Assignment-2
Course: Operating Systems
Course code: CSE 309

Submitted to:
Suman Saha
Assistant Professor
Dept. Of CSE
Bangladesh University of Business and Technology

Submitted by:
Syeda Nowshin Ibnat
ID: 17183103020
Intake: 39
Section: 1
Program: B.Sc. in CSE

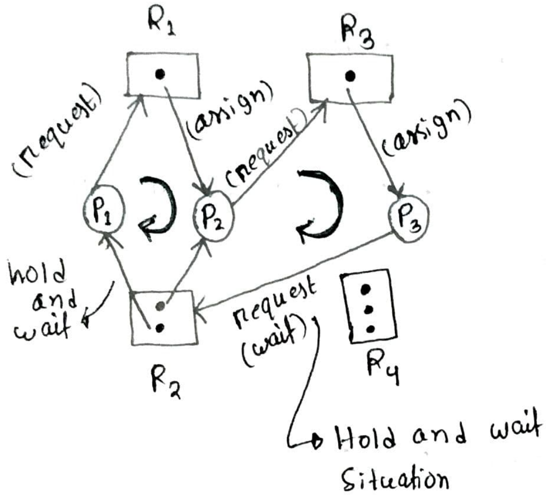
Date of submission: 15/03/2021

1 NO Question Answer

Part-1

Soln:

Resource Allocation Graph with a Deadlock



Here,

Vertices, V = 7

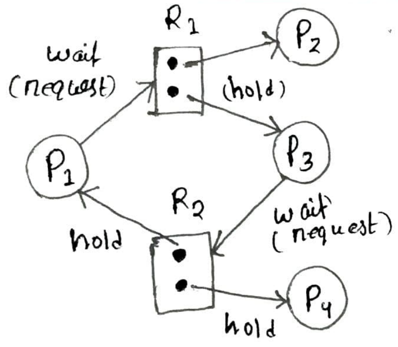
Assignment Edge = 4

Request Edge = 3

Instances: $R_1 = 1$
$R_2 = 2$
$R_3 = 1$
$R_4 = 3$

We can see from the graph that ~~here~~ there are two circular wait. So, there are deadlocks in this graph.

Resource Allocation Graph with a cycle but no deadlock



Here,

Vertices, V = 6

Assignment Edge = 4

Request Edge = 2

Instances: $R_1 = 2$
$R_2 = 2$

From the graph we can see P_4 is not waiting for any resource. ~~so~~, . So, it will release the instance which it is holding now. Now, if P_4 release the instance then P_3 can asign it (Hence, there are 2 instances). After P_4 releasing it and P_3 assiging it there will be no circular wait. So, there will be no deadlock.

Part - 2

There are two options for breaking Deadlock:

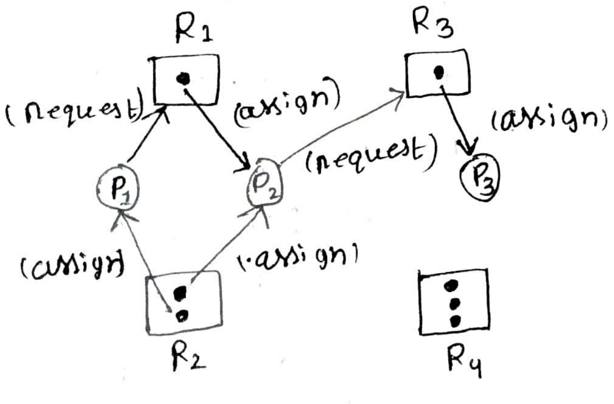
(1) One is simply to abort one or more processes to break the circular wait.

(2) The other is to preemt some resources from one or more of the ~~deak~~ deadlocked processes.

2 No question Answer

Soln: Yes, the system is deadlock free.

Proof by contradiction: Suppose the system is deadlock. This implies that each process is holding one resource and is waiting for one more. Since, there are three processes and four resources, one process must be able to obtain two resources. This process requires no more resources and therefore it will return its resources when done.



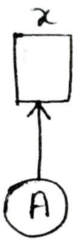
This system is deadlock free.

3 NO question Answer

Sol^n: Given, 3 processes: A, B, C

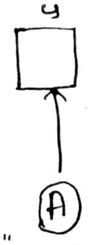
3 resources: x, y, z

(i)



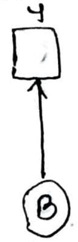
"A" requests x

(ii)



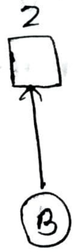
"A" requests y

(iii)



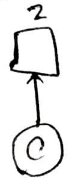
"B" requests y

(iv)



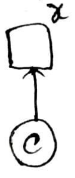
"B" requests 2

(v)



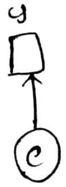
"C" requests z

(vi)



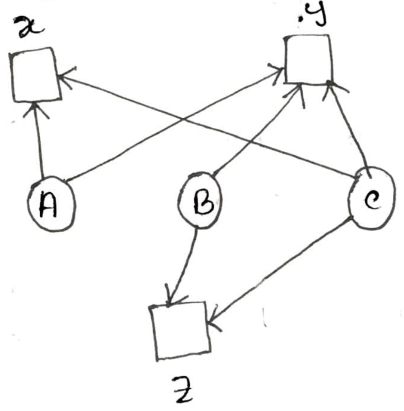
"C" requst x

(vii)



"C" requests y

Final Graph



Assuming that requested resources should always be allocated to the request process if it is available.

NO, this system is not in a deadlock.

4 No question Answer

Soln:

(i)

The values of Need of processes P_0 through P_4 respectively are (0,0,0,0), (0,7,5,0), (1,0,0,2), (0,0,2,0), (0,6,4,2)

$$\therefore \text{Need Matrix} : \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 7 & 5 & 0 \\ 1 & 0 & 0 & 2 \\ 0 & 0 & 2 & 0 \\ 0 & 6 & 4 & 2 \end{bmatrix}$$

We know,
Need = Max - Allocation

(ii)

Yes, with "available" be equal to (1,5,2,0) either process P_0 or P_3 could run. Once, process P_3 runs, it release it's resources, which allow all other existing processes to run.

(iii)

Yes, it can. This results in the value of "Available" being (1,1,0,0). One ordering of processes that can finish is P_0, P_2, P_3, P_1 and P_4.

5 No question

~~(a)~~ (i)

Soln:

The four necessary conditions for deadlock hold are:

1. Mutual exclusion: It is not required for sharable resources, must hold for non-sharable resources.
Here, only one car can occupy each intersection at a time.

2. Hold and wait: Here, cars can hold an intersection while waiting in a line for access to the next intersection.

3. No preemption: cars cann't be removed from their spot in the traffic flow, except by moving forward.

4. Circular wait: The set of cars in the deadlock situation includes the cars in the middle of the intersection.

(ii)

Soln: Install traffic lights that only allow flow in ~~one~~ one direction or other at a time.

We can still envision a deadlock if a city block is completely full of cars turning left or right. I think ~~you'd~~ we'd ~~will~~ would need to add a criteria to the problem requiring that cars will eventually leave the city block as well as prevent this.