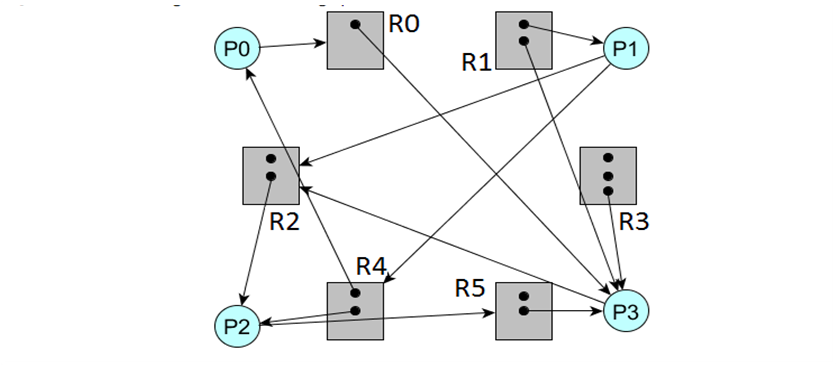
CSC310 Part 2



There is a cycle in this graph (being p2, to r5, to p3, to r2, back to p2), but this graph does not have a deadlock.

P0 has R4 and is requesting R0, and which is held by P3. P3 is requesting R2, and one is free and the other is taken by P2. The free slot is allocated to P3 and releases r2 and r0

P0 can get to r0 and r0 and r4 are freed up. Now p1 can get to r4 which was just freed and r2 has 2 free slots that p1 can use. P2 is requesting r5 which has free space so no problem and p3 is requesting for r2 which no other process is requesting at this point so there is no deadlock as everything is completed and no competing for resources.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Allocation | Max | Need | Available |
|  | A B C D | A B C D | A B C D | A B C D |
| P0 | 2 0 1 1 | 3 2 1 1 | 1 2 0 0 | 6 4 4 2 |
| P1 | 1 1 0 0 | 1 2 0 2 | 0 1 0 2 | 8 4 5 3 |
| P2 | 1 0 1 0 | 3 2 1 0 | 2 2 0 0 | 9 5 5 3 |
| P3 | 0 1 0 1 | 2 1 0 1 | 2 0 0 0 | 10 5 6 3 |

1. In need part of chart
2. Yes, as there is a safe order to do the processes, in this case, it’s actually in order being P0 P1 P2 P3, so it is in a safe state.
3. No, it is not deadlocked as the processes will execute in a safe order which in this case is the processes executing in order, as it is in a safe state, it will ensure that there is no deadlock as if a process cannot execute, it will go to the next one and come back later as to directly avoid deadlock.
4. No, it cannot as P3’s request of 2,1,0,0 is higher than the need as its need is 2,0,0,0 and its request need more resources than it actually needs so it will not be done.