Homework 8

余北辰 519030910245

8.3 Consider the following snapshot of a system:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
T_0	0 0 1 2	0 0 1 2	1 5 2 0
T_1	1 0 0 0	1 7 5 0	
T_2	1 3 5 4	2 3 5 6	
T_3	0 6 3 2	0 6 5 2	
T_4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix *Need*?

b. Is the system in a safe state?

c. If a request from thread T_1 arrives for $(0, 4, 2, 0)$, can the request be granted immediately?

a. Need = Max - Allocation

	Need
	A B C D
T_0	0 0 0 0
T_1	0 7 5 0
T_2	1 0 0 2
T_3	0 0 2 0
T_4	0 6 4 2

b.

先将资源分配给 T_0，结束后可用资源为 $(1, 5, 3, 2)$;

再将资源分配给 T_2，结束后可用资源为 $(2, 8, 8, 6)$;

再将资源分配给 T_3，结束后可用资源为 $(2, 14, 11, 8)$;

再将资源分配给 T_1，结束后可用资源为 $(3, 14, 11, 8)$；

再将资源分配给 T_4，结束后可用资源为 $(3, 14, 12, 12)$；

因此可以用银行家算法解决该问题，处于安全状态。

c. 如果该请求被立即准许：

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	0 0 1 2	0 0 1 2	0 0 0 0	1 1 0 0
T_1	1 4 2 0	1 7 5 0	0 3 3 0	
T_2	1 3 5 4	2 3 5 6	1 0 0 2	
T_3	0 6 3 2	0 6 5 2	0 0 2 0	
T_4	0 0 1 4	0 6 5 6	0 6 4 2	

先将资源分配给 T_0，结束后可用资源为 $(1, 1, 1, 2)$；

再将资源分配给 T_2，结束后可用资源为 $(2, 4, 6, 6)$；

再将资源分配给 T_3，结束后可用资源为 $(2, 10, 9, 8)$；

再将资源分配给 T_1，结束后可用资源为 $(3, 14, 11, 8)$；

再将资源分配给 T_4，结束后可用资源为 $(3, 14, 12, 12)$；

因此可以用银行家算法解决该问题，处于安全状态。

因此该请求可以被立即准许。

8.9 Consider the following snapshot of a system:

	Allocation	Max
	A B C D	A B C D
T_0	3 0 1 4	5 1 1 7
T_1	2 2 1 0	3 2 1 1
T_2	3 1 2 1	3 3 2 1
T_3	0 5 1 0	4 6 1 2
T_4	4 2 1 2	6 3 2 5

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

a. Available = $(0, 3, 0, 1)$

b. Available = $(1, 0, 0, 2)$

a.

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	3 0 1 4	5 1 1 7	2 1 0 3	0 3 0 1
T_1	2 2 1 0	3 2 1 1	1 0 0 1	
T_2	3 1 2 1	3 3 2 1	0 2 0 0	
T_3	0 5 1 0	4 6 1 2	4 1 0 2	
T_4	4 2 1 2	6 3 2 5	2 1 1 3	

先将资源分配给 T_2，结束后可用资源为 $(3, 4, 2, 2)$；

再将资源分配给 T_1，结束后可用资源为 $(5, 6, 3, 2)$；

再将资源分配给 T_3，结束后可用资源为 $(5, 11, 4, 2)$；

之后找不到可以分配的对象，因此出现死锁。

因此此状态非安全。

b.

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	3 0 1 4	5 1 1 7	2 1 0 3	1 0 0 2
T_1	2 2 1 0	3 2 1 1	1 0 0 1	
T_2	3 1 2 1	3 3 2 1	0 2 0 0	
T_3	0 5 1 0	4 6 1 2	4 1 0 2	
T_4	4 2 1 2	6 3 2 5	2 1 1 3	

