Tutorial deadlock

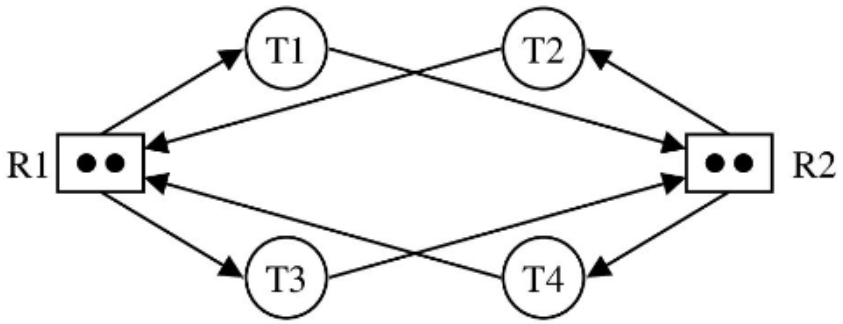
Problem 1

A system is having 3 user processes each requiring max 2 units of resource R.

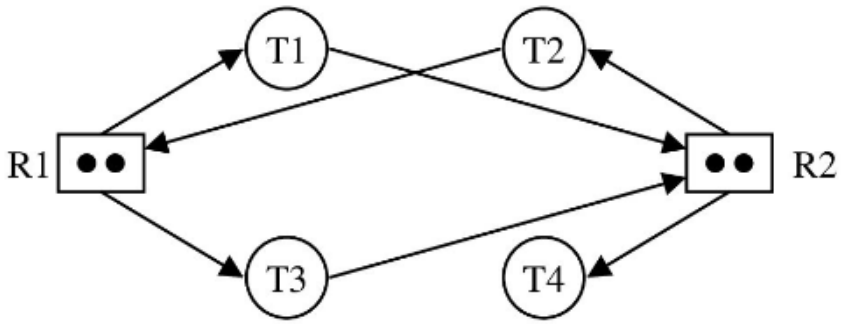
What is the minimum number of units of R such that no deadlock will occur?

Problem 2

Deadlock or not? Justify.



(a)



(b)

Problem 3

A single processor system has three resource types X, Y, and Z, which are shared by three processes. There are 5 units of each resource type.

Allocation

	X	Y	Z
P0	1	2	1
P1	2	0	1
P2	2	2	1

Request

	X	Y	Z
P0	1	0	3
P1	0	1	2
P2	1	2	0

(i) Is the system in a safe state? What is the safe sequence?
(ii) What will happen if process P_1 requests two additional instances of resource type C?
.

Answer:

(i) According to the question-
Total = [X Y Z] = [5 5 5] , Total _Allocation = [X Y Z] = [5 4 3]
Now, Available = Total – Total_Allocation = [5 5 5] – [5 4 3] = [0 1 2]

• Step: With the instances available currently, only the requirement of process P1 can be satisfied. So, process P1 is allocated the requested resources. It completes its execution and then frees up the instances of resources held by it.
(Then, Available = [0 1 2] + [2 0 1] = [2 1 3]

By repeating the above step, we will get the following

--→ P0, Available = [3 3 4]
-→ P2, Available = [5 5 5]
-→ There exists a safe sequence P1, P0, P2 in which all the processes can be executed.

(ii)New_Request = P1 [0 0 2], so Now, available becomes [0 1 0],

Problem 4

- S1=0, S2=0, S3=1;

P1 P2 P3

wait(S1); wait (S2); wait(S3)

--- ------

signal(S1); signal(S2); signal(S3)

Any deadlock?

Problem 5

- [Unsafe does not necessarily mean deadlock]
- One possibility: MaxNeed $_{ij}$ are upper bounds. Threads may or may not ask for so many resources.
- Assume that each T_i will ask for MaxNeed $_{ij}$ for each j (not necessarily together). Does this mean: Unsafe ⇒ Deadlock?

Suppose that there are two resources A and B with 12 and 1 instances, respectively. At some point of time, we have:

	MAX		ALLOC		NEED		AVAILABLE	
	A	B	A	B	A	B	A	B
P1	10	1	5	0	5	1	3	0
P2	4	1	2	0	2	1		
P3	9	1	2	1	7	0		

Unsafe?

Problem 5

- [Unsafe does not necessarily mean deadlock]
- One possibility: MaxNeed $_{ij}$ are upper bounds. Threads may or may not ask for so many resources.
- Assume that each T_i will ask for MaxNeed $_{ij}$ for each j (not necessarily together). Does this mean: Unsafe ⇒ Deadlock?

Suppose that there are two resources A and B with 12 and 1 instances, respectively. At some point of time, we have:

	MAX			ALLOC			NEED			AVAILABLE	
	A	B		A	B		A	B		A	B
P1	10	1	P1	5	0	P1	5	1		3	0
P2	4	1	P2	2	0	P2	2	1			
P3	9	1	P3	2	1	P3	7	0			

This is an unsafe state. However, the processes can still finish without encountering deadlock as follows.

P3 releases B. AVAILABLE = [3 1]
P2 requests [2 1], releases all the resources, and terminates. AVAILABLE = [5 1]
P1 requests [5 1], releases all the resources, and terminates. AVAILABLE = [10 1]
P3 requests [7 0], releases all the resources, and terminates. AVAILABLE = [12 1]

Here, P3 does not require [9 1] together. P1 and P2 hold their respective max needs at certain points of time.

Problem 6

Suppose that an OS uses deadlock avoidance using banker's algorithm. However, the OS restricts the threads to request for a resource of only one type at any point time. Multiple instances of that resource type are allowed in the request.

That is, each Request vector has exactly one non-zero entry.

In order to reduce deadlock-avoidance overhead, the OS runs banker's algorithm only on the requested resource type. That is, it grants the request if and only if all the threads can finish assuming that the requested resource type is the only resource type available. This reduces the running time from $O(mn^2)$ to $O(n^2)$.

Can this shortcut throw a system from a safe state to an unsafe state? Deadlock?

Suppose that there are two resources A and B, each with 12 instances. At some point of time, we have:

	MAX		ALLOC		NEED		AVAILABLE	
	A	B	A	B	A	B	A	B
P1	10	9	5	2	5	7	3	8
P2	4	4	2	2	2	2		
P3	9	10	2	0	7	10		

This is a safe state. A safe sequence is <P2, P1, P3>. Then, the following happens:

P3 asks for [0 5].

Suppose that there are two resources A and B, each with 12 instances. At some point of time, we have:

	MAX			ALLOC			NEED			AVAILABLE	
	A	B		A	B		A	B		A	B
P1	10	9	P1	5	2	P1	5	7		3	8
P2	4	4	P2	2	2	P2	2	2			
P3	9	10	P3	2	0	P3	7	10			

This is a safe state. A safe sequence is <P2, P1, P3>. Then, the following happens:

P3 asks for [0 5]. This request does not violate safety with respect to Resource B, and is therefore granted. Now, the state changes to:

	MAX			ALLOC			NEED			AVAILABLE	
	A	B		A	B		A	B		A	B
P1	10	9	P1	5	2	P1	5	7		3	3
P2	4	4	P2	2	2	P2	2	2			
P3	9	10	P3	2	5	P3	7	5			

This is unsafe. Only P2 can finish.