2类资源的实例总数  
 $= (2 + 2) = 4$
- 4个进程对资源的总需求  $L$   
 $L = (2 + 1 + 2 + 1) = 6$
- 导致发生死锁的  
 $K = L - n = 6 - 4 = 2$

- 目前，2类资源的实例总数  $4 \geq K + 1 = 3$ ，  
无死锁

- P2、P4没有处于环路中

有环路、无死锁

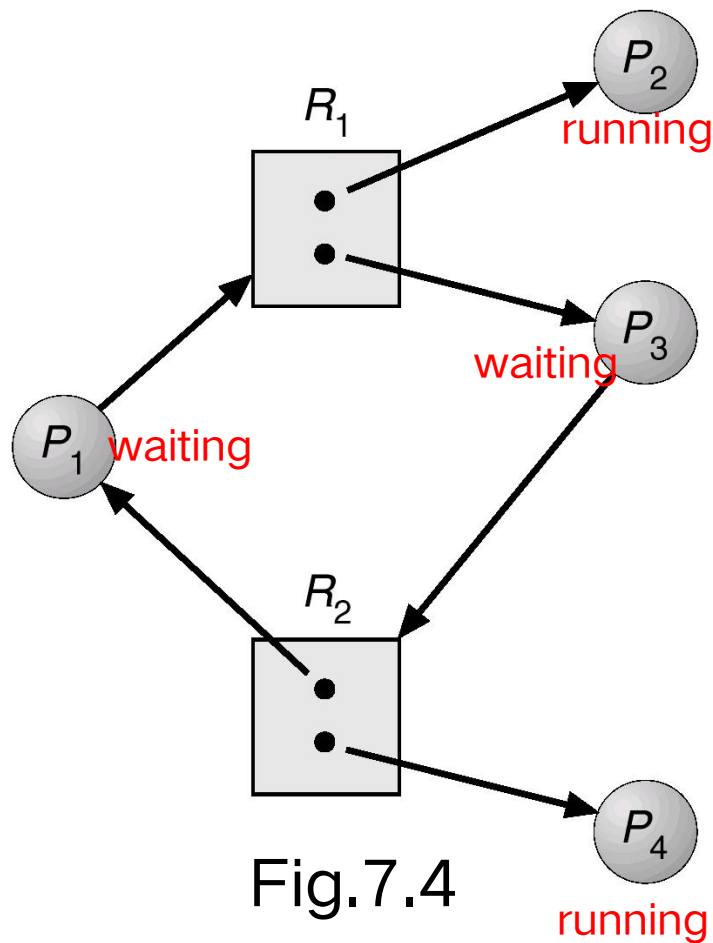


Fig.7.4

## 例4. 408选择题

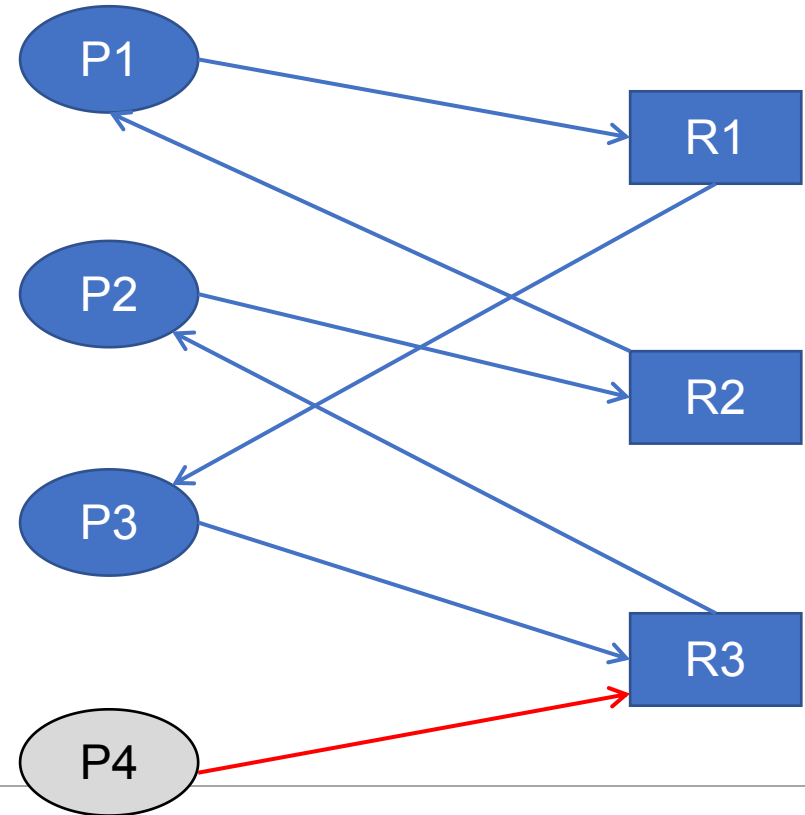
- 25. 系统中有3个不同的临界资源R1、R2和R3，被4个进程P1、P2、P3及P4共享。各进程对资源的需求为：P1申请R1和R2，P2申请R2和R3，P3申请R1和R3，P4申请R2。若系统出现死锁，则处于死锁状态的进程数至少是

