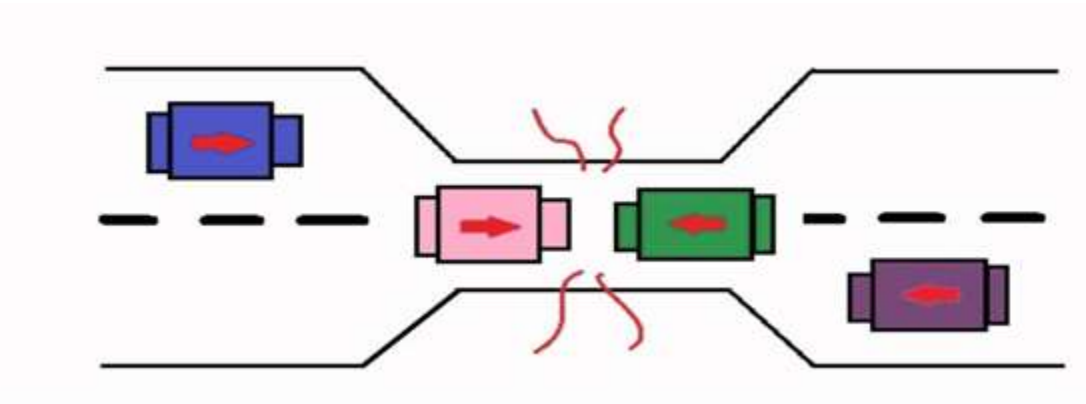
Deadlock

- A process is *deadlocked* if it is waiting for an event that will never occur

  Example 1

  - System has 2 tape drives. P1 and P2 each hold one tape drive and each needs the other one

Deadlock conditions(cont..)



- Assume traffic in one direction
  - Each section of the bridge is viewed as a resource
- If a deadlock occurs, it can be resolved only if one car backs up (preempt resources and rollback)
  - Several cars may have to be backed up if a deadlock occurs
  - Starvation is possible

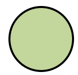
Deadlock conditions(cont..)

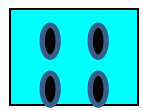
- The following 4 conditions are necessary and sufficient for deadlock (must hold simultaneously)
  - Mutual Exclusion:
    - Only one process at a time can use the resource
  - Hold and Wait:
    - Processes hold resources already allocated to them while waiting for other resources
  - No preemption:
    - Resources are released by processes holding them only after that process has completed its task
  - Circular wait:
    - A circular chain of processes exists in which each process waits for one or more resources held by the next process in the chain

Determine the deadlock occurrence in a system using resource-allocation graph

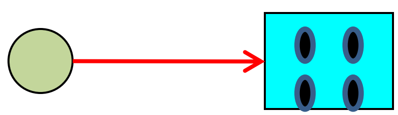
- A set of vertices V and a set of edges E
- V is partitioned into 2 types
  - P = {P1, P2,…,Pn} - the set of processes in the system
  - R = {R1, R2,…,Rn} - the set of resource types in the system
- Two kinds of edges
  - Request edge - Directed edge Pi ---> Rj
  - Assignment edge - Directed edge Rj ----> Pi

Resource Allocation Graph (cont..)

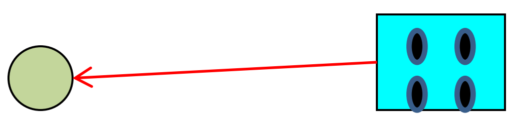
- Process 
- Resource type with 4 instances



- Pi requests instance of Rj



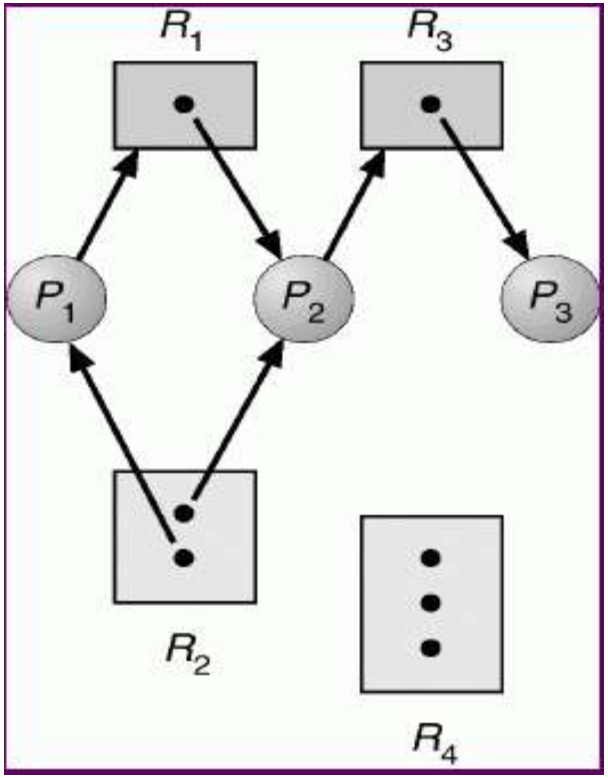
- Pi is holding an instance of Rj



Resource Allocation Graph (cont..)

