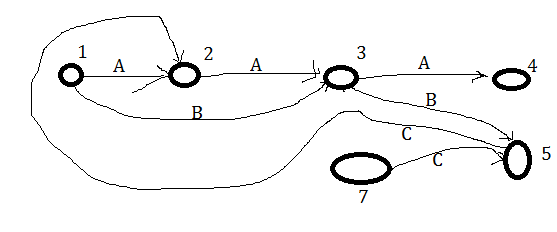
**Knowledge and Theory: Deadlock Prevention Problem**

**Question 16.4:** *Design a class which provides a lock only if there are no possible deadlocks*

1. We could request that all processes previously state which resource they will request and need later on. An example is shown below.

|  |
| --- |
| A = {1, *2, 3,* 4}  B = {1, 3, 5}  C = *{7, 5,* 2}   * In this case, there are three processes; A, B and C. A states that it will need resource {1,2,3, and 4}. When A is done with resource 1, it will request resource 2 (hold and and wait). Once A receives resource 2, it will release resource 1 and B will get that resource. * Note that each process takes with each process is unknown. Perhaps, B completes 1 and 3 before A is done with 2 or A completes 2 before B is done with resource 1. * Just by analyzing, you can find a deadlock situation. A could be locking resource 2 and waiting for resource 3. B could be locking resource 3 and waiting for resource 5. C could be locking resource 5 and waiting for resource 2. |

1. You can create where each resource is a vertex (so there are vertices 1,2,3,4,5 and 7) and each edge from resource 1 to resource 2 indicates that there is a process that will hold resource 1 and wait for resource 2. Let’s call this the deadlock detection graph.
2. If there is a cycle in the deadlock detection graph, that means there is a situation that deadlock can occur.



1. If there is a cycle in the deadlock detection graph, there is a situation where a deadlock can occur. For example, if there is a cycle {M,N,O,M…} this means that M will be holding and waiting for N, N will be holding and waiting for O, and O will be holding and waiting for M.
2. Now create a boolean method that takes in a graph and searches for cycles. If this returns false than you can return the lock.