A. 1    B. 2    C. 3    D. 4

- 答案

- D, 进程数目 $n=4$
- 可用资源数3, 资源总需求 $L=2+2+2+1=7$ ,  
 $L - n = 7 - 4 = 3$
- 4个进程, 每个进程均申请一个被其它进程占有的资源, 前3个形成环路:

$P1 \rightarrow R1 \rightarrow P3 \rightarrow R3 \rightarrow P2 \rightarrow R2 \rightarrow P1$



## 例5. Banker's Algorithm

- There are 5 processes  $P_0, P_1, P_2, P_3, P_4$ ; 3 resource types  $A$  with 10 instances (e.g. memory),  $B$  with 5 instances (disk), and  $C$  with 7 instances (tapes)
  - snapshot at time  $T_0$ :

	<u>Allocation</u>			<u>Max</u>		
	$A$	$B$	$C$	$A$	$B$	$C$
$P_0$	0	1	0	7	5	3
$P_1$	2	0	0	3	2	2
$P_2$	3	0	2	9	0	2
$P_3$	2	1	1	2	2	2
$P_4$	0	0	2	4	3	3

<u>Available</u>	<u>Need=MAX —Allocation</u>		
A B C	A B C		
3 3 2	7 4 3		
	1 2 2		
	6 0 0		
	0 1 1		
	4 3 1		

total : 6

total resources  
: (10 5 7)

## 例5. Banker's Algorithm

- total resources=(10 5 7)
- *Need* is defined to be  $Max - Allocation$   

$$Need = MAX - Allocation$$
- apply safety algorithm, for sequence  $\langle P_1, P_3, P_4, P_2, P_0 \rangle$

Need: A B C      Work

step5. $P_0$	7 4 3	< (7 4 5) + Allocat2(3 0 2) =(10 4 7)
step1. $P_1$	<u>1 2 2</u>	< (3 3 2)
Step4. $P_2$	6 0 0	< (7 4 3) + Allocat4(0 0 2) =(7 4 5)
step2. $P_3$	0 1 1	< (3 3 2) + Allocat 1 (2 0 0)=(5 3 2)
Step3. $P_4$	4 3 1	< (5 3 2) + Allocat3 (2 1 1)=(7 4 3)

- In the current situation, the system is in a safe state since the sequence  $\langle P_1, P_3, P_4, P_2, P_0 \rangle$  satisfies safety criteria

## 例5. Banker's Algorithm

- decide whether or not the  $P_1$ 's request(1, 0, 2) can be granted
  - request (1, 0, 2) < Need<sub>1</sub> = (1, 2, 2)
  - check that Request ≤ Available (that is, (1, 0, 2) ≤ (3, 3, 2) ⇒ true),  
(1 0 2) are allocated to  $P_1$ , and system enters **a new safety state**

	<u>Allocation</u>			<u>Need</u>			<u>Available</u>		
	A	B	C	A	B	C	A	B	C
$P_0$	0	1	0	7	4	3	2	3	0
$P_1$	3	0	2	0	2	0	3	3	2
$P_2$	3	0	1	6	0	0	2	-1	0
$P_3$	2	1	1	0	1	1	1	2	2
$P_4$	0	0	2	4	3	1	-1	0	0

