Resource Allocation Graph (RAG)

Set of Vertices & Edges

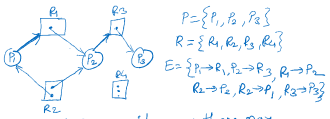
Set of Vertices (V)
- Set of process: $P = \{P_1, P_2, P_3 \ldots P_n\}$
- Set of Resources: $R = \{R_1, R_2, R_3, \ldots R_m\}$

Set of Edges
- $P_i \rightarrow R_j$
- $R_j \rightarrow P_i$

⇒ Resource type

⇒ process

R_1 (printer) (3 printers) — instances of Resource type



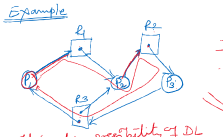
$P = \{P_1, P_2, P_3\}$

$R = \{R_1, R_2, R_3, R_4\}$

$E = \{P_1 \rightarrow R_1, P_2 \rightarrow R_3, R_1 \rightarrow P_2, R_2 \rightarrow P_2, R_2 \rightarrow P_1, R_3 \rightarrow P_3\}$

- If cycle exists in RAG, it means there may exists a Deadlock

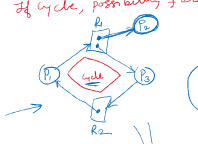
Example



Process	Allocation R1 R2 R3	Request R1 R2 R3	Availability R1 R2 R3
→ P_1	0 0 1	1 0 0	0 0 0
P_2	1 0 1	0 1 0	
P_3	0 1 0	0 0 1	

If cycle, possibility of DL

There is a Deadlock



Process	Allocation R1 R2	Request R1 R2	Availability R1 R2
P_1	0 1	1 0	0 0
P_2	1 0	0 0	1 0
P_3	1 0	0 1	1 1

Seq < P_2, P_1, P_3 >

NO Deadlock

Conclusion
1. If Graph contains no cycle ⇒ NO Deadlock
2. If Graph contains a cycle
   - If only one instance per resource, then Deadlock
   - If several instances per resource, then there is a possibility of Deadlock

✳ Methods to Handle Deadlock

1. Deadlock ignorance ✓
2. Deadlock prevention ✓
3. Deadlock Avoidance
4. Deadlock Detection & Recovery

1. Deadlock ignorance
- To add a code to OS
- Restart your m/c

2. Deadlock prevention

- ✳ Mutual Exclusion
- ✳ No preemption
- ✳ Hold & wait
- ✳ Circular wait

All (4)

Try to Remove all four condition OR any on condition

Share the Resources

Preemption

Time quantum = 4 unit