3.3) Draw a clear resource allocation graph based on the information below.

(The sets P - processes, R - resources, E- edges as follows)

P = {P1,P2,P3,P4}

R = {R1, R2, R3, R4, R5}

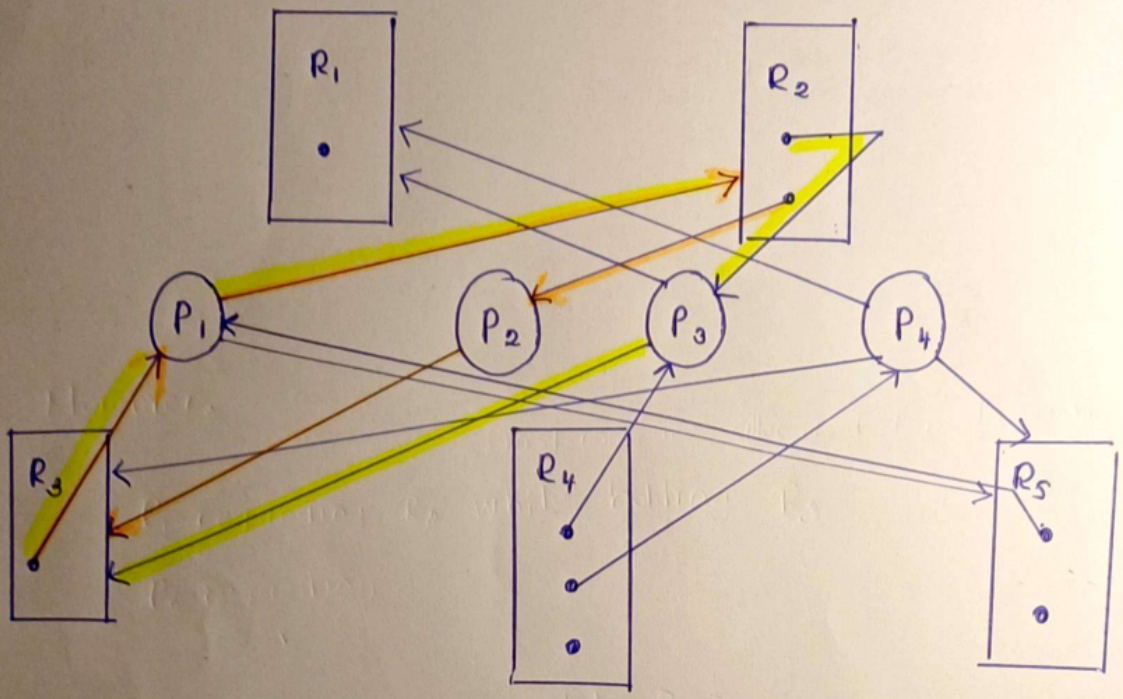
E = {(P1,R5) ,(P1,R2),(R3,P1),(R5,P1),(P2,R3),(R2,P2),(P3,R3),(P3,R1),(R2,P3), (R4,P3),(P4,R3),(P4,R1), (R4,P4), (P4, R5)}

The number of resource instances are

R1 has 1 instance
R2 has 2 instances
R3 has 1 instance
R4 has 3 instances
R5 has 2 instances

3.4) Using the resource graph that you have drawn in section 3.3, identify deadlock sequences (if any). Justify your answer.

Resource Allocation Graph:



This is the RAG we can draw based on the details provided in the question

---

To answer question 3.4, we need to identify whether there is a deadlock sequence or not. If a deadlock exists, we must justify the answer.

For that, we can use resource allocation graph reduction.

Therefore, let's ~~try to reduce the graph,~~ draw the resource allocation graph reduction diagram.

(*) Let's start with process that hold resources and don't request any resources.

Unfortunatly, there is no any processes, as I explain below :

- P_1 Holds R_3 and R_5

  BUT

  P_1 Requesting R_2 and R_5 [Yes, R_5 is already allocated to P_1, but still P_1 requesting R_2 & instances are allocated to P_2 & P_3]

- P_2 Holds R_2 BUT Requesting R_3 [P_2 can not be reduced yet. Since R_3 only has one instance & that is already allocated to P_1]

- P_3 Holds R_2 and R_4 BUT Requesting R_1 and R_3.

[It also can't reduced since R_3 only has one instance + that is already allocated to P_1]

- P_4 Holds R_4. BUT Requesting R_1, R_3 and R_5.

[It also can't reduced since R_3 only has one instance + that is already allocated to P_1]

Therefore, → (instances are allocated for P_2 and P_3)

P_1 requesting R_2 while holding R_3

P_2 requesting R_3 while holding R_2

→ (one instance allocated for P_1).

∴ there is a circular wait: $P_1 \rightarrow R_2 \rightarrow P_2 \rightarrow R_3 \rightarrow P_1$

P_1 requesting R_2, but R_2 allocated for P_2, but P_2 requesting R_3, but that is allocated for P_1.

P_1 not releasing R_3, since to complete the entire task of P_1 need R_2.

However, R_2 allocated for P_2, but P_2 can't completes its execution without R_3. Therefor P_1 + P_2 never releasing the resources.

Moreover,

P_1 requesting R_2 while holding R_3 → (instances are allocated for P_2 and P_3)

P_3 requesting R_3 while holding R_2 → (There is only one instance which is already allocated for P_1)

∴ there is a circular wait : $P_1 \rightarrow R_2 \rightarrow P_3 \rightarrow R_3 \rightarrow P_1$

P_1 requesting R_2, but R_2 allocated for P_3, but P_3 requesting R_3, but that is allocated for P1.

P_1 not releasing R_3, since to complete the entire task of P_1, need R_2. However, R_2 allocated for P_3, but P_3 can't complete its execution without R_3. Therefor P_1 & P_3 also never releasing the resources.

R_2 only has 2 instances
- given for P_2
- given for P_3

R_3 only has 1 instance
- given for P_1

} Therefor, these resources don't have free instances available for allocation when other processes are requesting them.

The system is in a deadlock state because there is a circular wait. Each process in the cycle is waiting for a resource held by the next process in the cycle. Therefore, no process can proceed.