Callouts showing calculations for Available resources:  
 - For  $P_1$ :  $2\ 3\ 0 + 3\ 3\ 2 = 5\ 6\ 2$  (Note: the image shows 2 3 0 and 3 3 2, resulting in 5 6 2)  
 - For  $P_2$ :  $1\ 2\ 2 + 6\ 0\ 0 = 7\ 2\ 2$  (Note: the image shows 1 2 2 and 6 0 0, resulting in 7 2 2)  
 - For  $P_4$ :  $2\ 0\ 0 + 4\ 3\ 1 = 6\ 3\ 1$  (Note: the image shows 2 0 0 and 4 3 1, resulting in 6 3 1)

- executing safety algorithm shows that sequence  $\langle P_1, P_3, P_4, P_0, P_2 \rangle$  satisfies safety requirement

## 补充作业1

- Can request for (3, 3, 0) by  $P_4$  be granted, and why?
- Can request for (0, 2, 0) by  $P_0$  be granted , and why?

## 例6. Banker's Algorithm

- For the system described in the following table
  - (a) is the system in a safe or unsafe state ? Why ?
  - (b) if  $P_3$  request resource of  $(0, 1, 0, 0)$ , can resources be allocated to it ? Why ?

process	Current allocation				Maximum allocation				Resource available			
	$R_1$	$R_2$	$R_3$	$R_4$	$R_1$	$R_2$	$R_3$	$R_4$	$R_1$	$R_2$	$R_3$	$R_4$
$P_1$	0	0	1	2	0	0	1	2	2	1	0	0
$P_2$	2	0	0	0	2	7	5	0				
$P_3$	0	0	3	4	6	6	5	6				
$P_4$	2	3	5	4	4	3	5	6				
$P_5$	0	3	3	2	0	6	5	2				



## 01 例6. Banker's Algorithm

- 答案

- ▪ 1

- the system is in a safe state, because there exists a safe process sequence  $\langle P_1, P_4, P_5, P_2, P_3 \rangle$
- the *Need* Matrix (=MAX - Allocation) is

	$R_1$	$R_2$	$R_3$	$R_4$
$P_1$	0	0	0	0
$P_2$	0	7	5	0
$P_3$	6	6	2	2
$P_4$	2	0	0	2
$P_5$	0	3	2	0

## 例6. Banker's Algorithm

- according to safety algorithm
  - firstly,  $Work := (2, 1, 0, 0)$ ,  $Finish[i] := \text{false}$   $i := 1, 2, 3, 4, 5$
  - for  $i := 1$ ,  $Need1 = (0, 0, 0, 0) < Work$ ,  $Finish[1] := \text{true}$ , and  $Work := (2, 1, 1, 2)$
  - for  $i := 4$ ,  $Need4 = (2, 0, 0, 2) < Work$ ,  $Finish[4] := \text{true}$ , and  $Work := (4, 4, 6, 6)$
  - for  $i := 5$ ,  $Need5 = (0, 3, 2, 0) < Work$ ,  $Finish[5] := \text{true}$ , and  $Work := (4, 7, 9, 8)$
  - for  $i := 2$ ,  $Need2 = (0, 7, 5, 0) < Work$ ,  $Finish[2] := \text{true}$ , and  $Work := (6, 7, 9, 8)$
  - for  $i := 3$ ,  $Need3 = (6, 6, 2, 2) < Work$ ,  $Finish[3] := \text{true}$ , and  $Work := (6, 7, 12, 12)$
  - $Finish[i] = \text{true}$  for  $i := 1, 2, 3, 4, 5$ , so the system is in a safe state.

## 例6. Banker's Algorithm

---

- Solution-(b)
  - No, because if the resources are allocated to  $P_3$ , then the system will be in a unsafe state
  - matrix *Allocation* , *Need* , and *Available* are as follows:

## 例6. Banker's Algorithm

proc ess	Current allocation				Need				Available			
	R <sub>1</sub>	R <sub>2</sub>	R <sub>3</sub>	R <sub>4</sub>	R <sub>1</sub>	R <sub>2</sub>	R <sub>3</sub>	R <sub>4</sub>	R <sub>1</sub>	R <sub>2</sub>	R <sub>3</sub>	R <sub>4</sub>
P <sub>1</sub>	0	0	1	2	0	0	0	0	2	0	0	0
P <sub>2</sub>	2	0	0	0	0	7	5	0				
P <sub>3</sub>	0	1	3	4	6	5	2	2				
P <sub>4</sub>	2	3	5	4	2	0	0	2				
P <sub>5</sub>	0	3	3	2	0	3	2	0				