先将资源分配给 T_1，结束后可用资源为 $(3, 2, 1, 2)$；

再将资源分配给 T_2，结束后可用资源为 $(6, 3, 3, 3)$；

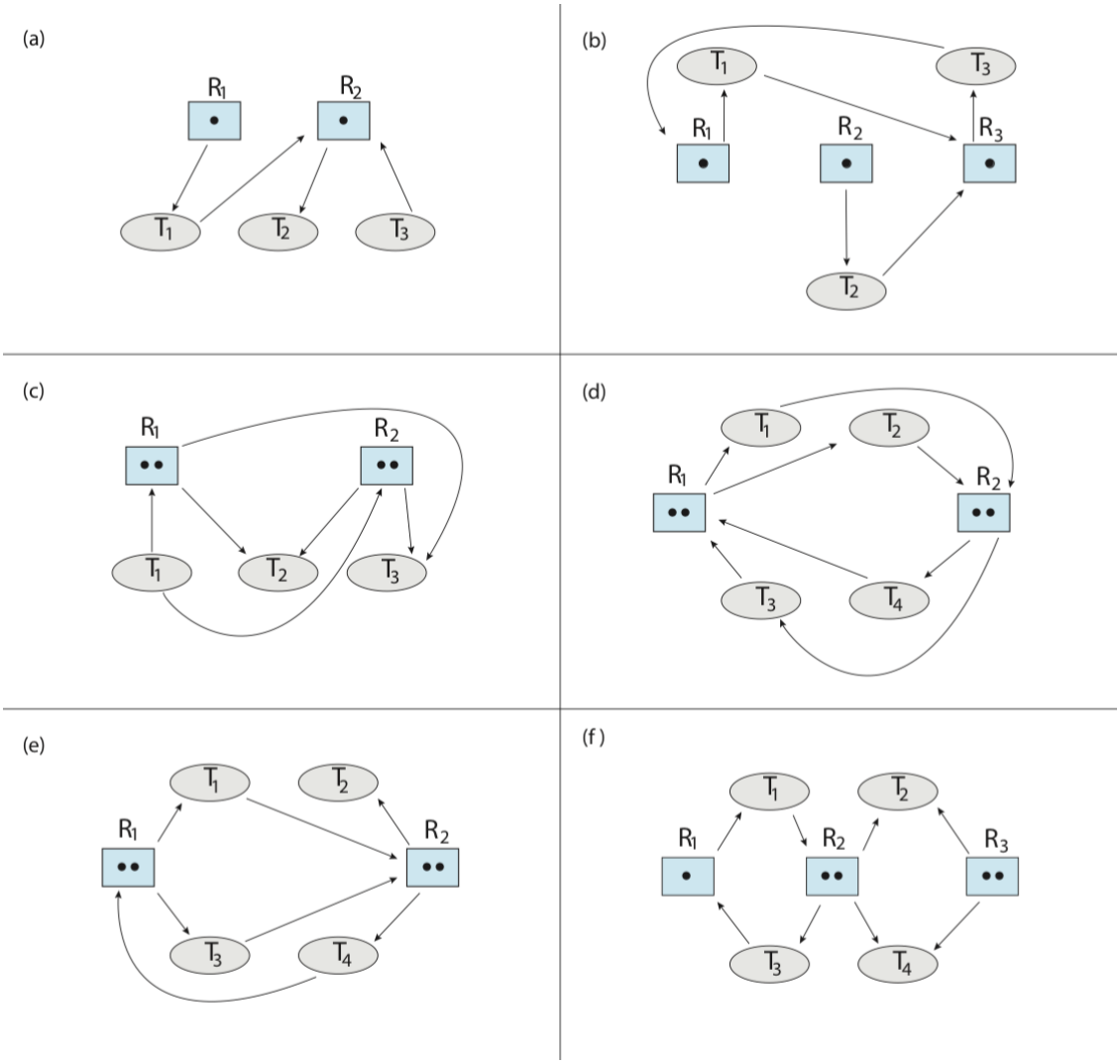
再将资源分配给 T_0，结束后可用资源为 $(9, 3, 4, 7)$；

再将资源分配给 T_3，结束后可用资源为 $(9, 8, 5, 7)$；

再将资源分配给 T_4，结束后可用资源为 $(13, 10, 6, 9)$；

因此可以用银行家算法解决该问题，处于安全状态。

8.18 Which of the six resource-allocation graphs shown in Figure 8.12 illustrate deadlock? For those situations that are deadlocked, provide the cycle of threads and resources. Where there is not a deadlock situation, illustrate the order in which the threads may complete execution.



