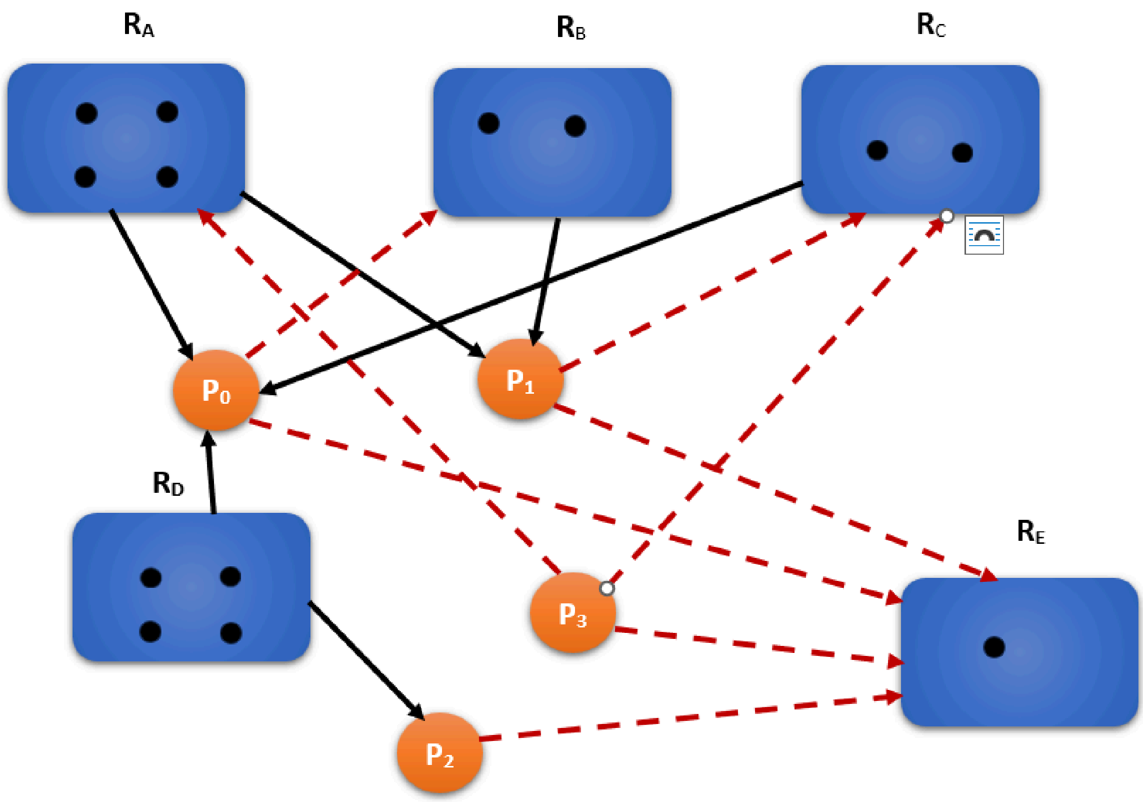
10.S Practical Solutions

Activity 1 - Resource Allocation Graph - Deadlock Detection



P0 Request = [0,1,0,0,1] ≤ Available = [2,1,1,2,1], we run P0 and reclaim it's Allocation.

New Allocation = Old Allocation +Available = [1,0,1,1,0] + [2,1,1,2,1] = [3,1,2,3,1]. The new matrix representation becomes:

PROCESS	ALLOCATION MATRIX A B C D E	REQUEST MATRIX A B C D E	AVAILABLE A B C D E
P0			**3 1 2 3 1**
P1	**1 1 0 0 0**	**0 0 1 0 1**	
P2	**0 0 0 1 0**	**0 0 0 0 1**	
P3	**0 0 0 0 0**	**1 0 1 0 1**	

Now, we have P1 Request = [0,0,1,0,1] ≤ Available = [3,1,2,3,1]. We can run P1, and reclaim its Allocation = [1,1,0,0,0].

New Available = [3,1,2,3,1] + [1,1,0,0,0] = [4,2,2,3,1]

P2 Request = [0,0,0,1,0] - Can execute and reclaim = [4,2,2,4,1]

P3 Request = [1,0,1,0,1] - Can execute and reclaim.

Activity 2 - Resource Allocation Graph - Deadlock Detection

A circuit does not necessarily mean a deadlock, although it will indicate the possibility of one. Here you'll notice that there is clearly a circuit from R5 to P1 to R2 to P2 back to R5. That means that there is circular wait. However, since R5 has two instances, each of which can be reserved, we need that second circuit from R5 to P0 to R2 to P2 back to R5. Since this completes a circuit which includes the same resource (R2), we have a deadlock.

