(a) Consider these three transactions:

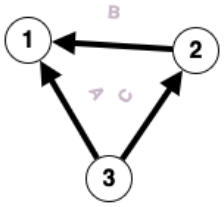
- *T1 : R1(A), R1(B), W1(A), W1(B), Co1*
- *T2 : R2(B), W2(B), R2(C), W2(C), Co2*
- *T3 : R3(C), W3(C), R3(A), W3(A), Co3*

i. Schedule 1:

R2(B), W2(B), R3(C), W3(C), R3(A), W3(A), Co3, R2(C), W2(C), Co2, R1(A), R1(B), W1(A), W1(B), Co1

Is this schedule conflict-serializable? If yes, indicate a serialization order.

Solution: yes: 3,2,1. No cycles in the graph.

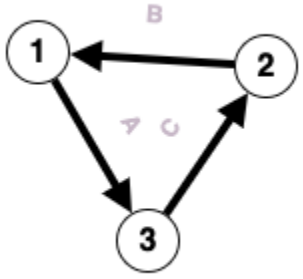


ii. Schedule 2:

R2(B), W2(B), R3(C), W3(C), R1(A), R1(B), W1(A), W1(B), Co1, R2(C), W2(C), Co2, R3(A), W3(A), Co3

Is this schedule conflict-serializable? If yes, indicate a serialization order.

Solution: no, Graph has cycles



(b) Consider the following three transactions:

- *T1 : R1(A), W1(B), C01*
- *T2 : R2(B), W2(C), C02*
- *T3 : R3(C), W3(D), C03*

Give an example of a conflict-serializable schedule that has the following properties: transaction T1 commits before transaction T3 starts, and the equivalent serial order is T3, T2, T1.

Solution:

- **R1(A), R2(B), W1(B), C01, R3(C), W2(C), C02, W3(D), C03**
- See slides for a visualization (starting slide 25): Section 6 - Hayoung Vidisha

Variations include:

- **Swap the first two reads (of A and B)**
- **Swap last two writes (of C and D, together with the commit order)**