(a) 无死锁。 无环。按照 $T_2 \to T_3 \to T_1$ 的顺序就能完成。

(b) 死锁。 $R_1 \to T_1 \to R_3 \to T_3 \to R_1$ 形成环，且 T_2 也在申请 R_3，因此无法解开。

(c) 无死锁。 无环。按照 $T_2 \to T_3 \to T_1$ 的顺序就能完成。

(d) 死锁。 存在两个环：$R_1 \to T_1 \to R_2 \to T_3 \to R_1$ 和 $R_1 \to T_2 \to R_2 \to T_4 \to R_1$，且无法解开。

(e) 无死锁。存在环：$R_1 \to T_1 \to R_2 \to T_4 \to R_1$，但是先将 R_2 的一个实例分配给 T_2，再将该实例分配给 T_1，就能解开。按照 $T_2 \to T_1 \to T_3 \to T_4$ 的顺序解开。

(f) 无死锁。存在环：$R_1 \to T_1 \to R_2 \to T_3$，但是先将 R_2 和 R_3 的一个实例分配给 T_2，再将 R_2 的一个实例分配给 T_1，再对 T_4 重复就能解开。按照 $T_2 \to T_4 \to T_1 \to T_3$ 的顺序解开。

8.27 Consider the following snapshot of a system:

	Allocation	Max
	A B C D	A B C D
T_0	1 2 0 2	4 3 1 6
T_1	0 1 1 2	2 4 2 4
T_2	1 2 4 0	3 6 5 1
T_3	1 2 0 1	2 6 2 3
T_4	1 0 0 1	3 1 1 2

Using the banker's algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

a. Available = $(2, 2, 2, 3)$

b. Available = $(4, 4, 1, 1)$

c. Available = $(3, 0, 1, 4)$

d. Available = $(1, 5, 2, 2)$

a.

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	1 2 0 2	4 3 1 6	3 1 1 4	2 2 2 3
T_1	0 1 1 2	2 4 2 4	2 3 1 2	
T_2	1 2 4 0	3 6 5 1	2 4 1 1	
T_3	1 2 0 1	2 6 2 3	1 4 2 2	
T_4	1 0 0 1	3 1 1 2	2 1 1 1	

先将资源分配给 T_4，结束后可用资源为 $(3, 2, 2, 4)$；

再将资源分配给 T_0，结束后可用资源为 $(4, 4, 2, 6)$；

再将资源分配给 T_1，结束后可用资源为 $(4, 5, 3, 8)$；

再将资源分配给 T_2，结束后可用资源为 $(5, 7, 7, 8)$；

再将资源分配给 T_3，结束后可用资源为 $(6, 7, 7, 9)$；

因此可以用银行家算法解决该问题，处于安全状态。

b.

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	1 2 0 2	4 3 1 6	3 1 1 4	4 4 1 1
T_1	0 1 1 2	2 4 2 4	2 3 1 2	
T_2	1 2 4 0	3 6 5 1	2 4 1 1	
T_3	1 2 0 1	2 6 2 3	1 4 2 2	
T_4	1 0 0 1	3 1 1 2	2 1 1 1	

先将资源分配给 T_2，结束后可用资源为 $(5,6,5,1)$；

再将资源分配给 T_4，结束后可用资源为 $(6,6,5,2)$；

再将资源分配给 T_1，结束后可用资源为 $(6,7,6,4)$；

再将资源分配给 T_0，结束后可用资源为 $(7,9,6,6)$；

再将资源分配给 T_3，结束后可用资源为 $(8,11,6,7)$；

因此可以用银行家算法解决该问题，处于安全状态。

c.

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	1 2 0 2	4 3 1 6	3 1 1 4	3 0 1 4
T_1	0 1 1 2	2 4 2 4	2 3 1 2	
T_2	1 2 4 0	3 6 5 1	2 4 1 1	
T_3	1 2 0 1	2 6 2 3	1 4 2 2	
T_4	1 0 0 1	3 1 1 2	2 1 1 1	

先将资源分配给 T_0，结束后可用资源为 $(4,2,1,6)$；

再将资源分配给 T_4，结束后可用资源为 $(5,2,1,7)$；

之后找不到可以分配的对象，因此出现死锁。

因此此状态非安全。

d.

