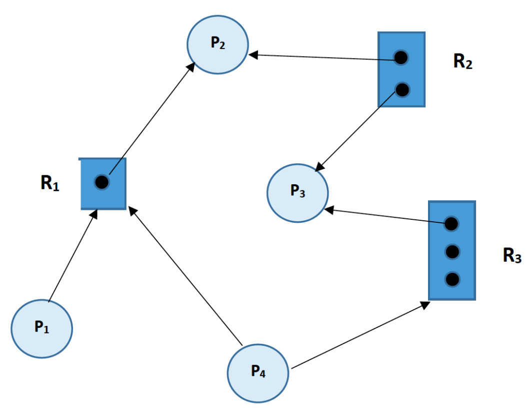
**Tutorial 5 – Deadlocks (Chapter 8)**

**Operating Systems Comp Sci 3SH3, Winter 2024**
Prof. Neerja Mhaskar

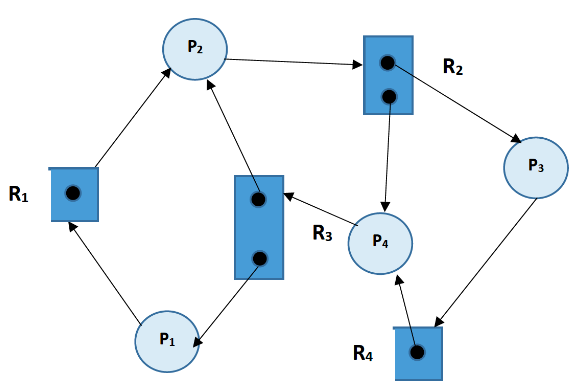
## Questions:

1. Consider the below resource allocation graph. Is the system in a deadlock state? If so, report the cycle(s) causing deadlock. If not, explain the order in which processes access the resources requested and complete execution.




**No deadlock. They complete execution in P2 -> P3 -> P1 -> P4**

2. Consider the below resource allocation graph. Is the system in a deadlocked state? If so, report the cycle(s) causing deadlock. If not, explain the order in which processes access the resources requested and complete execution.




**It is in a deadlock state. There is a cycles. It is**
**P1 -> P2 -> P3 -> P4 -> P1**
**P2 -> P3 -> P4 -> P2**

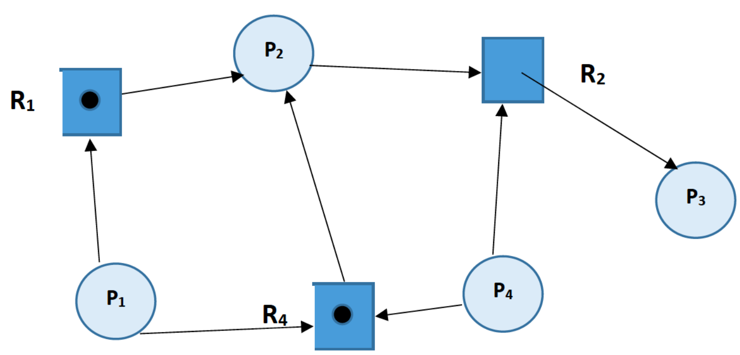
3. Consider the following snapshot of a system:

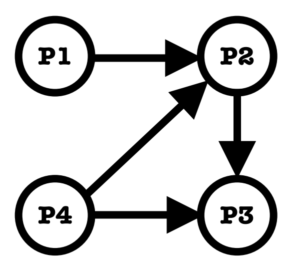
| | Allocation | Max | Available | Need |
|---|---|---|---|---|
| | A B C D | A B C D | A B C D | **A B C D** |
| $P_0$ | 0 0 1 2 | 0 0 1 2 | 1 5 2 0 | **0 0 0 0** |
| $P_1$ | 1 0 0 0 | 1 7 5 0 | | **0 7 5 0** |
| $P_2$ | 1 3 5 4 | 2 3 5 6 | | **1 0 0 2** |
| $P_3$ | 0 6 3 2 | 0 6 5 2 | | **0 0 2 0** |
| $P_4$ | 0 0 1 4 | 0 6 5 6 | | **0 6 4 2** |

Answer the following questions using the banker's algorithm:

a. What is the content of the matrix Need? **Answered above**
b. Is the system in a safe state? **Yes**
c. If a request from process $P_1$ arrives for (0,4,2,0), can the request be granted immediately? **Yes, because it's less than the work matrix and less than the max need the process requires**

4. Consider the below resource allocation graph. Construct the corresponding wait-for graph. Is the system in deadlock? If so, provide the cycle causing deadlock.




5. Consider the following snapshot of a system at time *T*0:

Five processes $P_0$ through $P_4$.

Three resource types A (10 instances), *B* (3 instances), and *C* (6 instances)

Snapshot at time $T_0$:

| | *Allocation* | *Request* | *Available* |
|---|---|---|---|
| | *A B C* | *A B C* | *A B C* |
| *P0* | 2 1 1 | 0 0 0 | 0 0 0 |
| *P1* | 2 1 2 | 2 0 2 | |
| *P2* | 4 0 0 | 0 0 1 | |
| *P3* | 2 1 1 | 1 0 0 | |
| *P4* | 0 0 2 | 0 0 2 | |

a) Is the system in deadlocked state? If no, provide a sequence of processes satisfying the safety requirement. If yes, explain why and list the processes involved in the deadlock.

b) Suppose process P1 makes an additional request of resource type B, the Request matrix is modified as follows:

| | *Request* |
|---|---|
| | *A B C* |
| $P_0$ | 0 0 0 |
| $P_1$ | 2 1 2 |
| $P_2$ | 0 0 1 |
| $P_3$ | 1 0 0 |
| $P_4$ | 0 0 2 |

c) Is the system in deadlocked state? If no, provide a sequence of processes satisfying the safety requirement. If yes, explain why and list the processes involved in the deadlock.

6. Consider a system consisting of four resources of the same type that are shared by three processes, each of which needs at most two resources. Show that the system is deadlock-free.

7. Consider the version of the dining-philosophers problem in which the chopsticks are placed at the center of the table and any two of them can be used by a philosopher. Assume that requests for chopsticks are made one at a time. Describe a simple rule for determining whether a particular request can be satisfied without causing deadlock given the current allocation of chopsticks to philosophers.