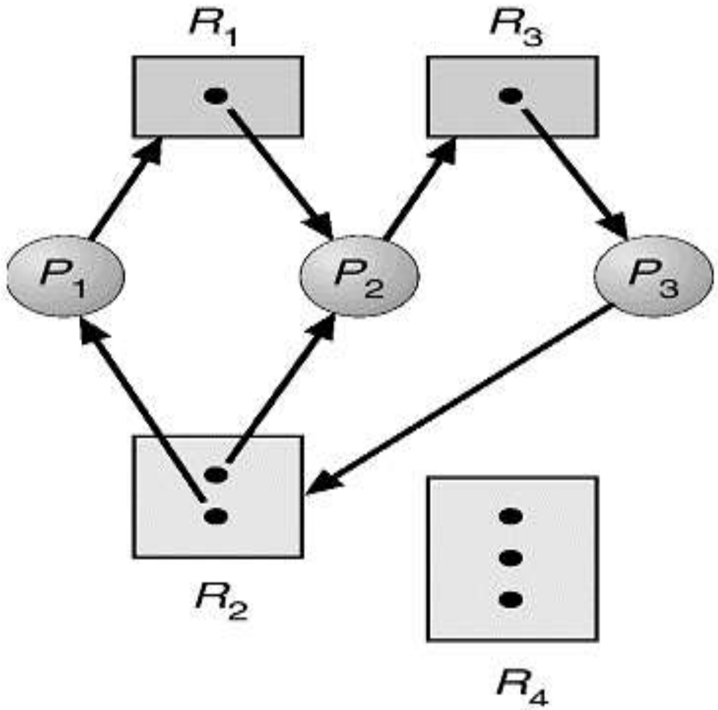
- Whenever there is no cycle in resource allocation graph ,deadlock will not occur.
- If there is a cycle in resource allocation graph deadlock may or may not occur.

Graph with no cycles



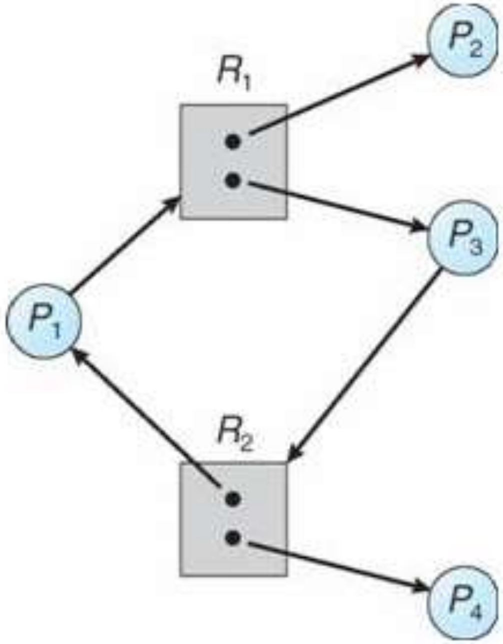
Graph with cycles and deadlock

Resource Allocation Graph with a Deadlock



Graph with cycle but no Deadlock

Resouce Allocation Graph with a cycle but no deadlock



- Prevention
  - Design the system in such a way that deadlocks can never occur
- Avoidance
  - Impose less stringent conditions than for prevention, allowing the possibility of deadlock but sidestepping it as it occurs
- Detection
  - Allow possibility of deadlock, determine if deadlock has occurred and which processes and resources are involved
- Recovery
  - After detection, clear the problem, allow processes to complete and resources to be reused. May involve destroying and restarting processes

Exemplify the deadlock prevention methods

Restrain ways in which requests can be made

Try to remove or make false all the condition

Try to remove or make false at least anyone of the condition

- Make Mutual Exclusion False
  - non-issue for sharable resources
  - cannot deny this for non-sharable resources
- Hold and Wait - guarantee that when a process requests a resource, it does not hold other resources
  - Force each process to acquire all the required resources at once. Process cannot proceed until all resources have been acquired
  - Low resource utilization, starvation possible

Deadlock prevention (cont..)

- No Preemption
  - Preempted resources are added to the list of resources for which the process is waiting
  - Process will be restarted only when it can regain its old resources as well as the new ones that it is requesting

Deadlock prevention (cont..)

- Circular Wait
  - Impose a total ordering of all resource types
  - Process can request the resources increasing order
  - Require that processes request resources in increasing order of enumeration; if a resource of type N is held, process can only request resources of types > N


Thank
You!!!