Activity 3 - Banker's Algorithm

PROCESS	ALLOCATION MATRIX A B C D	MAX MATRIX A B C D	AVAILABLE A B C D
P0	**0 1 1 0**	**0 2 1 0**	**1 5 2 0**
P1	**1 2 3 1**	**1 6 5 2**	
P2	**1 3 6 5**	**2 3 6 6**	
P3	**0 6 3 2**	**0 6 5 2**	
P4	**0 0 1 4**	**0 6 5 6**	

Calculate the ***Need Matrix. (Need = Max - Allocation)***

NEED MATRIX

PROCESS	NEED A B C D
P0	**0 1 0 0**

P1	**0 4 2 1**
P2	**1 0 0 1**
P3	**0 0 2 0**
P4	**0 6 4 2**

Initially **Work = Available = [1, 5, 2, 0]**

Finish = False for all processes

Check each process and find **$\mathbf{Need_i}$ <= Work -** if found;

- Execute process, change **$\mathbf{Finish_i}$ = True**
- Release allocated resources for process.
- Change **Work = $\mathbf{Allocated_i}$ + Work.**

$\mathbf{Need_0}$ [0,1,0,0] <= [1,5,2,0]. $\mathbf{P_0}$ = True. Work = [1,5,2,0] + [0,1,1,0] = [1,6,3,0]

$\mathbf{Need_1}$ [0,4,2,1] <= [1,6,3,0]. $\mathbf{P_1}$ = False.

$\mathbf{Need_2}$ [1,0,0,1] <= [1,6,3,0]. $\mathbf{P_2}$ = False.

$\mathbf{Need_3}$ [0,0,2,0] <= [1,6,3,0]. $\mathbf{P_3}$ = True. Work = [1,6,3,0] + [0,6,3,2] = [1,12,6,2]

$\mathbf{Need_4}$ [0,6,4,2] <= [1,12,6,2]. $\mathbf{P_4}$ = True. Work = [1,12,6,2] + [0,0,1,4] = [1,12,7,6]

$\mathbf{Need_1}$ [0,4,2,1] <= [1,12,7,6]. $\mathbf{P_1}$ = True. Work = [1,12,7,6] + [1,2,3,1] = [2,14,10,7]

$\mathbf{Need_2}$ [1,0,0,1] <= [2,14,10,7]. $\mathbf{P_2}$ = True. Work = [2,14,10,7] + [1,3,6,5] = [3,17,16,12]

System in safe state and process will execute in order: <<$\mathbf{P_0}$, $\mathbf{P_3}$, $\mathbf{P_4}$, $\mathbf{P_1}$, $\mathbf{P_2}$>>

Activity 4 - Deadlock Detection Algorithm

PROCESS	ALLOCATION MATRIX A B C D	REQUEST MATRIX A B C D	AVAILABLE A B C
P0	**0 3 0 0**	**3 2 1 0**	**2 3 0 1**
P1	**1 0 1 1**	**2 2 0 0**	

P2	**0 2 1 0**	**3 5 3 1**	
P3	**2 2 3 0**	**0 4 1 1**	

Initially **Work = Available = [2,3,0,1]**

Finish = False for all processes

Find an index ***i, such that Finish$_i$ = False, and Request$_i$ <= Work***

Request$_0$ [3,2,1,0] <= [2,3,0,1] P$_0$ = False.

Request$_1$ [2,2,0,0] <= [2,3,0,1] P$_1$ = True. Work = [2,3,0,1] + [1,0,1,1] = [3,3,1,2]

Request$_2$ [3,5,3,1] <= [3,3,1,2] P$_2$ = False.

Request$_3$ [0,4,1,1] <= [3,3,1,2] P$_3$ = False.

Request$_0$ [3,2,1,0] <= [3,3,1,2] P$_0$ = True. Work = [3,3,1,2] + [0,3,0,0] = [3,6,1,2]

Request$_2$ [3,5,3,1] <= [3,6,1,2] P$_2$ = False.

Request$_3$ [0,4,1,1] <= [3,6,1,2] P$_3$ = True. Work = [3,6,1,2] + [2,2,3,0] = [5,8,4,2]

Request$_2$ [3,5,3,1] <= [5,8,4,2] P$_2$ = True. Work = [5,8,4,2] + [0,2,1,0] = [5,10,5,2]

System in safe state and process will execute in order: <<P$_1$, P$_0$, P$_3$, P$_2$>>