## 例6. Banker's Algorithm

- According to safety algorithm

- firstly,  $Work := (2, 0, 0, 0)$   $Finish[i] := false$   $i := 1, 2, 3, 4, 5$
- for  $i := 1$ ,  $Need1 = (0, 0, 0, 0) < Work$ ,  $Finish[1] := true$ , and  $Work := (2, 0, 1, 2)$
- for  $i := 4$ ,  $Need4 = (2, 0, 0, 2) < Work$ ,  $Finish[4] := true$ , and  $Work := (4, 3, 6, 6)$
- for  $i := 5$ ,  $Need5 = (0, 3, 2, 0) < Work$ ,  $Finish[5] := true$ , and  $Work := (4, 6, 9, 8)$
- for  $i := 2$ ,  $Need2 = (0, 7, 5, 0) < Work$  is not true;  
and for  $i := 3$ ,  $Need3 = (6, 5, 2, 2) < Work$  is not true
- It is not true that  $Finish[i] = true$  for  $i := 1, 2, 3, 4, 5$

## 注意

关于安全状态、安全序列是针对死锁避免、不是针对死锁检测的  
死锁检测中无安全状态、安全序列的概念，只有死锁、非死锁状态

## Example 7. Deadlock Detection

- Five processes  $P_0$  through  $P_6$ ; three resource types A with 7 instances, B with 2 instances, and C with 6 instances
- Snapshot at time  $T_0$

	<i>Allocation</i>	<i>Request</i>	<i>Available</i>	<i>Finish</i>
	A B C	A B C	A B C	
$P_0$	0 1 0	0 0 0	0 0 0	
$P_1$	2 0 0	2 0 2		
$P_2$	3 0 3	0 0 0		
$P_3$	2 1 1	1 0 0		
$P_4$	0 0 2	0 0 2		
$P_5$	0 0 0	0 0 0		true
$P_6$	0 0 0	3 2 6		true

## Example 7. Deadlock Detection

- The system is not in a deadlock state, because
  - the sequence  $\langle P_0, P_2, P_3, P_1, P_4 \rangle$  and will result in  $\text{Finish}[i] = \text{true}$  for all  $i$ .  
Or:  $\langle P_2, P_0, P_3, P_1, P_4 \rangle$
- $P_2$  requests an additional instance of type C

	<u>Request</u>		
	A	B	C
$P_0$	0	0	0
$P_1$		2	0 2
$P_2$		0 0	<u>1</u>
$P_3$		1	0 0
$P_4$		0	0 2



## Example 7. Deadlock Detection

- then the system is in a deadlock state, processes  $P_1$ ,  $P_2$ ,  $P_3$ , and  $P_4$  is deadlocked



	Allocation	Request	Available
	A B C	A B C	A B C
$P_0$	0 1 0	0 0 0	0 0 0
$P_1$	2 0 0	2 0 2	
$P_2$	3 0 3	<u>0 0 1</u>	
$P_3$	2 1 1	1 0 0	
$P_4$	0 0 2	0 0 2	

## 补充作业2

- Five processes  $P_0$  through  $P_4$ ; three resource types A with 6 instances, B with 3 instances, and C with 6 instances
- Given the snapshot at time  $T_0$ :

	Allocation	Request	Available
	A B C	A B C	A B C
$P_0$	0 1 0	0 0 0	1 0 1
$P_1$	2 0 1	2 0 2	
$P_2$	1 1 1	0 0 2	
$P_3$	2 1 1	1 0 0	
$P_4$	0 0 2	0 0 2	

- Answer the following questions on the basis of deadlock-detecting algorithm
  - is the system in a deadlock-free state? and why?
  - if  $P_2$  requests **two additional instance of type A**, is there a deadlock in the system, and why? And if the system is in deadlock, which process is in deadlock?



---

(6 points) Consider a paging system with the physical memory of  $m$  frames that are shared by concurrent processes  $P_1$ ,  $P_2$ , and  $P_3$ . The frames, viewed as resources of the same type, can be requested and released by the processes only one at time. The maximum numbers of the frames needed by  $P_1$ ,  $P_2$ , and  $P_3$  are 3, 4 and 5 respectively.

It is supposed that the frames allocated to a process are not preempted by the other processes, and the pages of the process are not swapped out.

Requesting frames by the processes may result in deadlock. To guarantee the system is deadlock free, what is the minimum value of  $m$ , and why?