長庚大學106學年度第一學期 作業系統 第四次小考

系級: 姓名: 學號:

Banker's Algorithm is a deadlock avoidance algorithm. Assume there are 5 processes {P_0, P_1, P_2, P_3, P_4} and three types of shared resources {A, B, C} in the system, and the details are in the following table.

(1) By Banker's Algorithm, is the system in a safe state? If your answer is yes, please provide a safe sequence. If your answer is no, please provide the reason.

(2) Now, P_0 further has a request (1, 1 , 0) to use 1 more instance of A and 1 more instance of B. Should the request be granted? Again, provide the reason to support your answer.

(3) If we reject the request in (2), Now, P_3 further has a request (0, 1 , 1) to use 1 more instance of B and 1 more instance of C. Should the request be granted? Again, provide the reason to support your answer.

	Allocation			Max			Need			Available		
	A	B	C	A	B	C	A	B	C	A	B	C
P0	0	1	0	7	5	3	7	4	3	3	3	2
P1	1	0	1	2	4	3	1	4	2			
P2	3	0	2	9	0	2	6	0	0			
P3	0	1	1	0	2	2	0	1	1			
P4	2	1	1	6	4	2	4	3	1			

Answer:

(1) Yes.
Run the Banker's Algorithm: Available(3, 3, 2) → P3 Need(0, 1, 1) → Available(3, 4, 3) → P1 Need(1, 4, 2) → Available(4, 4, 4) → P4 Need(4, 3, 1) → Available(6, 5, 5) → P2 Need(6, 0, 0) → Available(9, 5, 7) → P0 Need(7, 4, 3)

(2) No.
Check 1: (1, 1 , 0) ≦ P0 Need(7, 4, 3)
Check 2: (1, 1 , 0) ≦ the current Available (3, 3, 2)
Check 3: Try to give the requested resources and run Banker's Algorithm
After the system grants the request: Available(3, 3, 2) → Available(2, 2, 2). P0 has Need(6, 3, 3) and Allocation(1, 2, 0).
We then run the Banker's Algorithm again: Available(2, 2, 2) → P3 Need(0, 1, 1) → Available(2, 3, 3) → No one can run now!

(3) Yes.
Check 1: (0, 1 , 1) ≦ P3 Need(0, 1, 1)
Check 2: (0, 1 , 1) ≦ the current Available (3, 3, 2)
Check 3: Try to give the requested resources and run Banker's Algorithm
After the system grants the request: Available(3, 3, 2) → Available(3, 2, 1). P3 has Need(0, 1, 1) and Allocation(0, 2, 2).
We then run the Banker's Algorithm again: Available(3, 2, 1) → P3 Need(0, 0, 0) → Available(3, 4, 3) → P1 Need(1, 4, 2) → Available(4, 4, 4) → P4 Need(4, 3, 1) → Available(6, 5, 5) → P2 Need(6, 0, 0) → Available(9, 5, 7) → P0 Need(7, 4, 3)