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	1 2 0 2	4 3 1 6	3 1 1 4	1 5 2 2
T_1	0 1 1 2	2 4 2 4	2 3 1 2	
T_2	1 2 4 0	3 6 5 1	2 4 1 1	
T_3	1 2 0 1	2 6 2 3	1 4 2 2	
T_4	1 0 0 1	3 1 1 2	2 1 1 1	

先将资源分配给 T_3，结束后可用资源为 $(2,7,2,3)$；

再将资源分配给 T_1，结束后可用资源为 $(2,8,3,5)$；

再将资源分配给 T_2，结束后可用资源为 $(3,10,7,5)$；

再将资源分配给 T_0，结束后可用资源为 $(4,12,7,7)$；

再将资源分配给 T_4，结束后可用资源为 $(5,12,7,8)$；

因此可以用银行家算法解决该问题，处于安全状态。

8.28 Consider the following snapshot of a system:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	2 2 2 4
T_1	2 1 0 2	4 2 3 2	
T_2	2 4 1 3	2 5 3 3	
T_3	4 1 1 0	6 3 3 2	
T_4	2 2 2 1	5 6 7 5	

Answer the following questions using the banker's algorithm:

a. Illustrate that the system is in a safe state by demonstrating an order in which the threads may complete.

b. If a request from thread T_4 arrives for $(2,2,2,4)$, can the request be granted immediately?

c. If a request from thread T_2 arrives for $(0,1,1,0)$, can the request be granted immediately?

d. If a request from thread T_3 arrives for $(2,2,1,2)$, can the request be granted immediately

a.

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	3 3 3 2	2 2 2 4
T_1	2 1 0 2	4 2 3 2	2 1 3 0	
T_2	2 4 1 3	2 5 3 3	0 1 2 0	
T_3	4 1 1 0	6 3 3 2	2 2 2 2	
T_4	2 2 2 1	5 6 7 5	3 4 5 4	

先将资源分配给 T_2，结束后可用资源为 $(4,6,3,7)$；

再将资源分配给 T_0，结束后可用资源为 $(7,7,7,8)$；

再将资源分配给 T_1，结束后可用资源为 $(9,8,7,10)$；

再将资源分配给 T_3，结束后可用资源为 $(13,9,8,10)$；

再将资源分配给 T_4，结束后可用资源为 $(15,11,10,11)$；

因此可以用银行家算法解决该问题，处于安全状态。

b. 如果该请求被立即准许：

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	3 3 3 2	0 0 0 0
T_1	2 1 0 2	4 2 3 2	2 1 3 0	
T_2	2 4 1 3	2 5 3 3	0 1 2 0	
T_3	4 1 1 0	6 3 3 2	2 2 2 2	
T_4	4 4 4 5	5 6 7 5	1 2 3 0	

找不到可以分配的对象，因此出现死锁。

因此此状态非安全。

因此该请求不能被立即准许。

c. 如果该请求被立即准许：

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	3 3 3 2	2 1 1 4
T_1	2 1 0 2	4 2 3 2	2 1 3 0	
T_2	2 5 2 3	2 5 3 3	0 0 1 0	
T_3	4 1 1 0	6 3 3 2	2 2 2 2	
T_4	2 2 2 1	5 6 7 5	3 4 5 4	

先将资源分配给 T_2，结束后可用资源为 $(4,6,3,7)$；

再将资源分配给 T_0，结束后可用资源为 $(7,7,7,8)$；

再将资源分配给 T_1，结束后可用资源为 $(9,8,7,10)$；

再将资源分配给 T_3，结束后可用资源为 $(13,9,8,10)$；

再将资源分配给 T_4，结束后可用资源为 $(15,11,10,11)$；

因此可以用银行家算法解决该问题，处于安全状态。

因此该请求可以被立即准许。

d. 如果该请求被立即准许：

	Allocation	Max	Need	Available
	A B C D	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	3 3 3 2	0 0 1 2
T_1	2 1 0 2	4 2 3 2	2 1 3 0	
T_2	2 4 1 3	2 5 3 3	0 1 2 0	
T_3	6 3 2 2	6 3 3 2	0 0 1 0	
T_4	2 2 2 1	5 6 7 5	3 4 5 4	

先将资源分配给 T_3，结束后可用资源为 $(6,3,3,4)$；

再将资源分配给 T_0，结束后可用资源为 $(9,4,7,5)$；

再将资源分配给 T_1，结束后可用资源为 $(11,5,7,7)$；

再将资源分配给 T_2，结束后可用资源为 $(13,9,7,10)$；

再将资源分配给 T_4，结束后可用资源为 $(15,11,10,10)$；

因此可以用银行家算法解决该问题，处于安全状态。

因此该请求可以被立即准许。